# Effective implementation of algorithms (Master Thesis)

Generated by Doxygen 1.7.4

Wed May 4 2011 21:39:13

# Contents

# Chapter 1

# Namespace Index

## 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

# Chapter 2

# Class Index

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

# Chapter 5

# Namespace Documentation

## 5.1 automakefile Namespace Reference

### Functions

- def get_dependencies
- def get_binary
- def print_compile_rule
- def print_compiletest_rule
- def print_headers

### Variables

- list EXCLUDES = ['gtest']
- string TESTLIB = "TESTLIB=../../gtest/gtest-all.o ../../gtest/gtest_main.o"
- string CC = "CC=mingw32-g++"
- string OPT = "OPT=-g -O2 -W -Wall -Werror -Wextra -mno-cygwin"
- tuple all_files = os.listdir('.')
- tuple unittests = filter(lambda file : re.match('.*unittest.cpp$', file), all_files)
- tuple tests = filter(lambda file : re.match('.*_test.cpp$', file), all_files)
- tuple benchmarks = filter(lambda file : re.match('.*benchmark.cpp$', file), all_files)
- tuple compiletest = filter(lambda file : re.match('.*compiletest.cpp$', file), all_files)
- compilable = unittests+benchmarks+tests;
- tuple b = get_binary(filename)

### 5.1.1 Function Documentation

#### 5.1.1.1 def automakefile::get_binary ( *filename* )

5.1.1.2 def automakefile::get_dependencies ( *filename, parent* )

5.1.1.3 def automakefile::print_compile_rule ( *filename* )

5.1.1.4 def automakefile::print_compiletest_rule ( *filename* )

5.1.1.5 def automakefile::print_headers ( )

### 5.1.2 Variable Documentation

5.1.2.1 tuple **automakefile::all_files** = os.listdir('.')

5.1.2.2 tuple **automakefile::b** = get_binary(filename)

5.1.2.3 tuple **automakefile::benchmarks** = filter(lambda file : re.match('.*benchmark.cpp$', file), **all_files**)

5.1.2.4 string **automakefile::CC** = "CC=mingw32-g++"

5.1.2.5 **automakefile::compilable** = **unittests+benchmarks+tests**;

5.1.2.6 tuple **automakefile::compiletest** = filter(lambda file : re.match('.*compiletest.cpp$', file), **all_files**)

5.1.2.7 list **automakefile::EXCLUDES** = ['gtest']

5.1.2.8 string **automakefile::OPT** = "OPT=-g -O2 -W -Wall -Werror -Wextra -mno-cygwin"

5.1.2.9 string **automakefile::TESTLIB** = "TESTLIB=../../gtest/gtest-all.o ../../gtest/gtest_main.o"

5.1.2.10 tuple **automakefile::tests** = filter(lambda file : re.match('.*_test.cpp$', file), **all_files**)

5.1.2.11 tuple **automakefile::unittests** = filter(lambda file : re.match('.*unittest.cpp$', file), **all_files**)

## 5.2 balanced_structures Namespace Reference

### Namespaces

- namespace skiplist

## 5.3 balanced_structures::skiplist Namespace Reference

### Namespaces

- namespace node_utils
- namespace trail

### Classes

- class Skiplist
- struct ConstIterator
- class Node

### Typedefs

- typedef short LevelType

### Variables

- static const int LEVELUP_PROB = 100 / 4
- static const LevelType MAXLEVEL = 15

## 5.3.1 Typedef Documentation

#### 5.3.1.1 typedef short balanced_structures::skiplist::LevelType

## 5.3.2 Variable Documentation

#### 5.3.2.1 const int balanced_structures::skiplist::LEVELUP_PROB = 100 / 4 [static]

*LEVELUP_PROB* is probability of "increasing" level of a node in the percent. Thus, node of level *L* is created with probability *LEVELUP_PROB^(L - 1) (1 - LEVELUP_PROB)* . The average level of nodes in skiplist will be *1 / 0LEVELUP_PROB*

#### 5.3.2.2 const LevelType balanced_structures::skiplist::MAXLEVEL = 15 [static]

Maximum allowed level of a node.

Note: current value of 15 is pretty good upper bound if you are using *LEVELUP_PROB* = 4 and less than 10^8 nodes.

## 5.4 balanced_structures::skiplist::node_utils Namespace Reference

### Functions

- template<typename T >
  T randomLevel (Rand *rand, int levelup_prob_percent, T max_level)

### 5.4.1 Function Documentation

#### 5.4.1.1 template<typename T > T balanced_structures::skiplist::node_utils::randomLevel ( Rand * *rand*, int *levelup_prob_percent*, T *max_level* )

Generate level of node randomly with geometric distribution with probablity *p* percent. Note: The resulting distribution is geometric distribution with cutoff at *max_level* (higher levels are accumulated to this last)

$0 < p < 100$ .

**Parameters**

| | |
|---|---|
| *rand* | random number generator |
| *levelup_prob_percent* | probability of *level+1* over *level* |
| *max_level* | level cutoff |

**Returns**

integer, level of a node, $1 <= level <= max\_level$

## 5.5 balanced_structures::skiplist::trail Namespace Reference

### Classes

- struct Trail
- class TrailFunction
- class LowerBoundTrailFunction
- class UpperBoundTrailFunction
- class KthTrailFunction

### Typedefs

- typedef size_t SizeType

### 5.5.1 Typedef Documentation

#### 5.5.1.1 typedef size_t balanced_structures::skiplist::trail::SizeType

## 5.6 color Namespace Reference

### Enumerations

- enum Color { BLUE = 34, PINK = 35, CYAN = 36 }

### Functions

- void colorPrintf (Color color, const char *fmt,...)

### 5.6.1 Enumeration Type Documentation

#### 5.6.1.1 enum color::Color

Available colors for color printing

**Enumerator:**

***BLUE***

***PINK***

***CYAN***

### 5.6.2 Function Documentation

#### 5.6.2.1 void color::colorPrintf ( Color *color,* const char * *fmt,* ... )

Prints message in color. The usage is same as for the printf.

**Parameters**

| | |
|---|---|
| *color* | in which text should be displayed |
| *fmt* | format string |

## 5.7 geometry Namespace Reference

### Namespaces

- namespace two_d

## 5.8 geometry::two_d Namespace Reference

## Classes

- class ConvexHull
- struct LineSegment
- class Point

## Enumerations

- enum Quadrant {
  CENTER = 0, TOP_RIGHT = 1, TOP_LEFT = 2, BOTTOM_LEFT = 3,
  BOTTOM_RIGHT = 4 }
- enum IntersectType { NO_INTERSECT, INTERSECT, TANGENCY, OVERLAY }

## Functions

- template<typename T >
  Quadrant getQuadrant (const Point< T > point)
- template<typename T >
  bool angleLess (const Point< T > point1, const Point< T > point2)
- template<typename T >
  T sqrDistancePointPoint (Point< T > p1, Point< T > p2)
- template<typename T >
  long double distancePointPoint (Point< T > p1, Point< T > p2)
- template<typename T >
  math::rational::Rational< T > sqrDistancePointLine (Point< T > p, LineSegment< T > line)
- template<typename T >
  long double distancePointLine (Point< T > p, LineSegment< T > line)
- template<typename T >
  math::rational::Rational< T > sqrDistancePointLineSegment (Point< T > p, LineSegment< T > line)
- template<typename T >
  long double distancePointLineSegment (Point< T > p, LineSegment< T > line)
- template<typename T >
  bool pointOnLine (Point< T > p, LineSegment< T > s)
- template<typename T >
  bool pointOnLineSegment (Point< T > p, LineSegment< T > s, bool acceptCorners)
- template<typename T >
  IntersectType intervalIntersect (T a1, T a2, T b1, T b2)
- template<typename T >
  IntersectType intersectLineLineSegment (const LineSegment< T > &line, const LineSegment< T > &segment)
- template<typename T >
  IntersectType intersectLineSegmentLineSegment (const LineSegment< T > &segment1, const LineSegment< T > &segment2)

- template<typename T >
  bool operator== (const Point< T > &a, const Point< T > &b)
- template<typename T >
  bool operator!= (const Point< T > &a, const Point< T > &b)
- template<typename T >
  Point< T > operator+ (const Point< T > &a, const Point< T > &b)
- template<typename T >
  Point< T > operator- (const Point< T > &a, const Point< T > &b)
- template<typename T >
  Point< T > operator- (const Point< T > &a)
- template<typename T >
  Point< T > operator* (const Point< T > &a, T scalar)
- template<typename T >
  Point< T > operator/ (const Point< T > &a, T scalar)
- template<typename T >
  int signum (T n)

### 5.8.1 Enumeration Type Documentation

#### 5.8.1.1 enum geometry::two_d::IntersectType

**Enumerator:**

***NO_INTERSECT***

***INTERSECT***

***TANGENCY***

***OVERLAY***

#### 5.8.1.2 enum geometry::two_d::Quadrant

Quadrant of an point

Note that quadrants are ordered with increasing angle starting with top right quadrant = 1

**Enumerator:**

***CENTER*** Point is in the center

***TOP_RIGHT*** Up right quadrant, including x-axis, excluding y-axis

***TOP_LEFT*** Up left quadrant, including y-axis, excluding x-axis

***BOTTOM_LEFT*** Down left quadrant, including x-axis, excluding y-axis

***BOTTOM_RIGHT*** Down right quadrant, including y-axis, excluding x-axis

### 5.8.2 Function Documentation

#### 5.8.2.1 template<typename T > bool geometry::two_d::angleLess ( const Point< T > *point1*, const Point< T > *point2* )

Compare two vectors by angle with x-axis

**Returns**

true if angle of the first vector is less than angle of the second

#### 5.8.2.2 template<typename T > long double geometry::two_d::distancePointLine ( Point< T > *p*, LineSegment< T > *line* )

Calculate distance between point and line.

Note: We reuse code from sqrDistancePointLine() because we do not need to compute fraction value and the whole Rational class (and we won't have problems with overflows).

**Returns**

distance

#### 5.8.2.3 template<typename T > long double geometry::two_d::distancePointLineSegment ( Point< T > *p*, LineSegment< T > *line* )

Calculate distance between point and line segment.

Note: We don't reuse sqrDistancePointPoint because this whole function can be calculated in long doubles and won't have problems with Rational overflows.

**Returns**

distance

#### 5.8.2.4 template<typename T > long double geometry::two_d::distancePointPoint ( Point< T > *p1*, Point< T > *p2* )

Calculate distance between two points

**Returns**

distance

5.8.2.5 template<typename T > **Quadrant** geometry::two_d::getQuadrant ( const Point< T > *point* )

5.8.2.6 template<typename T > **IntersectType** geometry::two_d::intersectLineLineSegment ( const LineSegment< T > & *line,* const LineSegment< T > & *segment* )

Determine whether line intersects with line segment. Warning: On integer types, be sure to compute this in double-sized type.

5.8.2.7 template<typename T > **IntersectType** geometry::two_d::intersectLineSegmentLineSegment ( const LineSegment< T > & *segment1,* const LineSegment< T > & *segment2* )

Determine whether two line segments intersects. Warning: On integer types, be sure to compute this in double-sized type.

5.8.2.8 template<typename T > **IntersectType** geometry::two_d::intervalIntersect ( T *a1,* T *a2,* T *b1,* T *b2* )

Determine whether two closed intervals intersects. Note that we allow any ordering on interval ranges;

5.8.2.9 template<typename T > bool geometry::two_d::operator!= ( const Point< T > & *a,* const Point< T > & *b* )

5.8.2.10 template<typename T > **Point**<T> geometry::two_d::operator* ( const Point< T > & *a,* T *scalar* )

5.8.2.11 template<typename T > **Point**<T> geometry::two_d::operator+ ( const Point< T > & *a,* const Point< T > & *b* )

5.8.2.12 template<typename T > **Point**<T> geometry::two_d::operator- ( const Point< T > & *a* )

5.8.2.13 template<typename T > **Point**<T> geometry::two_d::operator- ( const Point< T > & *a,* const Point< T > & *b* )

5.8.2.14 template<typename T > **Point**<T> geometry::two_d::operator/ ( const Point< T > & *a,* T *scalar* )

5.8.2.15 template<typename T > bool geometry::two_d::operator== ( const Point< T > & *a,* const Point< T > & *b* )

#### 5.8.2.16 template<typename T > bool geometry::two_d::pointOnLine ( Point< T > *p*, LineSegment< T > *s* )

Determine whether point lies on line. Warning: On float numbers use distanceToLine-Segment instead and check for zero with epsilon error. Warning: On integer types, be sure to compute this in double-sized type.

#### 5.8.2.17 template<typename T > bool geometry::two_d::pointOnLineSegment ( Point< T > *p*, LineSegment< T > *s*, bool *acceptCorners* )

Determine whether point lies on line. Warning: On float numbers use distanceToLine-Segment instead and check for zero with epsilon error. Warning: On integer types, be sure to compute this in double-sized type.

#### 5.8.2.18 template<typename T > int geometry::two_d::signum ( T *n* )

#### 5.8.2.19 template<typename T > math::rational::Rational<T> geometry::two_d::sqrDistancePointLine ( Point< T > *p*, LineSegment< T > *line* )

Calculate sqaure of distance between point and line.

**Returns**

square of the distance

#### 5.8.2.20 template<typename T > math::rational::Rational<T> geometry::two_d::sqrDistancePointLineSegment ( Point< T > *p*, LineSegment< T > *line* )

Calculate square of the distance between point and line segment.

**Returns**

square of the distance

#### 5.8.2.21 template<typename T > T geometry::two_d::sqrDistancePointPoint ( Point< T > *p1*, Point< T > *p2* )

Calculate square of the distance between two points.

**Returns**

square of the distance.

## 5.9 heap Namespace Reference

### Functions

- template<typename T >
  T left (T x)
- template<typename T >
  T right (T x)
- template<typename T >
  T parent (T x)
- template<typename T >
  bool isLeftChild (T x)
- template<typename T >
  bool isRightChild (T x)
- template<typename T >
  T sibling (T x)
- template<typename T >
  T nextPowerOfTwo (T x)

### 5.9.1 Function Documentation

#### 5.9.1.1 template<typename T > bool heap::isLeftChild ( T *x* ) [inline]

**Returns**

true if this is left child

#### 5.9.1.2 template<typename T > bool heap::isRightChild ( T *x* ) [inline]

**Returns**

true if this is right child

#### 5.9.1.3 template<typename T > T heap::left ( T *x* ) [inline]

**Parameters**

| | |
|---|---|
| *x* | node_position |

**Returns**

position of node corresponding to left child

#### 5.9.1.4 template<typename T > T heap::nextPowerOfTwo ( T *x* )

Find smallest power of two that is at least x. Example: 2->2, 3->4, 4->4, 5->8

**Returns**

next power of two

**Exceptions**

| | |
|---|---|
| *overflow_error* | |

#### 5.9.1.5 template<typename T > T heap::parent ( T *x* ) [inline]

**Parameters**

| | |
|---|---|
| *x* | node_position |

**Returns**

position of the parent node

#### 5.9.1.6 template<typename T > T heap::right ( T *x* ) [inline]

**Parameters**

| | |
|---|---|
| *x* | node_position |

**Returns**

position of node corresponding to right child

#### 5.9.1.7 template<typename T > T heap::sibling ( T *x* ) [inline]

**Returns**

position of sibling of this node

## 5.10 interval_trees Namespace Reference

### Namespaces

- namespace fenwick
- namespace simple

### Classes

- class FullBinaryTree

### 5.10.1 Detailed Description

This is implementation of full binary tree

## 5.11 interval_trees::fenwick Namespace Reference

### Classes

- class FenwickTree
- struct BinaryPlus
- class FenwickSumTree
- struct BinaryMax
- class FenwickMaxTree

### Enumerations

- enum FenwickDirection { TO_ZERO, TO_INFTY }

### 5.11.1 Enumeration Type Documentation

#### 5.11.1.1 enum interval_trees::fenwick::FenwickDirection

Type of a fenwick tree, determines which type of range cna Fenwick tree query

**Enumerator:**

***TO_ZERO***

***TO_INFTY***

## 5.12 interval_trees::simple Namespace Reference

### Classes

- class SimpleMaxTree

## 5.13 math Namespace Reference

### Namespaces

- namespace binsearch
- namespace factorize
- namespace gcd

- namespace modular_inverse
- namespace powermod
- namespace prime_sieve
- namespace primes
- namespace rational

### 5.13.1 Detailed Description

This file contains implementation of function root/minimum finding algorithms based on binary search

This file holds an implementation of factorizing method with supplied "oracle" which can give one factor for each composite.

This file holds an implementation of the modular inverse computation modulo prime p by using Fermat's little theorem.

This file holds an implementation of the modular inverse computation modulo prime p by using Extended Euclid's algorithm computing greatest common divisor.

This file holds an implementation of the computation of inverse numbers modulo prime p

This file contains fast computation of power of a to b modulo m.

This file implements basic version of eratosthenes sieve.

This file implements basic segmented eratosthenes sieve. It's purpose is to find all primes up to n with space complexity O(sqrt(n)).

Implementation is based on paper The Segmented Sieve of Eratosthenes and Primes in Arithmetic Progressions to 10^18 by Carter Bays and Richard H. Hudsom

We used mainly algorithm B from this article.

This pragma is for removal of compile warnings for "denominator $<$ 0" when denominator is unsigned!

## 5.14 math::binsearch Namespace Reference

### Classes

- class Function
- class ConvexFunction
- class FunctionBinsearch

### Functions

- template$<$typename T $>$
  T range_middle (T left, T right)

- template<typename ValueType , typename SizeType >
  SizeType lower_bound (ValueType pole[], SizeType left, SizeType right, ValueType value)
- template<typename ValueType , typename SizeType >
  SizeType upper_bound (ValueType pole[], SizeType left, SizeType right, ValueType value)

### 5.14.1 Function Documentation

#### 5.14.1.1 template<typename ValueType , typename SizeType > SizeType math::binsearch::lower_bound ( ValueType *pole[],* SizeType *left,* SizeType *right,* ValueType *value* )

Find first index in array range *[left, right)* where the value may be inserted without violating the ordering

Note that the definition is same as "index of first element which is >= value" except that the result is *right* if no such value exists

Example:

```
    a = 1 1 2 2 2 3 5
lb(1) = ^
lb(2) =     ^
lb(4) =           ^
lb(6) =             ^ (==right)
```

**Precondition**

- sorted array
- *SizeType* is integral type
- *(right-left)* will fit into type *SizeType*

**Parameters**

| | |
|---|---|
| *left* | start of the interval |
| *right* | index after the end of the interval |

**Returns**

index of the binsearched value

#### 5.14.1.2 template<typename T > T math::binsearch::range_middle ( T *left,* T *right* )

Finds the middle of the range *[left, right)*

Middle is defined as *floor((left + right) / 2)*

**Precondition**

- *T* is integral type
- *(right-left)* is representable in type *T*

**Parameters**

| | |
|---|---|
| *left* | start of the interval |
| *right* | first index after the end of the interval |

**Returns**

midde of the interval

#### 5.14.1.3 template<typename ValueType , typename SizeType > SizeType math::binsearch::upper_bound ( ValueType *pole[]*, SizeType *left*, SizeType *right*, ValueType *value* )

Finds last position from range *[left, right)* where the value may be inserted without violating ordering

Note that the definitions is the same as "index of first element that is greater than value" except that the result is *right* if no such value exists

Example:

```
       1 1 2 2 3 5
ub(6)              ^ (== right)
ub(2)          ^
ub(1)      ^
ub(0)  ^
```

**Precondition**

- *(right-left)* should fit into ValueType
- *SizeType* should be integral type

**Parameters**

| | |
|---|---|
| *left* | start of the interval |
| *right* | index after the end of the interval |

**Returns**

index of the binsearched value

## 5.15 math::factorize Namespace Reference

### Classes

- class FactorizeNaive_
- class FactorizeWithOracle_
- class OracleBrent_
- class OraclePollard_

### Typedefs

- typedef FactorizeNaive_< int > FactorizeNaive
- typedef FactorizeWithOracle_< int, OraclePollard > FactorizePollard
- typedef FactorizeWithOracle_< int, OracleBrent > FactorizeBrent
- typedef OracleBrent_< math::powermod::PowermodExtended > OracleBrent
- typedef OraclePollard_< math::powermod::PowermodExtended > OraclePollard

#### 5.15.1 Typedef Documentation

##### 5.15.1.1 typedef FactorizeWithOracle_<int, OracleBrent> math::factorize::FactorizeBrent

##### 5.15.1.2 typedef FactorizeNaive_<int> math::factorize::FactorizeNaive

Default naive factorization with *CountType=int*

##### 5.15.1.3 typedef FactorizeWithOracle_<int, OraclePollard> math::factorize::FactorizePollard

##### 5.15.1.4 typedef OracleBrent_<math::powermod::PowermodExtended> math::factorize::OracleBrent

##### 5.15.1.5 typedef OraclePollard_<math::powermod::PowermodExtended> math::factorize::OraclePollard

## 5.16 math::gcd Namespace Reference

### Classes

- class ExtendedGCD
- class ExtendedGCDLoop

### Functions

- template<typename T >
  T gcd (T a, T b)

### 5.16.1 Function Documentation

#### 5.16.1.1 template<typename T > T math::gcd::gcd ( T *a*, T *b* )

## 5.17 math::modular_inverse Namespace Reference

### Classes

- class ModularInverseFermat_
- class ModularInverseGcd
- class ModularInversePrecomputed_

### Typedefs

- typedef ModularInverseFermat_< math::powermod::PowermodSimple > ModularInverseFermat
- typedef ModularInversePrecomputed_< math::powermod::PowermodSimple > ModularInversePrecomputed

### 5.17.1 Typedef Documentation

#### 5.17.1.1 typedef ModularInverseFermat_<math::powermod::PowermodSimple> math::modular_inverse::ModularInverseFermat

#### 5.17.1.2 typedef ModularInversePrecomputed_<math::powermod::PowermodSimple> math::modular_inverse::ModularInversePrecomputed

## 5.18 math::powermod Namespace Reference

### Classes

- class MultmodExtended
- class MultmodExtendedOpt
- class MultmodSimple
- class Powermod_

### Typedefs

- typedef Powermod_< MultmodSimple > PowermodSimple
- typedef Powermod_< MultmodExtendedOpt > PowermodExtended

### 5.18.1 Typedef Documentation

#### 5.18.1.1 typedef Powermod_<MultmodExtendedOpt> math::powermod::PowermodExtended

#### 5.18.1.2 typedef Powermod_<MultmodSimple> math::powermod::PowermodSimple

## 5.19 math::prime_sieve Namespace Reference

### Classes

- class EratosthenesBasic
- class EratosthenesOptimized
- class SieveCallback
- class SegmentedSieve

## 5.20 math::primes Namespace Reference

### Classes

- class PrimesBasic
- class PrimesFast_
- class PrimesSlow

### Typedefs

- typedef PrimesFast_< math::powermod::PowermodExtended > PrimesFast

### 5.20.1 Typedef Documentation

#### 5.20.1.1 typedef PrimesFast_<math::powermod::PowermodExtended> math::primes::PrimesFast

## 5.21 math::rational Namespace Reference

### Classes

- class Rational

## Functions

- template<typename T >
  Rational< T > operator- (const Rational< T > &a)
- template<typename T >
  bool operator== (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool operator< (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool operator> (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool operator<= (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool operator>= (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  std::ostream & operator<< (std::ostream &out, const Rational< T > &a)

### 5.21.1 Function Documentation

#### 5.21.1.1 template<typename T > Rational<T> math::rational::operator- ( const Rational< T > & *a* )

Unary minus

**Returns**

-a

#### 5.21.1.2 template<typename T > bool math::rational::operator< ( const Rational< T > & *a*, const Rational< T > & *b* )

Test inequality of fractions

**Returns**

true iff *a* is less than *b*

#### 5.21.1.3 template<typename T > std::ostream& math::rational::operator<< ( std::ostream & *out*, const Rational< T > & *a* )

Write fraction to stream in form "a/b" where a/b is normalized fraction

**Precondition**

- Fraction should be normalized, this hold unless you modify class members directly.

**Parameters**

| | |
|---|---|
| *out* | stream to be used |
| *a* | number to be written |

**Returns**

stream *out*

#### 5.21.1.4 template<typename T > bool math::rational::operator<= ( const Rational< T > & *a*, const Rational< T > & *b* )

Test inequality of fractions

**Returns**

true iff *a* is less or equal than *b*

#### 5.21.1.5 template<typename T > bool math::rational::operator== ( const Rational< T > & *a*, const Rational< T > & *b* )

Test equality of fractions

**Precondition**

- Fractions should be normalized first, this is done automatically unless you modify fraction's member variables

**Returns**

true iff *a* is equal to *b*

#### 5.21.1.6 template<typename T > bool math::rational::operator> ( const Rational< T > & *a*, const Rational< T > & *b* )

Test inequality of fractions

**Returns**

true iff *a* is greater than *b*

#### 5.21.1.7 template<typename T > bool math::rational::operator>= ( const Rational< T > & *a*, const Rational< T > & *b* )

Test inequality of fractions

**Returns**

true iff *a* is greated or equals to *b*

## 5.22 strings Namespace Reference

### Namespaces

- namespace cyclic
- namespace lcs
- namespace search
- namespace search_callback
- namespace suffix_array
- namespace utils

### Classes

- class TestdataFiles
- class PatternFiles

### 5.22.1 Detailed Description

This file holds classes caculating longest common subsequence of two sequences.

Implementation of the suffix array search

Time: O(p log n) worst case where p is length of the pattern and n is length of the suffix array

This file holds Rabin-Karp string search algorithm Implementation of the Rabin-Karp algorithm from article Efficient randomized pattern-matching algorithms by Karp, Richard M. and Rabin, Michael O.

This file holds and rolling-hash function implementation used by Rabin-Karp string matching algorithm.

This file contains basic string-search algorithm using suffix arrays

This file holds a checker of suffix array consistency.

The check is based on paper Fast Lightweight Suffix Array Construction and Checking by Stefan Burkhardt and Juha Kärkkäinen

Implementation of suffix array LCP computation in linear time from article [ltLCP] Two space saving tricks for linear time LCP computation by Giovanni Manzini

This file holds implementation to O(n log n) suffix array generation from article Suffix arrays: A new method for on-line string searches by Udi Manber, Gene Myers

Implementation of the suffix array creation by naive sorting method.

Time: O(n^2 log n) worst case when there are suffixes with long same prefixes

Warning: This is only internal testing implementation and running time might be really big. Use other implementations instead.

This file contains testdata filename constants

## 5.23 strings::cyclic Namespace Reference

### Classes

- class Duval

## 5.24 strings::lcs Namespace Reference

### Classes

- class LCS
- class LCSHirschberg

## 5.25 strings::search Namespace Reference

### Classes

- class KMP
- class RabinKarp
- class RollingHash

## 5.26 strings::search_callback Namespace Reference

### Classes

- class SearchCallback

## 5.27 strings::suffix_array Namespace Reference

### Classes

- class SearchHelper
- class Binsearch
- class SuffixArrayChecker
- class LCPKasai
- class LCPManzini
- class LCPNaive
- class ManberMyersLog2_
- class ManberMyers
- class NaiveSuffixArray
- class SortHelper

### Typedefs

- typedef ManberMyersLog2_< int > ManberMyersLog2

#### 5.27.1 Typedef Documentation

##### 5.27.1.1 typedef ManberMyersLog2_<int> strings::suffix_array::ManberMyersLog2

## 5.28 strings::utils Namespace Reference

### Classes

- class SequenceHelper
- class SequenceLoader

## 5.29 testdata Namespace Reference

### Variables

- long long int prime_twins_count [][2]
- long long int prime_count_small [][2]
- long long int prime_count_big [][2]

#### 5.29.1 Detailed Description

This file holds some explicit counts of primes in specific ranges.

It is used only for unittesting implementations.

#### 5.29.2 Variable Documentation

##### 5.29.2.1 long long int testdata::prime_count_big[][2]

**Initial value:**

```
{
   {123456, 11601},
   {1234567, 95360},
   {12345678, 809227},
   {123456789, 7027260},
   {1234567890, 62106578},



   {0, 0}
 }
```

Number of primes in range [0, x)

Data based on direct computation with Mathematica 5

#### 5.29.2.2 long long int testdata::prime_count_small[ ][2]

**Initial value:**

```
{
   {3, 1},
   {4, 2},
   {5, 2},
   {6, 3},
   {7, 3},
   {8, 4},
   {20, 8},
   {155, 36},
   {3331, 469},
   {12345, 1474},
   {0, 0}
 }
```

Number of primes in range [2, x)

Data manually generated and generated with Mathematica 5

#### 5.29.2.3 long long int testdata::prime_twins_count[ ][2]

**Initial value:**

```
{
   {3, 0},
   {4, 0},
   {5, 0},
   {6, 1},
   {10, 2},
   {100, 8},
   {1000, 35},
   {10000, 205},
   {100000, 1224},
   {1000000, 8169},
   {10000000, 58980},
   {100000000, 440312},
   {1000000000, 3424506},
   {0, 0}
 }
```

Numer of twin primes (p and p+2 are both primes) up to specified number.

Data based on http://www.trnicely.net/twins/twins2.html

## 5.30 utils Namespace Reference

### Namespaces

- namespace benchmark
- namespace memory_usage
- namespace static_assert_
- namespace timer

## 5.31 utils::benchmark Namespace Reference

### Functions

- void printBenchmarkResults (long long int times, double run_time_sec, const char *function_str)

### Variables

- const double MIN_BENCHMARK_TIME = 1.5

### 5.31.1 Function Documentation

#### 5.31.1.1 void utils::benchmark::printBenchmarkResults ( long long int *times*, double *run_time_sec*, const char * *function_str* )

**Parameters**

| | |
|---|---|
| *times* | How many times the test was run |
| *run_time_sec* | Run time of the test in seconds |
| *function_str* | string containing the name of the function and arguments |

### 5.31.2 Variable Documentation

#### 5.31.2.1 const double utils::benchmark::MIN_BENCHMARK_TIME = 1.5

Minimum time (in seconds) for benchmark to have usedul results.

Note: current implementation of benchmark is using system timer to determine running time. This have the best resolution of 16ms and may wildly change, the present constant is conservative for reliable benchmarking

## 5.32 utils::memory_usage Namespace Reference

### Functions

- int getUsedMemoryKb ()

### 5.32.1 Function Documentation

#### 5.32.1.1 int utils::memory_usage::getUsedMemoryKb ( )

Return the current allocated memory.

**Warning**

this only counts memory allocated by malloc() (includes all STL structures and new() operator)
does not include shared objects
real memory usage may be bigger, because malloc() can keep some pieces of memory for later reuse

**Returns**

currently used memory in kilobytes

## 5.33 utils::static_assert_ Namespace Reference

### Classes

- struct STATIC_ASSERTION_FAILURE< true >
- struct static_assert_test

## 5.34 utils::timer Namespace Reference

### Classes

- class Timer

# Chapter 6

# Class Documentation

## 6.1 interval_trees::fenwick::BinaryMax< T > Struct Template Reference

`#include <fenwick.h>`

### Static Public Member Functions

- static T operation (const T &x, const T &y)

template<typename T> struct interval_trees::fenwick::BinaryMax< T >

### 6.1.1 Member Function Documentation

#### 6.1.1.1 template<typename T > static T interval_trees::fenwick::BinaryMax< T >::operation ( const T & *x*, const T & *y* ) [inline, static]

The documentation for this struct was generated from the following file:

- src/interval_trees/fenwick/fenwick.h

## 6.2 interval_trees::fenwick::BinaryPlus< T > Struct Template Reference

`#include <fenwick.h>`

### Static Public Member Functions

- static T operation (const T &x, const T &y)

### 6.2.1 Detailed Description

**template<typename T>struct interval_trees::fenwick::BinaryPlus< T >**

Fenwick operation for sums

### 6.2.2 Member Function Documentation

#### 6.2.2.1 template<typename T > static T interval_trees::fenwick::BinaryPlus< T >::operation ( const T & *x*, const T & *y* ) [inline, static]

The documentation for this struct was generated from the following file:

- src/interval_trees/fenwick/fenwick.h

## 6.3 strings::suffix_array::Binsearch Class Reference

`#include <binsearch.h>`

### Static Public Member Functions

- template<typename _Iterator , typename _PatternIterator >
  static void searchSuffixArray (_Iterator first, _Iterator last, const std::vector< int > &array, _PatternIterator pattern_first, _PatternIterator pattern_last, strings::search_callback::SearchCallback< _Iterator > *callback)

### 6.3.1 Detailed Description

Implementation of the suffix array search using bin-search algorithm

Time: <it>O(p log n)</it> worst case where *p* is the length of the pattern and *n* is the length of the suffix array

### 6.3.2 Member Function Documentation

#### 6.3.2.1 template<typename _Iterator , typename _PatternIterator > static void strings::suffix_array::Binsearch::searchSuffixArray ( _Iterator *first*, _Iterator *last*, const std::vector< int > & *array*, _PatternIterator *pattern_first*, _PatternIterator *pattern_last*, strings::search_callback::SearchCallback< _Iterator > * *callback* ) [inline, static]

Search suffix array and report any matches via #callback

**Precondition**

- <it> [first, last) </it> is valid range

- <it> [pattern_first, pattern_last) </it> is valid range
- *#array* is suffix array of sequence <it> [first, last) </it>

**Parameters**

| | |
|---|---|
| *first* | points to the start of the sequence |
| *last* | points the the element after the end of the sequence |
| *array* | suffix array corresponding to [first, last) |
| *pattern_first* | points to the start of the pattern |
| *pattern_last* | points to the element after the end of the pattern |
| *callback* | to be called for each match |

The documentation for this class was generated from the following file:

- src/strings/suffix_array_binsearch/binsearch.h

## 6.4 balanced_structures::skiplist::ConstIterator< T > Struct Template Reference

`#include <skiplist_iterator.h>`

### Public Types

- typedef const T value_type
- typedef const T & reference
- typedef const T * pointer
- typedef ConstIterator< T > self

### Public Member Functions

- ConstIterator ()
- ConstIterator (Node< T > *x)
- reference operator* () const
- Node< T > * getNode ()
- self & operator++ ()
- self operator++ (int)
- self & operator-- ()
- self operator-- (int)
- bool operator== (const self &x) const
- bool operator!= (const self &x) const

### Private Attributes

- Node< T > * node

### 6.4.1 Detailed Description

template<typename T>struct balanced_structures::skiplist::ConstIterator< T >

(Bidirectional) iterator over skiplist nodes.

