

Algorithms and Data Structures for Mathematicians

Lecture 2: Divide and Conquer

Peter Kostolányi

`kostolanyi at fmph and so on`

Room M-258

5 October 2017

Recapitulation

Some topics addressed in the first lecture:

- ▶ Algorithms and their description in pseudocode
- ▶ Worst-case time complexity analysis
- ▶ Asymptotic notation
- ▶ Sorting in $\Theta(n^2)$ worst-case (Insertion sort)

Questions that we have not dealt with so far:

- ▶ Are there any techniques for designing **efficient** algorithms?
- ▶ Can we do better than $\Theta(n^2)$ for sorting (in worst-case)?

Divide and Conquer

A paradigm for algorithm design:

Divide the problem into several **smaller** instances of the same problem

Conquer the subproblems by solving them recursively or directly if they are small enough

Combine the solutions of the subproblems to obtain a solution of the original problem

Merge Sort

- ▶ Let (S, \preceq) be a totally ordered set
- ▶ Given an array of n elements of S , we wish to sort them in increasing order

The Divide and Conquer Approach:

Divide the array into two (approximate) halves

Conquer the subproblems by sorting the respective subarrays

- ▶ If a subarray contains only one element, it is already sorted
- ▶ Otherwise divide once again (recursion)

Combine the two sorted subarrays by **merging** them into a single sorted array

Merge Procedure

- ▶ Let (S, \preceq) be a totally ordered set
- ▶ Let $\top \succ x$ for all x in S

MERGE(a, f, m, l):

Input : Integer $n \geq 1$; Array $a = \langle a[1], \dots, a[n] \rangle$ of elements of S ;
Integers f, m, l such that $1 \leq f \leq m < l \leq n$ and such that
 $\langle a[f], \dots, a[m] \rangle$ and $\langle a[m+1], \dots, a[l] \rangle$ are already sorted

Behaviour: Merges $\langle a[f], \dots, a[m] \rangle$ and $\langle a[m+1], \dots, a[l] \rangle$ into a sorted subarray

$i \leftarrow f; j \leftarrow m + 1; k \leftarrow 0;$

while $i \leq m$ **or** $j \leq l$ **do**

$k \leftarrow k + 1;$

if $i \leq m$ **then** $r \leftarrow a[i]$ **else** $r \leftarrow \top;$

if $j \leq l$ **then** $s \leftarrow a[j]$ **else** $s \leftarrow \top;$

if $r \preceq s$ **then** $c[k] \leftarrow r; i \leftarrow i + 1$ **else** $c[k] \leftarrow s; j \leftarrow j + 1;$

end

for $i \leftarrow f$ **to** l **do** $a[i] \leftarrow c[i - f + 1];$

- ▶ Time complexity: $\Theta(l - f)$

Merge Sort

MERGESORT(a, f, l):

Input : Integer $n \geq 1$; Array $a = \langle a[1], \dots, a[n] \rangle$ of elements of S ;
Integers f, l such that $1 \leq f \leq l \leq n$

Behaviour: Sorts the subarray $\langle a[f], \dots, a[l] \rangle$ in increasing order

if $f = l$ **then**

return

else

$m \leftarrow \lfloor (f + l)/2 \rfloor$;

MERGESORT(a, f, m);

MERGESORT($a, m + 1, l$);

MERGE(a, f, m, l);

end

- ▶ Time complexity $T(n)$ of **MERGESORT**($a, 1, n$)?
- ▶ Surely $T(1) = \Theta(1)$
- ▶ We have $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n)$ for $n \geq 2$
- ▶ We need to find a solution to the asymptotic recurrence above

Asymptotic Recurrences

We need a technique for solving recurrences of the form

$$\begin{aligned}T(n) &= \sum_{i=1}^a T(\varphi_i(n/b)) + \Theta(n^d) && \text{for } n \geq b, \\T(s) &= \Theta(1) && \text{for } s = 1, \dots, b-1,\end{aligned}$$

where:

- ▶ $a \geq 1$ is in \mathbb{N} , $b \geq 2$ is in \mathbb{N} , and $d > 0$ is in \mathbb{R}
- ▶ $\varphi_i(x) = \lfloor x \rfloor$ or $\varphi_i(x) = \lceil x \rceil$ for $i = 1, \dots, a$ and all x in \mathbb{R}

It is possible to prove that the (asymptotic) solution does not depend on the particular rounding functions φ_i

