

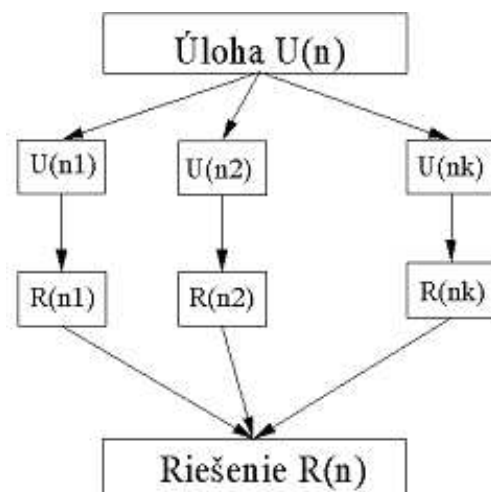
## Kapitola 2

# Základné metódy tvorby efektívnych algoritmov

### 2.1 Rozdeľuj a panuj

Táto metóda je jednou z najčastejšie používaných metód. Aplikujeme ju vtedy, ak riešenie úlohy väčšieho rozsahu vieme získať vhodným zložením výsledkov podúloh menšieho rozsahu ale toho istého charakteru. Úlohu potom riešime rekurzívne. Schématicky sa riešenie problémov metódou rozdeľuj a panuj dá popísať nasledovne

- Rozdeľ úlohu  $U(n)$  rozsahu  $n$  na niekoľko podúloh  $U(n_1), \dots, U(n_k)$  toho istého charakteru ale menšieho rozsahu
- Vyrieš jednotlivé úlohy  $U(n_1), \dots, U(n_k)$
- Ak sú rozsahy úloh  $U(n_1), \dots, U(n_k)$  príliš veľké, použi opakovane princíp rozdeľuj a panuj
- Zlož riešenie  $R(n)$  úlohy  $U(n)$  z riešení  $R(n_1), \dots, R(n_k)$  úloh  $U(n_1), \dots, U(n_k)$



Ak označíme  $T(n)$  zložitosť riešenia nášho problému pre rozsah vstupu  $n$ , tak

$$T(n) = \text{zložitosť rozkladu } U(n) \text{ na } U(n_1), \dots, U(n_k) + \\ + T(n_1) + T(n_2) + T(n_k) + \\ + \text{zložitosť zloženia výsledku.}$$

Skôr, ako si povieme o riešení uvedenej rovnice, jeden príklad.

**Príklad 2.1** *Majme problém MinMax*

*minMax*

*vstup:*  $S = \{a_1, a_2, \dots, a_n\}, a_i \in A$ , kde  $A$  je množina s usporiadaním  
*výstup:* Maximálny a minimálny prvok množiny  $S$   
*miera zložitosti:* počet porovnaní

---

**Algoritmus 1** Rozdeľuj a panuj

---

1: **if**  $|S| = \{a\}$  **then** return  $(a, a)$   
 2: **if**  $|S| = 2$  **then** return  $(a, b)$ , kde  $a, b \in S, a < b$   
 3: **else** rozdeľ  $S$  na dve rovnako veľké disjunktné množiny  $S_1, S_2$ ;  
 4:  $min1, max1 \leftarrow \mathbf{minMax}(S_1)$ ;  
 5:  $min2, max2 \leftarrow \mathbf{minMax}(S_2)$ ;  
 6: return  $(\min\{min1, min2\}, \max\{max1, max2\})$ ;

---

Zložitosť algoritmu budeme kvôli jednoduchosti analyzovať pre prípad  $n = 2^k$ .

$$T(n) = \begin{cases} 1, & \text{ak } n=2 \\ 2T(n/2) + 2 & \text{pre } n > 2 \end{cases}$$

Riešením tejto rekurentnej rovnice dostávame

$$\begin{aligned} T(n) &= 2T(n/2) + 2 = 2(2T(n/4) + 2) + 2 = \dots = \\ &= 2^{k-1}T(2) + \sum_{i=1}^{k-1} 2^i = 2^{k-1} + 2^k - 1 - 1 = \frac{3n}{2} - 2 \end{aligned}$$

Odvodili sme zložitosť najlepšieho i najhoršieho prípadu pre prípad  $n = 2^k$ . Dá sa ukázať, že zložitosť meraná počtom porovnaní sa vylepšiť nedá, a tak v tomto prípade použitie metódy rozdeľuj a panuj viedlo dokonca k optimálnemu algoritmu.