### 6.4.2 Member Typedef Documentation

#### 6.4.2.1 template<typename T > typedef const T* balanced_structures::skiplist::ConstIterator< T >::pointer

#### 6.4.2.2 template<typename T > typedef const T& balanced_structures::skiplist::ConstIterator< T >::reference

#### 6.4.2.3 template<typename T > typedef ConstIterator<T> balanced_structures::skiplist::ConstIterator< T >::self

#### 6.4.2.4 template<typename T > typedef const T balanced_structures::skiplist::ConstIterator< T >::value_type

### 6.4.3 Constructor & Destructor Documentation

#### 6.4.3.1 template<typename T > balanced_structures::skiplist::ConstIterator< T >::ConstIterator ( ) [inline]

Constructor

#### 6.4.3.2 template<typename T > balanced_structures::skiplist::ConstIterator< T >::ConstIterator ( Node< T > * x ) [inline, explicit]

Constructor, iterator pointing to specified node

### 6.4.4 Member Function Documentation

#### 6.4.4.1 template<typename T > Node<T>* balanced_structures::skiplist::ConstIterator< T >::getNode ( ) [inline]

**Returns**

corresponding skiplist node

#### 6.4.4.2 template<typename T > bool balanced_structures::skiplist::ConstIterator< T >::operator!= ( const self & *x* ) const [inline]

**Returns**

true iff these two iterators are pointing to different elements

#### 6.4.4.3 template<typename T > reference balanced_structures::skiplist::ConstIterator< T >::operator* ( ) const [inline]

dereferences an iterator, returns value

#### 6.4.4.4 template<typename T > self balanced_structures::skiplist::ConstIterator< T >::operator++ ( int ) [inline]

post-increment *iterator++*

#### 6.4.4.5 template<typename T > self& balanced_structures::skiplist::ConstIterator< T >::operator++ ( ) [inline]

pre-increment *++iterator*

#### 6.4.4.6 template<typename T > self balanced_structures::skiplist::ConstIterator< T >::operator-- ( int ) [inline]

post-decrement *iterator--*

#### 6.4.4.7 template<typename T > self& balanced_structures::skiplist::ConstIterator< T >::operator-- ( ) [inline]

pre-decrement *--iterator*

#### 6.4.4.8 template<typename T > bool balanced_structures::skiplist::ConstIterator< T >::operator== ( const self & *x* ) const [inline]

**Returns**

true if the iterators are pointing to the same element

### 6.4.5 Member Data Documentation

#### 6.4.5.1 template<typename T > Node<T>* balanced_structures::skiplist::ConstIterator< T >::node [private]

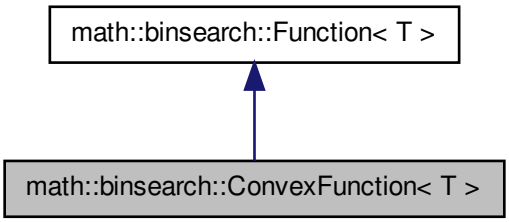
The documentation for this struct was generated from the following file:

- src/balanced_structures/skiplist/skiplist_iterator.h

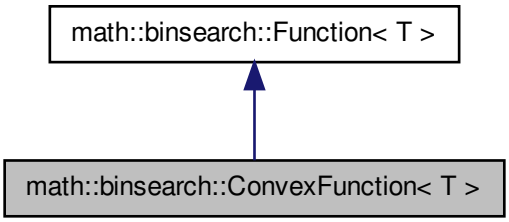
## 6.5 math::binsearch::ConvexFunction< T > Class Template Reference

`#include <function_binsearch.h>`

Inheritance diagram for math::binsearch::ConvexFunction< T >:




Collaboration diagram for math::binsearch::ConvexFunction< T >:



### 6.5.1 Detailed Description

template<typename T>class math::binsearch::ConvexFunction< T >

Base class of all convex functions

Function is convex, iff *f(t * x1 + (1 - t) * x2) <= t * f(x1) + (1 - t) * f(x2)* for every *x1 != x2* and *0 < t < 1*

The documentation for this class was generated from the following file:

- src/math/binsearch/function_binsearch.h

## 6.6 geometry::two_d::ConvexHull< T > Class Template Reference

```
#include <convex_hull.h>
```

### Classes

- class PointCompare

### Public Types

- typedef Point< T > PointType

### Public Member Functions

- void clear ()
- void addPoint (const PointType point)
- std::vector< PointType > convexHull ()

### Private Member Functions

- std::vector< PointType > computeChain (const std::vector< PointType > data)
- std::vector< PointType > rotate180 (const std::vector< PointType > data)

### Private Attributes

- std::vector< PointType > data

### 6.6.1 Detailed Description

template<typename T>class geometry::two_d::ConvexHull< T >

Class computing the convex hull of a set of points

### 6.6.2 Member Typedef Documentation

#### 6.6.2.1 template<typename T > typedef Point<T> geometry::two_d::ConvexHull< T >::PointType

### 6.6.3 Member Function Documentation

#### 6.6.3.1 template<typename T > void geometry::two_d::ConvexHull< T >::addPoint ( const PointType *point* ) [inline]

Add a new point.

#### 6.6.3.2 template<typename T > void geometry::two_d::ConvexHull< T >::clear ( ) [inline]

Removes all points.

#### 6.6.3.3 template<typename T > std::vector<PointType> geometry::two_d::ConvexHull< T >::computeChain ( const std::vector< PointType > *data* ) [inline, private]

Returns monotone chain

Chain starts at leftmost point (in case of more leftmost points the one with smallest y coordinate), it is upper half of the convex hull and ends in rightmost (in case of tie the topmost one of them) point

**Precondition**

- data sorted ascending by *x* coordinate, then *y* coordinate

**Returns**

upper monotone chain

#### 6.6.3.4 template<typename T > std::vector<PointType> geometry::two_d::ConvexHull< T >::convexHull ( ) [inline]

Compute convex hull

Note: Convex hull of an empty set is an empty set.

#### 6.6.3.5 template<typename T > std::vector<PointType> geometry::two_d::ConvexHull< T >::rotate180 ( const std::vector< PointType > *data* ) [inline, private]

Rotates all points by 180 degrees

### 6.6.4 Member Data Documentation

#### 6.6.4.1 template<typename T > std::vector<PointType> geometry::two_d::ConvexHull< T >::data [private]

inserted points.

The documentation for this class was generated from the following file:

- src/geometry/two_d/convex_hull.h

## 6.7 strings::cyclic::Duval< T > Class Template Reference

`#include <duval.h>`

### Public Types

- typedef int SizeType

### Static Public Member Functions

- static void minimumSuffixes (T *sequence, SizeType length, std::vector< int > *out)
- static int leastCyclicShift (T *sequence, int length)
- static int leastCyclicShiftEmaxx (T *sequence, int length)

template<typename T> class strings::cyclic::Duval< T >

### 6.7.1 Member Typedef Documentation

#### 6.7.1.1 template<typename T > typedef int strings::cyclic::Duval< T >::SizeType

### 6.7.2 Member Function Documentation

#### 6.7.2.1 template<typename T > static int strings::cyclic::Duval< T >::leastCyclicShift ( T * *sequence*, int *length* ) [inline, static]

Find lexicographically minimal cyclic shift of the sequence.

This is almost identical to "factor" algorithm 2.1 from article, but we incorporated concatenation(sequence, sequence) into the algorithm instead of explicitly concatening sequences and running original algorithm. The idea is based on the last proposition in the article

**Parameters**

| | |
|---|---|
| *sequence* | |
| *length* | |

**Returns**

index of the start of the minimal lexicographic cyclic shift of the sequence. For example result 2 means that smallest shift is sequence(s[2],s[3], ..., s[0],s[1])

#### 6.7.2.2 template<typename T > static int strings::cyclic::Duval< T >::leastCyclicShiftEmaxx ( T * *sequence*, int *length* ) [inline, static]

Find lexicographically minimal cyclic shift of the sequence.

This implementation is based on http://e-maxx.ru/algo/duval_algorithm

**Parameters**

| | |
|---|---|
| *sequence* | |
| *length* | |

**Returns**

index of the start of the minimal lexicographic cyclic shift of the sequence. For example result 2 means that smallest shift is sequence(s[2],s[3], ..., s[0],s[1])

#### 6.7.2.3 template<typename T > static void strings::cyclic::Duval< T >::minimumSuffixes ( T * *sequence*, SizeType *length*, std::vector< int > * *out* ) [inline, static]

Compute lexicographically minimal suffix for each prefix of the input sequence.

Based on pseudocode of algorithm 3.1 from the Duval's article. Note that "end of the string" is smaller than any letter, i.e. "x" < "xa"

**Parameters**

| | |
|---|---|
| *sequence* | |
| *length* | length of the sequence |
| *out* | start positions of suffixes of each prefix. For example, out[2] holds starting index of smallest suffix for sequence seq[0],seq[1],seq[2] |

The documentation for this class was generated from the following file:

- src/strings/cyclic/duval.h

## 6.8 math::prime_sieve::EratosthenesBasic Class Reference

`#include <eratosthenes_basic.h>`

### Public Member Functions

- template<typename T >
  void initialize (T t_size)
- template<typename T >
  bool isPrime (T p)

### Private Types

- typedef std::vector< bool >::size_type SizeType

### Private Attributes

- std::vector< bool > data

## 6.8.1 Detailed Description

The basic implementation of eratosthenes sieve.

Usage: EratosthenesBasic e; e.initialize(100); cout << "7:"<< e.isPrime(7) << " 10:" << e.isPrime(10);

## 6.8.2 Member Typedef Documentation

### 6.8.2.1 typedef std::vector<bool>::size_type math::prime_sieve::EratosthenesBasic::SizeType [private]

Internal type of the size index

## 6.8.3 Member Function Documentation

### 6.8.3.1 template<typename T > void math::prime_sieve::EratosthenesBasic::initialize ( T *t_size* ) [inline]

Initialize eratosthenes sieve to use size up to specified size.

**Parameters**

| | |
|---|---|
| *T* | size size of the input |

#### 6.8.3.2 template<typename T > bool math::prime_sieve::EratosthenesBasic::isPrime ( T *p* ) [inline]

**Parameters**

| | |
|---|---|
| *p* | - number to test |

**Returns**

true if p is prime

### 6.8.4 Member Data Documentation

#### 6.8.4.1 std::vector<bool> math::prime_sieve::EratosthenesBasic::data [private]

The documentation for this class was generated from the following file:

- src/math/prime_sieve/eratosthenes_basic.h

## 6.9 math::prime_sieve::EratosthenesOptimized Class Reference

`#include <eratosthenes_optimized.h>`

### Public Member Functions

- void initialize (int size_)
- bool isPrime (int p)

### Private Attributes

- std::vector< bool > data
- int size

### 6.9.1 Member Function Documentation

#### 6.9.1.1 void math::prime_sieve::EratosthenesOptimized::initialize ( int *size_* ) [inline]

#### 6.9.1.2 bool math::prime_sieve::EratosthenesOptimized::isPrime ( int *p* ) [inline]

### 6.9.2 Member Data Documentation

#### 6.9.2.1 std::vector<bool> math::prime_sieve::EratosthenesOptimized::data [private]

#### 6.9.2.2 int math::prime_sieve::EratosthenesOptimized::size [private]

The documentation for this class was generated from the following file:

- src/math/prime_sieve/eratosthenes_optimized.h

## 6.10 math::gcd::ExtendedGCD Class Reference

`#include <extended_gcd.h>`

### Static Public Member Functions

- template<typename T >
  static std::pair< T, T > extended_gcd_positive (T a, T b)
- template<typename T >
  static std::pair< T, T > extended_gcd (T a, T b)

### 6.10.1 Detailed Description

Class computing extended GCD using recursion

### 6.10.2 Member Function Documentation

#### 6.10.2.1 template<typename T > static std::pair<T, T> math::gcd::ExtendedGCD::extended_gcd ( T *a*, T *b* ) [inline, static]

Returns extended gcd, i.e. *pair(x, y) such that* $x * a + y * b == gcd(a, b)$

*Note: that returned values* ***can*** *be negative*

*Note: for both zero arguments function will return (0,0)*

***Warning***

*type T can be only a signed integer*
*Note that computation of* $a*x+b*y$ *may overflow for partial results but the result of whole computation is correct!*

***Parameters***

| | |
|---|---|
| a | |
| b | |

***Returns***

*(x,y) such that* $a*x+b*y==gcd(a,b)$

#### 6.10.2.2 template<typename T > static std::pair<T, T> math::gcd::ExtendedGCD::extended_gcd_positive ( T *a,* T *b* ) [inline, static]

Returns extended gcd, i.e. *pair(x, y) such that x ∗ a + y ∗ b == gcd(a, b)*

*Note: that returned values* ***can*** *be negative*

*Note: for both zero arguments function will return (0,0)*

***Warning***

*type T can be only a signed integer*
*Note that computation of a∗x+b∗y may overflow for partial results but the result of whole computation is correct!*

***Parameters***

| | |
|---|---|
| a | *non-negative integer* |
| b | *non-negative integer* |

***Returns***

*(x,y) such that a∗x+b∗y==gcd(a,b)*

The documentation for this class was generated from the following file:

- src/math/gcd/extended_gcd.h

## 6.11 math::gcd::ExtendedGCDLoop Class Reference

`#include <extended_gcd_loop.h>`

### Static Public Member Functions

- template<typename T >
  static std::pair< T, T > extended_gcd_positive (T a, T b)

### 6.11.1 Detailed Description

Class computing extended GCD using loop

### 6.11.2 Member Function Documentation

#### 6.11.2.1 template<typename T > static std::pair<T, T> math::gcd::ExtendedGCDLoop::extended_gcd_positive ( T *a,* T *b* ) [inline, static]

Returns extended gcd, i.e. *pair(x, y) such that x ∗ a + y ∗ b == gcd(a, b)*

*Note: that returned values **can** be negative*

*Note: for both zero arguments function will return (0,0)*

***Warning***

*T can be only to signed integer*

***Parameters***

| | |
|---|---|
| a | *non-negative integer* |
| b | *non-negative integer* |

***Returns***

*(x,y) such that a∗x+b∗y==gcd(a,b)*

The documentation for this class was generated from the following file:

- src/math/gcd/extended_gcd_loop.h

## 6.12 math::factorize::FactorizeNaive_< CountType > Class Template Reference

`#include <factorize_naive.h>`

### Static Public Member Functions

- template<typename T >
  static std::vector< std::pair< T, CountType > > factorize (T number)

### 6.12.1 Detailed Description

**template<typename CountType>class math::factorize::FactorizeNaive_< CountType >**

Naive factorization method. Running time is *O(m)* , where *m* is greatest prime factor, or *O(sqrt(n))* if the number is prime.

### 6.12.2 Member Function Documentation

#### 6.12.2.1 template<typename CountType > template<typename T > static std::vector<std::pair<T, CountType>> math::factorize::FactorizeNaive_< CountType >::factorize ( T *number* ) [inline, static]

Factorize number.

*T* should be integral type

**Parameters**

| | |
|---|---|
| *number* | integer to factorize, should be greater than 1 |

**Returns**

factorization, factors in increasing order

The documentation for this class was generated from the following file:

- src/math/factorize/factorize_naive.h

## 6.13 math::factorize::FactorizeWithOracle_< CountType, Oracle, Primes > Class Template Reference

`#include <factorize_with_oracle.h>`

### Static Public Member Functions

- template<typename T >
  static std::vector< std::pair< T, CountType > > factorize (T number)

**template<typename CountType, class Oracle, class Primes = math::primes::PrimesFast> class math::factorize::FactorizeWithOracle_< CountType, Oracle, Primes >**

### 6.13.1 Member Function Documentation

#### 6.13.1.1 template<typename CountType , class Oracle , class Primes = math::primes::PrimesFast> template<typename T > static std::vector<std::pair<T, CountType> > math::factorize::FactorizeWithOracle_< CountType, Oracle, Primes >::factorize ( T *number* ) [inline, static]

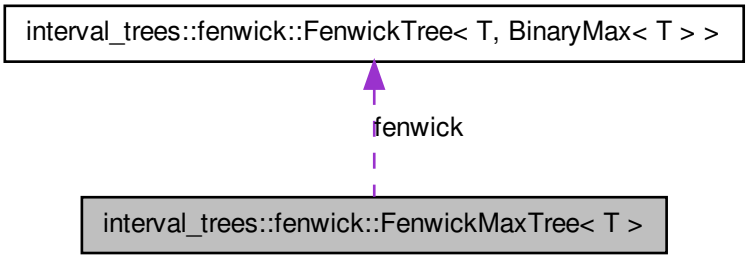
The documentation for this class was generated from the following file:

- src/math/factorize/factorize_with_oracle.h

## 6.14 interval_trees::fenwick::FenwickMaxTree< T > Class Template Reference

`#include <fenwick.h>`

Collaboration diagram for interval_trees::fenwick::FenwickMaxTree< T >:




## Public Member Functions

- void initialize (FenwickDirection type, size_t size)
- void update (int pos, T value)
- T get_max (int pos)

## Private Types

- typedef FenwickTree< T, BinaryMax< T > > FenwickType

## Private Attributes

- FenwickType fenwick

template<typename T> class interval_trees::fenwick::FenwickMaxTree< T >

### 6.14.1 Member Typedef Documentation

#### 6.14.1.1 template<typename T > typedef FenwickTree<T, BinaryMax<T> > interval_trees::fenwick::FenwickMaxTree< T >::FenwickType [private]

### 6.14.2 Member Function Documentation

#### 6.14.2.1 template<typename T > T interval_trees::fenwick::FenwickMaxTree< T >::get_max ( int *pos* ) [inline]

#### 6.14.2.2 template<typename T > void interval_trees::fenwick::FenwickMaxTree< T >::initialize ( FenwickDirection *type*, size_t *size* ) [inline]

#### 6.14.2.3 template<typename T > void interval_trees::fenwick::FenwickMaxTree< T >::update ( int *pos*, T *value* ) [inline]

### 6.14.3 Member Data Documentation

#### 6.14.3.1 template<typename T > FenwickType interval_trees::fenwick::FenwickMaxTree< T >::fenwick [private]

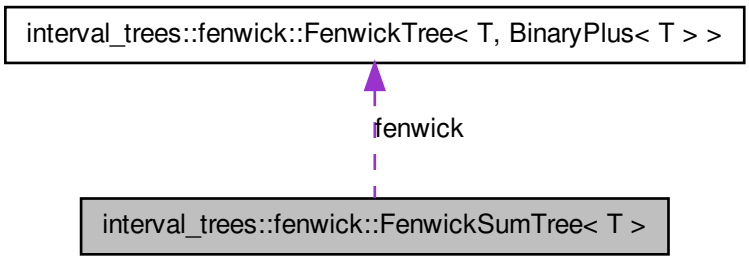
The documentation for this class was generated from the following file:

- src/interval_trees/fenwick/fenwick.h

## 6.15 interval_trees::fenwick::FenwickSumTree< T > Class Template Reference

`#include <fenwick.h>`

Collaboration diagram for interval_trees::fenwick::FenwickSumTree< T >:




### Public Member Functions

- void initialize (SizeType size)
- void increment (SizeType pos, T value)
- T get_prefix_sum (SizeType pos)

### Private Types

- typedef class FenwickTree< T, BinaryPlus< T > > FenwickType
- typedef FenwickType::SizeType SizeType

### Private Attributes

- FenwickType fenwick

template<typename T> class interval_trees::fenwick::FenwickSumTree< T >

## 6.15.1 Member Typedef Documentation

### 6.15.1.1 template<typename T > typedef class FenwickTree< T, BinaryPlus< T > > interval_trees::fenwick::FenwickSumTree< T >::FenwickType [private]

Underlying Fenwick tree type

### 6.15.1.2 template<typename T > typedef FenwickType::SizeType interval_trees::fenwick::FenwickSumTree< T >::SizeType [private]

Type of indexes

## 6.15.2 Member Function Documentation

### 6.15.2.1 template<typename T > T interval_trees::fenwick::FenwickSumTree< T >::get_prefix_sum ( SizeType *pos* ) [inline]

### 6.15.2.2 template<typename T > void interval_trees::fenwick::FenwickSumTree< T >::increment ( SizeType *pos*, T *value* ) [inline]

### 6.15.2.3 template<typename T > void interval_trees::fenwick::FenwickSumTree< T >::initialize ( SizeType *size* ) [inline]

## 6.15.3 Member Data Documentation

### 6.15.3.1 template<typename T > FenwickType interval_trees::fenwick::FenwickSumTree< T >::fenwick [private]

Underlying fenwick tree implementation

The documentation for this class was generated from the following file:

- src/interval_trees/fenwick/fenwick.h

## 6.16 interval_trees::fenwick::FenwickTree< ValueType, Operation > Class Template Reference

`#include <fenwick.h>`

### Public Types

- typedef size_t SizeType

### Public Member Functions

- void initialize (FenwickDirection type_, size_t size, const ValueType &initial)
- void update (SizeType pos, ValueType value)
- ValueType get_on_interval (SizeType pos)

### Private Member Functions

- SizeType _advance (SizeType pos, FenwickDirection direction)
- SizeType last_one (SizeType x)

### Private Attributes

- std::vector< ValueType > data
- FenwickDirection type

### 6.16.1 Detailed Description

**template<typename ValueType, class Operation>class interval_trees::fenwick::FenwickTree< ValueType, Operation >**

General Fenwick interval tree.

can update position pos by applying "Operation" can report an "Operation" over values at interval [0..pos] or [pos..size) (not both at once)

Note: sensible operations are only plus, min, max

**Precondition**

- If you need a custom operation, you will need following concept: operation on [a..b) == operation( operation on [a..c), operation on [c..b) )

### 6.16.2 Member Typedef Documentation

#### 6.16.2.1 template<typename ValueType, class Operation> typedef size_t interval_trees::fenwick::FenwickTree< ValueType, Operation >::SizeType

### 6.16.3 Member Function Documentation

#### 6.16.3.1 template<typename ValueType, class Operation> SizeType interval_trees::fenwick::FenwickTree< ValueType, Operation >::_advance ( SizeType *pos,* FenwickDirection *direction* ) [inline, private]

advance to next position in the structure

#### 6.16.3.2 template<typename ValueType, class Operation> ValueType interval_trees::fenwick::FenwickTree< ValueType, Operation >::get_on_interval ( SizeType *pos* ) [inline]

Return value of operation on interval [0, pos] or [pos, size) depending on the type of a tree

#### 6.16.3.3 template<typename ValueType, class Operation> void interval_trees::fenwick::FenwickTree< ValueType, Operation >::initialize ( FenwickDirection *type_,* size_t *size,* const ValueType & *initial* ) [inline]

Initialize fenwick tree

#### 6.16.3.4 template<typename ValueType, class Operation> SizeType interval_trees::fenwick::FenwickTree< ValueType, Operation >::last_one ( SizeType *x* ) [inline, private]

**Returns**

last set bit of an integer

#### 6.16.3.5 template<typename ValueType, class Operation> void interval_trees::fenwick::FenwickTree< ValueType, Operation >::update ( SizeType *pos,* ValueType *value* ) [inline]

Update value on position by *value* using operator Operation::operation(old, value);

### 6.16.4 Member Data Documentation

#### 6.16.4.1 template<typename ValueType, class Operation> std::vector<ValueType> interval_trees::fenwick::FenwickTree< ValueType, Operation >::data [private]

#### 6.16.4.2 template<typename ValueType, class Operation> FenwickDirection interval_trees::fenwick::FenwickTree< ValueType, Operation >::type [private]

The documentation for this class was generated from the following file:

- src/interval_trees/fenwick/fenwick.h

## 6.17 interval_trees::FullBinaryTree< NodeType > Class Template Reference

`#include <full_binary_tree.h>`

### Classes

- class Traverser

### Public Member Functions

- FullBinaryTree ()
- void _clear ()
- void initialize (Tpos size)
- void initialize (Tpos size_, const Tpos &default_value)
- Traverser root ()
- Traverser leaf (Tpos pos)

### Private Types

- typedef size_t Tpos

### Private Attributes

- std::vector< NodeType > data

### 6.17.1 Detailed Description

template<typename NodeType>class interval_trees::FullBinaryTree< NodeType >

Simple interval tree with get/set value and get_max over interval

### 6.17.2 Member Typedef Documentation

#### 6.17.2.1 template<typename NodeType > typedef size_t interval_trees::FullBinaryTree< NodeType >::Tpos [private]

### 6.17.3 Constructor & Destructor Documentation

#### 6.17.3.1 template<typename NodeType > interval_trees::FullBinaryTree< NodeType >::FullBinaryTree ( ) [inline]

### 6.17.4 Member Function Documentation

#### 6.17.4.1 template<typename NodeType > void interval_trees::FullBinaryTree< NodeType >::_clear ( ) [inline]

Clear tree and set it's size to zero.

#### 6.17.4.2 template<typename NodeType > void interval_trees::FullBinaryTree< NodeType >::initialize ( Tpos *size* ) [inline]

Shorthand for initialization

**See also**

initialize(size, default_value

#### 6.17.4.3 template<typename NodeType > void interval_trees::FullBinaryTree< NodeType >::initialize ( Tpos *size_*, const Tpos & *default_value* ) [inline]

Initialize tree

#### 6.17.4.4 template<typename NodeType > Traverser interval_trees::FullBinaryTree< NodeType >::leaf ( Tpos *pos* ) [inline]

#### 6.17.4.5 template<typename NodeType > Traverser interval_trees::FullBinaryTree< NodeType >::root ( ) [inline]

### 6.17.5 Member Data Documentation

#### 6.17.5.1 template<typename NodeType > std::vector<NodeType> interval_trees::FullBinaryTree< NodeType >::data [private]

number of leaves in heap structure. Also, pos+base is the index of leaf in data

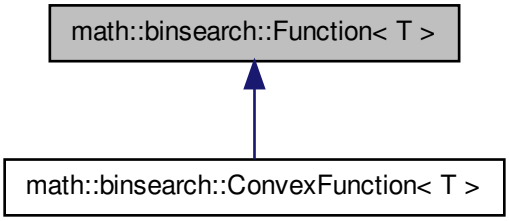
The documentation for this class was generated from the following file:

- src/interval_trees/full_binary_tree/full_binary_tree.h

## 6.18 math::binsearch::Function< T > Class Template Reference

`#include <function_binsearch.h>`

Inheritance diagram for math::binsearch::Function< T >:




### Public Member Functions

- virtual T operator() (T x)=0

### 6.18.1 Detailed Description

**template<typename T>class math::binsearch::Function< T >**

Base class of all unary functions

### 6.18.2 Member Function Documentation

#### 6.18.2.1 template<typename T> virtual T math::binsearch::Function< T >::operator() ( T *x* ) [pure virtual]

The documentation for this class was generated from the following file:

- src/math/binsearch/function_binsearch.h

## 6.19 math::binsearch::FunctionBinsearch< T > Class Template Reference

`#include <function_binsearch.h>`

## Public Member Functions

- T root (Function< T > *f, T left, T right, T precision)
- T convex_min (ConvexFunction< T > *f, T left, T right, T precision)

## Private Member Functions

- int number_of_iterations (T left, T right, T precision, double division)

template<typename T> class math::binsearch::FunctionBinsearch< T >

### 6.19.1 Member Function Documentation

#### 6.19.1.1 template<typename T > T math::binsearch::FunctionBinsearch< T >::convex_min ( ConvexFunction< T > * *f*, T *left*, T *right*, T *precision* ) [inline]

Finds a minimum value of convex function on given interval *[left, right)*

**Warning**

Real error can be greated than precision if precision is small compared to magnitude of the result,

**See also**

root()

**Precondition**

- function should be convex

**Parameters**

| | |
|---|---|
| *left* | left end of the interval |
| *right* | right end of the interval |
| *precision* | required precision |

**Returns**

argmin

#### 6.19.1.2 template<typename T > int math::binsearch::FunctionBinsearch< T >::number_of_iterations ( T *left*, T *right*, T *precision*, double *division* ) [inline, private]

Calculate the number of iterations needed to obtain specified precision if original range is *[left, right)* and each iteration the range is reduced to *1/division* of its size.

**Warning**

This is exact-math result, in floating-point numbers it may not be possible to obtain required precision. In that case the iteration count should be hovewer sufficient as after the required number of iterations, the computation will be stalled

#### 6.19.1.3 template<typename T > T math::binsearch::FunctionBinsearch< T >::root ( Function< T > * *f*, T *left*, T *right*, T *precision* ) [inline]

Finds any root of function on interval *[left, right)* with specified precision

**Precondition**

- You must provide such an interval, that function values are of opposite sign at the interval ends.

**Warning**

Real error can be greater than precision in case that precision is smaller than distance between two floating-point values at required range

**Returns**

any root on the interval.

The documentation for this class was generated from the following file:

- src/math/binsearch/function_binsearch.h

## 6.20 IntervalMaxArray< ValueType > Class Template Reference

`#include <interval_array.h>`

### Public Member Functions

- void initialize (SizeType size)
- void set (SizeType start, SizeType end, ValueType value)
- void update (SizeType start, SizeType end, ValueType value)
- ValueType get_max (SizeType start, SizeType end)

### Private Types

- typedef std::vector< ValueType >::size_type SizeType

### Private Attributes

- std::vector< ValueType > data

### 6.20.1 Detailed Description

template<typename ValueType>class IntervalMaxArray< ValueType >

Interval-tree-like implementation done in a simple array.

**Warning**

This class is inefficient, use only for testing!

### 6.20.2 Member Typedef Documentation

#### 6.20.2.1 template<typename ValueType > typedef std::vector<ValueType>::size_type IntervalMaxArray< ValueType >::SizeType [private]

### 6.20.3 Member Function Documentation

#### 6.20.3.1 template<typename ValueType > ValueType IntervalMaxArray< ValueType >::get_max ( SizeType *start,* SizeType *end* ) [inline]

get maximum of interval [start, end)

#### 6.20.3.2 template<typename ValueType > void IntervalMaxArray< ValueType >::initialize ( SizeType *size* ) [inline]

#### 6.20.3.3 template<typename ValueType > void IntervalMaxArray< ValueType >::set ( SizeType *start,* SizeType *end,* ValueType *value* ) [inline]

sets value on whole interval [start, end)

#### 6.20.3.4 template<typename ValueType > void IntervalMaxArray< ValueType >::update ( SizeType *start,* SizeType *end,* ValueType *value* ) [inline]

update interval with new max [start, end)

### 6.20.4 Member Data Documentation

#### 6.20.4.1 template<typename ValueType > std::vector<ValueType> IntervalMaxArray< ValueType >::data [private]

The documentation for this class was generated from the following file:

- src/interval_trees/array/interval_array.h

## 6.21 IntervalSumArray< ValueType > Class Template Reference

`#include <interval_array.h>`

### Public Member Functions

- void initialize (SizeType size)
- void increment (SizeType start, SizeType end, ValueType value)
- ValueType get_sum (SizeType start, SizeType end)

### Private Types

- typedef std::vector< ValueType >::size_type SizeType

### Private Attributes

- std::vector< ValueType > data

### 6.21.1 Detailed Description

**template<typename ValueType>class IntervalSumArray< ValueType >**

Interval-tree-like implementation done in a simple array.

**Warning**

This class is inefficient, use only for testing!