We shall thus simply write:

$$T(n) = aT(n/b) + \Theta(n^d)$$

- ▶ We shall find a solution assuming that $n = b^k$, where k is in \mathbb{N}
- ▶ It is possible to extend the analysis to deal with non-exact-powers of b as well

Asymptotic Recurrences

Theorem

Let $T(n)$ be a function satisfying

$$\begin{aligned} T(n) &\leq aT(n/b) + cn^d && \text{for } n = b^k, k = 1, 2, 3, \dots, \\ T(1) &= \Theta(1), \end{aligned}$$

where $a \geq 1$ and $b \geq 2$ are in \mathbb{N} , and $c, d > 0$ are in \mathbb{R} . Then

$$T(n) = \left\{ \begin{array}{ll} O(n^{\log_b a}) & \text{if } d < \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^d) & \text{if } d > \log_b a \end{array} \right\} \text{ for } n = b^k, k = 1, 2, 3, \dots$$

Asymptotic Recurrences

Theorem

Let $T(n)$ be a function satisfying

$$\begin{aligned} T(n) &\geq aT(n/b) + cn^d && \text{for } n = b^k, k = 1, 2, 3, \dots, \\ T(1) &= \Theta(1), \end{aligned}$$

where $a \geq 1$ and $b \geq 2$ are in \mathbb{N} , and $c, d > 0$ are in \mathbb{R} . Then

$$T(n) = \left\{ \begin{array}{ll} \Omega(n^{\log_b a}) & \text{if } d < \log_b a \\ \Omega(n^d \log n) & \text{if } d = \log_b a \\ \Omega(n^d) & \text{if } d > \log_b a \end{array} \right\} \text{ for } n = b^k, k = 1, 2, 3, \dots$$

Asymptotic Recurrences

Corollary

Let $T(n)$ be a function satisfying

$$\begin{aligned}T(n) &= aT(n/b) + \Theta(n^d) && \text{for } n = b^k, k = 1, 2, 3, \dots, \\T(1) &= \Theta(1),\end{aligned}$$

where $a \geq 1$ and $b \geq 2$ are in \mathbb{N} , and $d > 0$ is in \mathbb{R} . Then

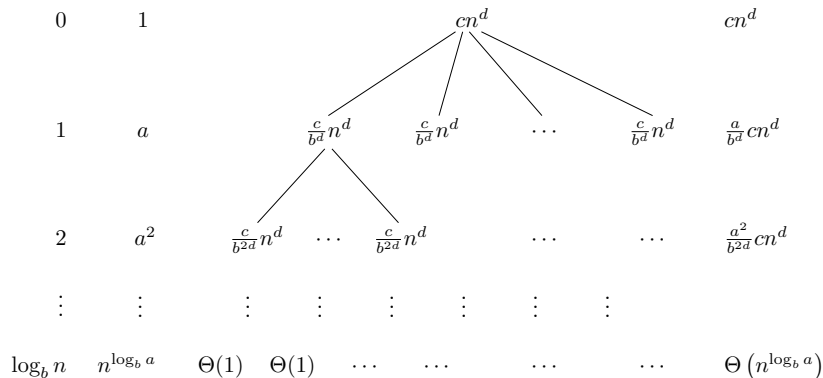
$$T(n) = \left\{ \begin{array}{ll} \Theta(n^{\log_b a}) & \text{if } d < \log_b a \\ \Theta(n^d \log n) & \text{if } d = \log_b a \\ \Theta(n^d) & \text{if } d > \log_b a \end{array} \right\} \text{ for } n = b^k, k = 1, 2, 3, \dots$$

Asymptotic Recurrences

Proof

Level Nodes

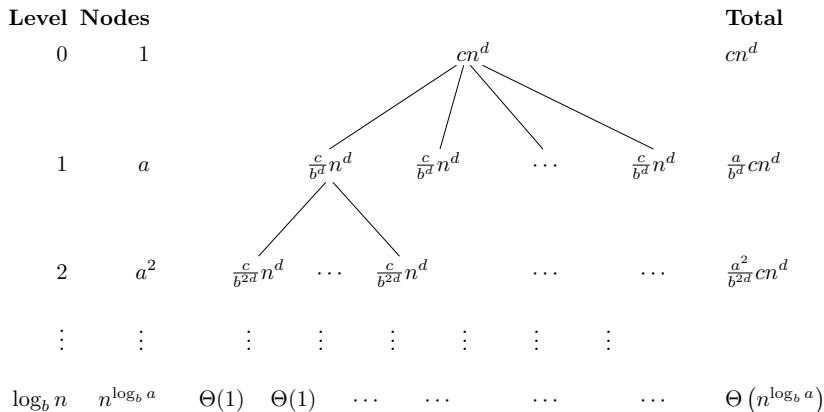
Total



$$T(n) \leq \left(\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} cn^d \right) + \Theta(n^{\log_b a}) = cn^d \left(\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} \right) + \Theta(n^{\log_b a})$$