**Veta 2.1 (metóda rozdeľuj a panuj)** *Nech  $a, b, c$  sú nezáporné racionálne čísla,  $n = c^k$ . Potom riešením rekurentnej rovnice*

$$T(n) = \begin{cases} b & \text{ak } n=1 \\ aT(n/c) + bn & \text{pre } n > 1 \end{cases} \quad \text{je} \quad T(n) = \begin{cases} O(n) & \text{pre prípad } a < c \\ O(n \log n) & \text{pre prípad } a = c \\ O(n^{\log_c a}) & \text{pre prípad } a > c \end{cases}$$

**Dôkaz:** Všimnime si niekoľko prvých členov súčtu

$$\begin{aligned} T(n) &= aT(n/c) + bn = a(aT(n/c^2) + bn/c) = a^2T(n/c^2) + bna/c + bn = \\ &= a^2(aT(n/c^3) + bn/c^2) + abn/c + bn = \dots = \\ &= a^{\log_c n} b + bn \sum_{i=0}^{\log_c(n-1)} \left(\frac{a}{c}\right)^i \end{aligned}$$

Rovnicu doriešime podľa vzťahu  $a, c$ .

$$a < c \quad \sum_{i=0}^{\log_c n-1} \left(\frac{a}{c}\right)^i \text{ konverguje} \Rightarrow O(n)$$

$$a = c \quad \left(\frac{a}{c}\right) = 1, \text{ preto } \sum_{i=0}^{\log_c n-1} \left(\frac{a}{c}\right)^i = \log n \Rightarrow O(n \log n)$$

$$a > c \quad \sum_{i=0}^{\log_c n-1} \left(\frac{a}{c}\right)^i \text{ je geometrický rad so súčtom } O(n^{\log_c a})$$

□

Prečo hovoríme tejto vete, že je to veta o metóde rozdeľuj a panuj?

**ak**

- delením získame  $a$  podúloh rovnakej veľkosti - rozsahu  $n/c$

- rozklad a zloženie výsledku vieme spraviť v lineárnom čase

**tak** zložitosť získaného algoritmu je vyjadrená práve vzťahom z vety 2.1 a je preto nanaajvyš polynomiálna.

Bez dôkazu uvedieme všeobecnejšie znenie vety 2.1.

**Veta 2.2 (master theorem)** *Nech  $a \geq 1, b > 1$  sú konštanty,  $f(n)$  je funkcia a nech  $T(n)$  je funkcia na nezáporných celých číslach definovaná nasledujúcou rekurenciou:*

$$T(n) = \begin{cases} c & \text{ak } n=1 \\ aT(n/b) + f(n) & \text{pre } n > 1 \end{cases}$$

Potom  $T(n)$  asymptoticky odhadneme nasledovne:

$$T(n) = \begin{cases} f(n) = O(n^{\log_b a - \varepsilon}), \text{ pre nejakú konštantu } \varepsilon & \text{potom } T(n) = \Theta(n^{\log_b a}) \\ f(n) = O(n^{\log_b a}) & \text{potom } T(n) = \Theta(n^{\log_b a} \ln n) \\ f(n) = O(n^{\log_b a + \varepsilon}), af(n/b) \leq df(n), c < 1 & \text{potom } T(n) = O(f(n)) \end{cases}$$

### 2.1.1 Násobenie veľkých čísel

Uvažujme násobenie  $n$ -bitových čísel v situácii, keď v jednotkovom čase vieme násobiť len jednociferné čísla (alebo čísla ohraničenej dĺžky). Klasický školský algoritmus vedie k zložitosti  $O(n^2)$  aritmetických operácií. Aplikujme metódu rozdeľuj a panuj.