### 6.21.2 Member Typedef Documentation

#### 6.21.2.1 template<typename ValueType > typedef std::vector<ValueType>::size_type IntervalSumArray< ValueType >::SizeType [private]

### 6.21.3 Member Function Documentation

#### 6.21.3.1 template<typename ValueType > ValueType IntervalSumArray< ValueType >::get_sum ( SizeType *start,* SizeType *end* ) [inline]

get sum of interval [start, end)

#### 6.21.3.2 template<typename ValueType > void IntervalSumArray< ValueType >::increment ( SizeType *start,* SizeType *end,* ValueType *value* ) [inline]

increment interval [start, end)

#### 6.21.3.3 template<typename ValueType > void IntervalSumArray< ValueType >::initialize ( SizeType *size* ) [inline]

Initialize whole array

### 6.21.4 Member Data Documentation

#### 6.21.4.1 template<typename ValueType > std::vector<ValueType> IntervalSumArray< ValueType >::data [private]

The documentation for this class was generated from the following file:

- src/interval_trees/array/interval_array.h

## 6.22 strings::search::KMP Class Reference

`#include <kmp.h>`

### Static Public Member Functions

- template<typename _Iterator , typename _PatternIterator >
  static void search (_Iterator first, _Iterator last, _PatternIterator pattern_first, _PatternIterator pattern_last, strings::search_callback::SearchCallback< _Iterator > *callback)

### Static Private Member Functions

- template<typename _PatternIterator , typename IndexType >
  static void prepare (_PatternIterator pattern_first, _PatternIterator pattern_last, std::vector< IndexType > *out_)
- template<typename _Iterator , typename _PatternIterator , typename IndexType >
  static void search (_Iterator first, _Iterator last, _PatternIterator pattern_first, _PatternIterator pattern_last, std::vector< IndexType > &data, strings::search_callback::SearchCallback< _Iterator > *callback)

### 6.22.1 Member Function Documentation

#### 6.22.1.1 template<typename _PatternIterator , typename IndexType > static void strings::search::KMP::prepare ( _PatternIterator *pattern_first*, _PatternIterator *pattern_last*, std::vector< IndexType > * *out_* ) [inline, static, private]

6.22.1.2 template<typename _Iterator , typename _PatternIterator > static void strings::search::KMP::search ( _Iterator *first*, _Iterator *last*, _PatternIterator *pattern_first*, _PatternIterator *pattern_last*, strings::search_callback::SearchCallback< _Iterator > * *callback* ) [inline, static]

6.22.1.3 template<typename _Iterator , typename _PatternIterator , typename IndexType > static void strings::search::KMP::search ( _Iterator *first*, _Iterator *last*, _PatternIterator *pattern_first*, _PatternIterator *pattern_last*, std::vector< IndexType > & *data*, strings::search_callback::SearchCallback< _Iterator > * *callback* ) [inline, static, private]

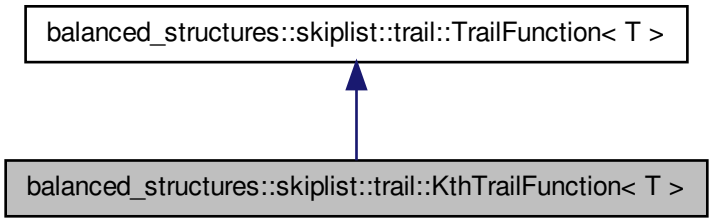
The documentation for this class was generated from the following file:

- src/strings/search_kmp/kmp.h

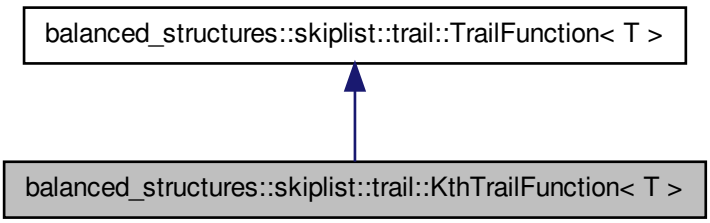
## 6.23 balanced_structures::skiplist::trail::KthTrailFunction< T > Class Template Reference

`#include <skiplist_trail.h>`

Inheritance diagram for balanced_structures::skiplist::trail::KthTrailFunction< T >:



Collaboration diagram for balanced_structures::skiplist::trail::KthTrailFunction< T >:




### Public Member Functions

- KthTrailFunction (SizeType pos_)
- virtual bool goFurther (const Node< T > *UNUSED(node), SizeType position)

### Private Attributes

- SizeType pos

## 6.23.1 Detailed Description

template<typename T>class balanced_structures::skiplist::trail::KthTrailFunction< T >

Find k-th element

## 6.23.2 Constructor & Destructor Documentation

### 6.23.2.1 template<typename T> balanced_structures::skiplist::trail::KthTrailFunction< T >::KthTrailFunction ( SizeType *pos_* ) [inline]

Constructor

## 6.23.3 Member Function Documentation

#### 6.23.3.1 template<typename T> virtual bool balanced_structures::skiplist::trail::KthTrailFunction< T >::goFurther ( const Node< T > * *UNUSEDnode,* SizeType *position* ) [inline, virtual]

goes further until we encounter node with specified position

### 6.23.4 Member Data Documentation

#### 6.23.4.1 template<typename T> SizeType balanced_structures::skiplist::trail::KthTrailFunction< T >::pos [private]

index of an element we are searching for

The documentation for this class was generated from the following file:

- src/balanced_structures/skiplist/skiplist_trail.h

## 6.24 strings::suffix_array::LCPKasai Class Reference

`#include <lcp_kasai.h>`

### Static Public Member Functions

- template<typename _Iterator , typename _SAIterator >
  static void getHeightArray (_Iterator seq_first, _Iterator seq_last, _SAIterator sa_first, _SAIterator sa_last, std::vector< int > *out)

### 6.24.1 Member Function Documentation

#### 6.24.1.1 template<typename _Iterator , typename _SAIterator > static void strings::suffix_array::LCPKasai::getHeightArray ( _Iterator *seq_first,* _Iterator *seq_last,* _SAIterator *sa_first,* _SAIterator *sa_last,* std::vector< int > * *out* ) [inline, static]

Calculate "Height array", which is defined as follows height[i] = longest_common_prefix( sequence[pos[k-1]..n), sequence[pos[k]..n)) i.e. it is longest common prefix of two adjacent suffixes (in sorted order).

Rank is inverse array to pos, i.e. rank[pos[i]]=i. This actually means, that if we sort suffixes, the i-th position in the sorted array will belong to suffix starting at rank[i]

The documentation for this class was generated from the following file:

- src/strings/suffix_array_lcp_kasai/lcp_kasai.h

## 6.25 strings::suffix_array::LCPManzini Class Reference

`#include <lcp_manzini.h>`

### Static Public Member Functions

- template<typename _Iterator , typename _SAIterator >
  static void getHeightArray (_Iterator seq_first, _Iterator seq_last, _SAIterator sa_-
  first, _SAIterator sa_last, int alphabet_size, std::vector< int > *out)

### Private Member Functions

- DISALLOW_EVIL_CONSTRUCTORS (LCPManzini)

### Static Private Member Functions

- template<typename _Iterator >
  static void compute_counts (_Iterator first, _Iterator last, int alphabet_size, std::vector<
  int > *out)
- template<typename _Iterator , typename _SAIterator >
  static int compute_rank_next (_Iterator seq_first, _Iterator seq_last, _SAIterator
  sa_first, _SAIterator sa_last, int alphabet_size, std::vector< int > *rank_next)

### 6.25.1 Member Function Documentation

#### 6.25.1.1 template<typename _Iterator > static void strings::suffix_-array::LCPManzini::compute_counts ( _Iterator *first*, _Iterator *last*, int *alphabet_size*, std::vector< int > * *out* ) [inline, static, private]

#### 6.25.1.2 template<typename _Iterator , typename _SAIterator > static int strings::suffix_array::LCPManzini::compute_rank_next ( _Iterator *seq_first*, _Iterator *seq_last*, _SAIterator *sa_first*, _SAIterator *sa_last*, int *alphabet_size*, std::vector< int > * *rank_next* ) [inline, static, private]

Compute the "rankNext" map as defined in [ltLCP] RankNext is defined as following RankNext[i] = Rank[sa[i] + 1] for i=[0..n), i != Rank[n] for i = Rank[i] is RankNext[i] undefined as it is the last element;

**Returns**

value i which is not amongs values of RankNext array (one value is missing)

#### 6.25.1.3 strings::suffix_array::LCPManzini::DISALLOW_EVIL_CONSTRUCTORS ( LCPManzini ) [private]

#### 6.25.1.4 template<typename _Iterator , typename _SAIterator > static void strings::suffix_array::LCPManzini::getHeightArray ( _Iterator *seq_first,* _Iterator *seq_last,* _SAIterator *sa_first,* _SAIterator *sa_last,* int *alphabet_size,* std::vector< int > * *out* ) [inline, static]

The documentation for this class was generated from the following file:

- src/strings/suffix_array_lcp_manzini/lcp_manzini.h

## 6.26 strings::suffix_array::LCPNaive Class Reference

`#include <lcp_naive.h>`

### Static Public Member Functions

- template<typename _Iterator , typename _SAIterator >
  static void getHeightArray (_Iterator seq_first, _Iterator seq_last, _SAIterator sa_-first, _SAIterator sa_last, std::vector< int > *out)

### Static Private Member Functions

- template<typename _Iterator >
  static int lcp (_Iterator pos1, _Iterator pos2, _Iterator last)

### 6.26.1 Detailed Description

Calculates longest common prefixes of suffix arrays Note: This is trivial naive implementation used mainly for testing. It may have serious performance problems when used with long lcp-s. Use LCPKasai or LCPManzini for real data.

### 6.26.2 Member Function Documentation

#### 6.26.2.1 template<typename _Iterator , typename _SAIterator > static void strings::suffix_array::LCPNaive::getHeightArray ( _Iterator *seq_first,* _Iterator *seq_last,* _SAIterator *sa_first,* _SAIterator *sa_last,* std::vector< int > * *out* ) [inline, static]

Calculate "Height array", which is defined as follows height[i] = longest_common_prefix( sequence[pos[k-1]..n), sequence[pos[k]..n)) i.e. it is longest common prefix of two adjacent suffixes (in sorted order).

6.26.2.2 template<typename _Iterator > static int strings::suffix_array::LCPNaive::lcp ( _Iterator *pos1,* _Iterator *pos2,* _Iterator *last* ) [inline, static, private]

The documentation for this class was generated from the following file:

- src/strings/suffix_array_lcp_naive/lcp_naive.h

## 6.27 strings::lcs::LCS< T > Class Template Reference

`#include <lcs.h>`

### Static Public Member Functions

- static int length (const T *seq1, int len1, const T *seq2, int len2)
- static int subsequence (const T *seq1, int len1, const T *seq2, int len2, std::vector< T > *out)

### 6.27.1 Detailed Description

template<typename T>class strings::lcs::LCS< T >

Trivial dynamic programming computation of LCS

### 6.27.2 Member Function Documentation

6.27.2.1 template<typename T > static int strings::lcs::LCS< T >::length ( const T * *seq1,* int *len1,* const T * *seq2,* int *len2* ) [inline, static]

Compute the length of the longest common subsequence of two sequences.

Running time is *O(nm)* , memory is *O(n+m)*

**Returns**

LCS length

6.27.2.2 template<typename T > static int strings::lcs::LCS< T >::subsequence ( const T * *seq1,* int *len1,* const T * *seq2,* int *len2,* std::vector< T > * *out* ) [inline, static]

Computes the longest common subsequence of two sequences.

Running time *O(nm)* , memory *O(nm)*

**See also**

length() if you only need length of the LCS

**Parameters**

| | |
|---|---|
| *out* | LCS will be written here |

**Returns**

length of the LCS

The documentation for this class was generated from the following file:

- src/strings/lcs/lcs.h

## 6.28 strings::lcs::LCSHirschberg< T > Class Template Reference

`#include <lcs_hirschberg.h>`

### Static Public Member Functions

- static int subsequence (const T *seq1, int len1, const T *seq2, int len2, std::vector< T > *out)

### Static Private Member Functions

- static void saveBest (utils::SequenceHelper< const T > seq1, utils::SequenceHelper< const T > seq2, std::vector< int > *out)
- static void recurse (utils::SequenceHelper< const T > seq1, utils::SequenceHelper< const T > seq2, std::vector< T > *out)

### 6.28.1 Detailed Description

**template<typename T>class strings::lcs::LCSHirschberg< T >**

LCS computation in *O(nm)* time and *O(n+m)space based on paper by Hirschberg.*

### 6.28.2 Member Function Documentation

#### 6.28.2.1 template<typename T > static void strings::lcs::LCSHirschberg< T >::recurse ( utils::SequenceHelper< const T > *seq1*, utils::SequenceHelper< const T > *seq2*, std::vector< T > * *out* ) [inline, static, private]

6.28.2.2 template<typename T > static void strings::lcs::LCSHirschberg< T >::saveBest ( utils::SequenceHelper< const T > *seq1,* utils::SequenceHelper< const T > *seq2,* std::vector< int > * *out* ) [inline, static, private]

6.28.2.3 template<typename T > static int strings::lcs::LCSHirschberg< T >::subsequence ( const T * *seq1,* int *len1,* const T * *seq2,* int *len2,* std::vector< T > * *out* ) [inline, static]

The documentation for this class was generated from the following file:

- src/strings/lcs/lcs_hirschberg.h

## 6.29 geometry::two_d::LineSegment< T > Struct Template Reference

`#include <linesegment.h>`

### Public Member Functions

- LineSegment ()
- LineSegment (Point< T > b, Point< T > e)

### Public Attributes

- Point< T > begin
- Point< T > end

template<typename T> struct geometry::two_d::LineSegment< T >

### 6.29.1 Constructor & Destructor Documentation

6.29.1.1 template<typename T> geometry::two_d::LineSegment< T >::LineSegment ( ) [inline]

6.29.1.2 template<typename T> geometry::two_d::LineSegment< T >::LineSegment ( Point< T > *b,* Point< T > *e* ) [inline]

### 6.29.2 Member Data Documentation

6.29.2.1 template<typename T> Point<T> geometry::two_d::LineSegment< T >::begin

#### 6.29.2.2 template<typename T> Point<T> geometry::two_d::LineSegment< T >::end

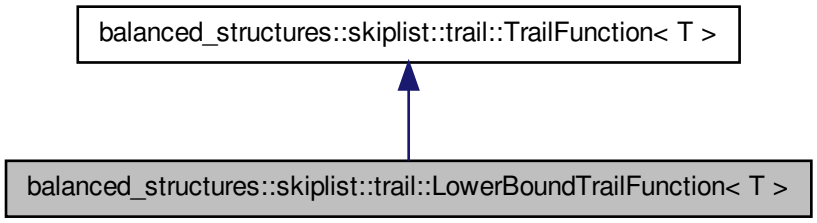
The documentation for this struct was generated from the following file:

- src/geometry/two_d/linesegment.h

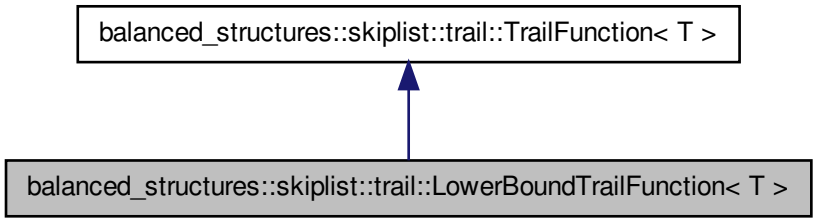
## 6.30 balanced_structures::skiplist::trail::LowerBoundTrailFunction< T > Class Template Reference

`#include <skiplist_trail.h>`

Inheritance diagram for balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >:




Collaboration diagram for balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >:



### Public Member Functions

- LowerBoundTrailFunction (const T &value_)
- virtual bool goFurther (const Node< T > *node, SizeType UNUSED(position))

### Private Attributes

- T value

## 6.30.1 Detailed Description

template<typename T>class balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >

Trail function which points to the first position where element can be inserted without violating ordering.

Note that this is equivalent to the first element >= *value*

## 6.30.2 Constructor & Destructor Documentation

### 6.30.2.1 template<typename T> balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >::LowerBoundTrailFunction ( const T & *value_* ) [inline]

Constructor

## 6.30.3 Member Function Documentation

### 6.30.3.1 template<typename T> virtual bool balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >::goFurther ( const Node< T > * *node*, SizeType *UNUSEDposition* ) [inline, virtual]

Goes further until we encounter a *node* with value greater or equal than *value*

## 6.30.4 Member Data Documentation

### 6.30.4.1 template<typename T> T balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >::value [private]

Value we are searching for

The documentation for this class was generated from the following file:

- src/balanced_structures/skiplist/skiplist_trail.h

## 6.31 strings::suffix_array::ManberMyers Class Reference

`#include <manber_myers.h>`

### Static Private Member Functions

- template<typename _Iterator >
  static void sortByFirstCharacter (_Iterator first, _Iterator last, int sigma, std::vector< int > *out)

### 6.31.1 Detailed Description

Constructs suffix array in O(n log n) time with Mamber-Myers algorithm.

The implementation is optimized to use as little memory as possible without great increase in code complexity, in this case it is 3N ints and 2N bools of storage. Note that authors of the paper claims it can be done in 2N ints, but I haven't been able to fully comprehend and write bug-free code for that version. Note: another implementation and some discussion can be found at http://forums.topcoder.com/?module=RevisionHistory&me

### 6.31.2 Member Function Documentation

#### 6.31.2.1 template<typename _Iterator > static void strings::suffix_array::ManberMyers::sortByFirstCharacter ( _Iterator *first*, _Iterator *last*, int *sigma*, std::vector< int > * *out* ) [inline, static, private]

Sort suffixes by first character. Note: This implementation is count-sort, because for reasonable small alphabets quicksort is awfully slow.

The documentation for this class was generated from the following file:

- src/strings/suffix_array_myers/manber_myers.h

## 6.32 strings::suffix_array::ManberMyersLog2_< IndexType > Class Template Reference

`#include <manber_myers_log2.h>`

### Classes

- struct Suffix

### 6.32.1 Detailed Description

template<typename IndexType>class strings::suffix_array::ManberMyersLog2_< IndexType >

Constructs suffix array in O(n (log n)^2) time with Mamber-Myers algorithm, but using std::sort instead of countsorts.

The implementation uses 4N ints of memory. The code is based on original from Michal Forisek.

The documentation for this class was generated from the following file:

- src/strings/suffix_array_log2/manber_myers_log2.h

## 6.33 math::modular_inverse::ModularInverseFermat_< PowerModImpl, checkPrimality > Class Template Reference

`#include <modular_inverse_fermat.h>`

### Static Public Member Functions

- template<typename T >
  static T getInverse (T i, T p)

### 6.33.1 Detailed Description

template<typename PowerModImpl, bool checkPrimality = true>class math::modular_inverse::ModularInverseFermat_< PowerModImpl, checkPrimality >

Compute modular inverses using Fermat's little theorem.

**Precondition**

- PowerModImpl contains static function *powermod(base,exponent,modulo)*

**Parameters**

| | |
|---|---|
| *PowerModImpl* | implementation of powermod |
| *checkPrimality* | whether to check that the modulo is a prime number |

### 6.33.2 Member Function Documentation

#### 6.33.2.1 template<typename PowerModImpl , bool checkPrimality = true> template<typename T > static T math::modular_inverse::ModularInverseFermat_< PowerModImpl, checkPrimality >::getInverse ( T *i*, T *p* ) [inline, static]

Compute modular inverse of number i over field Z_p.

The result is based on the fact, that a^(p-2) = a^(-1) mod p based on Fermat's little theorem.

Note: More general case is Euler's theorem: a^(fi(m)-1) = a^(-1) mod m but in this case, calculating fi(m) needs to factorize m and it is therefore inefficient. If you have good implementation of euler's fi, you may tweak this implementation. Anyway, consider using ModularInverseGcd for inverses for non-primes.

**Parameters**

| | |
|---|---|
| *i* | number for which the inverse should be calculated |
| *p* | prime modulus |

**Returns**

inverse of the number, i.e. number such that (inverse * i) mod p == 1, or zero if such a number does not exists (case i==0)

The documentation for this class was generated from the following file:

- src/math/modular_inverse/modular_inverse_fermat.h

## 6.34 math::modular_inverse::ModularInverseGcd Class Reference

`#include <modular_inverse_gcd.h>`

### Static Public Member Functions

- template<typename T >
  static T getInverse (T i, T p)

### 6.34.1 Detailed Description

Computes modular inverse of number over any (even non-prime) modulus.

The implementation is O(1) for initialization (no initialization) and O(log n) for query, but the queries are generally fast.

If you need to precompute modular inverses over all elements of field Z_p where p is prime, then you may use ModularInverse

### 6.34.2 Member Function Documentation

#### 6.34.2.1 template<typename T > static T math::modular_inverse::ModularInverseGcd::getInverse ( T *i*, T *p* ) [inline, static]

Compute modular inverse of number i modulo p.

**Returns**

inverse of the number i or zero if such number does not exists

The documentation for this class was generated from the following file:

- src/math/modular_inverse/modular_inverse_gcd.h

## 6.35 math::modular_inverse::ModularInversePrecomputed_< PowerModImpl > Class Template Reference

`#include <modular_inverse_precomputed.h>`

### Public Member Functions

- template<typename T >
  void initialize (T t_size)
- template<typename T >
  SizeType getInverse (T t_pos)

### Private Types

- typedef std::vector< bool >::size_type SizeType

### Private Attributes

- std::vector< SizeType > inverses

### 6.35.1 Detailed Description

template<class PowerModImpl>class math::modular_inverse::ModularInversePrecomputed_< PowerModImpl >

Calculate the modular inverses of all numbers modulo **prime** p in O(p) time and 2∗p space.

The initialization is in O(p), query is in O(1). The implementation is thus useful for medium-sized p (limited memory), for which you will ask for many different inverses. If you will ask only limited number of questions, but the p is big, it is probably better to use ModularInverseGcd class

### 6.35.2 Member Typedef Documentation

#### 6.35.2.1 template<class PowerModImpl > typedef std::vector<bool>::size_type math::modular_inverse::ModularInversePrecomputed_< PowerModImpl >::SizeType [private]

### 6.35.3 Member Function Documentation

#### 6.35.3.1 template<class PowerModImpl > template<typename T > SizeType math::modular_inverse::ModularInversePrecomputed_< PowerModImpl >::getInverse ( T *t_pos* ) [inline]

Return inverse of number pos

#### 6.35.3.2 template<class PowerModImpl > template<typename T > void math::modular_inverse::ModularInversePrecomputed_< PowerModImpl >::initialize ( T *t_size* ) [inline]

### 6.35.4 Member Data Documentation

#### 6.35.4.1 template<class PowerModImpl > std::vector<SizeType> math::modular_inverse::ModularInversePrecomputed_< PowerModImpl >::inverses [private]

The documentation for this class was generated from the following file:

- src/math/modular_inverse/modular_inverse_precomputed.h

## 6.36 math::powermod::MultmodExtended< shift > Class Template Reference

`#include <multmod_extended.h>`

### Static Public Member Functions

- template<typename T >
  static T max_argument (T UNUSED(x))
- template<typename T >
  static T multmod (T a, T b, T modulo)

### Private Member Functions

- STATIC_ASSERT (shift > 0,"")

### 6.36.1 Detailed Description

**template<int shift>class math::powermod::MultmodExtended< shift >**

Extended modular multiplication. Does not require bigger types for temporaries.

The computation is done by using some type, the drawback is that we need to reserve several bits for our computations.

**Parameters**

| | |
|---|---|
| *shift* | number of bits reserved for the computation |

### 6.36.2 Member Function Documentation

#### 6.36.2.1 template<int shift> template<typename T > static T math::powermod::MultmodExtended< shift >::max_argument ( T *UNUSEDx* ) [inline, static]

#### 6.36.2.2 template<int shift> template<typename T > static T math::powermod::MultmodExtended< shift >::multmod ( T *a*, T *b*, T *modulo* ) [inline, static]

**Precondition**

- T integer type

#### 6.36.2.3 template<int shift> math::powermod::MultmodExtended< shift >::STATIC_ASSERT ( shift , 0 , "" ) [private]

The documentation for this class was generated from the following file:

- src/math/powermod/multmod_extended.h

## 6.37 math::powermod::MultmodExtendedOpt Class Reference

`#include <multmod_extended.h>`

### Static Public Member Functions

- template<typename T >
  static T max_argument (T UNUSED(x))

- template<typename T >
  static T multmod (T a, T b, T modulo)

### 6.37.1 Detailed Description

Optimized version of MultmodExtended

This is previous version with fixed shift=1, but we optimized modular arithmetic using only additions/subtractions

### 6.37.2 Member Function Documentation

#### 6.37.2.1 template<typename T > static T math::powermod::MultmodExtendedOpt::max_argument ( T *UNUSEDx* ) [inline, static]

#### 6.37.2.2 template<typename T > static T math::powermod::MultmodExtendedOpt::multmod ( T *a,* T *b,* T *modulo* ) [inline, static]

The documentation for this class was generated from the following file:

- src/math/powermod/multmod_extended.h

## 6.38 math::powermod::MultmodSimple Class Reference

`#include <multmod_simple.h>`

### Static Public Member Functions

- template<typename T >
  static T max_argument (T UNUSED(x))
- static BaseType multmod (BaseType a, BaseType b, BaseType modulo)

### Private Types

- typedef int BaseType
- typedef long long DoubleType

### Private Member Functions

- STATIC_ASSERT (std::numeric_limits< BaseType >::digits *2<=std::numeric_limits< DoubleType >::digits,"The long long is not enough for temporary ""computations")

### 6.38.1 Member Typedef Documentation

#### 6.38.1.1 typedef int math::powermod::MultmodSimple::BaseType [private]

#### 6.38.1.2 typedef long long math::powermod::MultmodSimple::DoubleType [private]

### 6.38.2 Member Function Documentation

#### 6.38.2.1 template<typename T > static T math::powermod::MultmodSimple::max_argument ( T *UNUSEDx* ) [inline, static]

#### 6.38.2.2 static BaseType math::powermod::MultmodSimple::multmod ( BaseType *a,* BaseType *b,* BaseType *modulo* ) [inline, static]

#### 6.38.2.3 math::powermod::MultmodSimple::STATIC_ASSERT ( std::numeric_limits< BaseType >::digits *2<=std::numeric_limits< DoubleType >::digits , "The long long is not enough for temporary ""computations" ) [private]

The documentation for this class was generated from the following file:

- src/math/powermod/multmod_simple.h

## 6.39 strings::suffix_array::NaiveSuffixArray Class Reference

`#include <naive.h>`

### Static Public Member Functions

- template<typename _Iterator >
  static void buildSuffixArray (_Iterator first, _Iterator last, std::vector< int > *out)

### 6.39.1 Member Function Documentation

#### 6.39.1.1 template<typename _Iterator > static void strings::suffix_array::NaiveSuffixArray::buildSuffixArray ( _Iterator *first,* _Iterator *last,* std::vector< int > * *out* ) [inline, static]

Build suffix array with naive sorting.

_Iterator random access iterator

**Parameters**

| | |
|---|---|
| *input* | sequence |
| *length* | length of the sequence |
| *out* | constructed suffix array, will be owerwrittent |

The documentation for this class was generated from the following file:

- src/strings/suffix_array_naive/naive.h

## 6.40 balanced_structures::skiplist::Node< T > Class Template Reference

`#include <skiplist_node.h>`

Collaboration diagram for balanced_structures::skiplist::Node< T >:


### Public Member Functions

- Node (SizeType level_)
- Self * next () const
- Self * prev () const
- ~Node ()

### Public Attributes

- T value
- LevelType level
- Self ** forward
- SizeType * forward_length
- Self * previous

### Private Types

- typedef int SizeType
- typedef Node< T > Self

### 6.40.1 Detailed Description

template<typename T>class balanced_structures::skiplist::Node< T >

Skiplist node

### 6.40.2 Member Typedef Documentation

#### 6.40.2.1 template<typename T> typedef Node<T> balanced_structures::skiplist::Node< T >::Self [private]

Type of this object

#### 6.40.2.2 template<typename T> typedef int balanced_structures::skiplist::Node< T >::SizeType [private]

Type of position in skiplist

### 6.40.3 Constructor & Destructor Documentation

#### 6.40.3.1 template<typename T> balanced_structures::skiplist::Node< T >::Node ( SizeType *level_* ) [inline]

#### 6.40.3.2 template<typename T> balanced_structures::skiplist::Node< T >::~Node ( ) [inline]

Destructor

### 6.40.4 Member Function Documentation

#### 6.40.4.1 template<typename T> Self* balanced_structures::skiplist::Node< T >::next ( ) const [inline]

Return next node

If this node is last node (skiplist 'tail'), it will return NULL

**Returns**

pointer to the next node

#### 6.40.4.2 template<typename T> Self* balanced_structures::skiplist::Node< T >::prev ( ) const [inline]

Return previous node

If this node is first node (skiplist 'head'), it will return NULL

**Returns**

pointer to the next node

### 6.40.5 Member Data Documentation

#### 6.40.5.1 template<typename T> Self** balanced_structures::skiplist::Node< T >::forward

Forward links to next nodes.

#### 6.40.5.2 template<typename T> SizeType* balanced_structures::skiplist::Node< T >::forward_length

Lengths of forward links.

#### 6.40.5.3 template<typename T> LevelType balanced_structures::skiplist::Node< T >::level

Level of the node.

Also size of the *forward* and *forward_length* arrays

#### 6.40.5.4 template<typename T> Self* balanced_structures::skiplist::Node< T >::previous

Backward link.

#### 6.40.5.5 template<typename T> T balanced_structures::skiplist::Node< T >::value

A value this node is holding.

The documentation for this class was generated from the following file:

- src/balanced_structures/skiplist/skiplist_node.h

## 6.41 math::factorize::OracleBrent_< Powermod > Class Template Reference

`#include <oracle_brent.h>`

### Static Public Member Functions

- template<typename T >
  static T findFactor (T number)

### Static Private Member Functions

- template<typename T >
  static T advance (T x, T n, T a)
- template<typename T >
  static T brent (T number, T start, T a)

template<class Powermod> class math::factorize::OracleBrent_< Powermod >

### 6.41.1 Member Function Documentation

#### 6.41.1.1 template<class Powermod > template<typename T > static T math::factorize::OracleBrent_< Powermod >::advance ( T *x*, T *n*, T *a* ) [inline, static, private]

#### 6.41.1.2 template<class Powermod > template<typename T > static T math::factorize::OracleBrent_< Powermod >::brent ( T *number*, T *start*, T *a* ) [inline, static, private]

#### 6.41.1.3 template<class Powermod > template<typename T > static T math::factorize::OracleBrent_< Powermod >::findFactor ( T *number* ) [inline, static]

The documentation for this class was generated from the following file:

- src/math/factorize/oracle_brent.h

## 6.42 math::factorize::OraclePollard_< Powermod > Class Template Reference

`#include <oracle_pollard.h>`

### Static Public Member Functions

- template<typename T >
  static T findFactor (T number)

### Static Private Member Functions

- template<typename T >
  static T advance (T x, T n, T a)
- template<typename T >
  static T pollard (T number, T start, T a)

template<class Powermod> class math::factorize::OraclePollard_< Powermod >

## 6.42.1 Member Function Documentation

### 6.42.1.1 template<class Powermod > template<typename T > static T math::factorize::OraclePollard_< Powermod >::advance ( T *x*, T *n*, T *a* ) [inline, static, private]

### 6.42.1.2 template<class Powermod > template<typename T > static T math::factorize::OraclePollard_< Powermod >::findFactor ( T *number* ) [inline, static]

### 6.42.1.3 template<class Powermod > template<typename T > static T math::factorize::OraclePollard_< Powermod >::pollard ( T *number*, T *start*, T *a* ) [inline, static, private]

The documentation for this class was generated from the following file:

- src/math/factorize/oracle_pollard.h

# 6.43 strings::PatternFiles Class Reference

`#include <testdata.h>`

### Static Public Attributes

- static const char * SEARCH_PATTERNS []

## 6.43.1 Detailed Description

Contains name of the testdata used as search patterns

## 6.43.2 Member Data Documentation

### 6.43.2.1 const char * strings::PatternFiles::SEARCH_PATTERNS [static]

**Initial value:**

```
 {
  "testdata/patterns/aaa.txt",
  "testdata/patterns/alphabet.txt",
  "testdata/patterns/and.txt",
  "testdata/patterns/nnn.txt",
  "testdata/patterns/php.txt",
  "testdata/patterns/php_line.txt",
  "testdata/patterns/random_genome.txt",
  "testdata/patterns/random_genome2.txt",
  "testdata/patterns/random_keyboard.txt",
  NULL
}
```

The documentation for this class was generated from the following file:

- src/strings/testdata.h

## 6.44 geometry::two_d::Point< T > Class Template Reference

`#include <point.h>`

### Public Member Functions

- Point ()
- Point (T x, T y)
- Point (std::complex< T >)
- void operator= (const Point< T > &b)
- T dot (const Point< T > &b) const
- T cross (const Point< T > &b) const
- T x () const
- T y () const
- void swap ()
- std::complex< T > _point () const
- operator Point< long double > () const

### Private Attributes

- std::complex< T > point

**template<typename T> class geometry::two_d::Point< T >**

### 6.44.1 Constructor & Destructor Documentation

#### 6.44.1.1 template<typename T > geometry::two_d::Point< T >::Point ( )

#### 6.44.1.2 template<typename T > geometry::two_d::Point< T >::Point ( T *x*, T *y* )

6.44.1.3 template<typename T > geometry::two_d::Point< T >::Point ( std::complex< T > *p* )

### 6.44.2 Member Function Documentation

6.44.2.1 template<typename T> std::complex<T> geometry::two_d::Point< T >::_point ( ) const [inline]

6.44.2.2 template<typename T> T geometry::two_d::Point< T >::cross ( const Point< T > & *b* ) const [inline]

6.44.2.3 template<typename T> T geometry::two_d::Point< T >::dot ( const Point< T > & *b* ) const [inline]

6.44.2.4 template<typename T> geometry::two_d::Point< T >::operator Point< long double > ( ) const [inline]

6.44.2.5 template<typename T> void geometry::two_d::Point< T >::operator= ( const Point< T > & *b* ) [inline]

6.44.2.6 template<typename T> void geometry::two_d::Point< T >::swap ( ) [inline]

6.44.2.7 template<typename T> T geometry::two_d::Point< T >::x ( ) const [inline]

6.44.2.8 template<typename T> T geometry::two_d::Point< T >::y ( ) const [inline]

### 6.44.3 Member Data Documentation

6.44.3.1 template<typename T> std::complex<T> geometry::two_d::Point< T >::point [private]

The documentation for this class was generated from the following file:

- src/geometry/two_d/point.h

## 6.45 geometry::two_d::ConvexHull< T >::PointCompare Class Reference

### Public Member Functions

- bool operator() (const PointType &p1, const PointType &p2)

### 6.45.1 Detailed Description

template<typename T>class geometry::two_d::ConvexHull< T >::PointCompare

Comparison function.

Points are sorted according *x* coordinate, than *y*.

### 6.45.2 Member Function Documentation

#### 6.45.2.1 template<typename T > bool geometry::two_d::ConvexHull< T >::PointCompare::operator() ( const PointType & *p1,* const PointType & *p2* ) [inline]

The documentation for this class was generated from the following file:

- src/geometry/two_d/convex_hull.h

## 6.46 math::powermod::Powermod_< MultModImpl > Class Template Reference

`#include <powermod.h>`

### Static Public Member Functions

- template<typename T >
  static T powermod (T base, T power, T modulo)
- template<typename T >
  static T multmod (T a, T b, T modulo)

template<class MultModImpl> class math::powermod::Powermod_< MultModImpl >

### 6.46.1 Member Function Documentation

#### 6.46.1.1 template<class MultModImpl > template<typename T > static T math::powermod::Powermod_< MultModImpl >::multmod ( T *a,* T *b,* T *modulo* ) [inline, static]

#### 6.46.1.2 template<class MultModImpl > template<typename T > static T math::powermod::Powermod_< MultModImpl >::powermod ( T *base,* T *power,* T *modulo* ) [inline, static]

The documentation for this class was generated from the following file:

- src/math/powermod/powermod.h

## 6.47 Preconditions Class Reference

`#include <preconditions.h>`

### Static Public Member Functions

- static void check (bool expression)
- static void check (bool expression, const char *message)
- template<typename T >
  static void checkRange (T index, T size)
- template<typename T >
  static void checkRange (T index, T low, T high)
- template<typename T >
  static void checkNotNull (const T *ptr)

### 6.47.1 Detailed Description

Preconditions is a helper class containing static methods that can check the validity of function arguments (function pre-conditions). In case there is a failure, Preconditions will throw an std::invalid_argument error.

Note: Do not overuse the throwing/catching exceptions. Failed preconditions are running much slower because of branch prediction and slowness of exceptions.