Asymptotic Recurrences

Proof



$$T(n) \geq \left(\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} cn^d \right) + \Theta(n^{\log_b a}) = cn^d \left(\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} \right) + \Theta(n^{\log_b a})$$

Asymptotic Recurrences

Let

$$\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =: S(n)$$

1. If $d < \log_b a$, then $a/b^d > 1$ and

$$\begin{aligned} S(n) &= \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i = \frac{(a/b^d)^{\log_b n} - 1}{a/b^d - 1} = \\ &= \frac{n^{\log_b (a/b^d)} - 1}{a/b^d - 1} = \Theta(n^{\log_b a - d}). \end{aligned}$$

As a result,

$$T(n) \leq cn^d S(n) + \Theta(n^{\log_b a}) = O(n^{\log_b a}).$$

Asymptotic Recurrences

Let

$$\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =: S(n)$$

1. If $d < \log_b a$, then $a/b^d > 1$ and

$$\begin{aligned} S(n) &= \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i = \frac{(a/b^d)^{\log_b n} - 1}{a/b^d - 1} = \\ &= \frac{n^{\log_b (a/b^d)} - 1}{a/b^d - 1} = \Theta(n^{\log_b a - d}). \end{aligned}$$

As a result,

$$T(n) \geq cn^d S(n) + \Theta(n^{\log_b a}) = \Omega(n^{\log_b a}).$$

Asymptotic Recurrences

Let

$$\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =: S(n)$$

2. If $d = \log_b a$, then $a/b^d = 1$ and

$$S(n) = \log_b n = \Theta(\log n).$$

As a result,

$$T(n) \leq cn^d S(n) + \Theta(n^{\log_b a}) = O(n^d \log n).$$

Asymptotic Recurrences

Let

$$\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =: S(n)$$

2. If $d = \log_b a$, then $a/b^d = 1$ and

$$S(n) = \log_b n = \Theta(\log n).$$

As a result,

$$T(n) \geq cn^d S(n) + \Theta(n^{\log_b a}) = \Omega(n^d \log n).$$

Asymptotic Recurrences

Let

$$\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =: S(n)$$

3. If $d > \log_b a$, then $a/b^d < 1$ and

$$S(n) = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i \leq \sum_{i=0}^{\infty} \left(\frac{a}{b^d}\right)^i = O(1).$$

Moreover,

$$S(n) \geq 1 = \Omega(1).$$

Hence,

$$S(n) = \Theta(1).$$

As a result,

$$T(n) \leq cn^d S(n) + \Theta(n^{\log_b a}) = O(n^d).$$

Asymptotic Recurrences

Let

$$\sum_{i=0}^{\log_b n - 1} \frac{a^i}{b^{id}} = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =: S(n)$$

3. If $d > \log_b a$, then $a/b^d < 1$ and

$$S(n) = \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i \leq \sum_{i=0}^{\infty} \left(\frac{a}{b^d}\right)^i = O(1).$$

Moreover,

$$S(n) \geq 1 = \Omega(1).$$

Hence,

$$S(n) = \Theta(1).$$

As a result,

$$T(n) \geq cn^d S(n) + \Theta(n^{\log_b a}) = \Omega(n^d).$$

Asymptotic Recurrences

Corollary

Let $T(n)$ be a function satisfying

$$\begin{aligned}T(n) &= aT(n/b) + \Theta(n^d) && \text{for } n = b^k, k = 1, 2, 3, \dots, \\T(1) &= \Theta(1),\end{aligned}$$

where $a \geq 1$ and $b \geq 2$ are in \mathbb{N} , and $d > 0$ is in \mathbb{R} . Then

$$T(n) = \left\{ \begin{array}{ll} \Theta(n^{\log_b a}) & \text{if } d < \log_b a \\ \Theta(n^d \log n) & \text{if } d = \log_b a \\ \Theta(n^d) & \text{if } d > \log_b a \end{array} \right\} \text{ for } n = b^k, k = 1, 2, 3, \dots$$

- ▶ Holds for non-exact-powers of b as well
- ▶ More general statement: [Master theorem](#) (see Cormen et al.)