Pri priamočiarom použití metódy rozdeľuj a panuj vyjadríme každý z reťazcov  $X, Y$  priamočiare použítie dĺžky  $n$  pomocou dvoch reťazcov dĺžky  $n/2$ . Ich vynásobením dostávame algoritmus, ktorého zložitosť ostala  $O(n^2)$ .

$$X = X_1X_2 \quad X = X_12^{n/2} + X_2 \quad Y = Y_1Y_2 \quad Y = Y_12^{n/2} + Y_2$$

$$X \cdot Y = (X_12^{n/2} + X_2)(Y_12^{n/2} + Y_2) = X_1Y_12^n + (X_1Y_2 + X_2Y_1)2^{n/2} + X_2Y_2$$

↓

$$T(n) = 4T(n/2) + bn = O(n^2)$$

Zdá sa, že nám tento prístup nepomohol. Aby metóda rozdeľuj a panuj viedla k efektívnejšiemu algoritmu, museli by sme použiť menej ako 4 násobenia.

Keď si všimneme, že

$$(a + b)(c + d) = ac + (bc + ad) + bd$$

rozumnejšie použítie

mohli by sme postupovať nasledovne:

$$U \leftarrow X_1Y_1 \quad V \leftarrow X_2Y_2 \quad W \leftarrow (X_1 + X_2)(Y_1 + Y_2)$$

↓

$$X \cdot Y = U2^n + (W - U - V)2^{n/2} + V$$

Pre zložitosť tohto prístupu platí  $T(n) = 2T(n/2) + T(n/2 + 1) + cn$ . Ostalo násobenie  $n/2$  a  $n/2 + 1$  bitových čísel. Tu však môžeme postupovať podobne —  $n/2 + 1$  bitové čísla môžeme "rozložiť" na 1 bit a  $n/2$  bitov. To vedie k nahradeniu násobenia  $n/2 + 1$  bitových čísel jedným násobením  $n/2$ -bitových čísel.

$$\begin{aligned}(X_1 + X_2) &= A_1 A_2, \text{ pričom } A_1 \text{ je jednobitové číslo} \\ (Y_1 + Y_2) &= B_1 B_2, \text{ pričom } B_1 \text{ je jednobitové číslo}\end{aligned}$$

Potom

$$\begin{aligned}(X_1 + X_2)(Y_1 + Y_2) &= (A_1 2^{n/2} + A_2)(B_1 2^{n/2} + B_2) \\ &= A_1 B_1 2^n + A_2 B_1 2^{n/2} + A_1 B_2 2^{n/2} + A_2 B_2\end{aligned}$$

$$T(n) = 3T(n/2) + cn = \mathbf{O}(n^{\log_2 3})$$

### 2.1.2 Strassenov algoritmus násobenia matíc

Uvažujme problém násobenia dvoch štvorcových matíc stupňa  $n \times n$ . Ak postupujeme podľa definície násobenia matíc, získaný algoritmus je zložitosti  $O(n^3)$  aritmetických operácií. Ak uvažujeme, že prvky matíc sú z okruhu, môžeme sa pokúsiť použiť metódu rozdeľuj a panuj:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$\begin{aligned}C_{11} &= A_{11}B_{11} + A_{12}B_{21} & C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} & C_{22} &= A_{21}B_{12} + A_{22}B_{22}\end{aligned}$$

Priamočiare použitie metódy rozdeľuj a panuj opäť vedie k zložitosti, ktorá nie je vylepšením.

$$T(n) = 8T(n/2) + cn^2 = \dots = O(n^3)$$

Vylepšenie získame ušetrením jedného násobenie na úkor nárastu počtu sčítaní a odčítaní. Násobenie dvoch matíc rozmeru  $2 \times 2$  môžeme spraviť nasledovne:

$$\begin{aligned}P &= (a_{11} + a_{22})(b_{11} + b_{22}) & C_{11} &= P + S - T + V \\ Q &= (a_{21} + a_{22})b_{11} & C_{12} &= R + T \\ R &= a_{11}(b_{12} - b_{22}) & C_{21} &= Q + S \\ S &= a_{22}(b_{21} - b_{11}) & C_{22} &= P + R - Q + U \\ T &= (a_{11} + a_{12})b_{22} \\ U &= (a_{21} - a_{11})(b_{11} + b_{12}) \\ V &= (a_{12} - a_{22})(b_{21} + b_{22})\end{aligned}$$

Aplikovaním uvedených vzťahov a metódy rozdeľuj a panuj dostávame rekurzívny algoritmus, ktorého zložitosť vyjadríme rekurentnou rovnicou

$$T(n) = \begin{cases} 7T(n/2) + an^2, & n > 2 \\ b, & n \leq 2 \end{cases}$$

Riešením tejto rovnice je  $O(n^{\log_2 7})$ .