### 6.47.2 Member Function Documentation

#### 6.47.2.1 static void Preconditions::check ( bool *expression* ) [inline, static]

Checks whether the expression is true

**Parameters**

| | |
|---|---|
| *expression* | value that is expected to be true |

**Exceptions**

| | |
|---|---|
| *invalid_argument* | if expression is false |

#### 6.47.2.2 static void Preconditions::check ( bool *expression*, const char * *message* ) [inline, static]

Checks whether the expression is true

**Parameters**

| | |
|---|---|
| *expression* | value that is expected to be true |
| *message* | of the thrown expection if case of failure |

**Exceptions**

| | |
|---|---|
| *invalid_argument* | if expression is false |

#### 6.47.2.3 template<typename T > static void Preconditions::checkNotNull ( const T * *ptr* ) [inline, static]

Checks that pointer is not NULL.

**Exceptions**

| | |
|---|---|
| *invalid_argument* | in case of an error |

#### 6.47.2.4 template<typename T > static void Preconditions::checkRange ( T *index*, T *size* ) [inline, static]

Check that index is in range [0, size)

Warning: never ever override this template for using two different types - it may end up with nasty results because of casting like checkRange(unsigned int, int) and check will be done in ints!

**Exceptions**

| | |
|---|---|
| *invalid_argument* | in case of an error |

#### 6.47.2.5 template<typename T > static void Preconditions::checkRange ( T *index*, T *low*, T *high* ) [inline, static]

Check that index is in range [low, high)

**See also**

warnings for checkRange(index, size) implementation

**Exceptions**

| | |
|---|---|
| *invalid_argument* | in case of an error |

The documentation for this class was generated from the following file:

- src/utils/preconditions/preconditions.h

## 6.48 math::primes::PrimesBasic Class Reference

```
#include <primes_basic.h>
```

### Public Types

- typedef long long int BaseType

### Static Public Member Functions

- static bool isPrime (BaseType p)

#### 6.48.1 Detailed Description

Primality testing using trial division up to square root

#### 6.48.2 Member Typedef Documentation

##### 6.48.2.1 typedef long long int math::primes::PrimesBasic::BaseType

Basic type for all computations

#### 6.48.3 Member Function Documentation

##### 6.48.3.1 static bool math::primes::PrimesBasic::isPrime ( BaseType *p* ) [inline, static]

Checks whether a number is a prime number.

**Parameters**

| | |
|---|---|
| *p* | number to test |

**Returns**

true iff p is a prime

The documentation for this class was generated from the following file:

- src/math/primes/primes_basic.h

## 6.49 math::primes::PrimesFast_< PowerModImpl > Class Template Reference

`#include <primes_fast.h>`

### Static Public Member Functions

- static bool isPrime (BaseType p)

### Private Types

- typedef long long int BaseType

### Private Member Functions

- STATIC_ASSERT (std::numeric_limits< BaseType >::digits > 50,"We need at least 50-bit integer as a base tyep")

## 6.49.1 Detailed Description

**template<class PowerModImpl>class math::primes::PrimesFast_< PowerModImpl >**

Deterministic version of Miller-Rabbin primality test.

The implementation is based on article ON STRONG PSEUDOPRIMES TO SEVERAL BASES by GERHARD JAESCHKE It's running time is O(log(n)^2) (multiplication is supposed to be O(log n))

**Parameters**

| | |
|---|---|
| *PowerModImpl* | implementation of modular exponentiation |

**Precondition**

- PowerModImpl have functions
  - powermod(base, exponent, modulus)
  - multmod(a, b, modulus)
- long long int is 64bit

## 6.49.2 Member Typedef Documentation

### 6.49.2.1 template<class PowerModImpl > typedef long long int math::primes::PrimesFast_< PowerModImpl >::BaseType [private]

Base type for our computation, at least 50 bits

## 6.49.3 Member Function Documentation

### 6.49.3.1 template<class PowerModImpl > static bool math::primes::PrimesFast_< PowerModImpl >::isPrime ( BaseType *p* ) [inline, static]

Miller-Rabin test with fixed withesses which is correct up to 3*10^14. (but the implementation is only for integers.)

**Parameters**

| | |
|---|---|
| *p* | number to test, should be positive |

**Returns**

true iff *p* is prime

#### 6.49.3.2 template<class PowerModImpl > math::primes::PrimesFast_< PowerModImpl >::STATIC_ASSERT ( std::numeric_limits< BaseType >::digits , 50 , "We need at least 50-bit integer as a base tyep" ) [private]

The documentation for this class was generated from the following file:

- src/math/primes/primes_fast.h

## 6.50 math::primes::PrimesSlow Class Reference

`#include <primes_slow.h>`

### Static Public Member Functions

- template<typename BaseType >
  static bool isPrime (BaseType p)

### 6.50.1 Detailed Description

Very naive implementation of primality testing.

**Warning**

: This is terribly slow! Use other methods instead, this is implementation is just for testing.

### 6.50.2 Member Function Documentation

#### 6.50.2.1 template<typename BaseType > static bool math::primes::PrimesSlow::isPrime ( BaseType *p* ) [inline, static]

Checks whether a specified number is prime

**Precondition**

- *BaseType* is integral typpe

**Parameters**

| | |
|---|---|
| *p* | number to be tested |

**Returns**

true iff p is a prime number

The documentation for this class was generated from the following file:

- src/math/primes/primes_slow.h

## 6.51 strings::search::RabinKarp Class Reference

`#include <rabin_karp.h>`

### Static Public Member Functions

- template<typename _Iterator , typename _PatternIterator >
  static void search (_Iterator first, _Iterator last, _PatternIterator pattern_first, _PatternIterator pattern_last, bool checkFalsePositives, strings::search_callback::SearchCallback< _Iterator > *callback)

### Static Private Member Functions

- template<typename _Iterator , typename _PatternIterator >
  static bool checkMatch (_Iterator start, _PatternIterator pattern_first, _PatternIterator pattern_last)

### 6.51.1 Member Function Documentation

#### 6.51.1.1 template<typename _Iterator , typename _PatternIterator > static bool strings::search::RabinKarp::checkMatch ( _Iterator *start*, _PatternIterator *pattern_first*, _PatternIterator *pattern_last* ) [inline, static, private]

#### 6.51.1.2 template<typename _Iterator , typename _PatternIterator > static void strings::search::RabinKarp::search ( _Iterator *first*, _Iterator *last*, _PatternIterator *pattern_first*, _PatternIterator *pattern_last*, bool *checkFalsePositives*, strings::search_callback::SearchCallback< _Iterator > * *callback* ) [inline, static]

The documentation for this class was generated from the following file:

- src/strings/search_rabin_karp/rabin_karp.h

## 6.52 Rand Class Reference

`#include <rand.h>`

### Public Member Functions

- Rand (unsigned int seed)
- unsigned int rand ()
- int rand (int min, int max)
- int exprand (unsigned int min, unsigned int max)
- double randdouble (double min, double max)
- double expranddouble (double min, double max)

### Private Attributes

- uint64_t my_seed

### 6.52.1 Constructor & Destructor Documentation

#### 6.52.1.1 Rand::Rand ( unsigned int *seed* )

### 6.52.2 Member Function Documentation

#### 6.52.2.1 int Rand::exprand ( unsigned int *min,* unsigned int *max* )

#### 6.52.2.2 double Rand::expranddouble ( double *min,* double *max* )

#### 6.52.2.3 int Rand::rand ( int *min,* int *max* )

#### 6.52.2.4 unsigned int Rand::rand ( )

Vygeneruj nahodne cislo s exponencialnym rozlozenim (rovnomerne rozlozenie v log-hodnotach) z intervalu <min, max)

#### 6.52.2.5 double Rand::randdouble ( double *min,* double *max* )

### 6.52.3 Member Data Documentation

#### 6.52.3.1 uint64_t Rand::my_seed [private]

The documentation for this class was generated from the following files:

- src/utils/rand/rand.h
- src/utils/rand/rand.cpp

## 6.53 math::rational::Rational< T > Class Template Reference

`#include <rational.h>`

### Public Member Functions

- C_ASSERT (NumericType< T >::isInt)
- Rational ()
- Rational (T value)
- Rational (T n, T d)
- Rational (const Rational< T > &value)
- Rational< T > inverted () const
- T numerator () const
- T denominator () const
- void normalize ()

### Private Attributes

- T num
- T den

### Related Functions

(Note that these are not member functions.)

- template<typename T >
  Rational< T > operator* (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  Rational< T > operator/ (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  Rational< T > operator+ (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  Rational< T > operator- (const Rational< T > &a, const Rational< T > &b)

**template<typename T> class math::rational::Rational< T >**

### 6.53.1 Constructor & Destructor Documentation

#### 6.53.1.1 template<typename T> math::rational::Rational< T >::Rational ( ) [inline]

Constructor, creates zero

#### 6.53.1.2 template<typename T> math::rational::Rational< T >::Rational ( T *value* ) [inline]

Constructor,

**Parameters**

| | |
|---|---|
| *value* | integer value of the rational number |

#### 6.53.1.3 template<typename T> math::rational::Rational< T >::Rational ( T *n*, T *d* ) [inline]

Constructor

**Parameters**

| | |
|---|---|
| *n* | numerator |
| *d* | denominator |

#### 6.53.1.4 template<typename T> math::rational::Rational< T >::Rational ( const Rational< T > & *value* ) [inline]

Copy constructor

### 6.53.2 Member Function Documentation

#### 6.53.2.1 template<typename T> math::rational::Rational< T >::C_ASSERT ( NumericType< T >::isInt )

#### 6.53.2.2 template<typename T> T math::rational::Rational< T >::denominator ( ) const [inline]

**Returns**

denominator

#### 6.53.2.3 template<typename T> Rational<T> math::rational::Rational< T >::inverted ( ) const [inline]

**Returns**

inverse number 1/x

#### 6.53.2.4 template<typename T> void math::rational::Rational< T >::normalize ( ) [inline]

Normalize rational number. After this operation, numerator and demoninator will be coprime and denominator will be positive.

#### 6.53.2.5 template<typename T> T math::rational::Rational< T >::numerator ( ) const [inline]

**Returns**

numerator

### 6.53.3 Friends And Related Function Documentation

#### 6.53.3.1 template<typename T > Rational< T > operator∗ ( const Rational< T > & *a,* const Rational< T > & *b* ) [related]

Multiply two fractions

Note: there exists slight optimization of multiplication - for normalized fractions (a/b) and (c/d) multiplication (a/b) ∗ (c/d) may be written as normalize(c/b) ∗ normalize(a/d) hovewer we will not use it here.

**Returns**

a∗b

#### 6.53.3.2 template<typename T > Rational< T > operator+ ( const Rational< T > & *a,* const Rational< T > & *b* ) [related]

Add two fractions

**Returns**

a + b

#### 6.53.3.3 template<typename T > Rational< T > operator- ( const Rational< T > & *a,* const Rational< T > & *b* ) [related]

Subtract two fractions

**Returns**

a - b

#### 6.53.3.4 template<typename T > Rational< T > operator/ ( const Rational< T > & *a*, const Rational< T > & *b* ) [related]

Divide two fractions

**Returns**

a / b

### 6.53.4 Member Data Documentation

#### 6.53.4.1 template<typename T> T math::rational::Rational< T >::den [private]

denominator

#### 6.53.4.2 template<typename T> T math::rational::Rational< T >::num [private]

numerator

The documentation for this class was generated from the following file:

- src/math/rational/rational.h

## 6.54 strings::search::RollingHash< BaseType > Class Template Reference

`#include <rolling_hash.h>`

### Public Member Functions

- RollingHash (SizeType length, BaseType modulus_, BaseType c_)
- template<typename ValueType >
  BaseType roll (ValueType new_element, ValueType discarded_element)
- BaseType getHash () const

### Private Types

- typedef size_t SizeType

### Private Attributes

- BaseType modulus
- BaseType c
- BaseType hash

- BaseType c_len
- SizeType length

### 6.54.1 Detailed Description

template<typename BaseType>class strings::search::RollingHash< BaseType >

Rolling hash implementation -- can efficiently compute hash of sequence of fixed length.

### 6.54.2 Member Typedef Documentation

#### 6.54.2.1 template<typename BaseType> typedef size_t strings::search::RollingHash< BaseType >::SizeType [private]

### 6.54.3 Constructor & Destructor Documentation

#### 6.54.3.1 template<typename BaseType> strings::search::RollingHash< BaseType >::RollingHash ( SizeType *length*, BaseType *modulus_*, BaseType *c_* ) [inline]

Create rolling hash and initialize it to zero sequence.

The rolling hash value is defined as hash = sum_{i=0}^{length-1} seq[i] * c_^(length-1-i) mod modulus_

**Parameters**

| | |
|---|---|
| *length* | length of the sequence |
| *modulus_* | modulus is best to be a prime |
| *c_* | |

### 6.54.4 Member Function Documentation

#### 6.54.4.1 template<typename BaseType> BaseType strings::search::RollingHash< BaseType >::getHash ( ) const [inline]

**Returns**

current hash value

#### 6.54.4.2 template<typename BaseType> template<typename ValueType > BaseType strings::search::RollingHash< BaseType >::roll ( ValueType *new_element*, ValueType *discarded_element* ) [inline]

Moves the hashing function window by one character

Note: This implementation is very conservative in considering overflows. If you know

about your data limitations, it can be probably optimized to two multiplications and one modular division.

**Precondition**

- discarded elements should be consistent with the #new_elements when rolling length times

**Parameters**

| | |
|---|---|
| *new_element* | new element of the sequence |
| *discarded_element* | element of the sequence which "fell out" |

**Returns**

new hash value

### 6.54.5 Member Data Documentation

#### 6.54.5.1 template<typename BaseType> BaseType strings::search::RollingHash< BaseType >::c [private]

TODO

#### 6.54.5.2 template<typename BaseType> BaseType strings::search::RollingHash< BaseType >::c_len [private]

value $c^{length-1}$

#### 6.54.5.3 template<typename BaseType> BaseType strings::search::RollingHash< BaseType >::hash [private]

current value of the hash

#### 6.54.5.4 template<typename BaseType> SizeType strings::search::RollingHash< BaseType >::length [private]

TODO: potrebne?

#### 6.54.5.5 template<typename BaseType> BaseType strings::search::RollingHash< BaseType >::modulus [private]

modulus of the hash function

The documentation for this class was generated from the following file:

- src/strings/search_rabin_karp/rolling_hash.h

## 6.55 strings::search_callback::SearchCallback< _Iterator > Class Template Reference

`#include <search_callback.h>`

### Public Member Functions

- virtual void foundMatch (const _Iterator &start)=0

### 6.55.1 Detailed Description

**template<typename _Iterator>class strings::search_callback::SearchCallback< _Iterator >**

Callback that reports match in string search problem

### 6.55.2 Member Function Documentation

#### 6.55.2.1 template<typename _Iterator> virtual void strings::search_callback::SearchCallback< _Iterator >::foundMatch ( const _Iterator & *start* ) [pure virtual]

The documentation for this class was generated from the following file:

- src/strings/search_callback/search_callback.h

## 6.56 strings::suffix_array::SearchHelper< _Iterator > Class Template Reference

`#include <binsearch.h>`

### Public Member Functions

- SearchHelper (const _Iterator &first, const _Iterator &last_)
- template<typename _PatternIterator >
  int compare (const std::pair< _PatternIterator, _PatternIterator > &pattern, const int &a)
- template<typename _PatternIterator >
  bool operator() (const std::pair< _PatternIterator, _PatternIterator > &pattern, const int &a)

- template<typename _PatternIterator >
  bool operator() (const int &a, const std::pair< _PatternIterator, _PatternIterator > &pattern)

### Private Attributes

- _Iterator base
- _Iterator last

## 6.56.1 Detailed Description

**template<typename _Iterator>class strings::suffix_array::SearchHelper< _Iterator >**

Helper for searching Can compare two suffixes

## 6.56.2 Constructor & Destructor Documentation

### 6.56.2.1 template<typename _Iterator > strings::suffix_array::SearchHelper< _Iterator >::SearchHelper ( const _Iterator & *first,* const _Iterator & *last_* ) [inline]

**Parameters**

| | |
|---|---|
| *input* | whole sequence |
| *length* | length of the sequence |

## 6.56.3 Member Function Documentation

### 6.56.3.1 template<typename _Iterator > template<typename _PatternIterator > int strings::suffix_array::SearchHelper< _Iterator >::compare ( const std::pair< _PatternIterator, _PatternIterator > & *pattern,* const int & *a* ) [inline]

**Returns**

-1 if pattern < suffix 0 if pattern = suffix 1 if pattern > suffix

### 6.56.3.2 template<typename _Iterator > template<typename _PatternIterator > bool strings::suffix_array::SearchHelper< _Iterator >::operator() ( const int & *a,* const std::pair< _PatternIterator, _PatternIterator > & *pattern* ) [inline]

### 6.56.3.3 template<typename _Iterator > template<typename _PatternIterator > bool strings::suffix_array::SearchHelper< _Iterator >::operator() ( const std::pair< _PatternIterator, _PatternIterator > & *pattern,* const int & *a* ) [inline]

Returns which of the suffixes is lexicographically smaller. Note that end of sequence is less than any of the characters, i.e. "x" < "xa"

**Returns**

true iff pattern < suffix[a]

### 6.56.4 Member Data Documentation

#### 6.56.4.1 template<typename _Iterator > _Iterator strings::suffix_array::SearchHelper< _Iterator >::base [private]

#### 6.56.4.2 template<typename _Iterator > _Iterator strings::suffix_array::SearchHelper< _Iterator >::last [private]

The documentation for this class was generated from the following file:

- src/strings/suffix_array_binsearch/binsearch.h

## 6.57 math::prime_sieve::SegmentedSieve Class Reference

`#include <segmented_sieve.h>`

### Static Public Member Functions

- static void findPrimes (long long int n, SieveCallback *callback)

### Static Private Member Functions

- static void sieve (const vector< int > &primes, long long int segment_start, long long int delta, long long int n, SieveCallback *callback)

### 6.57.1 Detailed Description

Implementation of segmented sieve

The basic idea is to a) find all primes up to sqrt(n) (inclusive) b) sieve sucessive intervals of length sqrt(n)

- each interval can be sieved in the same way as in normal sieve, we just need to maintain for each prime[i] position L[i] of last crossed-out number

### 6.57.2 Member Function Documentation

#### 6.57.2.1 static void math::prime_sieve::SegmentedSieve::findPrimes ( long long int *n*, SieveCallback * *callback* ) [inline, static]

Find all primes less than n and report them to the supplied callback

**Parameters**

| | |
|---|---|
| *n* | |
| *callback* | use this callback on each found prime |

#### 6.57.2.2 static void math::prime_sieve::SegmentedSieve::sieve ( const vector< int > & *primes*, long long int *segment_start*, long long int *delta*, long long int *n*, SieveCallback * *callback* ) [inline, static, private]

Interval sieving

The documentation for this class was generated from the following file:

- src/math/prime_sieve/segmented_sieve.h

## 6.58 strings::utils::SequenceHelper< T > Class Template Reference

`#include <sequence_helper.h>`

### Public Member Functions

- SequenceHelper (T *base_, int length_)
- SequenceHelper reversed () const
- SequenceHelper subsequence (int start_, int end_) const
- int size () const
- const T & operator[] (int pos) const
- T & operator[] (int pos)

### Private Member Functions

- SequenceHelper (T *base_, int start_, int length_)

### Private Attributes

- T * base
- int start
- int length

### 6.58.1 Detailed Description

template<typename T>class strings::utils::SequenceHelper< T >

This is a simple helper class. It takes an pointer to the sequence array and can be used to access this array. Moreover, there are some handy functions like Creating subsequences from the original sequence or reversing the direction.

Note that SequenceHelper holds only pointers to the data, not the actual data itself. This means that you shouldn't deallocate the array itself if you have active instance of SequenceHelper you may edit the data and the changes will be reflected in SequenceHelper SequenceHelper is memory-cheap (no big arrays copying, can be on the stack

### 6.58.2 Constructor & Destructor Documentation

#### 6.58.2.1 template<typename T> strings::utils::SequenceHelper< T >::SequenceHelper ( T * *base_*, int *start_*, int *length_* ) [inline, private]

#### 6.58.2.2 template<typename T> strings::utils::SequenceHelper< T >::SequenceHelper ( T * *base_*, int *length_* ) [inline]

Construct the helper.

**Parameters**

| | |
|---|---|
| *base_* | pointer to the start of the sequence (index 0) |
| *length_* | length of the sequence |

### 6.58.3 Member Function Documentation

#### 6.58.3.1 template<typename T> const T& strings::utils::SequenceHelper< T >::operator[] ( int *pos* ) const [inline]

[] operator like in arrays.

**Parameters**

| | |
|---|---|
| *pos* | index of the element we want to retrieve. |

#### 6.58.3.2 template<typename T> T& strings::utils::SequenceHelper< T >::operator[] ( int *pos* ) [inline]

[] operator like in arrays.

**Parameters**

| | |
|---|---|
| *pos* | index of the element we want to retrieve. |

#### 6.58.3.3 template<typename T> SequenceHelper strings::utils::SequenceHelper< T >::reversed ( ) const [inline]

Reverse SequenceHelper, i.e. the resulting object will satisfy new[0] = old[size - 1], ... , new[size - 1] = old[0]

**Returns**

reversed SequenceHelper

#### 6.58.3.4 template<typename T> int strings::utils::SequenceHelper< T >::size ( ) const [inline]

Returns the size of the sequence

#### 6.58.3.5 template<typename T> SequenceHelper strings::utils::SequenceHelper< T >::subsequence ( int *start_*, int *end_* ) const [inline]

Create subsequence of this sequence, i.e. the resulting object will be reindexed like this SequenceHelper([0, 1, 2, 3, 4, 5]).subsequence(2, 5) is equal to SequenceHelper([2, 3, 4])

The resulting subsequence will be representing elements [start_, end_) of the original sequence

**Parameters**

| | |
|---|---|
| *start_* | starting index of the subsequence (will become zero in result) |
| *end_* | index after the last position of the subsequence |

**Returns**

subsequence SequenceHelper

### 6.58.4 Member Data Documentation

#### 6.58.4.1 template<typename T> T* strings::utils::SequenceHelper< T >::base [private]

#### 6.58.4.2 template<typename T> int strings::utils::SequenceHelper< T >::length [private]

#### 6.58.4.3 template<typename T> int strings::utils::SequenceHelper< T >::start [private]

The documentation for this class was generated from the following file:

- src/strings/utils/sequence_helper.h

## 6.59 strings::utils::SequenceLoader Class Reference

`#include <sequence_loader.h>`

### Static Public Member Functions

- static std::vector< unsigned char > loadSequence (const char *filename, const std::map< unsigned char, unsigned char > &conversion_map)

### 6.59.1 Detailed Description

Implementation of file reader which can re-map input characters according to same map.

This class is useful for testing string manipulations, as you can re-map 256 character alphabet to anything you want.

### 6.59.2 Member Function Documentation

#### 6.59.2.1 static std::vector<unsigned char> strings::utils::SequenceLoader::loadSequence ( const char * *filename*, const std::map< unsigned char, unsigned char > & *conversion_map* ) [inline, static]

Load sequence of re-mapped characters from input file

**Parameters**

| | |
|---|---|
| *filename* | |
| *map* | which characters match to which value, if the character read on input is not included in the map, it won't appear in the resulting sequence. |

**Returns**

re-mapped sequence read from file

The documentation for this class was generated from the following file:

- src/strings/utils/sequence_loader.h

## 6.60 math::prime_sieve::SieveCallback Class Reference

`#include <segmented_sieve.h>`

### Public Member Functions

- virtual void foundPrime (long long int p)=0

### 6.60.1 Detailed Description

Because we can't store all primes found in segmented sieve due to space complexity of result, we need way to "stream" primes.

### 6.60.2 Member Function Documentation

#### 6.60.2.1 virtual void math::prime_sieve::SieveCallback::foundPrime ( long long int *p* ) [pure virtual]

The documentation for this class was generated from the following file:

- src/math/prime_sieve/segmented_sieve.h

## 6.61 interval_trees::simple::SimpleMaxTree< T > Class Template Reference

`#include <simple_max.h>`

### Public Member Functions

- SimpleMaxTree ()
- void _clear ()
- void initialize (SizeType size)
- void initialize (SizeType size_, const T &default_value)
- T get (SizeType pos)
- void set (SizeType pos, T value)
- T get_max (SizeType left, SizeType right)

### Private Types

- typedef std::vector< T >::size_type SizeType

### Private Attributes

- SizeType base
- SizeType original_size
- std::vector< T > data

### 6.61.1 Detailed Description

template<typename T>class interval_trees::simple::SimpleMaxTree< T >

This implementation of simple interval tree with following operations:

- can change element at position i
- can compute maximum over some range [left, right)

Implementation - specific details: This implementation is solely non-recursive and uses bottom-up approach instead of standard top-bottom recursive calls. Because of lack of recursion, you can't (easily) adapt this implementation to change whole intervals.

### 6.61.2 Member Typedef Documentation

#### 6.61.2.1 template<typename T > typedef std::vector<T>::size_type interval_trees::simple::SimpleMaxTree< T >::SizeType [private]

SizeType is enough to store index to vector.

### 6.61.3 Constructor & Destructor Documentation

#### 6.61.3.1 template<typename T > interval_trees::simple::SimpleMaxTree< T >::SimpleMaxTree ( ) [inline]

### 6.61.4 Member Function Documentation

#### 6.61.4.1 template<typename T > void interval_trees::simple::SimpleMaxTree< T >::_clear ( ) [inline]

Clear tree and set it's size to zero.

#### 6.61.4.2 template<typename T > T interval_trees::simple::SimpleMaxTree< T >::get ( SizeType *pos* ) [inline]

get value at position pos

#### 6.61.4.3 template<typename T > T interval_trees::simple::SimpleMaxTree< T >::get_max ( SizeType *left,* SizeType *right* ) [inline]

get maximum over interval [left, right)

Note: right should be greater than left!

Implementation details: we walk simultaneously from the left and right up the heap-structure until we meet at the same point

#### 6.61.4.4 template<typename T > void interval_trees::simple::SimpleMaxTree< T >::initialize ( SizeType *size* ) [inline]

Shorthand for initialization

**See also**

initialize(size, default_value

#### 6.61.4.5 template<typename T > void interval_trees::simple::SimpleMaxTree< T >::initialize ( SizeType *size_*, const T & *default_value* ) [inline]

Initialize tree

#### 6.61.4.6 template<typename T > void interval_trees::simple::SimpleMaxTree< T >::set ( SizeType *pos*, T *value* ) [inline]

set value at position pos

### 6.61.5 Member Data Documentation

#### 6.61.5.1 template<typename T > SizeType interval_trees::simple::SimpleMaxTree< T >::base [private]

Number of leaves in heap structure.

Also, pos+base is the index of leaf in data. Note that 2∗base is the size of the whole tree.

#### 6.61.5.2 template<typename T > std::vector<T> interval_trees::simple::SimpleMaxTree< T >::data [private]

The data of the tree

#### 6.61.5.3 template<typename T > SizeType interval_trees::simple::SimpleMaxTree< T >::original_size [private]

The requested size of the structure, we won't allow access outside this range

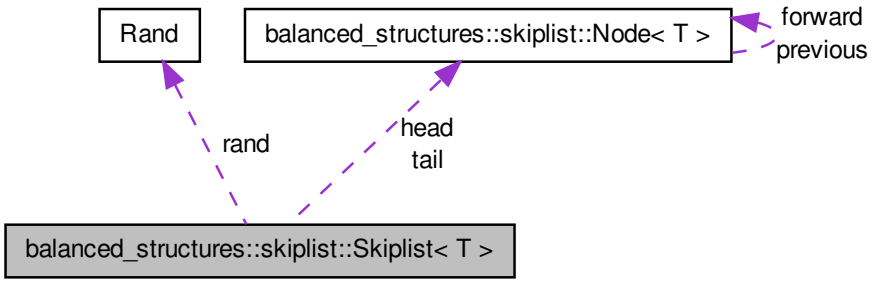
The documentation for this class was generated from the following file:

- src/interval_trees/simple/simple_max.h

# 6.62 balanced_structures::skiplist::Skiplist< T > Class Template Reference

`#include <skiplist.h>`

Collaboration diagram for balanced_structures::skiplist::Skiplist< T >:




## Public Types

- typedef skiplist::ConstIterator< T > iterator

## Public Member Functions

- Skiplist (Rand rand_)
- ~Skiplist ()
- iterator begin () const
- iterator end () const
- TrailType generic_trail (trail::TrailFunction< T > *function) const
- iterator lower_bound (const T &value) const
- iterator upper_bound (const T &value) const
- iterator find (const T &value) const
- iterator insert (const T &value)
- SizeType nodePosition (const NodeType *node) const
- SizeType xth (iterator it) const
- void erase (iterator it)
- iterator kth (SizeType k) const
- SizeType size () const

### Private Types

- typedef Node< T > NodeType
- typedef size_t SizeType
- typedef short LevelType
- typedef trail::Trail< T > TrailType

### Private Member Functions

- DISALLOW_EVIL_CONSTRUCTORS (Skiplist)

### Private Attributes

- NodeType * head
- NodeType * tail
- SizeType size_
- Rand rand

## 6.62.1 Detailed Description

**template<typename T>class balanced_structures::skiplist::Skiplist< T >**

Skiplist is sorted range container with following operations

- insert() in expected *O(log N)*
- erase() in expected *O(log N)*
- find() element by value in expected *O(log N)*
- kth() element in expected *O(log N)*
- xth() find position of current element in expected *O(log N)*

  Skiplist:

```
Nodes:
level: 3    1    4    2    1    3    4

  # -----(3)---> # -------(4)------> #
  # -> # --(2)-> # ----(3)----> # -> #
  # -> # --(2)-> # -> # --(2)-> # -> #
  # -> # -> # -> # -> # -> # -> # -> #
HEAD   1    2    7   14   19   26   TAIL
```

### 6.62.2 Member Typedef Documentation

#### 6.62.2.1 template<typename T > typedef skiplist::ConstIterator<T> balanced_structures::skiplist::Skiplist< T >::iterator

Type of the iterator

#### 6.62.2.2 template<typename T > typedef short balanced_structures::skiplist::Skiplist< T >::LevelType [private]

#### 6.62.2.3 template<typename T > typedef Node<T> balanced_structures::skiplist::Skiplist< T >::NodeType [private]

Type of the node

#### 6.62.2.4 template<typename T > typedef size_t balanced_structures::skiplist::Skiplist< T >::SizeType [private]

Type of lengths

#### 6.62.2.5 template<typename T > typedef trail::Trail<T> balanced_structures::skiplist::Skiplist< T >::TrailType [private]

Current TrailType type

### 6.62.3 Constructor & Destructor Documentation

#### 6.62.3.1 template<typename T > balanced_structures::skiplist::Skiplist< T >::Skiplist ( Rand *rand_* ) [inline]

Constructor, creates empty skiplist

#### 6.62.3.2 template<typename T > balanced_structures::skiplist::Skiplist< T >::~Skiplist ( ) [inline]

Destructor

### 6.62.4 Member Function Documentation

#### 6.62.4.1 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::begin ( ) const [inline]

Iterator to the first element

#### 6.62.4.2 template<typename T > balanced_structures::skiplist::Skiplist< T >::DISALLOW_EVIL_CONSTRUCTORS ( Skiplist< T > ) [private]

#### 6.62.4.3 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::end ( ) const [inline]

Iterator to the element after the last element

#### 6.62.4.4 template<typename T > void balanced_structures::skiplist::Skiplist< T >::erase ( iterator *it* ) [inline]

Remove element from skiplist

**Precondition**

- *it* should be a valid iterator

**Postcondition**

only iterators pointing to the deleted element will be invalidated

**Parameters**

| | |
|---|---|
| *it* | iterator |

#### 6.62.4.5 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::find ( const T & *value* ) const [inline]

**Returns**

iterator to first element equals to *value* or end() if not found

#### 6.62.4.6 template<typename T > TrailType balanced_structures::skiplist::Skiplist< T >::generic_trail ( trail::TrailFunction< T > * *function* ) const [inline]

**Returns**

trail to element determined by a TrailFunction<T>

#### 6.62.4.7 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::insert ( const T & *value* ) [inline]

inserts a new element into a skiplist

**Returns**

iterator to newly inserted element

#### 6.62.4.8 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::kth ( SizeType *k* ) const [inline]

Returns iterator to the k-th element of the skiplist

**Precondition**

- *0 <= k < size()*

#### 6.62.4.9 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::lower_bound ( const T & *value* ) const [inline]

returns iterator to last node which is < value

#### 6.62.4.10 template<typename T > SizeType balanced_structures::skiplist::Skiplist< T >::nodePosition ( const NodeType * *node* ) const [inline]

**Returns**

distance of the *node* from the start

#### 6.62.4.11 template<typename T > SizeType balanced_structures::skiplist::Skiplist< T >::size ( ) const [inline]

**Returns**

number of elements in the skiplist

#### 6.62.4.12 template<typename T > iterator balanced_structures::skiplist::Skiplist< T >::upper_bound ( const T & *value* ) const [inline]

returns iterator to first node which is > value

#### 6.62.4.13 template<typename T > SizeType balanced_structures::skiplist::Skiplist< T >::xth ( iterator *it* ) const [inline]

**Returns**

distance of the iterator from the start of the list

### 6.62.5 Member Data Documentation

#### 6.62.5.1 template<typename T > NodeType* balanced_structures::skiplist::Skiplist< T >::head [private]

Head of the skiplist, does not contain data

#### 6.62.5.2 template<typename T > Rand balanced_structures::skiplist::Skiplist< T >::rand [private]

Random generator used for node level generation

#### 6.62.5.3 template<typename T > SizeType balanced_structures::skiplist::Skiplist< T >::size_ [private]

size of the skiplist

#### 6.62.5.4 template<typename T > NodeType* balanced_structures::skiplist::Skiplist< T >::tail [private]

Tail of the skiplist, does not contain data

The documentation for this class was generated from the following file:

- src/balanced_structures/skiplist/skiplist.h

## 6.63 strings::suffix_array::SortHelper< _Iterator > Class Template Reference

`#include <sort_helper.h>`

### Public Member Functions

- SortHelper (const _Iterator first, const _Iterator last_)
- bool operator() (const int &a, const int &b)

### Private Attributes

- _Iterator base
- _Iterator last

## 6.63.1 Detailed Description

template<typename _Iterator>class strings::suffix_array::SortHelper< _Iterator >

Helper for sorting Can compare two suffixes

## 6.63.2 Constructor & Destructor Documentation

### 6.63.2.1 template<typename _Iterator> strings::suffix_array::SortHelper< _Iterator >::SortHelper ( const _Iterator *first*, const _Iterator *last_* ) [inline]

**Parameters**

| | |
|---|---|
| *input* | whole sequence |
| *length* | length of the sequence |

## 6.63.3 Member Function Documentation

### 6.63.3.1 template<typename _Iterator> bool strings::suffix_array::SortHelper< _Iterator >::operator() ( const int & *a*, const int & *b* ) [inline]

Returns which of the suffixes is lexicographically smaller. Note that end of sequence is less than any of the characters, i.e. "x" < "xa"

**Returns**

true iff suffix[a] < suffix[b]

## 6.63.4 Member Data Documentation

### 6.63.4.1 template<typename _Iterator> _Iterator strings::suffix_array::SortHelper< _Iterator >::base [private]

### 6.63.4.2 template<typename _Iterator> _Iterator strings::suffix_array::SortHelper< _Iterator >::last [private]

The documentation for this class was generated from the following file:

- src/strings/suffix_array_naive/sort_helper.h

## 6.64 utils::static_assert_::static_assert_test< x > Struct Template Reference

`#include <static_assert.h>`

### 6.64.1 Detailed Description

template<int x>struct utils::static_assert_::static_assert_test< x >

Helper structure for static assertions

The documentation for this struct was generated from the following file:

- src/utils/static_assert/static_assert.h

## 6.65 utils::static_assert_::STATIC_ASSERTION_FAILURE< true > Struct Template Reference

`#include <static_assert.h>`

### Public Types

- enum { value = 1 }

### 6.65.1 Detailed Description

template<>struct utils::static_assert_::STATIC_ASSERTION_FAILURE< true >

specialization of helper structure for valid expressions

### 6.65.2 Member Enumeration Documentation

#### 6.65.2.1 anonymous enum

**Enumerator:**

***value***

The documentation for this struct was generated from the following file:

- src/utils/static_assert/static_assert.h

# 6.66 strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix Struct Reference

## Public Member Functions

- bool operator< (const Suffix &other) const

## Public Attributes

- IndexType index
- IndexType pos_n
- IndexType pos_2n

### 6.66.1 Detailed Description

template<typename IndexType>struct strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix

Suffix is helper for sorting suffixes.