Analysis of Merge Sort

MERGESORT(a, f, l):

Input : Integer $n \geq 1$; Array $a = \langle a[1], \dots, a[n] \rangle$ of elements of S ;
Integers f, l such that $1 \leq f \leq l \leq n$

Behaviour: Sorts the subarray $\langle a[f], \dots, a[l] \rangle$ in increasing order

if $f = l$ **then**

return

else

$m \leftarrow \lfloor (f + l) / 2 \rfloor$;

MERGESORT(a, f, m);

MERGESORT($a, m + 1, l$);

MERGE(a, f, m, l);

end

- ▶ Time complexity $T(n)$ of **MERGESORT**($a, 1, n$):

$$T(n) = 2T(n/2) + \Theta(n)$$

$$T(1) = \Theta(1)$$

- ▶ Solution: $T(n) = \Theta(n \log n)$
- ▶ Better than Insertion sort...

Multiplication of Square Matrices

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \cdot \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{pmatrix} =$$
$$= \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{pmatrix}$$

$$c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}, \quad i, j = 1, \dots, n$$

Naive Matrix Multiplication

Input : Integer $n \geq 1$; Matrices $A = (a[i,j] \mid 1 \leq i,j \leq n)$,
 $B = (b[i,j] \mid 1 \leq i,j \leq n)$ over some semiring

Output: The matrix $A \cdot B = (c[i,j] \mid 1 \leq i,j \leq n)$

```
for  $i \leftarrow 1$  to  $n$  do
  for  $j \leftarrow 1$  to  $n$  do
     $c[i,j] \leftarrow 0$ ;
    for  $k \leftarrow 1$  to  $n$  do
       $c[i,j] \leftarrow c[i,j] + a[i,k] * b[k,j]$ ;
    end
  end
end
```

- ▶ $\Theta(n^3)$ arithmetic operations
- ▶ Can we do better?

Matrix Multiplication: Divide and Conquer?

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

$$C_{1,1} = A_{1,1} \cdot B_{1,1} + A_{1,2} \cdot B_{2,1}$$

$$C_{1,2} = A_{1,1} \cdot B_{1,2} + A_{1,2} \cdot B_{2,2}$$

$$C_{2,1} = A_{2,1} \cdot B_{1,1} + A_{2,2} \cdot B_{2,1}$$

$$C_{2,2} = A_{2,1} \cdot B_{1,2} + A_{2,2} \cdot B_{2,2}$$

- ▶ Suppose that the blocks are square matrices of sizes $\lfloor n/2 \rfloor \times \lfloor n/2 \rfloor$ or $\lceil n/2 \rceil \times \lceil n/2 \rceil$
- ▶ (If not, we can simply forget about at most one row or column, which we may compute separately in $\Theta(n^2)$ operations)
- ▶ $T(n) = 8T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$
- ▶ Solution: $T(n) = \Theta(n^3)$
- ▶ This is no better than the naive approach

Strassen's Algorithm

- Works for matrices over rings

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

$$C_{1,1} = M_1 + M_2 - M_4 + M_6$$

$$C_{1,2} = M_4 + M_5$$

$$C_{2,1} = M_6 + M_7$$

$$C_{2,2} = M_2 - M_3 + M_5 - M_7$$

where

$$M_1 = (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2})$$

$$M_2 = (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2})$$

$$M_3 = (A_{1,1} - A_{2,1}) \cdot (B_{1,1} + B_{1,2})$$

$$M_4 = (A_{1,1} + A_{1,2}) \cdot B_{2,2}$$

$$M_5 = A_{1,1} \cdot (B_{1,2} - B_{2,2})$$

$$M_6 = A_{2,2} \cdot (B_{2,1} - B_{1,1})$$

$$M_7 = (A_{2,1} + A_{2,2}) \cdot B_{1,1}$$

Strassen's Algorithm

- ▶ Time complexity (in arithmetic operations):

$$T(n) = 7T(n/2) + \Theta(n^2)$$

$$T(1) = \Theta(1)$$

- ▶ Solution: $T(n) = \Theta(n^{\log_2 7})$, where $\log_2 7$ is approximately 2.807
- ▶ The constant factor is pretty large (not ideal for small matrices)
- ▶ There are some asymptotically faster algorithms, but with extremely large constants (limited practical value)
- ▶ Current record of Le Gall (2014): $\Theta(n^{2.373\dots})$