Pritom sme predpokladali, že  $n = 2^k$ . Čo v prípade, ak to nie je pravda

- doplniť nulovými riadkami a stĺpcami na najbližšiu mocninu dvojky;
- pri párnom  $n$  priame použitie, pri nepárnom  $n$  doplniť jeden riadok a stĺpec
- doplniť na najbližšie také číslo  $m$ , že  $m = 2^r p$  a potom po rozmer  $p \times p$  aplikuj rozdeľuj a panuj, pri rozmere  $p \times p$  priamo vynásob

### 2.1.3 Vyhľadávanie $k$ -teho najmenšieho prvku

**vstup:**  $S = \{a_1, a_2, \dots, a_n\}, a_i \in A$ , kde  $A$  je množina s usporiadaním,  
 $k \in \mathbb{N}$

**výstup:**  $k$ -ty najmenší prvok množiny  $S$

**miera zložitosti:** počet porovnaní

Zaujímá nás teda  $k$ -ty najmenší prvok, nevyžadujeme, aby množina bola utriedená. Na riešenie by sme mohli použiť jednoduché algoritmy.

1. utried' a vyber  $k$ -ty najmenší prvok  $O(n \log n)$
2.  $k$ -krát hľadaj minimum  $O(kn)$
3. resp. porovnaním vstupných hodnôt  $k, \log n$  zistiť, ktorý z uvedených algoritmov je pre daný vstup výhodnejší.  $O(\min\{k, \log n\} \cdot n)$

Využitím metódy rozdeľuj a panuj získame algoritmus  $Select(k, S)$ .

---

**Algoritmus 2**  $Select(k, S)$ 


---

```

1: if  $S < 50$  then utried' a najdi  $k$ -ty najmenší
2: else
3:   rozdeľ  $S$  do 5-tíc a utried' v rámci 5-tíc
4:   vytvor množinu  $M$  stredných prvkov
5:    $m := SELECT(M/2, M)$ 
6:   vytvor množiny  $S_1, S_2$ 
7:    $S_1 = \{a \in S; a < m\}$ 
8:    $S_2 = \{a \in S; a > m\}$ 
9:   if  $|S_1| \geq k$  then return  $SELECT(k, S_1)$ 
10:  else
11:    if  $|S_1 + 1| = k$  then return  $m$ 
12:    else return  $SELECT(k, S_2)$ 

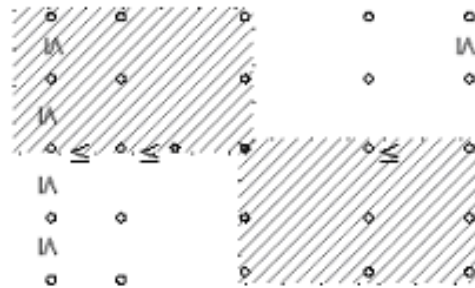
```

---

**Korektnosť algoritmu** ľahko ukážeme indukciou vzhľadomk veľkosti množiny  $S$ .

**Zložitosť algoritmu:** označme  $T(n)$  zložitosť pre  $|S| = n$ . Potom

- 3,4      zložitosť  $O(n)$   
5           $T(n/5)$   
6-8         $O(n)$   
9-12       $O(\max\{T(|S_1|), T(|S_2|)\})$



Keďže  $|S_1|, |S_2| \leq 3|S|/4$ , dostávame

$$T(n) \leq \begin{cases} T(n/5) + T(3n/4) + cn, & 50 \leq n \\ cn, & n \leq 50 \end{cases}$$

Matematickou indukciou sa dá ukázať:  $T(n) \leq 20cn$ .