Warning: this struct does not verify any arguments, it expects that caller uses it correctly!

### 6.66.2 Member Function Documentation

#### 6.66.2.1 template<typename IndexType > bool strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix::operator< ( const Suffix & *other* ) const [inline]

### 6.66.3 Member Data Documentation

#### 6.66.3.1 template<typename IndexType > IndexType strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix::index

#### 6.66.3.2 template<typename IndexType > IndexType strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix::pos_2n

#### 6.66.3.3 template<typename IndexType > IndexType strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix::pos_n

The documentation for this struct was generated from the following file:

- src/strings/suffix_array_log2/manber_myers_log2.h

## 6.67 strings::suffix_array::SuffixArrayChecker< T > Class Template Reference

`#include <suffix_array_check.h>`

### Static Public Member Functions

- static bool isValidSuffixArray (const T s[], const int SA[], int length)
- static bool isValidSuffixArrayInverses (const T s[], const int SA[], int length)

### Static Protected Member Functions

- static bool checkCondition1Holds (const int SA[], int length)
- static bool checkCondition2Holds (const T s[], const int SA[], int length)
- static bool checkCondition3HoldsKarkkainen (const T s[], const int SA[], int length)
- static bool checkCondition3HoldsInverses (const T s[], const int SA[], int length)

### Private Member Functions

- FRIEND_TEST (SuffixArrayCheck, condition1)

### 6.67.1 Detailed Description

**template<typename T>class strings::suffix_array::SuffixArrayChecker< T >**

Class that can check validity of suffix array conditions.

### 6.67.2 Member Function Documentation

#### 6.67.2.1 template<typename T > static bool strings::suffix_array::SuffixArrayChecker< T >::checkCondition1Holds ( const int *SA[],* int *length* ) [inline, static, protected]

Checks that: For all i [0,length), SA[i] [0,length).

**Returns**

true if condition holds

#### 6.67.2.2 template<typename T > static bool strings::suffix_array::SuffixArrayChecker< T >::checkCondition2Holds ( const T *s[],* const int *SA[],* int *length* ) [inline, static, protected]

Checks that: For all i [1,length), s[SA[i - 1]] <= s[SA[i]]. Warning: there is no out-of bounds checking in this function, so be sure to call checkCondition1Holds() first!

**Returns**

true if condition holds

#### 6.67.2.3 template<typename T > static bool strings::suffix_array::SuffixArrayChecker< T >::checkCondition3HoldsInverses ( const T *s[],* const int *SA[],* int *length* ) [inline, static, protected]

Check that: For all i [1,length) such that s[SA[i - 1]] = s[SA[i]] and SA[i - 1] != n - 1, there exists j, k [0,n) such that SA[j] = SA[i - 1]+1, SA[k] = SA[i]+1 and j<k.

This algorithm requires linear memory becouse it needs inverse array to suffix array.

Warning: this function does not check for out-of-bounds, check condition 1 first.

**Returns**

true if condition holds

#### 6.67.2.4 template<typename T > static bool strings::suffix_array::SuffixArrayChecker< T >::checkCondition3HoldsKarkkainen ( const T *s[],* const int *SA[],* int *length* ) [inline, static, protected]

Checks following reformulation of 3. condition: For all characters c alphabet: If SA[a, b] contains the suffixes starting with the character c, then SA[a]+1, SA[a+1]+1,... , SA[b]+1 occur in SA in this order (but not consecutively in general), except that the first entry SA[a] + 1 is missing when c = s[n - 1].

Warning: this function does not check for out-of-the bounds, so you should check condition 1 first!

Note that memory O(sigma) where sigma is size of the alphabet. If your alphabet is positive-indexed and not sparse, you may change std::map to array in this implementation to gain speed and less memory storage.

Note: reformulation of 3rd condition is equal to the 3rd condition itself only when the first two conditions hold. The results of checkCondition3HoldsKarkkainen may be therefore different from checkCondition3HoldsInverses unless you assert the condition 1,2.

Based on pseudocode from the paper.

**Returns**

true if condition holds

#### 6.67.2.5 template<typename T > strings::suffix_array::SuffixArrayChecker< T >::FRIEND_TEST ( SuffixArrayCheck , condition1 ) [private]

#### 6.67.2.6 template<typename T > static bool strings::suffix_array::SuffixArrayChecker< T >::isValidSuffixArray ( const T *s[]*, const int *SA[]*, int *length* ) [inline, static]

Liner-time check of suffix-array invariant. Uses O(alphabet) memory.

**Parameters**

| | |
|---|---|
| *s* | original sequence |
| *sa* | suffix array to be checked |
| *length* | length of the sequence (and suffix array) |

**Returns**

true if the suffix array is correct for the sequence

#### 6.67.2.7 template<typename T > static bool strings::suffix_array::SuffixArrayChecker< T >::isValidSuffixArrayInverses ( const T *s[]*, const int *SA[]*, int *length* ) [inline, static]

Liner-time check of suffix-array invariant. Uses O(n) memory.

**Parameters**

| | |
|---|---|
| *s* | original sequence |
| *sa* | suffix array to be checked |
| *length* | length of the sequence (and suffix array) |

**Returns**

true if the suffix array is correct for the sequence

The documentation for this class was generated from the following file:

- src/strings/suffix_array_check/suffix_array_check.h

## 6.68 strings::TestdataFiles Class Reference

`#include <testdata.h>`

### Static Public Attributes

- static const char * ARTIFICIAL_RANDOM = "testdata/artificial/random.txt"

- static const char * ARTIFICIAL_ALPHABET_SMALL = "testdata/artificial/alphabet.small.txt"
- static const char * ARTIFICIAL_ALPHABET_BIG = "testdata/artificial/alphabet.txt"
- static const char * ARTIFICIAL_AAA_SMALL = "testdata/artificial/aaa.small.txt"
- static const char * ARTIFICIAL_AAA_BIG = "testdata/artificial/aaa.txt"
- static const char * TEXT_BIBLE = "testdata/larger/bible.txt"
- static const char * TEXT_FACTBOOK = "testdata/larger/world192.txt"
- static const char * GENOME_ECOLI = "testdata/larger/E.coli"
- static const char * ARTIFICIAL_PI = "testdata/misc/pi.txt"
- static const char * GENOME_SHORT = "testdata/genome/chrUn_gl000211.fa"
- static const char * GENOME_CHROMOSOME_Y = "testdata/genome/chrY.fa"
- static const char * SOURCE_CODE_PHP = "testdata/big/php-5.3.5.tar"
- static const char * TEXT_APACHE_LOGS = "testdata/big/access.log"

### 6.68.1 Detailed Description

Contains names of the test files

### 6.68.2 Member Data Documentation

#### 6.68.2.1 const char * strings::TestdataFiles::ARTIFICIAL_AAA_BIG = "testdata/artificial/aaa.txt" [static]

File with repeated "a", size = 100k

#### 6.68.2.2 const char * strings::TestdataFiles::ARTIFICIAL_AAA_SMALL = "testdata/artificial/aaa.small.txt" [static]

File with repeated "a", size = 30k

#### 6.68.2.3 const char * strings::TestdataFiles::ARTIFICIAL_ALPHABET_BIG = "testdata/artificial/alphabet.txt" [static]

File with repeated pattern "abcd....z", size = 100k

#### 6.68.2.4 const char * strings::TestdataFiles::ARTIFICIAL_ALPHABET_SMALL = "testdata/artificial/alphabet.small.txt" [static]

File with repeated pattern "abcd....z", size = 30k

#### 6.68.2.5 const char * strings::TestdataFiles::ARTIFICIAL_PI = "testdata/misc/pi.txt" [static]

Text representation of Pi number up to 1M digits

#### 6.68.2.6 const char * strings::TestdataFiles::ARTIFICIAL_RANDOM = "testdata/artificial/random.txt" [static]

File with random base64 characters, size = 1M

#### 6.68.2.7 const char * strings::TestdataFiles::GENOME_CHROMOSOME_Y = "testdata/genome/chrY.fa" [static]

Human genome for Y chromosome, cca 60M

#### 6.68.2.8 const char * strings::TestdataFiles::GENOME_ECOLI = "testdata/larger/E.coli" [static]

E.coli genome sequence, size = 4.6M

#### 6.68.2.9 const char * strings::TestdataFiles::GENOME_SHORT = "testdata/genome/chrUn_gl000211.fa" [static]

Short human genome sequence

#### 6.68.2.10 const char * strings::TestdataFiles::SOURCE_CODE_PHP = "testdata/big/php-5.3.5.tar" [static]

Source code tar archive for php, cca 93M

#### 6.68.2.11 const char * strings::TestdataFiles::TEXT_APACHE_LOGS = "testdata/big/access.log" [static]

Apache logs, cca 16M

#### 6.68.2.12 const char * strings::TestdataFiles::TEXT_BIBLE = "testdata/larger/bible.txt" [static]

Text file with bible, size = 4M

#### 6.68.2.13 const char * strings::TestdataFiles::TEXT_FACTBOOK = "testdata/larger/world192.txt" [static]

Text file with world factbook, size = 2M

The documentation for this class was generated from the following file:

- src/strings/testdata.h

## 6.69 utils::timer::Timer Class Reference

`#include <timer.h>`

### Public Member Functions

- Timer ()
- void reset ()
- double elapsed_time_sec ()

### Private Attributes

- clock_t start_time

### 6.69.1 Detailed Description

Basic measurement of time intervals

Usage:

```
Timer t;
hard_work();
cout << t.get_elapsed_time();
t.reset();
hard_work();
cout << t.get_elapsed_time();
```

**Warning**

On some systems long times may be wrapped!

### 6.69.2 Constructor & Destructor Documentation

#### 6.69.2.1 utils::timer::Timer::Timer ( ) [inline]

Construct the Timer, the time is measured from this moment

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | on failure |

### 6.69.3 Member Function Documentation

#### 6.69.3.1 double utils::timer::Timer::elapsed_time_sec ( ) [inline]

Returns elapsed time since last reset()

**Warning**

on sime systems long times may get wrapped

**Returns**

elapsed time in seconds

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | on failure |

#### 6.69.3.2 void utils::timer::Timer::reset ( ) [inline]

Reset the timer. Measure time from this instant.

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | on failure |

### 6.69.4 Member Data Documentation

#### 6.69.4.1 clock_t utils::timer::Timer::start_time [private]

Last time of the reset

The documentation for this class was generated from the following file:

- src/utils/timer/timer.h

## 6.70 balanced_structures::skiplist::trail::Trail< T > Struct Template Reference

`#include <skiplist_trail.h>`

### Public Attributes

- Node< T > * node [MAXLEVEL]
- SizeType position [MAXLEVEL]

### 6.70.1 Detailed Description

template<typename T>struct balanced_structures::skiplist::trail::Trail< T >

Trail is a collection of nodes directly preceding the node we selected.

The trail holds useful information for insertions and deletions.

```
Example trail for node 19 (denoted @):

  # ------------> @ -----------------> #
  # -> # -------> @ ------------> # -> #
  # -> # -------> # -> @ -------> # -> #
  # -> # -> # -> # -> # -> @ -> # -> #
HEAD   1    2    7   14   19   26   TAIL

Trail nodes: 19, 14, 7, 7
Trail position: 5, 4, 3, 3
```

### 6.70.2 Member Data Documentation

#### 6.70.2.1 template<typename T > Node<T>* balanced_structures::skiplist::trail::Trail< T >::node[MAXLEVEL]

#### 6.70.2.2 template<typename T > SizeType balanced_structures::skiplist::trail::Trail< T >::position[MAXLEVEL]

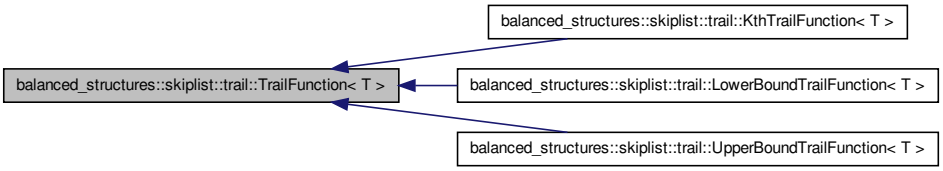
The documentation for this struct was generated from the following file:

- src/balanced_structures/skiplist/skiplist_trail.h

## 6.71 balanced_structures::skiplist::trail::TrailFunction< T > Class Template Reference

`#include <skiplist_trail.h>`

Inheritance diagram for balanced_structures::skiplist::trail::TrailFunction< T >:



### Public Member Functions

- virtual bool goFurther (const Node< T > *node, SizeType position)=0
- virtual ~TrailFunction ()

## 6.71.1 Detailed Description

template<typename T>class balanced_structures::skiplist::trail::TrailFunction< T >

Trail function is a function which determines the node which should be a result of the search.

**Precondition**

- The function should be a binary predicate returning True on all elements less/equal than the correct one, and returning False on all elements greater.

## 6.71.2 Constructor & Destructor Documentation

### 6.71.2.1 template<typename T> virtual balanced_structures::skiplist::trail::TrailFunction< T >::~TrailFunction ( ) [inline, virtual]

Destructor

## 6.71.3 Member Function Documentation

### 6.71.3.1 template<typename T> virtual bool balanced_structures::skiplist::trail::TrailFunction< T >::goFurther ( const Node< T > * *node*, SizeType *position* ) [pure virtual]

Determine if the search should continue

**Parameters**

| | |
|---|---|
| *node* | pointer to the current node |
| *position* | distance of the current node from the start of the skiplist |

**Returns**

true if the search should continue

The documentation for this class was generated from the following file:

- src/balanced_structures/skiplist/skiplist_trail.h

## 6.72 interval_trees::FullBinaryTree< NodeType >::Traverser Class Reference

`#include <full_binary_tree.h>`

### Public Member Functions

- Traverser (std::vector< NodeType > *data_, Tpos pos_, Tpos left_, Tpos right_)
- NodeType & operator* ()
- const NodeType & operator* () const
- Traverser left ()
- Traverser right ()
- Traverser parent ()
- Tpos range_left () const
- Tpos range_right () const

### Private Attributes

- std::vector< NodeType > * data_ptr
- Tpos pos
- Tpos r_left
- Tpos r_right

template<typename NodeType> class interval_trees::FullBinaryTree< NodeType >::Traverser

### 6.72.1 Constructor & Destructor Documentation

#### 6.72.1.1 template<typename NodeType > interval_trees::FullBinaryTree< NodeType >::Traverser::Traverser ( std::vector< NodeType > * *data_*, Tpos *pos_*, Tpos *left_*, Tpos *right_* ) [inline, explicit]

### 6.72.2 Member Function Documentation

#### 6.72.2.1 template<typename NodeType > Traverser interval_trees::FullBinaryTree< NodeType >::Traverser::left ( ) [inline]

#### 6.72.2.2 template<typename NodeType > const NodeType& interval_trees::FullBinaryTree< NodeType >::Traverser::operator* ( ) const [inline]

#### 6.72.2.3 template<typename NodeType > NodeType& interval_trees::FullBinaryTree< NodeType >::Traverser::operator* ( ) [inline]

#### 6.72.2.4 template<typename NodeType > Traverser interval_trees::FullBinaryTree< NodeType >::Traverser::parent ( ) [inline]

#### 6.72.2.5 template<typename NodeType > Tpos interval_trees::FullBinaryTree< NodeType >::Traverser::range_left ( ) const [inline]

#### 6.72.2.6 template<typename NodeType > Tpos interval_trees::FullBinaryTree< NodeType >::Traverser::range_right ( ) const [inline]

#### 6.72.2.7 template<typename NodeType > Traverser interval_trees::FullBinaryTree< NodeType >::Traverser::right ( ) [inline]

### 6.72.3 Member Data Documentation

#### 6.72.3.1 template<typename NodeType > std::vector<NodeType>* interval_trees::FullBinaryTree< NodeType >::Traverser::data_ptr [private]

#### 6.72.3.2 template<typename NodeType > Tpos interval_trees::FullBinaryTree< NodeType >::Traverser::pos [private]

#### 6.72.3.3 template<typename NodeType > Tpos interval_trees::FullBinaryTree< NodeType >::Traverser::r_left [private]

#### 6.72.3.4 template<typename NodeType > Tpos interval_trees::FullBinaryTree< NodeType >::Traverser::r_right [private]

The documentation for this class was generated from the following file:

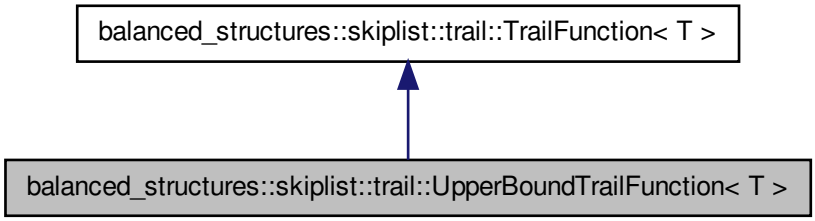
- src/interval_trees/full_binary_tree/full_binary_tree.h

## 6.73 balanced_structures::skiplist::trail::UpperBoundTrailFunction< T > Class Template Reference

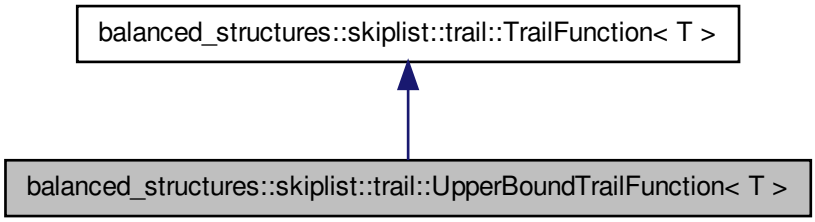
`#include <skiplist_trail.h>`

Inheritance diagram for balanced_structures::skiplist::trail::UpperBoundTrailFunction<

T >:




Collaboration diagram for balanced_structures::skiplist::trail::UpperBoundTrailFunction< T >:




## Public Member Functions

- UpperBoundTrailFunction (const T &value_)
- virtual bool goFurther (const Node< T > *node, SizeType UNUSED(position))

## Private Attributes

- T value

template<typename T> class balanced_structures::skiplist::trail::UpperBoundTrailFunction< T >

### 6.73.1 Constructor & Destructor Documentation

#### 6.73.1.1 template<typename T> balanced_structures::skiplist::trail::UpperBoundTrailFunction< T >::UpperBoundTrailFunction ( const T & *value_* ) [inline]

### 6.73.2 Member Function Documentation

#### 6.73.2.1 template<typename T> virtual bool balanced_structures::skiplist::trail::UpperBoundTrailFunction< T >::goFurther ( const Node< T > * *node*, SizeType *UNUSEDposition* ) [inline, virtual]

Goes further until we encounter a *node* with value greater than *value*

### 6.73.3 Member Data Documentation

#### 6.73.3.1 template<typename T> T balanced_structures::skiplist::trail::UpperBoundTrailFunction< T >::value [private]

Value we are searching for

The documentation for this class was generated from the following file:

- src/balanced_structures/skiplist/skiplist_trail.h

# Chapter 7

# File Documentation

## 7.1 src/automakefile.py File Reference

### Namespaces

- namespace automakefile

### Functions

- def automakefile::get_dependencies
- def automakefile::get_binary
- def automakefile::print_compile_rule
- def automakefile::print_compiletest_rule
- def automakefile::print_headers

### Variables

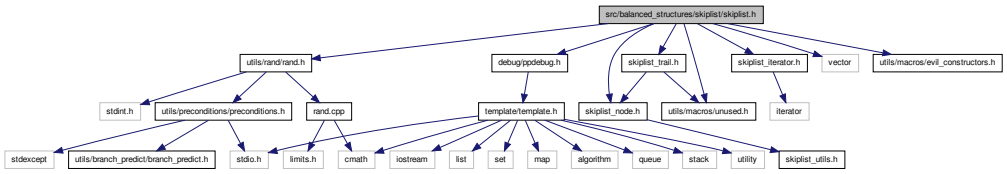
- list automakefile::EXCLUDES = ['gtest']
- string automakefile::TESTLIB = "TESTLIB=../../gtest/gtest-all.o ../../gtest/gtest_-main.o"
- string automakefile::CC = "CC=mingw32-g++"
- string automakefile::OPT = "OPT=-g -O2 -W -Wall -Werror -Wextra -mno-cygwin"
- tuple automakefile::all_files = os.listdir('.')
- tuple automakefile::unittests = filter(lambda file : re.match('.*unittest.cpp$', file), all_files)
- tuple automakefile::tests = filter(lambda file : re.match('.*_test.cpp$', file), all_-files)
- tuple automakefile::benchmarks = filter(lambda file : re.match('.*benchmark.cpp$', file), all_files)
- tuple automakefile::compiletest = filter(lambda file : re.match('.*compiletest.cpp$', file), all_files)

- automakefile::compilable = unittests+benchmarks+tests;
- tuple automakefile::b = get_binary(filename)

## 7.2 src/balanced_structures/skiplist/skiplist.h File Reference

```
#include "utils/rand/rand.h"
#include "debug/ppdebug.h"
#include "skiplist_node.h"
#include "skiplist_iterator.h"
#include "skiplist_trail.h"
#include <vector>
#include "utils/macros/unused.h"
#include "utils/macros/evil_constructors.h"
```

Include dependency graph for skiplist.h:




### Classes

- class balanced_structures::skiplist::Skiplist< T >

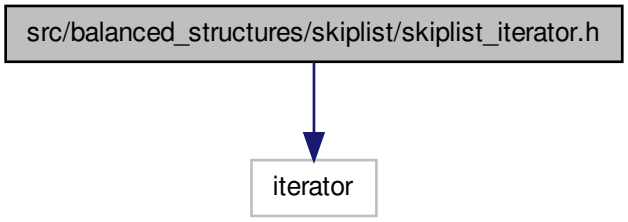
### Namespaces

- namespace balanced_structures
- namespace balanced_structures::skiplist

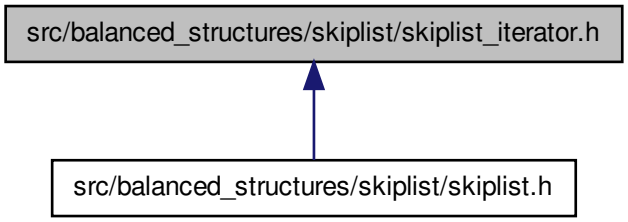
## 7.3 src/balanced_structures/skiplist/skiplist_iterator.h File Reference

```
#include <iterator>
```

Include dependency graph for skiplist_iterator.h:




This graph shows which files directly or indirectly include this file:




### Classes

- struct balanced_structures::skiplist::ConstIterator< T >

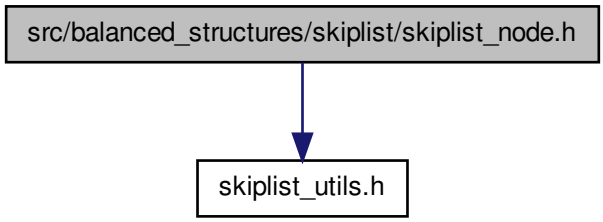
### Namespaces

- namespace balanced_structures
- namespace balanced_structures::skiplist

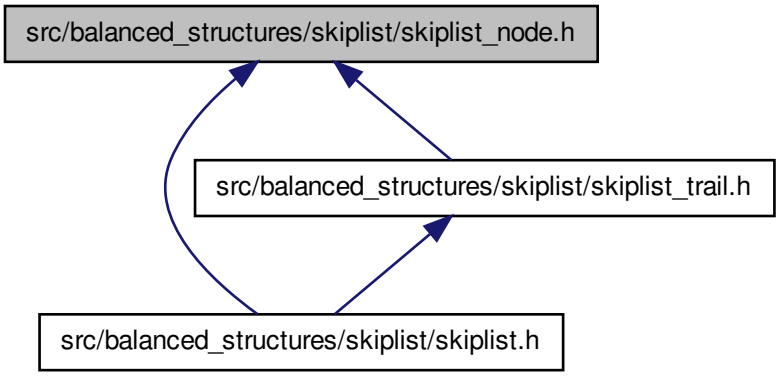
## 7.4 src/balanced_structures/skiplist/skiplist_node.h File Reference

```
#include "skiplist_utils.h"
```

Include dependency graph for skiplist_node.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class balanced_structures::skiplist::Node< T >

## Namespaces

- namespace balanced_structures
- namespace balanced_structures::skiplist

### Typedefs

- typedef short balanced_structures::skiplist::LevelType

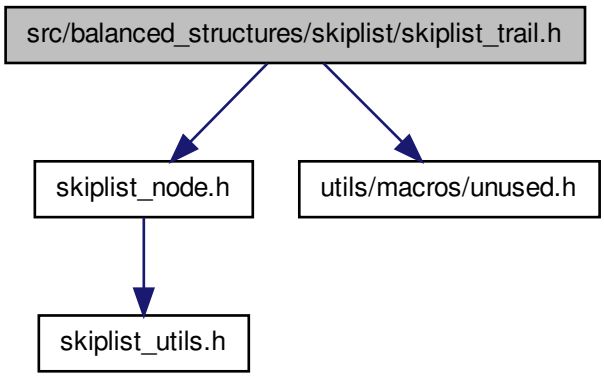
### Variables

- static const int balanced_structures::skiplist::LEVELUP_PROB = 100 / 4
- static const LevelType balanced_structures::skiplist::MAXLEVEL = 15

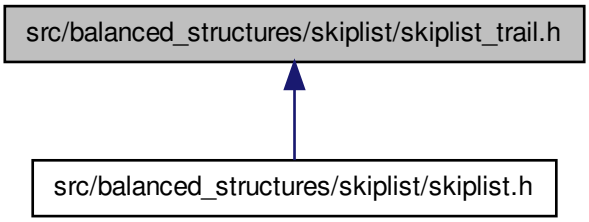
## 7.5 src/balanced_structures/skiplist/skiplist_trail.h File Reference

```
#include "skiplist_node.h"
#include "utils/macros/unused.h"
```

Include dependency graph for skiplist_trail.h:



This graph shows which files directly or indirectly include this file:




## Classes

- struct balanced_structures::skiplist::trail::Trail< T >
- class balanced_structures::skiplist::trail::TrailFunction< T >
- class balanced_structures::skiplist::trail::LowerBoundTrailFunction< T >
- class balanced_structures::skiplist::trail::UpperBoundTrailFunction< T >
- class balanced_structures::skiplist::trail::KthTrailFunction< T >

## Namespaces

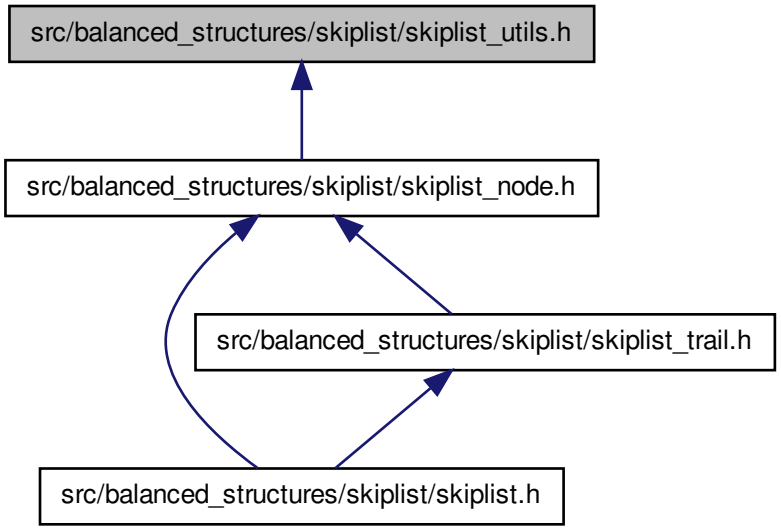
- namespace balanced_structures
- namespace balanced_structures::skiplist
- namespace balanced_structures::skiplist::trail

## Typedefs

- typedef size_t balanced_structures::skiplist::trail::SizeType

## 7.6 src/balanced_structures/skiplist/skiplist_utils.h File Reference

This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace balanced_structures
- namespace balanced_structures::skiplist
- namespace balanced_structures::skiplist::node_utils

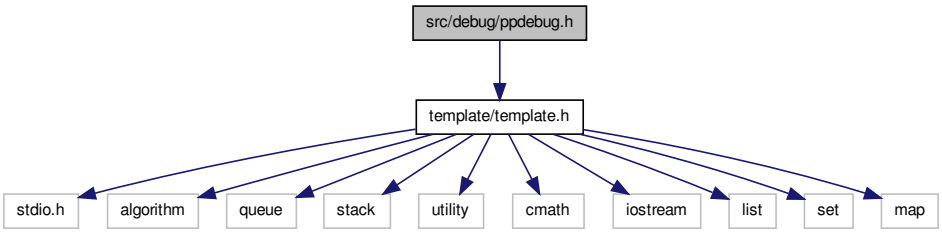
### Functions

- template<typename T >
  T balanced_structures::skiplist::node_utils::randomLevel (Rand *rand, int levelup_-
  prob_percent, T max_level)

## 7.7 src/debug/ppdebug.h File Reference

```
#include "template/template.h"
```

Include dependency graph for ppdebug.h:




This graph shows which files directly or indirectly include this file:

## Defines

- #define D(var) { cout << #var << ": " << (var) << endl; }
- #define TPL_T template <class T>
- #define TPL_ST template <class S, class T>
- #define OSTREAM(X...) ostream& operator << (ostream &out, const X& temp)
- #define _OUT(X, Y...)