### 2.1.4 Násobenie Booleovských matic

Majme problém násobenia dvoch booleovských matic. Nakoľko booleovské matice netvoria okruh, nemôžeme priamo použiť Strassenov algoritmus násobenia matic. Môžeme však vnoriť booleovské matice do okruhu, tam použiť Strassenov algoritmus a potom sa vrátiť naspäť.

Majme booleovské matice  $A, B$ . Nech súčinom  $A \cdot B = C$ . Predstavme si, že budeme matice  $A$  a  $B$  násobiť v **okruhu**  $Z_{n+1}$ , kde  $n$  je rozmer matic  $A, B$ . Výsledkom tohto násobenia nech je matica  $D$ . Aký je vzťah medzi maticami  $C$  a  $D$ ?

$$\begin{aligned} D(i,j) = 0 &\Leftrightarrow C(i,j) = 0 \\ D(i,j) \neq 0 &\Leftrightarrow C(i,j) = 1 \end{aligned}$$

Takže z matice  $D$  vieme požadovanú maticu  $C$  získať v čase úmernom veľkosti matice  $C, D$ .

Nasleduje algoritmus, ktorý je lepší ako klasický školský algoritmus násobenia, hoci väčšej zložitosti ako Strassenov algoritmus. Rozdelíme maticu  $A$  na stĺpce  $A_1, \dots, A_m$  hrúbky  $\log n$  a maticu  $B$  na riadky  $B_1, \dots, B_m$  hrúbky  $\log n$ , kde  $m = n/\log n$ . Potom

$$A \cdot B = \sum_{i=1}^m A_i \cdot B_i = \sum_{i=1}^m C_i$$

Pokiaľ by sa nám podarilo získať súčin  $A_i \cdot B_i$  v čase  $O(n^2)$ , získame algoritmus zložitosti  $O(n^3/\log n)$ . Uvedomme si, že  $j$ -ty riadok matice  $C_i$  vznikol tak, že sa sčítali tie riadky matice  $B_i$ , ktoré odpovedali jednotkám v  $j$ -tom riadku matice  $A_i$ . Keďže všetkých možných riadkov matice  $A_i$  je  $n$ , je aj počet všetkých možných riadkov matice  $C_i$  len  $n$ . Najprv všetky tieto potenciálne riadky predvypočítame do tabuľky TAB, a potom použijeme ten, ktorý potrebujeme.

Pri fixovanom  $i$  TAB počítame nasledovne:

$$TAB(0) = 0000\dots 0$$

$$TAB(j) = TAB(j - 2^k) + B_i(k + 1), \text{ kde } 2^k \leq j < 2^{k+1} \text{ a } B_i(t) \text{ označuje}$$

$t$ -ty riadok odspodu matice  $B_i$  (každé číslo považujeme za binárny reťazec dĺžky  $\log n$ . Potom odčítanie  $2^k$  odpovedá nahradeniu vodiacej jednotky v binárnom zápise čísla  $j$  nulou)

Ako pomocou TAB získame maticu  $C$ ? Nech  $bin(v)$  označuje číslo, ktorého binárnym zápisom je binárny reťazec  $v$ . Potom

$$C_i(j) = TAB(bin(A_i(j)))$$

---

### Algoritmus 3 BoolNásobenie

---

- 1: rozdeľ maticu  $A$  na stĺpcové podmatice  $A_1, A_2, \dots, A_m$
  - 2: rozdeľ maticu  $B$  na riadkové podmatice  $B_1, B_2, \dots, B_m$
  - 3: vytvor nulovú maticu  $C$
  - 4: **for**  $i=1$  to  $m$  **do**
  - 5:     vypočítaj TAB
  - 6:     **for**  $j = 1$  to  $n$  **do**
  - 7:          $C(j) = C(j) \oplus TAB(bin(A_i(j)))$
- 

Aká je zložitosť tohto algoritmu?

- riadok 5 vieme realizovať so zložitou  $O(n^2)$
- cyklus na riadku 6 sa realizuje v čase  $O(n^2)$
- cyklus na riadku 4 opakujeme  $m$ , čiže  $n/\log n$  krát

Preto je výsledná zložitosť  $O(n^3/\log n)$ .