## Functions

- TPL_ST OSTREAM (pair< S, T >)
- TPL_T OSTREAM (pair< T *, T * >)
- _OUT (TPL_T, vector< T >)
- _OUT (TPL_T, list< T >)
- _OUT (TPL_T, set< T >)
- _OUT (TPL_T, multiset< T >)
- _OUT (TPL_ST, map< S, T >)
- _OUT (TPL_ST, multimap< S, T >)
- _OUT (TPL_ST, set< S, T >)

### 7.7.1 Define Documentation

#### 7.7.1.1 #define _OUT( *X, Y...* )

**Value:**

```
X OSTREAM(Y) { \
        out << "[ "; FOREACH(it, temp) out << *it << ", "; out << "]"; \
        return out; };
```

#### 7.7.1.2 #define D( *var* ) { cout << #var << ": " << (var) << endl; }

#### 7.7.1.3 #define OSTREAM( *X...* ) ostream& operator << (ostream &out, const X& temp)

#### 7.7.1.4 #define TPL_ST template <class S, class T>

#### 7.7.1.5 #define TPL_T template <class T>

### 7.7.2 Function Documentation

#### 7.7.2.1 _OUT ( TPL_T , vector< T > )

#### 7.7.2.2 _OUT ( TPL_ST , set< S, T > )

#### 7.7.2.3 _OUT ( TPL_T , list< T > )

#### 7.7.2.4 _OUT ( TPL_ST , map< S, T > )

#### 7.7.2.5 _OUT ( TPL_ST , multimap< S, T > )

#### 7.7.2.6 _OUT ( TPL_T , set< T > )

#### 7.7.2.7 _OUT ( TPL_T , multiset< T > )

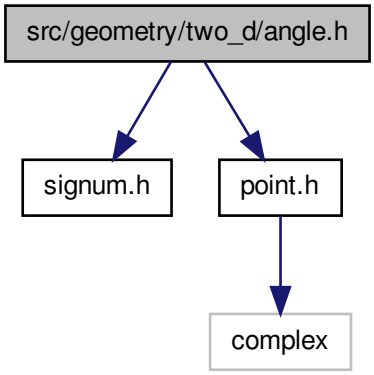
#### 7.7.2.8 TPL_T OSTREAM ( pair< T *, T * > )

#### 7.7.2.9 TPL_ST OSTREAM ( pair< S, T > )

## 7.8 src/geometry/two_d/angle.h File Reference

```
#include "signum.h"
#include "point.h"
```

Include dependency graph for angle.h:




### Namespaces

- namespace geometry
- namespace geometry::two_d

### Enumerations

- enum geometry::two_d::Quadrant {

  geometry::two_d::CENTER = 0, geometry::two_d::TOP_RIGHT = 1, geometry::two_d::TOP_LEFT = 2, geometry::two_d::BOTTOM_LEFT = 3,

  geometry::two_d::BOTTOM_RIGHT = 4 }

### Functions

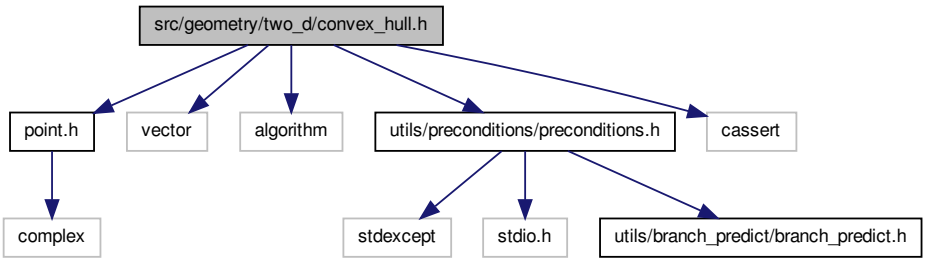
- template<typename T >
  Quadrant geometry::two_d::getQuadrant (const Point< T > point)
- template<typename T >
  bool geometry::two_d::angleLess (const Point< T > point1, const Point< T > point2)

## 7.9 src/geometry/two_d/convex_hull.h File Reference

```
#include "point.h"
```

```
#include <vector>
#include <algorithm>
#include "utils/preconditions/preconditions.h"
#include <cassert>
```

Include dependency graph for convex_hull.h:




### Classes

- class geometry::two_d::ConvexHull< T >
- class geometry::two_d::ConvexHull< T >::PointCompare

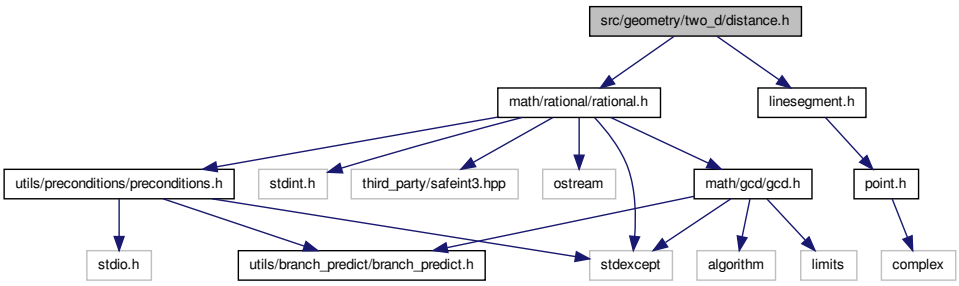
### Namespaces

- namespace geometry
- namespace geometry::two_d

## 7.10 src/geometry/two_d/distance.h File Reference

```
#include "math/rational/rational.h"
#include "linesegment.h"
```

Include dependency graph for distance.h:




## Namespaces

- namespace geometry
- namespace geometry::two_d

## Functions

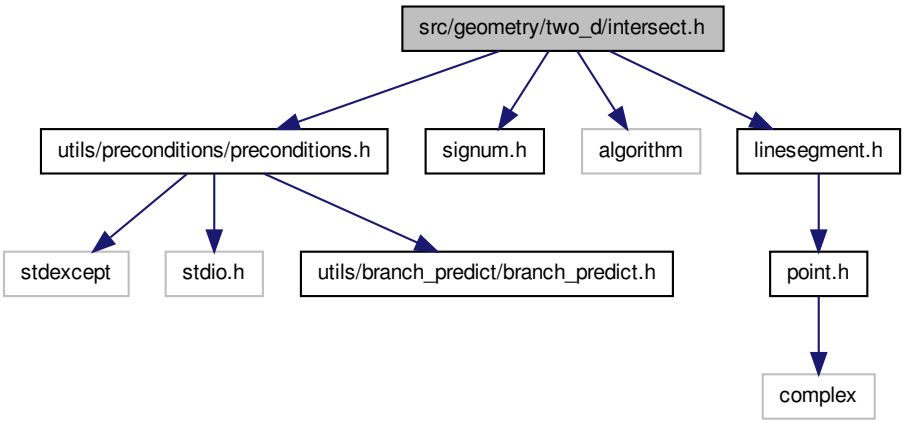
- template<typename T >
  T geometry::two_d::sqrDistancePointPoint (Point< T > p1, Point< T > p2)
- template<typename T >
  long double geometry::two_d::distancePointPoint (Point< T > p1, Point< T > p2)
- template<typename T >
  math::rational::Rational< T > geometry::two_d::sqrDistancePointLine (Point< T > p, LineSegment< T > line)
- template<typename T >
  long double geometry::two_d::distancePointLine (Point< T > p, LineSegment< T > line)
- template<typename T >
  math::rational::Rational< T > geometry::two_d::sqrDistancePointLineSegment (Point< T > p, LineSegment< T > line)
- template<typename T >
  long double geometry::two_d::distancePointLineSegment (Point< T > p, LineSegment< T > line)

# 7.11 src/geometry/two_d/intersect.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include "signum.h"
```

```
#include <algorithm>
#include "linesegment.h"
```

Include dependency graph for intersect.h:




## Namespaces

- namespace geometry
- namespace geometry::two_d

## Enumerations

- enum geometry::two_d::IntersectType { geometry::two_d::NO_INTERSECT, geometry::two_-d::INTERSECT, geometry::two_d::TANGENCY, geometry::two_d::OVERLAY }

## Functions

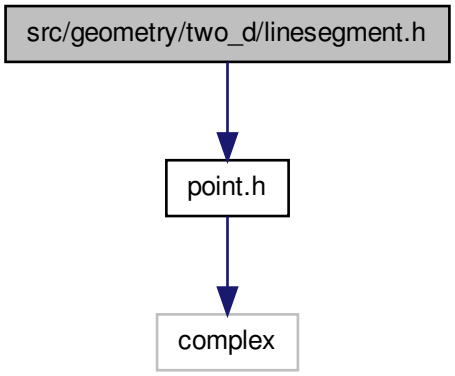
- template<typename T >
  bool geometry::two_d::pointOnLine (Point< T > p, LineSegment< T > s)
- template<typename T >
  bool geometry::two_d::pointOnLineSegment (Point< T > p, LineSegment< T > s, bool acceptCorners)
- template<typename T >
  IntersectType geometry::two_d::intervalIntersect (T a1, T a2, T b1, T b2)
- template<typename T >
  IntersectType geometry::two_d::intersectLineLineSegment (const LineSegment< T > &line, const LineSegment< T > &segment)

- template<typename T >
  IntersectType geometry::two_d::intersectLineSegmentLineSegment (const LineSegment< T > &segment1, const LineSegment< T > &segment2)

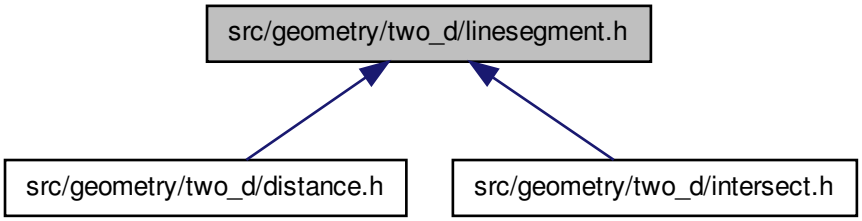
## 7.12 src/geometry/two_d/linesegment.h File Reference

```
#include "point.h"
```

Include dependency graph for linesegment.h:




This graph shows which files directly or indirectly include this file:



### Classes

- struct geometry::two_d::LineSegment< T >

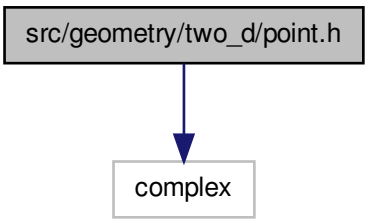
### Namespaces

- namespace geometry
- namespace geometry::two_d

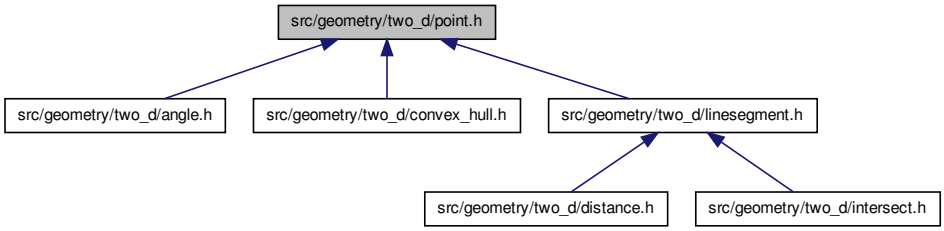
## 7.13 src/geometry/two_d/point.h File Reference

```
#include <complex>
```

Include dependency graph for point.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class geometry::two_d::Point< T >

### Namespaces

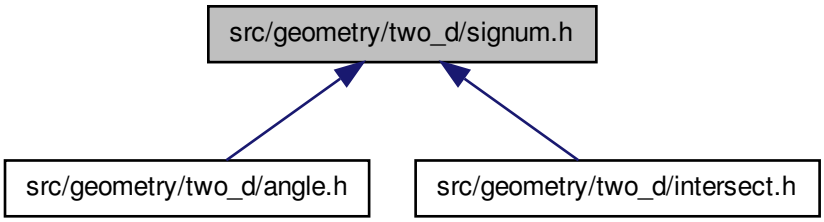
- namespace geometry
- namespace geometry::two_d

### Functions

- template<typename T >
  bool geometry::two_d::operator== (const Point< T > &a, const Point< T > &b)
- template<typename T >
  bool geometry::two_d::operator!= (const Point< T > &a, const Point< T > &b)
- template<typename T >
  Point< T > geometry::two_d::operator+ (const Point< T > &a, const Point< T > &b)
- template<typename T >
  Point< T > geometry::two_d::operator- (const Point< T > &a, const Point< T > &b)
- template<typename T >
  Point< T > geometry::two_d::operator- (const Point< T > &a)
- template<typename T >
  Point< T > geometry::two_d::operator* (const Point< T > &a, T scalar)
- template<typename T >
  Point< T > geometry::two_d::operator/ (const Point< T > &a, T scalar)

## 7.14 src/geometry/two_d/signum.h File Reference

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace geometry
- namespace geometry::two_d

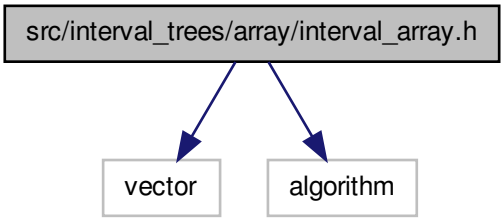
### Functions

- template<typename T >
  int geometry::two_d::signum (T n)

## 7.15 src/interval_trees/array/interval_array.h File Reference

```
#include <vector>
#include <algorithm>
```

Include dependency graph for interval_array.h:




### Classes

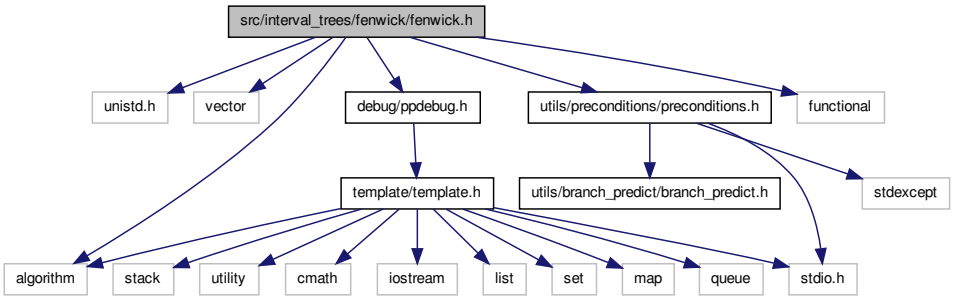
- class IntervalSumArray< ValueType >
- class IntervalMaxArray< ValueType >

## 7.16 src/interval_trees/fenwick/fenwick.h File Reference

```
#include <unistd.h>
#include <vector>
#include <algorithm>
#include "utils/preconditions/preconditions.h"
```

```
#include <functional>
#include "debug/ppdebug.h"
```

Include dependency graph for fenwick.h:




### Classes

- class interval_trees::fenwick::FenwickTree< ValueType, Operation >
- struct interval_trees::fenwick::BinaryPlus< T >
- class interval_trees::fenwick::FenwickSumTree< T >
- struct interval_trees::fenwick::BinaryMax< T >
- class interval_trees::fenwick::FenwickMaxTree< T >

### Namespaces

- namespace interval_trees
- namespace interval_trees::fenwick

### Enumerations

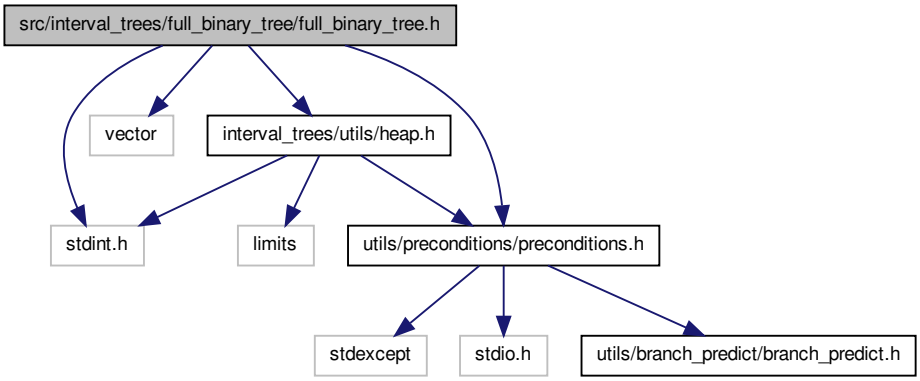
- enum interval_trees::fenwick::FenwickDirection { interval_trees::fenwick::TO_ZERO, interval_trees::fenwick::TO_INFTY }

## 7.17 src/interval_trees/full_binary_tree/full_binary_tree.h File Reference

```
#include <stdint.h>
#include <vector>
#include "utils/preconditions/preconditions.h"
```

```
#include "interval_trees/utils/heap.h"
```

Include dependency graph for full_binary_tree.h:




### Classes

- class interval_trees::FullBinaryTree< NodeType >
- class interval_trees::FullBinaryTree< NodeType >::Traverser

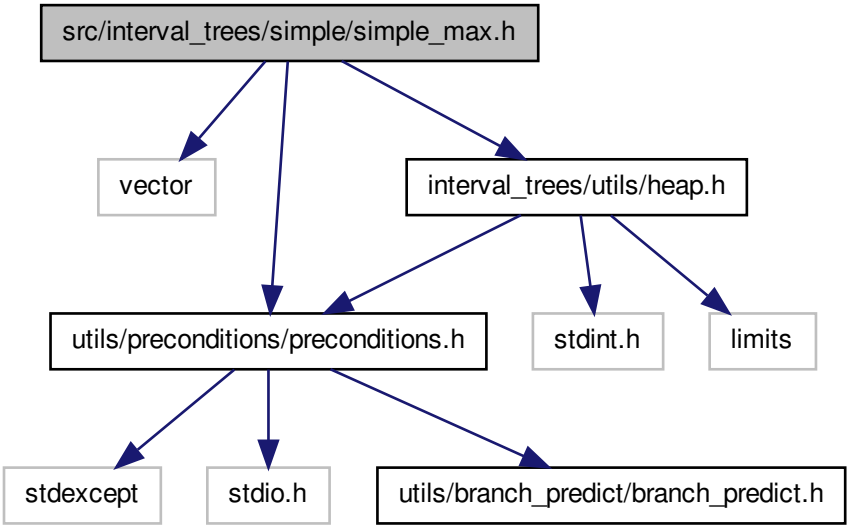
### Namespaces

- namespace interval_trees

## 7.18 src/interval_trees/simple/simple_max.h File Reference

```
#include <vector>
#include "utils/preconditions/preconditions.h"
#include "interval_trees/utils/heap.h"
```

Include dependency graph for simple_max.h:




### Classes

- class interval_trees::simple::SimpleMaxTree< T >

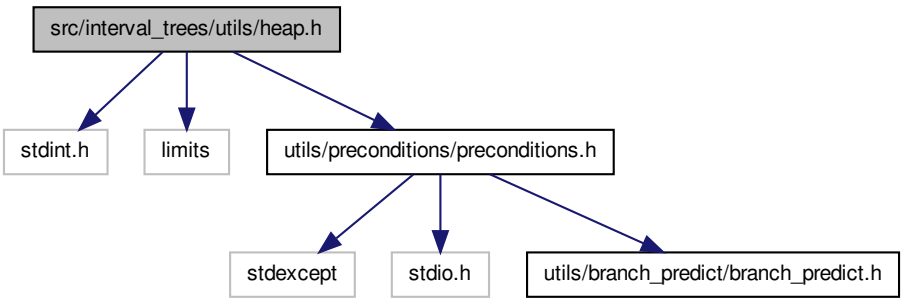
### Namespaces

- namespace interval_trees
- namespace interval_trees::simple

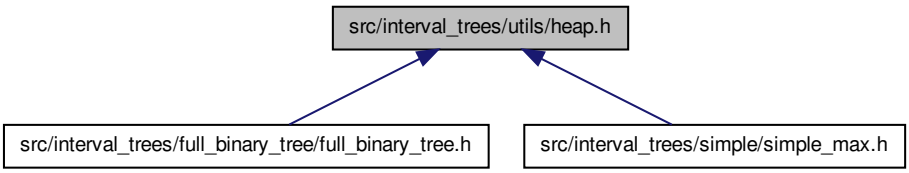
## 7.19 src/interval_trees/utils/heap.h File Reference

```
#include <stdint.h>
#include <limits>
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for heap.h:




This graph shows which files directly or indirectly include this file:




## Namespaces

- namespace heap

## Functions

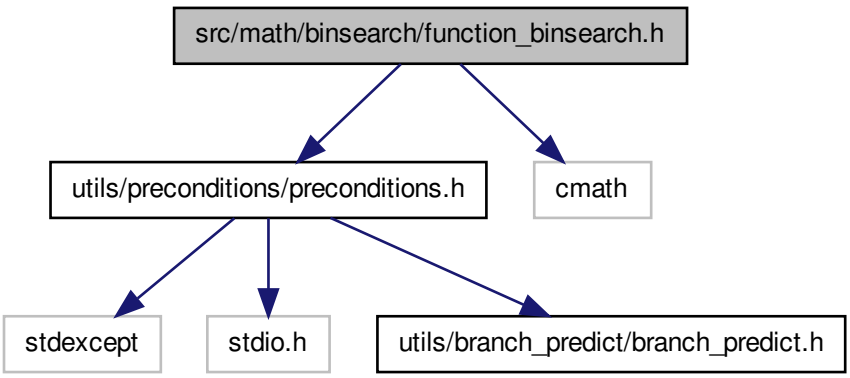
- template<typename T >
  T heap::left (T x)
- template<typename T >
  T heap::right (T x)
- template<typename T >
  T heap::parent (T x)
- template<typename T >
  bool heap::isLeftChild (T x)

- template<typename T >
  bool heap::isRightChild (T x)
- template<typename T >
  T heap::sibling (T x)
- template<typename T >
  T heap::nextPowerOfTwo (T x)

## 7.20 src/math/binsearch/function_binsearch.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <cmath>
```

Include dependency graph for function_binsearch.h:




### Classes

- class math::binsearch::Function< T >
- class math::binsearch::ConvexFunction< T >
- class math::binsearch::FunctionBinsearch< T >

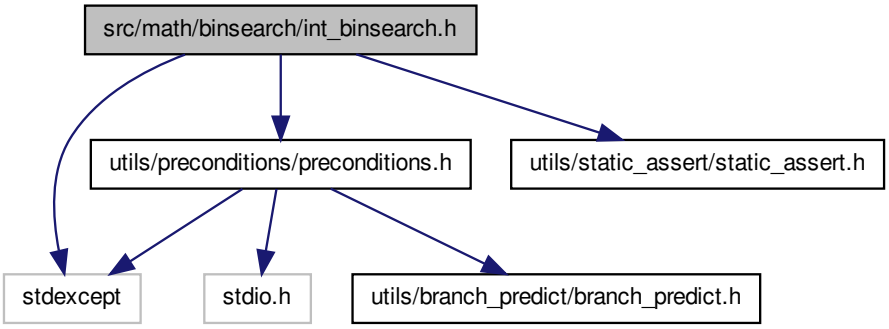
### Namespaces

- namespace math
- namespace math::binsearch

## 7.21 src/math/binsearch/int_binsearch.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include "utils/static_assert/static_assert.h"
#include <stdexcept>
```

Include dependency graph for int_binsearch.h:




### Namespaces

- namespace math
- namespace math::binsearch

### Functions

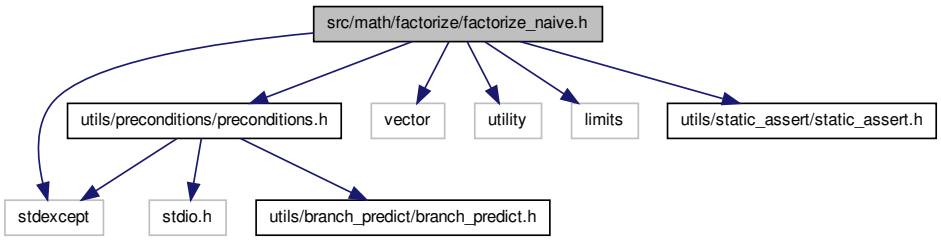
- template<typename T >
  T math::binsearch::range_middle (T left, T right)
- template<typename ValueType , typename SizeType >
  SizeType math::binsearch::lower_bound (ValueType pole[], SizeType left, SizeType right, ValueType value)
- template<typename ValueType , typename SizeType >
  SizeType math::binsearch::upper_bound (ValueType pole[], SizeType left, SizeType right, ValueType value)

## 7.22 src/math/factorize/factorize_naive.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

```
#include <vector>
#include <utility>
#include <limits>
#include <stdexcept>
#include "utils/static_assert/static_assert.h"
```

Include dependency graph for factorize_naive.h:




### Classes

- class math::factorize::FactorizeNaive_< CountType >

### Namespaces

- namespace math
- namespace math::factorize

### Typedefs

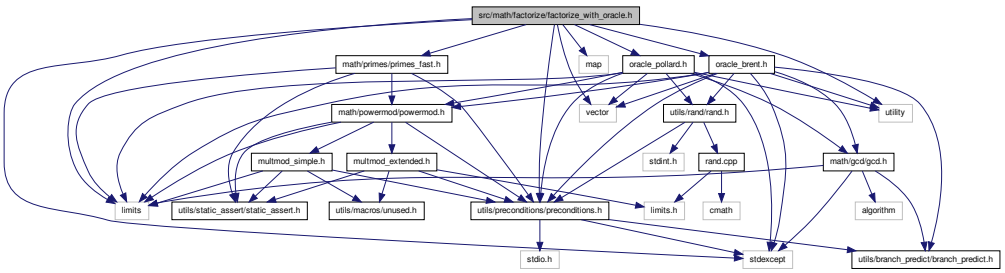
- typedef FactorizeNaive_< int > math::factorize::FactorizeNaive

## 7.23 src/math/factorize/factorize_with_oracle.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <utility>
#include <limits>
#include <map>
```

```
#include <stdexcept>
#include "math/primes/primes_fast.h"
#include "oracle_pollard.h"
#include "oracle_brent.h"
```

Include dependency graph for factorize_with_oracle.h:




### Classes

- class math::factorize::FactorizeWithOracle_< CountType, Oracle, Primes >

### Namespaces

- namespace math
- namespace math::factorize

### Typedefs

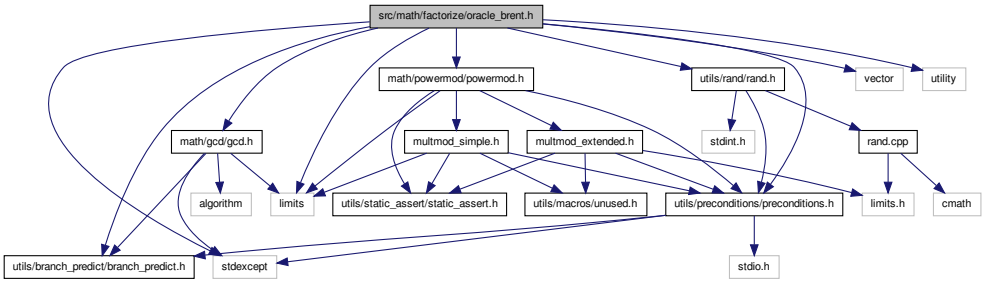
- typedef FactorizeWithOracle_< int, OraclePollard > math::factorize::FactorizePollard
- typedef FactorizeWithOracle_< int, OracleBrent > math::factorize::FactorizeBrent

## 7.24 src/math/factorize/oracle_brent.h File Reference

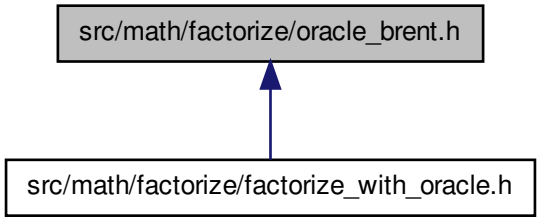
```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <utility>
#include <limits>
#include <stdexcept>
#include "math/powermod/powermod.h"
```

```
#include "math/gcd/gcd.h"
#include "utils/rand/rand.h"
#include "utils/branch_predict/branch_predict.h"
```

Include dependency graph for oracle_brent.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class math::factorize::OracleBrent_< Powermod >

## Namespaces

- namespace math
- namespace math::factorize

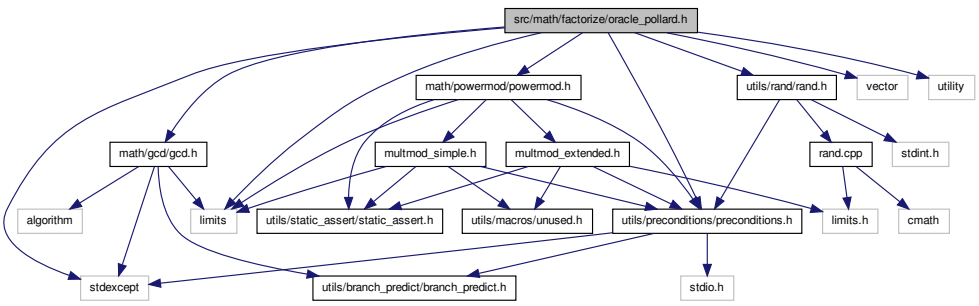
### Typedefs

- typedef OracleBrent_< math::powermod::PowermodExtended > math::factorize::OracleBrent

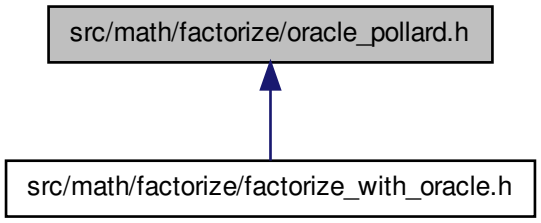
## 7.25 src/math/factorize/oracle_pollard.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <utility>
#include <limits>
#include <stdexcept>
#include "math/powermod/powermod.h"
#include "math/gcd/gcd.h"
#include "utils/rand/rand.h"
```

Include dependency graph for oracle_pollard.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class math::factorize::OraclePollard_< Powermod >

### Namespaces

- namespace math
- namespace math::factorize

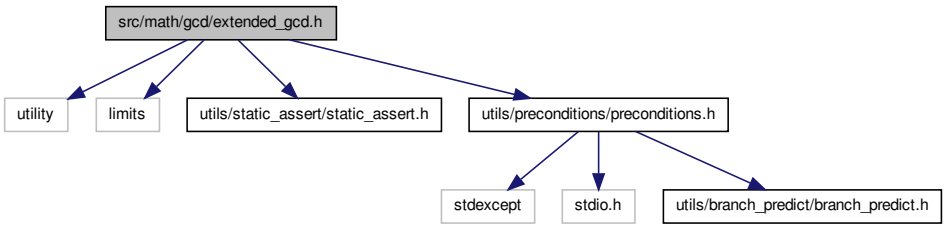
### Typedefs

- typedef OraclePollard_< math::powermod::PowermodExtended > math::factorize::OraclePollard

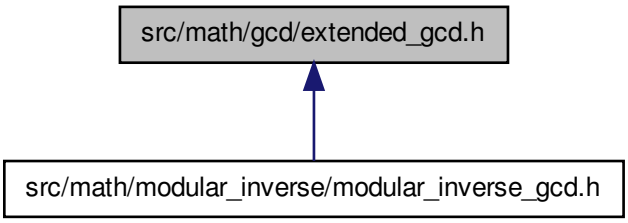
## 7.26 src/math/gcd/extended_gcd.h File Reference

```
#include <utility>
#include <limits>
#include "utils/static_assert/static_assert.h"
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for extended_gcd.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class math::gcd::ExtendedGCD

### Namespaces

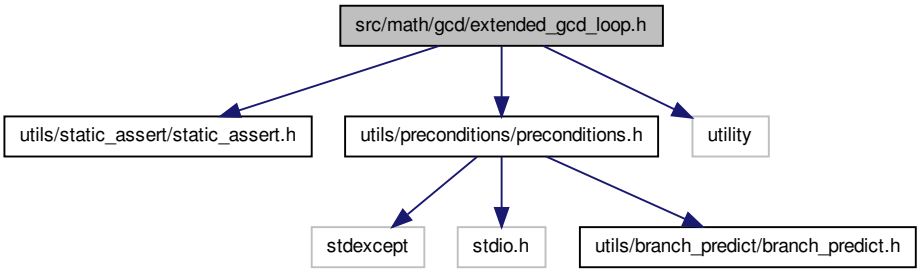
- namespace math
- namespace math::gcd

## 7.27 src/math/gcd/extended_gcd_loop.h File Reference

```
#include "utils/static_assert/static_assert.h"
#include "utils/preconditions/preconditions.h"
```

```
#include <utility>
```

Include dependency graph for extended_gcd_loop.h:




### Classes

- class math::gcd::ExtendedGCDLoop

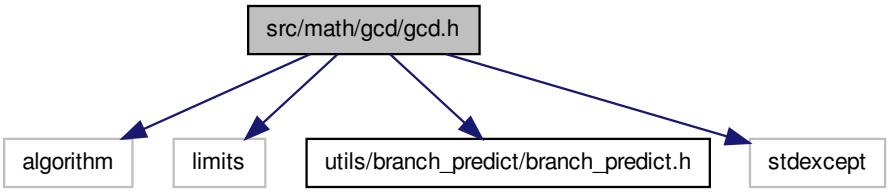
### Namespaces

- namespace math
- namespace math::gcd

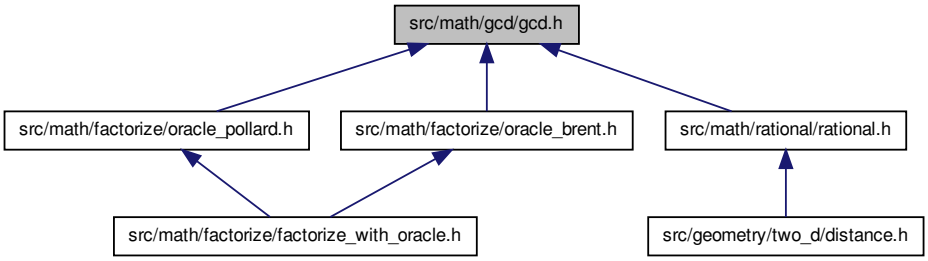
## 7.28 src/math/gcd/gcd.h File Reference

```
#include <algorithm>
#include <limits>
#include "utils/branch_predict/branch_predict.h"
#include <stdexcept>
```

Include dependency graph for gcd.h:




This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace math
- namespace math::gcd

### Functions

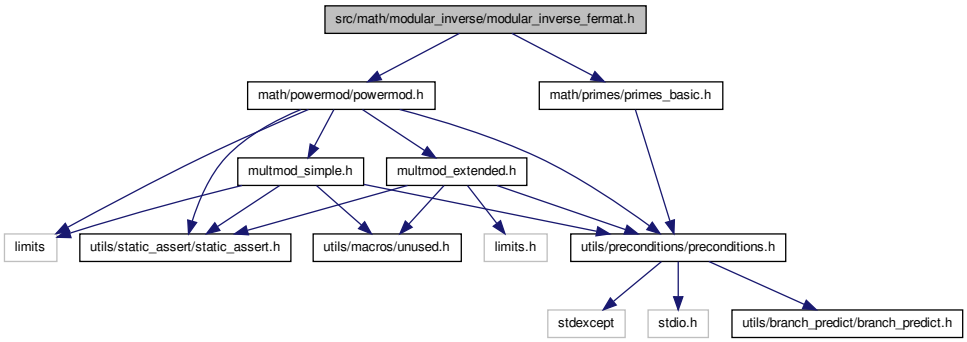
- template<typename T >
  T math::gcd::gcd (T a, T b)

## 7.29 src/math/modular_inverse/modular_inverse_fermat.h File Reference

```
#include "math/primes/primes_basic.h"
```

```
#include "math/powermod/powermod.h"
```

Include dependency graph for modular_inverse_fermat.h:




## Classes

- class math::modular_inverse::ModularInverseFermat_< PowerModImpl, checkPrimality >

## Namespaces

- namespace math
- namespace math::modular_inverse

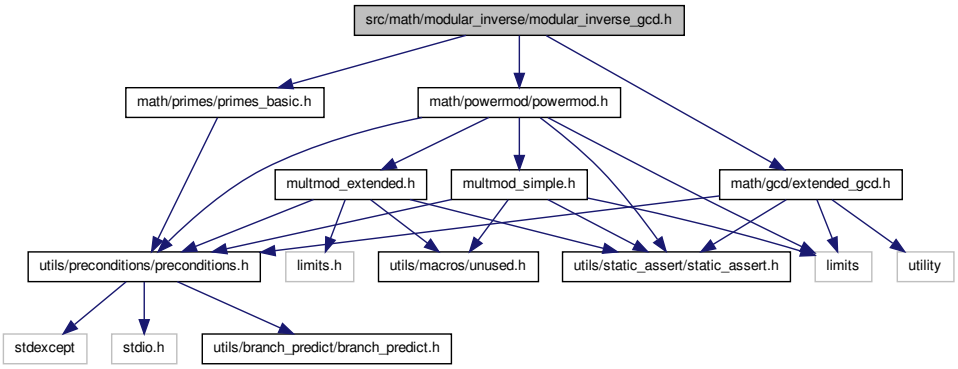
## Typedefs

- typedef ModularInverseFermat_< math::powermod::PowermodSimple > math::modular_inverse::ModularInverseFermat

# 7.30 src/math/modular_inverse/modular_inverse_gcd.h File Reference

```
#include "math/primes/primes_basic.h"
#include "math/powermod/powermod.h"
#include "math/gcd/extended_gcd.h"
```

Include dependency graph for modular_inverse_gcd.h:




### Classes

- class math::modular_inverse::ModularInverseGcd

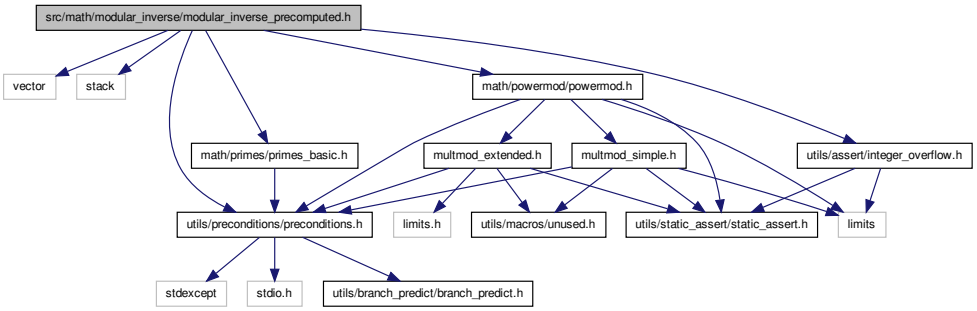
### Namespaces

- namespace math
- namespace math::modular_inverse

## 7.31 src/math/modular_inverse/modular_inverse_precomputed.h File Reference

```
#include <vector>
#include <stack>
#include <utils/preconditions/preconditions.h>
#include <utils/assert/integer_overflow.h>
#include <math/powermod/powermod.h>
#include <math/primes/primes_basic.h>
```

Include dependency graph for modular_inverse_precomputed.h:




## Classes

- class math::modular_inverse::ModularInversePrecomputed_< PowerModImpl >

## Namespaces

- namespace math
- namespace math::modular_inverse

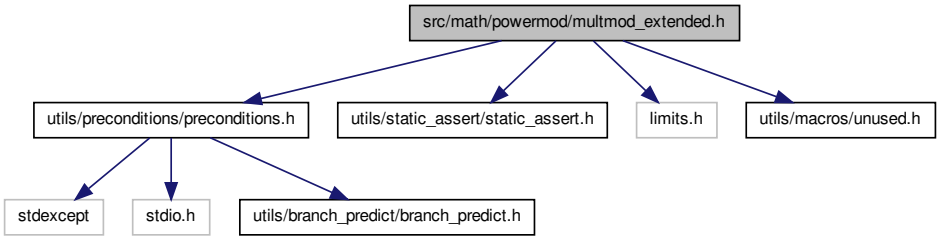
## Typedefs

- typedef ModularInversePrecomputed_< math::powermod::PowermodSimple > math::modular_inverse::ModularInversePrecomputed

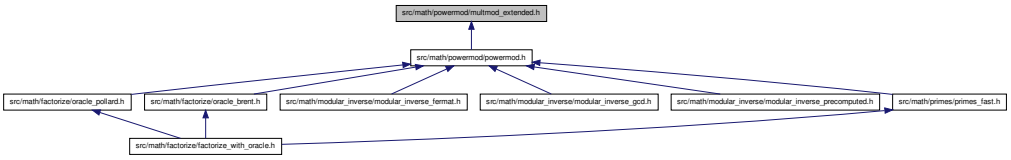
# 7.32 src/math/powermod/multmod_extended.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include "utils/static_assert/static_assert.h"
#include <limits.h>
#include "utils/macros/unused.h"
```

Include dependency graph for multmod_extended.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class math::powermod::MultmodExtended< shift >
- class math::powermod::MultmodExtendedOpt

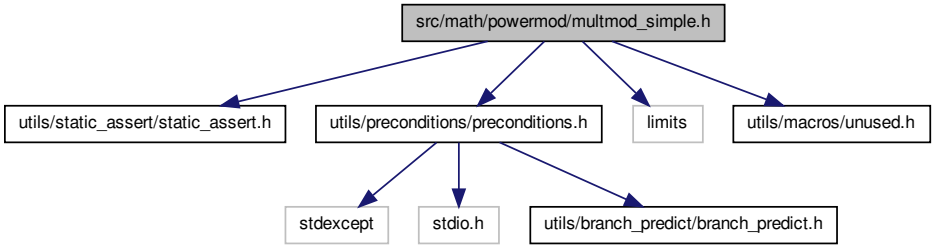
### Namespaces

- namespace math
- namespace math::powermod

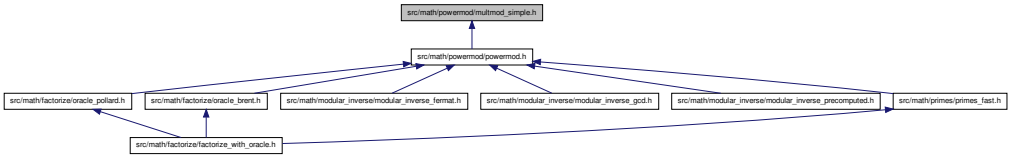
## 7.33 src/math/powermod/multmod_simple.h File Reference

```
#include "utils/static_assert/static_assert.h"
#include "utils/preconditions/preconditions.h"
#include <limits>
#include "utils/macros/unused.h"
```

Include dependency graph for multmod_simple.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class math::powermod::MultmodSimple

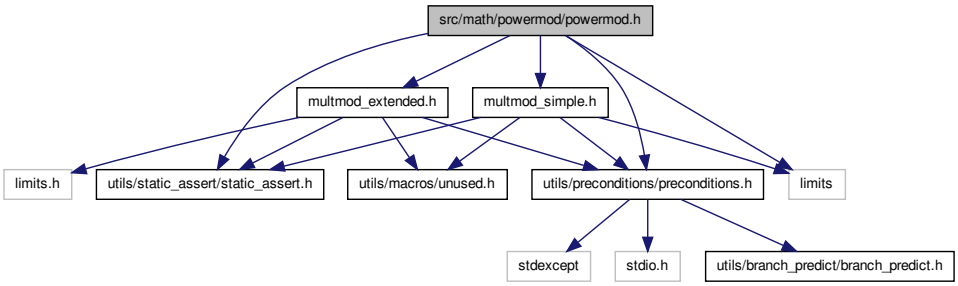
## Namespaces

- namespace math
- namespace math::powermod

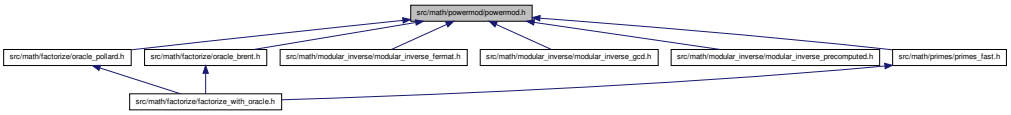
# 7.34 src/math/powermod/powermod.h File Reference

```
#include "utils/static_assert/static_assert.h"
#include "utils/preconditions/preconditions.h"
#include <limits>
#include "multmod_simple.h"
#include "multmod_extended.h"
```

Include dependency graph for powermod.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class math::powermod::Powermod_< MultModImpl >

### Namespaces

- namespace math
- namespace math::powermod

### Typedefs

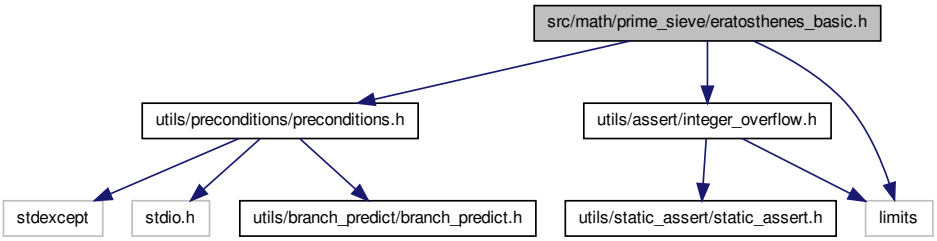
- typedef Powermod_< MultmodSimple > math::powermod::PowermodSimple
- typedef Powermod_< MultmodExtendedOpt > math::powermod::PowermodExtended

## 7.35 src/math/prime_sieve/eratosthenes_basic.h File Reference

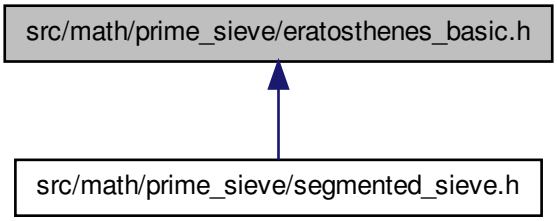
```
#include "utils/preconditions/preconditions.h"
#include "utils/assert/integer_overflow.h"
```

```
#include <limits>
```

Include dependency graph for eratosthenes_basic.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class math::prime_sieve::EratosthenesBasic

### Namespaces

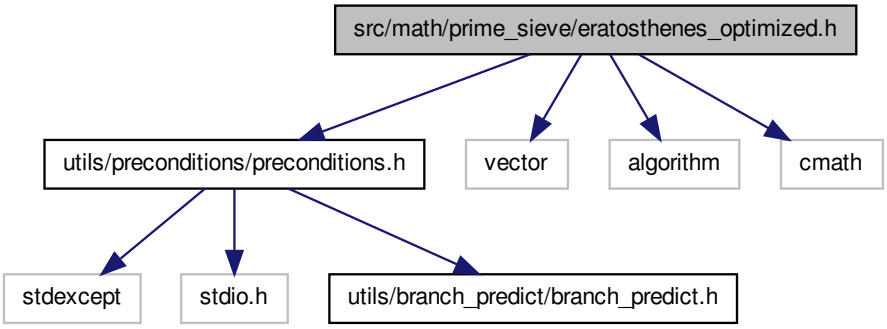
- namespace math
- namespace math::prime_sieve

## 7.36 src/math/prime_sieve/eratosthenes_optimized.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

```
#include <vector>
#include <algorithm>
#include <cmath>
```

Include dependency graph for eratosthenes_optimized.h:




### Classes

- class math::prime_sieve::EratosthenesOptimized

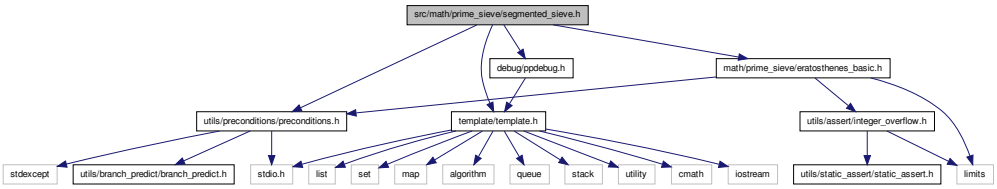
### Namespaces

- namespace math
- namespace math::prime_sieve

## 7.37 src/math/prime_sieve/segmented_sieve.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include "debug/ppdebug.h"
#include "template/template.h"
#include "math/prime_sieve/eratosthenes_basic.h"
```

Include dependency graph for segmented_sieve.h:




### Classes

- class math::prime_sieve::SieveCallback
- class math::prime_sieve::SegmentedSieve

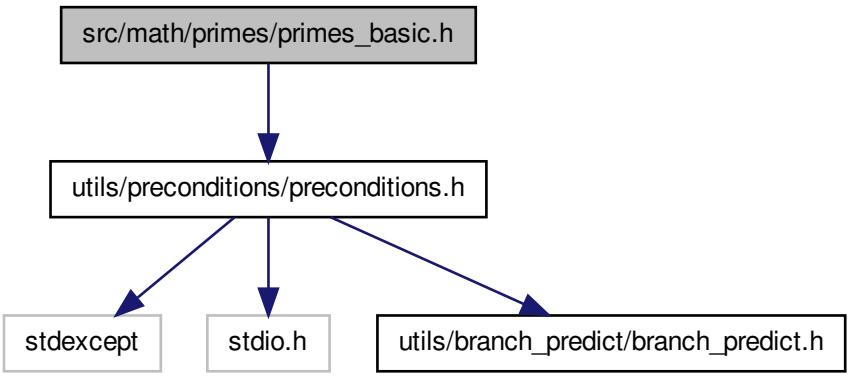
### Namespaces

- namespace math
- namespace math::prime_sieve

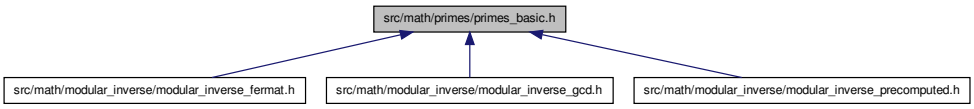
## 7.38 src/math/primes/primes_basic.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for primes_basic.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class math::primes::PrimesBasic

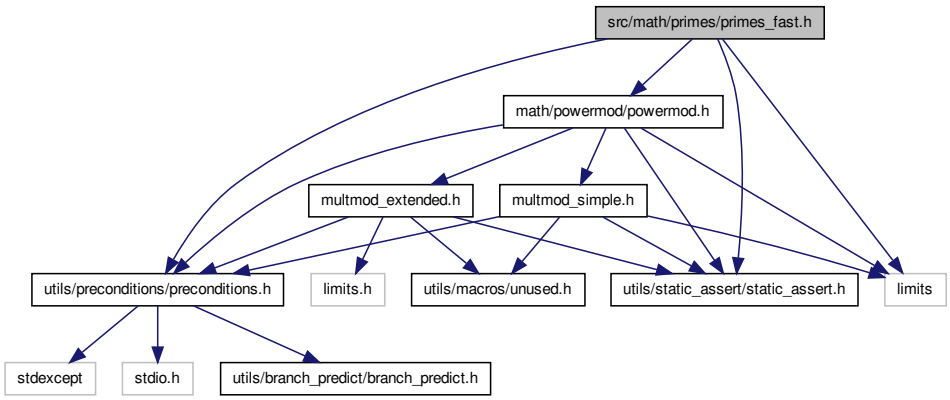
### Namespaces

- namespace math
- namespace math::primes

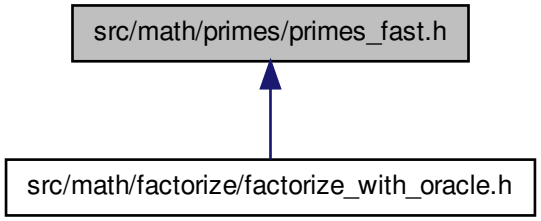
## 7.39 src/math/primes/primes_fast.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include "math/powermod/powermod.h"
#include "utils/static_assert/static_assert.h"
#include <limits>
```

Include dependency graph for primes_fast.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class math::primes::PrimesFast_< PowerModImpl >

### Namespaces

- namespace math
- namespace math::primes

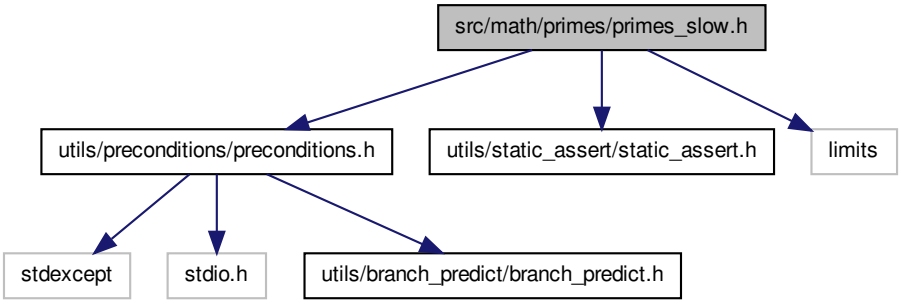
### Typedefs

- typedef PrimesFast_< math::powermod::PowermodExtended > math::primes::PrimesFast

## 7.40 src/math/primes/primes_slow.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include "utils/static_assert/static_assert.h"
#include <limits>
```

Include dependency graph for primes_slow.h:




### Classes

- class math::primes::PrimesSlow

### Namespaces

- namespace math
- namespace math::primes

## 7.41 src/math/primes/primes_test_data.h File Reference

### Namespaces

- namespace testdata

### Variables

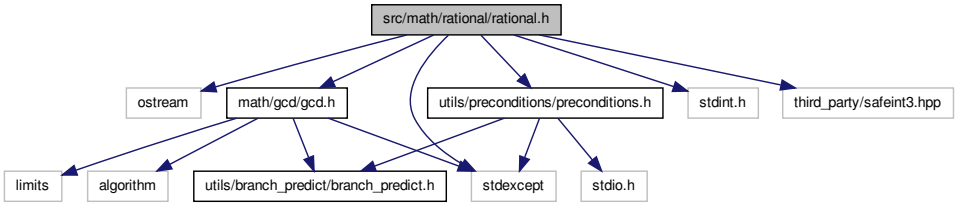
- long long int testdata::prime_twins_count [][2]
- long long int testdata::prime_count_small [][2]
- long long int testdata::prime_count_big [][2]

## 7.42 src/math/rational/rational.h File Reference

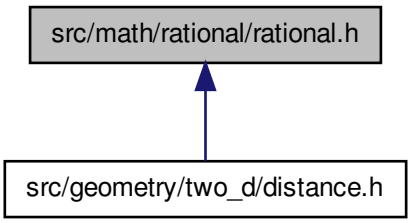
```
#include <ostream>
```

```
#include <stdexcept>
#include <stdint.h>
#include "math/gcd/gcd.h"
#include "utils/preconditions/preconditions.h"
#include "third_party/safeint3.hpp"
```

Include dependency graph for rational.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class math::rational::Rational< T >

## Namespaces

- namespace math
- namespace math::rational

### Defines

- #define NEEDS_INT_DEFINED

### Functions

- template<typename T >
  Rational< T > math::rational::operator- (const Rational< T > &a)
- template<typename T >
  bool math::rational::operator== (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool math::rational::operator< (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool math::rational::operator> (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool math::rational::operator<= (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  bool math::rational::operator>= (const Rational< T > &a, const Rational< T > &b)
- template<typename T >
  std::ostream & math::rational::operator<< (std::ostream &out, const Rational< T > &a)

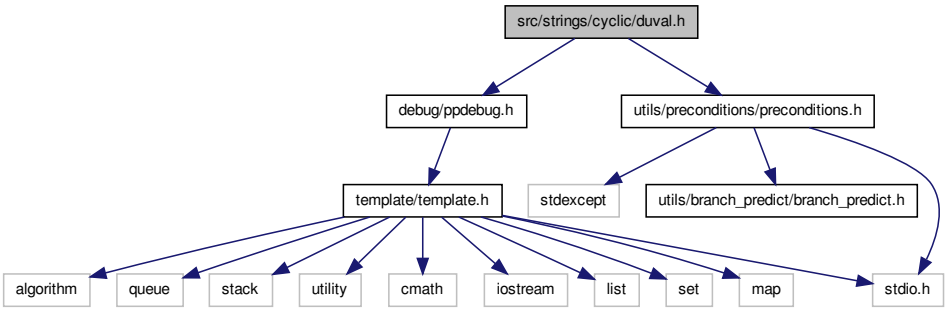
### 7.42.1 Define Documentation

#### 7.42.1.1 #define NEEDS_INT_DEFINED

## 7.43 src/strings/cyclic/duval.h File Reference

```
#include "debug/ppdebug.h"
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for duval.h:




### Classes

- class strings::cyclic::Duval< T >

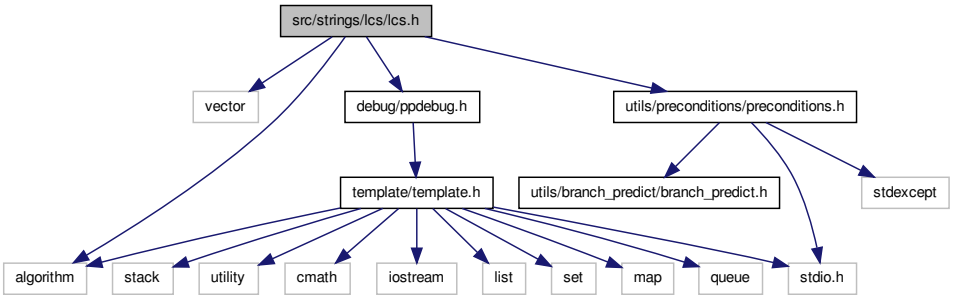
### Namespaces

- namespace strings
- namespace strings::cyclic

## 7.44 src/strings/lcs/lcs.h File Reference

```
#include <vector>
#include <algorithm>
#include "utils/preconditions/preconditions.h"
#include "debug/ppdebug.h"
```

Include dependency graph for lcs.h:




### Classes

- class strings::lcs::LCS< T >

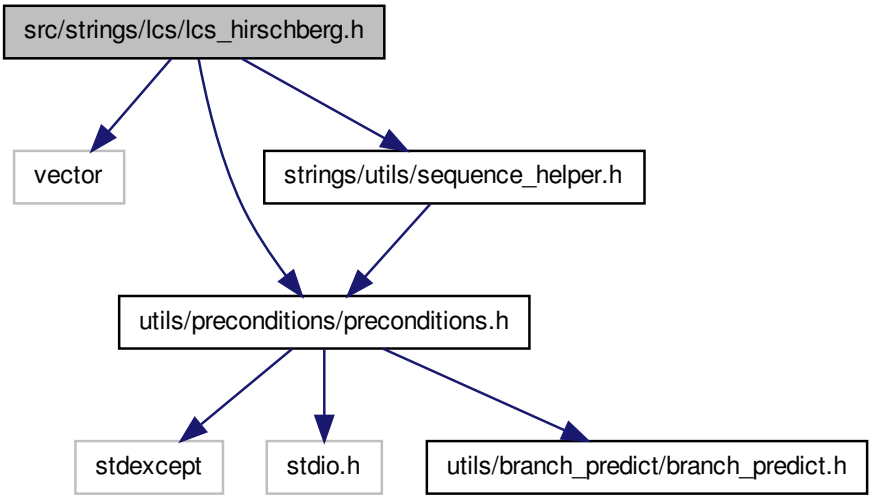
### Namespaces

- namespace strings
- namespace strings::lcs

## 7.45 src/strings/lcs/lcs_hirschberg.h File Reference

```
#include <vector>
#include "utils/preconditions/preconditions.h"
#include "strings/utils/sequence_helper.h"
```

Include dependency graph for lcs_hirschberg.h:




## Classes

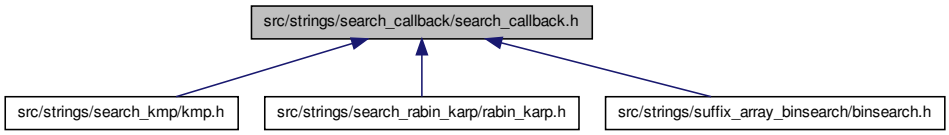
- class strings::lcs::LCSHirschberg< T >

## Namespaces

- namespace strings
- namespace strings::lcs

## 7.46 src/strings/search_callback/search_callback.h File Reference

This graph shows which files directly or indirectly include this file:




### Classes

- class strings::search_callback::SearchCallback< _Iterator >

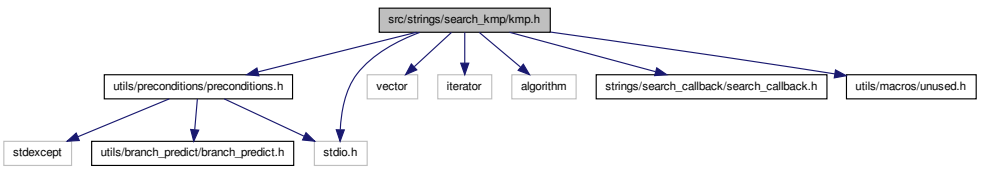
### Namespaces

- namespace strings
- namespace strings::search_callback

## 7.47 src/strings/search_kmp/kmp.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <iterator>
#include <algorithm>
#include "strings/search_callback/search_callback.h"
#include "utils/macros/unused.h"
#include <stdio.h>
```

Include dependency graph for kmp.h:



### Classes

- class strings::search::KMP

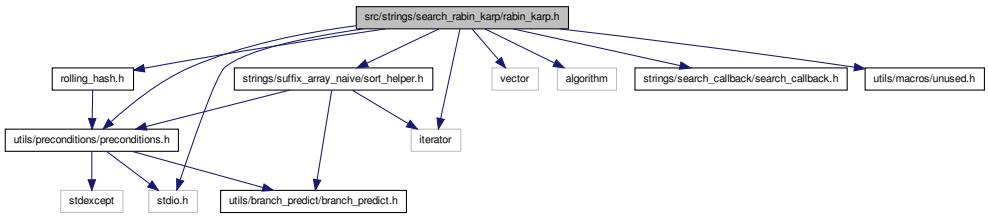
### Namespaces

- namespace strings
- namespace strings::search

## 7.48 src/strings/search_rabin_karp/rabin_karp.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <iterator>
#include <algorithm>
#include "strings/suffix_array_naive/sort_helper.h"
#include "strings/search_callback/search_callback.h"
#include "utils/macros/unused.h"
#include <stdio.h>
#include "rolling_hash.h"
```

Include dependency graph for rabin_karp.h:




### Classes

- class strings::search::RabinKarp

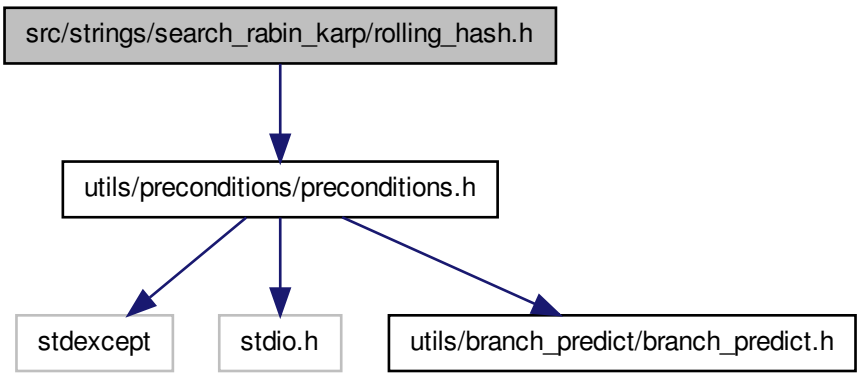
### Namespaces

- namespace strings
- namespace strings::search

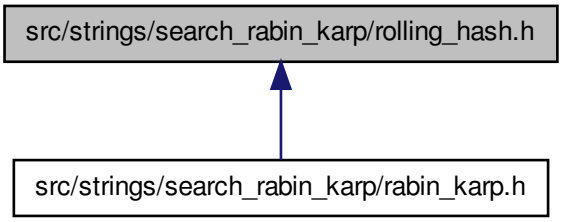
## 7.49 src/strings/search_rabin_karp/rolling_hash.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for rolling_hash.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class strings::search::RollingHash< BaseType >

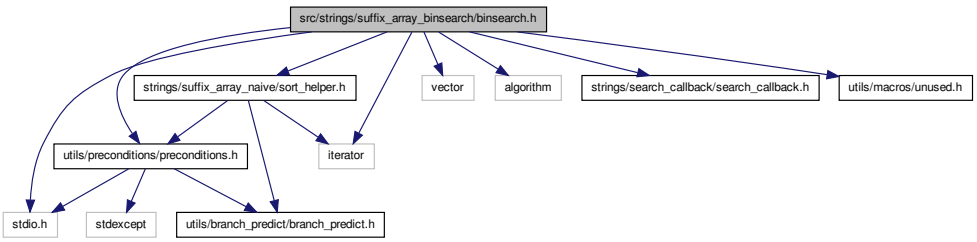
### Namespaces

- namespace strings
- namespace strings::search

## 7.50 src/strings/suffix_array_binsearch/binsearch.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <iterator>
#include <algorithm>
#include "strings/suffix_array_naive/sort_helper.h"
#include "strings/search_callback/search_callback.h"
#include "utils/macros/unused.h"
#include <stdio.h>
```

Include dependency graph for binsearch.h:




### Classes

- class strings::suffix_array::SearchHelper< _Iterator >
- class strings::suffix_array::Binsearch

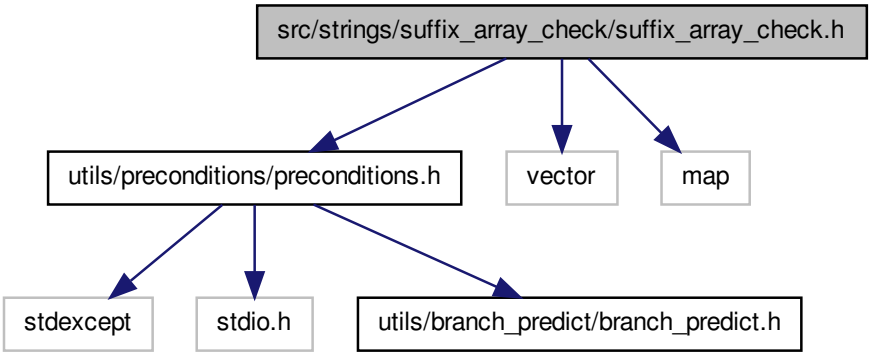
### Namespaces

- namespace strings
- namespace strings::suffix_array

## 7.51 src/strings/suffix_array_check/suffix_array_check.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <map>
```

Include dependency graph for suffix_array_check.h:




### Classes

- class strings::suffix_array::SuffixArrayChecker< T >

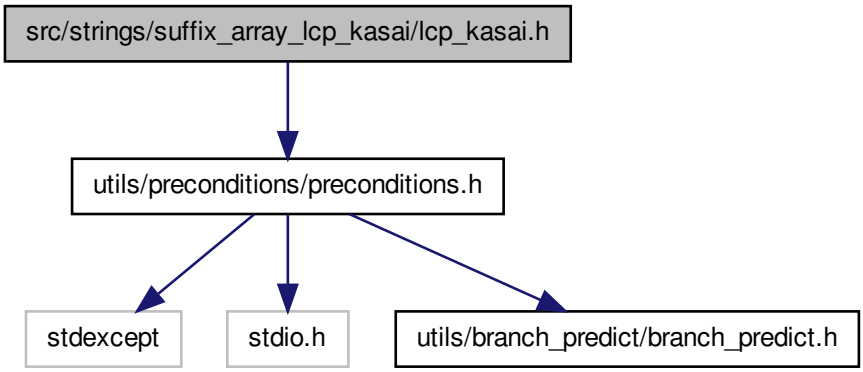
### Namespaces

- namespace strings
- namespace strings::suffix_array

## 7.52 src/strings/suffix_array_lcp_kasai/lcp_kasai.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for lcp_kasai.h:




### Classes

- class strings::suffix_array::LCPKasai

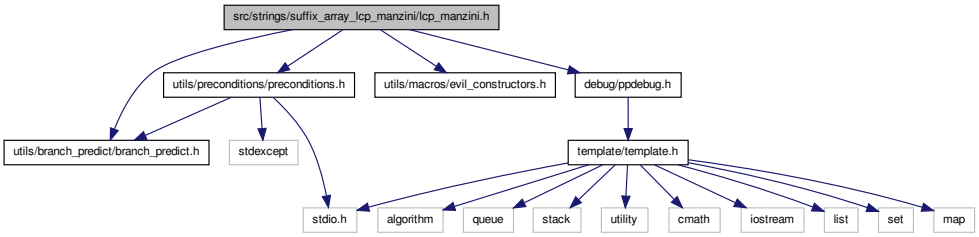
### Namespaces

- namespace strings
- namespace strings::suffix_array

## 7.53 src/strings/suffix_array_lcp_manzini/lcp_manzini.h File Reference

```
#include "utils/branch_predict/branch_predict.h"
#include "utils/preconditions/preconditions.h"
#include "utils/macros/evil_constructors.h"
#include "debug/ppdebug.h"
```

Include dependency graph for lcp_manzini.h:




### Classes

- class strings::suffix_array::LCPManzini

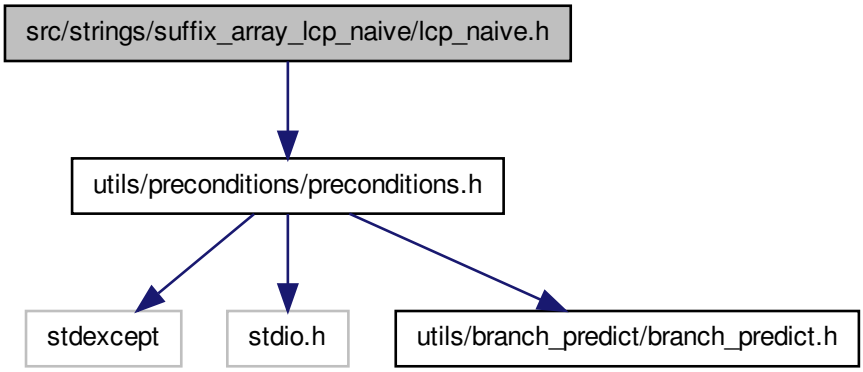
### Namespaces

- namespace strings
- namespace strings::suffix_array

## 7.54 src/strings/suffix_array_lcp_naive/lcp_naive.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for lcp_naive.h:




### Classes

- class strings::suffix_array::LCPNaive

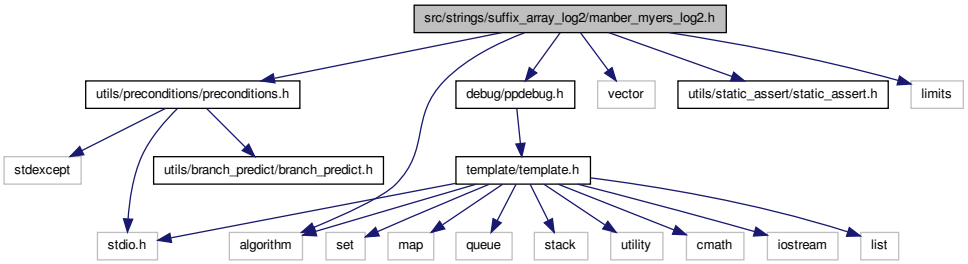
### Namespaces

- namespace strings
- namespace strings::suffix_array

## 7.55 src/strings/suffix_array_log2/manber_myers_log2.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <algorithm>
#include "debug/ppdebug.h"
#include "utils/static_assert/static_assert.h"
#include <limits>
```

Include dependency graph for manber_myers_log2.h:




## Classes

- class strings::suffix_array::ManberMyersLog2_< IndexType >
- struct strings::suffix_array::ManberMyersLog2_< IndexType >::Suffix

## Namespaces

- namespace strings
- namespace strings::suffix_array

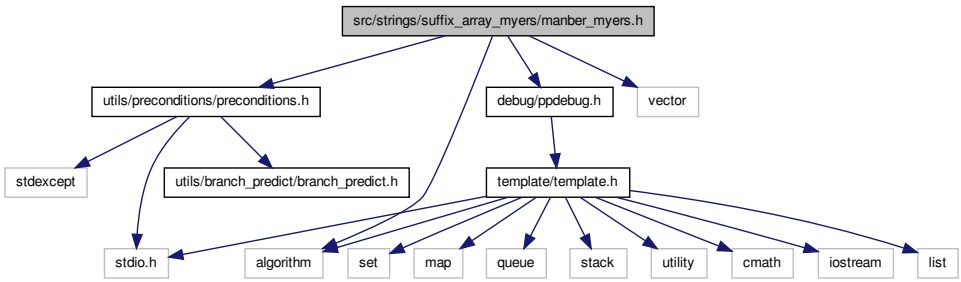
## Typedefs

- typedef ManberMyersLog2_< int > strings::suffix_array::ManberMyersLog2

# 7.56 src/strings/suffix_array_myers/manber_myers.h File Reference

```
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <algorithm>
#include "debug/ppdebug.h"
```

Include dependency graph for manber_myers.h:




## Classes

- class strings::suffix_array::ManberMyers

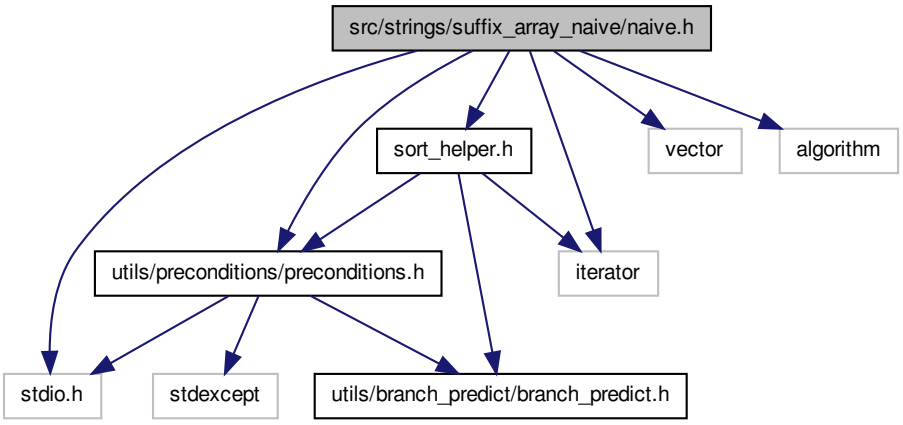
## Namespaces

- namespace strings
- namespace strings::suffix_array

# 7.57 src/strings/suffix_array_naive/naive.h File Reference

```
#include <stdio.h>
#include "utils/preconditions/preconditions.h"
#include <vector>
#include <iterator>
#include <algorithm>
#include "sort_helper.h"
```

Include dependency graph for naive.h:




### Classes

- class strings::suffix_array::NaiveSuffixArray

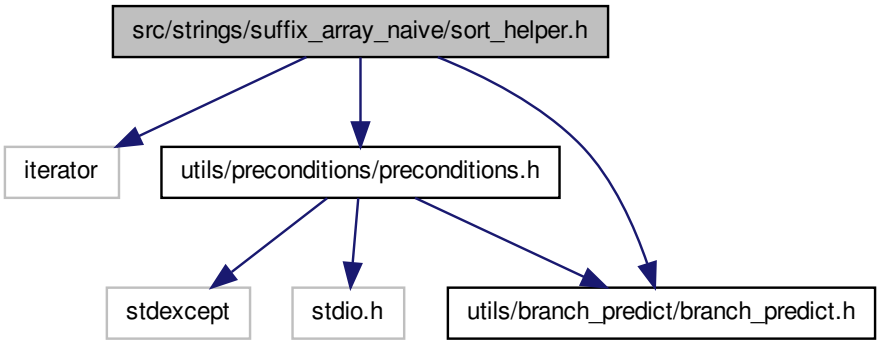
### Namespaces

- namespace strings
- namespace strings::suffix_array

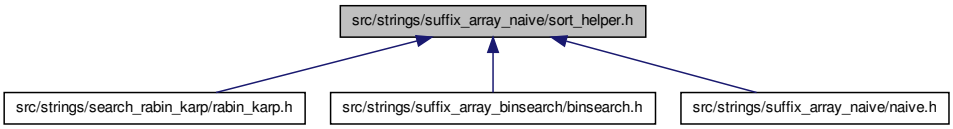
## 7.58 src/strings/suffix_array_naive/sort_helper.h File Reference

```
#include <iterator>
#include "utils/preconditions/preconditions.h"
#include "utils/branch_predict/branch_predict.h"
```

Include dependency graph for sort_helper.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class strings::suffix_array::SortHelper< _Iterator >

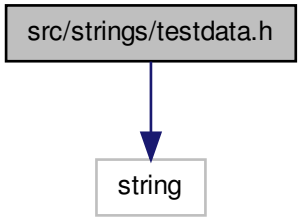
### Namespaces

- namespace strings
- namespace strings::suffix_array

## 7.59 src/strings/testdata.h File Reference

```
#include <string>
```

Include dependency graph for testdata.h:




### Classes

- class strings::TestdataFiles
- class strings::PatternFiles

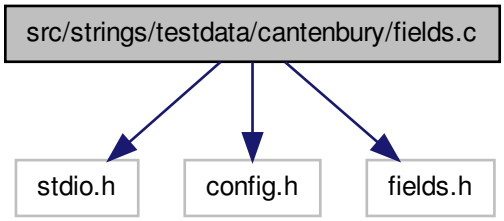
### Namespaces

- namespace strings

## 7.60 src/strings/testdata/cantenbury/fields.c File Reference

```
#include <stdio.h>
#include "config.h"
#include "fields.h"
```

Include dependency graph for fields.c:




## Defines

- #define strchr index

## Functions

- field_t *fieldread P ((FILE *file, char *delims, int flags, int maxf))
- field_t *fieldmake P ((char *line, int allocated, char *delims, int flags, int maxf))
- static field_t *fieldparse P ((field_t *fieldp, char *line, char *delims, int flags, int maxf))
- static int fieldbackch P ((char *str, char **out, int strip))
- int fieldwrite P ((FILE *file, field_t *fieldp, int delim))
- void fieldfree P ((field_t *fieldp))
- void free ()
- char * malloc ()
- char * realloc ()
- char * strchr ()
- int strlen ()
- field_t * fieldread (FILE *file, char *delims, int flags, int maxf)
- field_t * fieldmake (char *line, int allocated, char *delims, int flags, int maxf)
- static field_t * fieldparse (field_t *fieldp, char *line, char *delims, int flags, int maxf)
- static int fieldbackch (char *str, char **out, int strip)
- int fieldwrite (FILE *file, field_t *fieldp, int delim)
- void fieldfree (field_t *fieldp)

## Variables

- static char Rcs_Id [] = "$Id: fields.c,v 1.7 1994/01/06 05:26:37 geoff Exp $"
- unsigned int field_field_inc = 20
- unsigned int field_line_inc = 512

### 7.60.1 Define Documentation

#### 7.60.1.1 #define strchr index

### 7.60.2 Function Documentation

#### 7.60.2.1 static int fieldbackch ( char * *str,* char ** *out,* int *strip* ) [static]

#### 7.60.2.2 void fieldfree ( field_t * *fieldp* )

#### 7.60.2.3 field_t* fieldmake ( char * *line,* int *allocated,* char * *delims,* int *flags,* int *maxf* )

#### 7.60.2.4 static field_t* fieldparse ( field_t * *fieldp,* char * *line,* char * *delims,* int *flags,* int *maxf* ) [static]

#### 7.60.2.5 field_t* fieldread ( FILE * *file,* char * *delims,* int *flags,* int *maxf* )

#### 7.60.2.6 int fieldwrite ( FILE * *file,* field_t * *fieldp,* int *delim* )

#### 7.60.2.7 void free ( )

#### 7.60.2.8 char* malloc ( )

#### 7.60.2.9 field_t* fieldread P ( (FILE *file, char *delims, int flags, int maxf) )

#### 7.60.2.10 field_t* fieldmake P ( (char *line, int allocated, char *delims, int flags, int maxf) )

#### 7.60.2.11 int fieldwrite P ( (FILE *file, field_t *fieldp, int delim) )

#### 7.60.2.12 static int fieldbackch P ( (char *str, char **out, int strip) ) [static]

#### 7.60.2.13 static field_t* fieldparse P ( (field_t *fieldp, char *line, char *delims, int flags, int maxf) ) [static]

#### 7.60.2.14 void fieldfree P ( (field_t *fieldp) )

#### 7.60.2.15 char* realloc ( )

#### 7.60.2.16 char* strchr ( )

#### 7.60.2.17 int strlen ( )

### 7.60.3 Variable Documentation

#### 7.60.3.1 unsigned int field_field_inc = 20

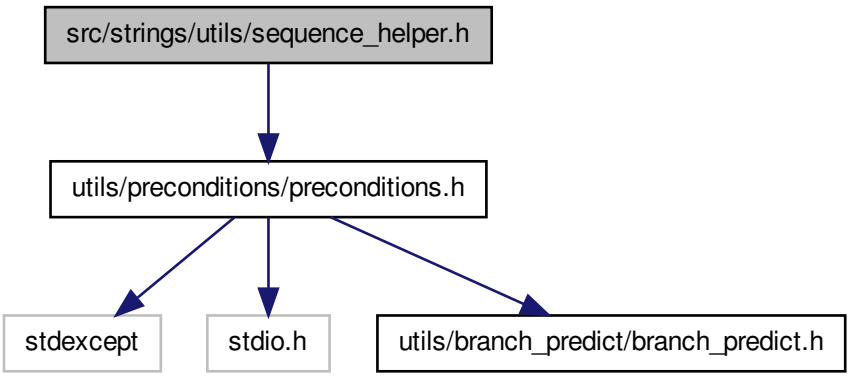
#### 7.60.3.2 unsigned int field_line_inc = 512

#### 7.60.3.3 char Rcs_Id[ ] = "$Id: fields.c,v 1.7 1994/01/06 05:26:37 geoff Exp $" [static]

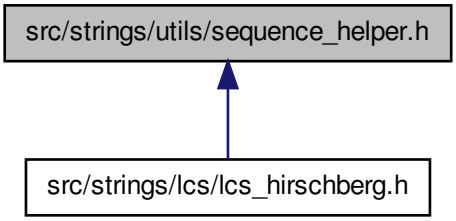
## 7.61 src/strings/utils/sequence_helper.h File Reference

```
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for sequence_helper.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class strings::utils::SequenceHelper< T >

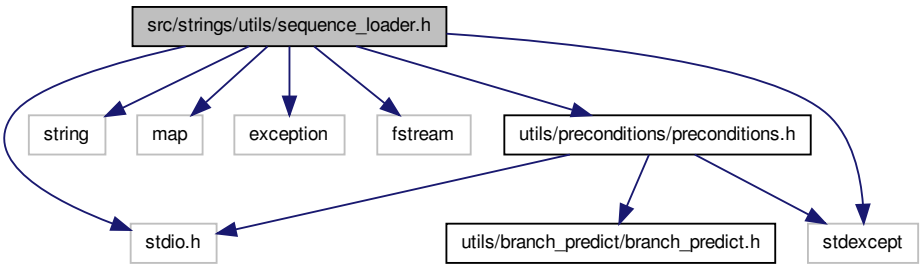
### Namespaces

- namespace strings
- namespace strings::utils

## 7.62 src/strings/utils/sequence_loader.h File Reference

```
#include <stdio.h>
#include <string>
#include <map>
#include <exception>
#include <fstream>
#include <stdexcept>
#include "utils/preconditions/preconditions.h"
```

Include dependency graph for sequence_loader.h:




### Classes

- class strings::utils::SequenceLoader

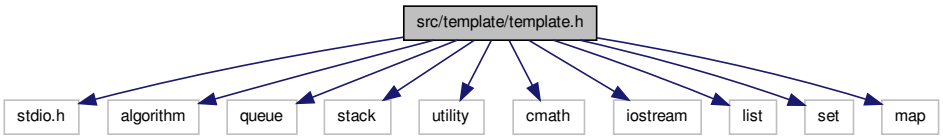
### Namespaces

- namespace strings
- namespace strings::utils

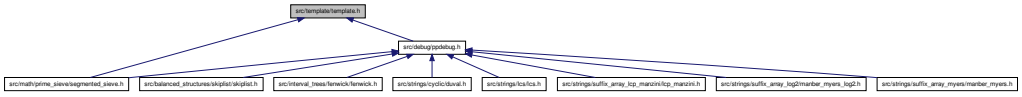
## 7.63 src/template/template.h File Reference

```
#include <stdio.h>
#include <algorithm>
#include <queue>
#include <stack>
#include <utility>
#include <cmath>
#include <iostream>
#include <list>
#include <set>
#include <map>
```

Include dependency graph for template.h:




This graph shows which files directly or indirectly include this file:



## Defines

- #define FOR(q, n) for(int q = 0; q < (int) n; ++q)
- #define FOREACH(it, container)
- #define fi first
- #define se second
- #define mp make_pair
- #define pb push_back

## Typedefs

- typedef long long int ll
- typedef long double ld
- typedef pair< int, int > PII

### 7.63.1 Define Documentation

#### 7.63.1.1 #define fi first

#### 7.63.1.2 #define FOR( *q, n* ) for(int q = 0; q < (int) n; ++q)

#### 7.63.1.3 #define FOREACH( *it, container* )

**Value:**

```
for( \
        __typeof(container.begin()) it = container.begin(); \
        it != container.end(); ++it)
```

#### 7.63.1.4 #define mp make_pair

#### 7.63.1.5 #define pb push_back

#### 7.63.1.6 #define se second

### 7.63.2 Typedef Documentation

#### 7.63.2.1 typedef long double ld

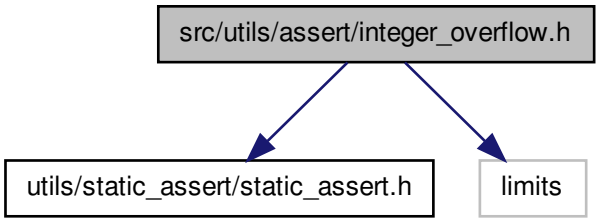
#### 7.63.2.2 typedef long long int ll

#### 7.63.2.3 typedef pair<int,int> pii

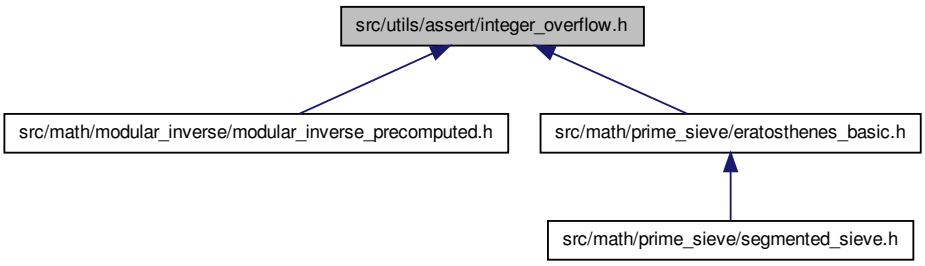
## 7.64 src/utils/assert/integer_overflow.h File Reference

```
#include "utils/static_assert/static_assert.h"
#include <limits>
```

Include dependency graph for integer_overflow.h:



This graph shows which files directly or indirectly include this file:




## Functions

- template<typename SuppliedType , typename RequiredType >
  void STATIC_ASSERT_CHECK_INTEGER_OVERFLOW ()

### 7.64.1 Function Documentation

#### 7.64.1.1 template<typename SuppliedType , typename RequiredType > void STATIC_ASSERT_CHECK_INTEGER_OVERFLOW ( ) [inline]

This template is used to ensure (at compile type) that user can't pass integral value of bigger type than specified. This is very useful to remind users about possible overflows. Note that this template is also checking unsigned->signed

Note: this template is quite aggresive, you may want to consider using run-time checks (especially signed constants versus unsigned expected)

Basic usage is:

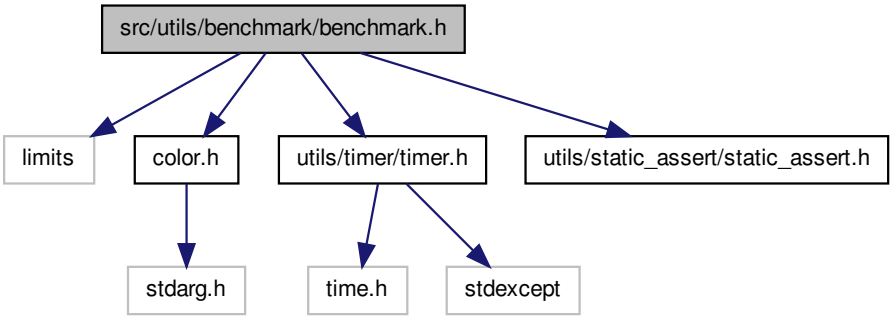
template <typename t>=""> userFunction(T arg1) { STATIC_ASSERT_CHECK_INTEGER_-OVERFLOW<T, short>(); short tmp = (short) arg1; }

and this code won't compile for example for T=32bit int (and short = 16bit int)

## 7.65 src/utils/benchmark/benchmark.h File Reference

```
#include <limits>
#include "color.h"
#include "utils/timer/timer.h"
#include "utils/static_assert/static_assert.h"
```

Include dependency graph for benchmark.h:




## Namespaces

- namespace utils
- namespace utils::benchmark

## Defines

- #define BENCHMARK(times, code)
- #define AUTO_BENCHMARK(code)

## Functions

- void utils::benchmark::printBenchmarkResults (long long int times, double run_-time_sec, const char *function_str)

## Variables

- const double utils::benchmark::MIN_BENCHMARK_TIME = 1.5

### 7.65.1 Define Documentation

#### 7.65.1.1 #define AUTO_BENCHMARK( *code* )

**Value:**

```
{ \
    ::utils::timer::Timer timer = ::utils::timer::Timer(); \
    long long int times = 0; \
    int i = 0; \
    while ((timer.elapsed_time_sec() < ::utils::benchmark::MIN_BENCHMARK_TIME) &&
        \
        (i < std::numeric_limits<long long int>::digits)) { \
      long long int t = 1LL << i; \
        for (long long q__ = 0; q__ < t; q__++) { \
            code; \
        } \
      times += t; \
      i++; \
    } \
    ::utils::benchmark::printBenchmarkResults(\
        times, timer.elapsed_time_sec(), #code); \
  }
```

Macro for benchmarking the code The benchmark runs the code several times depenging on the speed of execution and prints nicely formatted report to stdout. Usage:

```
AUTO_BENCHMARK(my_function());
```

calls the function my_function for approximately *MIN_BENCHMARK_TIME* seconds.

Note: This macro uses exponential decay to guess the correct number of iterations, and so that instrumentation is not big.

**Warning**

Sometimes compiler optimizes out parts of you code!

**Parameters**

| | |
|---|---|
| *code* | code that should be run |

#### 7.65.1.2 #define BENCHMARK( *times, code* )

**Value:**

```
{ \
    STATIC_ASSERT(std::numeric_limits<typeof(times)>::is_integer, "") \
    ::utils::timer::Timer timer = ::utils::timer::Timer(); \
    for (typeof(times) q__= 0; q__ < times; q__++) { \
        code;\
    } \
    ::utils::benchmark::printBenchmarkResults(\
        times, timer.elapsed_time_sec(), #code); \
  }
```

Macro for benchmarking code.

The benchmark runs specified *times* the code and prints nicely formatted report to stdout. Usage: `BENCHMARK(100, my_function());` calls 100 times function my_function.

Note: "function" does not need to be the function, in fact, it may be sequence of statements of event the {} block.

**Warning**

Sometimes compiler optimizes out parts of you code!

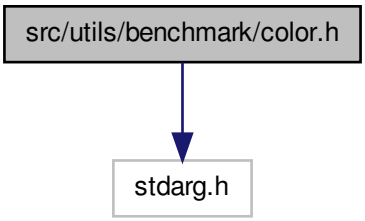
**Parameters**

| | |
|---|---|
| *times* | number of times the *code* should be run |
| *code* | code that should be run |

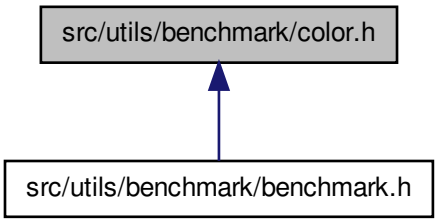
## 7.66 src/utils/benchmark/color.h File Reference

```
#include <stdarg.h>
```

Include dependency graph for color.h:



This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace color

### Enumerations

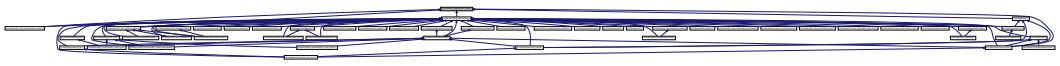
- enum color::Color { color::BLUE = 34, color::PINK = 35, color::CYAN = 36 }

### Functions

- void color::colorPrintf (Color color, const char *fmt,...)

## 7.67 src/utils/branch_predict/branch_predict.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define LIKELY(x) (x)
- #define UNLIKELY(x) (x)

### 7.67.1 Define Documentation

#### 7.67.1.1 #define LIKELY( *x* ) (x)

#### 7.67.1.2 #define UNLIKELY( *x* ) (x)

## 7.68 src/utils/macros/array_size.h File Reference

### Defines

- #define ARRAY_SIZE(array) (sizeof(ArraySizeHelper(array)))

### Functions

- template<typename T , unsigned int N>
  char(& ArraySizeHelper (T(&array)[N]))[N]
- template<typename T , unsigned int N>
  char(& ArraySizeHelper (const T(&array)[N]))[N]

### 7.68.1 Define Documentation

#### 7.68.1.1 #define ARRAY_SIZE( *array* ) (sizeof(ArraySizeHelper(array)))

Macro which can determine size of the array.

Basic usage: int a[5]; cout >> ARRAY_SIZE(a); // 5

int funct(int b*) { cout >> ARRAY_SIZE(b); // compilation error

### 7.68.2 Function Documentation

#### 7.68.2.1 template<typename T , unsigned int N> char(& ArraySizeHelper ( T(&) *array[N]* ))[N]

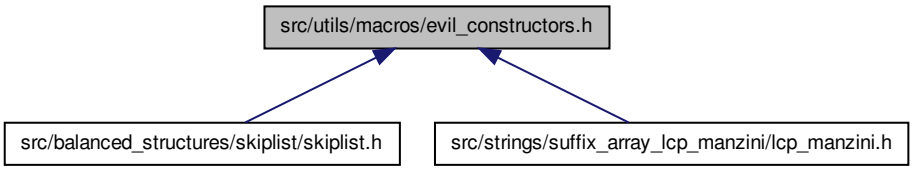
This file contains macro which can determine size of an array

Macro is taken from chrome's basictypes.h. ARRAY_SIZE relies on template matching failure if the argument is not an array.

#### 7.68.2.2 template<typename T , unsigned int N> char(& ArraySizeHelper ( const T(&) *array[N]* ))[N]

## 7.69 src/utils/macros/evil_constructors.h File Reference

This graph shows which files directly or indirectly include this file:




### Defines

- #define DISALLOW_EVIL_CONSTRUCTORS(type)

### 7.69.1 Define Documentation

#### 7.69.1.1 #define DISALLOW_EVIL_CONSTRUCTORS( *type* )

**Value:**

```
void operator=(type const &); \
  type(type const &);
```

This macro disallows "evil" constructors, i.e. defines constructors that are created implicitly and should not be. This is very useful if you are not intending to class be a copiable.

Warning: This macro will not define the constructors as private, you must put it into the right section! Note: not including in the private section will result to linking problems if the constructors are used.

correct usage: class MyClass { private: DISALLOW_EVIL_CONSTRUCTORS(MyClass) }

## 7.70 src/utils/macros/foreach.h File Reference

### Defines

- #define FOREACH(it, container)

### 7.70.1 Define Documentation

#### 7.70.1.1 #define FOREACH( *it, container* )

**Value:**

```
for( \
        __typeof(container.begin()) it = container.begin(); \
        it != container.end(); ++it)
```

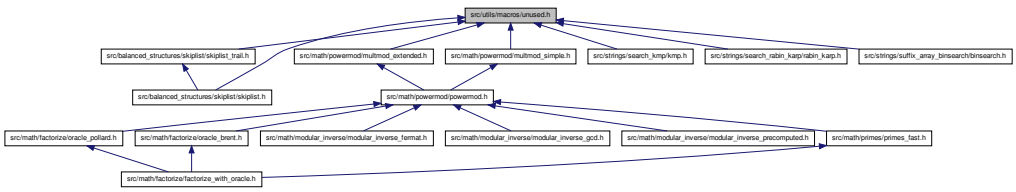
foreach macro helps with enumerating stl containers. It iterates with newly-created iterator variable from begin of the container to the end.

Warning: FOREACH macro does not cache the container itself. This means, that calling FOREACH(it, functionReturningContainer()) will have unexpected behaviour!

Example usage: void annotate(const vector<string>& lines) { FOREACH(it, lines) { // it is of type vector<string>::iterator cout << "annotated line:" << *it; } }

## 7.71 src/utils/macros/unused.h File Reference

This graph shows which files directly or indirectly include this file:




### Defines

- #define UNUSED(x) x __attribute__((unused))

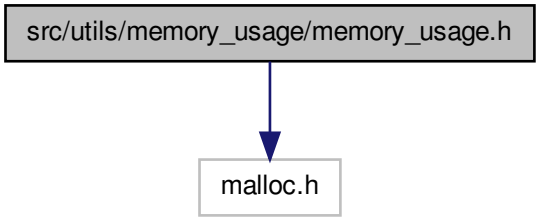
### 7.71.1 Define Documentation

#### 7.71.1.1 #define UNUSED( *x* ) x __attribute__((unused))

## 7.72 src/utils/memory_usage/memory_usage.h File Reference

```
#include <malloc.h>
```

Include dependency graph for memory_usage.h:




### Namespaces

- namespace utils
- namespace utils::memory_usage

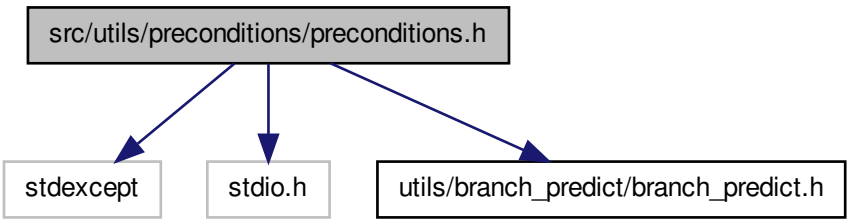
### Functions

- int utils::memory_usage::getUsedMemoryKb ()

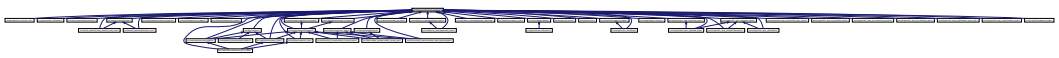
## 7.73 src/utils/preconditions/preconditions.h File Reference

```
#include <stdexcept>
#include <stdio.h>
#include "utils/branch_predict/branch_predict.h"
```

Include dependency graph for preconditions.h:




This graph shows which files directly or indirectly include this file:



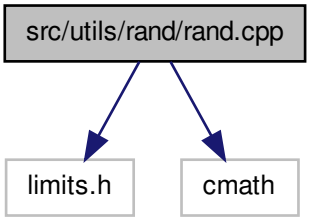
### Classes

- class Preconditions

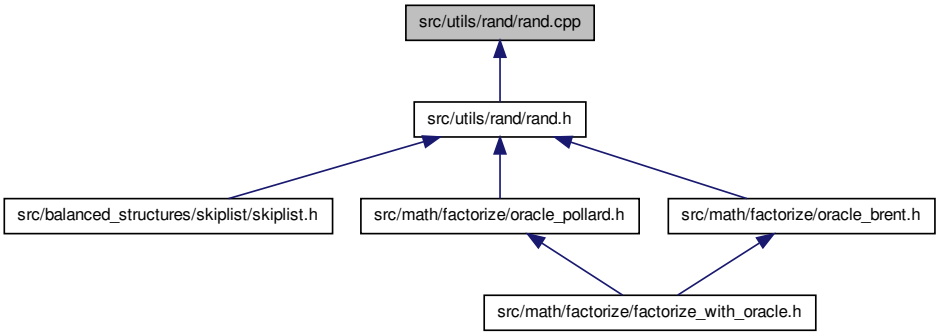
## 7.74 src/utils/rand/rand.cpp File Reference

```
#include <limits.h>
#include <cmath>
```

Include dependency graph for rand.cpp:




This graph shows which files directly or indirectly include this file:




### Variables

- const unsigned int RandMax = (1u<<31) - 2

### 7.74.1 Variable Documentation

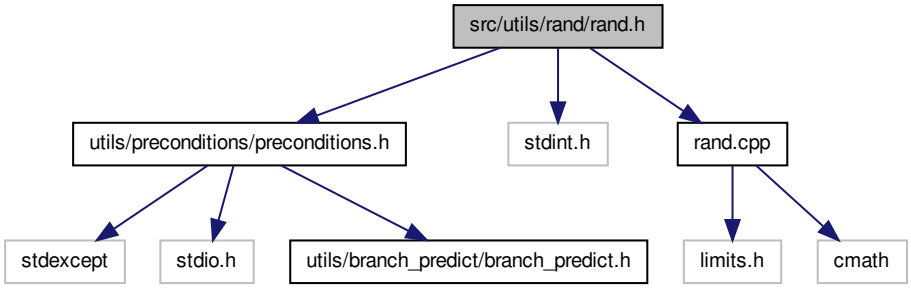
#### 7.74.1.1 const unsigned int RandMax = (1u<<31) - 2

## 7.75 src/utils/rand/rand.h File Reference

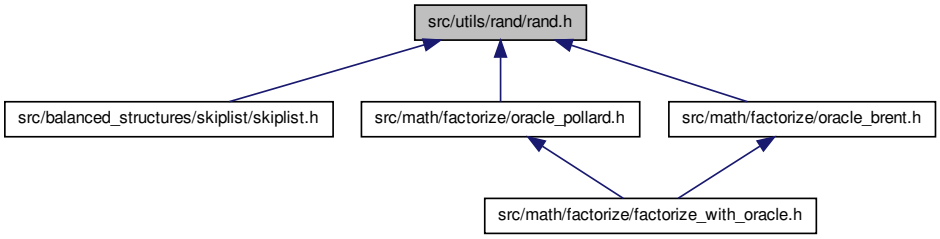
```
#include "utils/preconditions/preconditions.h"
```

```
#include <stdint.h>
#include "rand.cpp"
```

Include dependency graph for rand.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class Rand

## 7.76 src/utils/si_units/si_units.h File Reference

### Variables

- const long long Ki = 1000
- const long long Mi = 1000 ∗ Ki

- const long long Gi = 1000 ∗ Mi

### 7.76.1 Variable Documentation

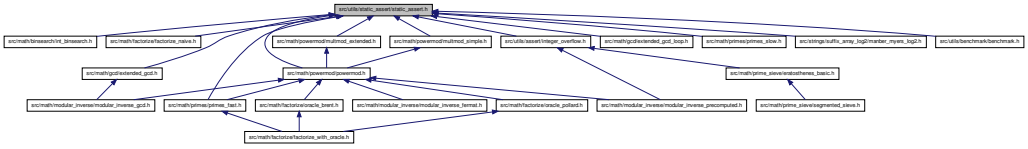
#### 7.76.1.1 const long long Gi = 1000 ∗ Mi

#### 7.76.1.2 const long long Ki = 1000

#### 7.76.1.3 const long long Mi = 1000 ∗ Ki

## 7.77 src/utils/static_assert/static_assert.h File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- struct utils::static_assert_::STATIC_ASSERTION_FAILURE< true >
- struct utils::static_assert_::static_assert_test< x >

### Namespaces

- namespace utils
- namespace utils::static_assert_

### Defines

- #define __JOIN(X, Y) __JOIN2(X, Y)
- #define __JOIN2(X, Y) X##Y
- #define STATIC_ASSERT(B, MSG)

### 7.77.1 Define Documentation

#### 7.77.1.1 #define __JOIN( *X, Y* ) __JOIN2(X, Y)

#### 7.77.1.2 #define __JOIN2( *X, Y* ) X##Y

#### 7.77.1.3 #define STATIC_ASSERT( *B, MSG* )

**Value:**

```
typedef ::utils::static_assert_::static_assert_test<\
      sizeof(::utils::static_assert_::STATIC_ASSERTION_FAILURE<(bool) (B)>)> \
      __JOIN(static_assert_on_line_, __LINE__);
```

Static assertion. You can assert on any compile-time expressions. Note: if you try to pass non-compile-time expression, compilation will be aborted

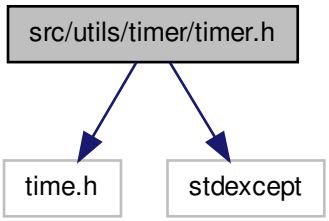
**Parameters**

| | |
|---|---|
| *B* | boolean expression |
| *MSG* | (ignored) message - it is for compatibility with C++0x static_assert |

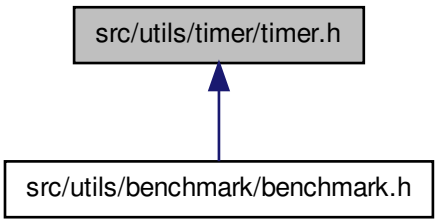
## 7.78 src/utils/timer/timer.h File Reference

```
#include <time.h>
#include <stdexcept>
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class utils::timer::Timer

## Namespaces

- namespace utils
- namespace utils::timer

# Index