Curs 1

Timp de rulare = nr. de operații

Notațiile asimptotice: $o \to$ „o mic"

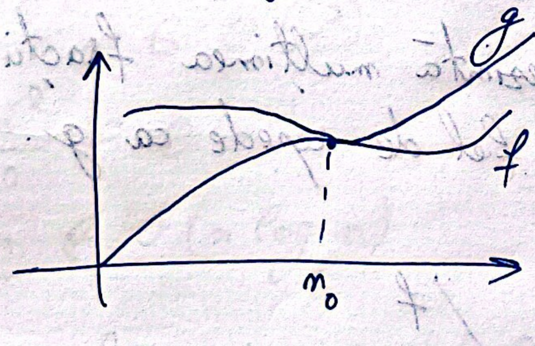
$O \to$ „O mare"

$\Omega \to$ „omega"

$\omega \to$ „omega mic"

$\theta \to$ „Teta"

$O(g)$: mulțimea funcțiilor care cresc mai încet sau la fel de încet ca g.



Def.: Spunem că o fct. $f \in O(g)$ dacă $(\exists)\ n_0, c > 0$ a.î. $(\forall)\ n \geq n_0$ avem $f(n) \leq c \cdot g(n)$

Exemplu: • $2n \in O(n^2)$

$f(n) = 2n$

$g(n) = n^2$

$n_0 = 2$

$c = 1$

$\forall\, n \geq 2 \quad 2n \leq 1 \cdot n^2$

• $200 n^2 \in O(n^2)$ A

$n_0 = 1$ $\quad (\forall)\, n \geq 1 \quad 200 n^2 \leq c \cdot n^2$

$c = 200$

• $n^2 \notin O(n)$

P.p. prin reducere la absurd că $n^2 \in O(n)$

$\Rightarrow (\exists)\, n_0, c > 0$ a.î. $n^2 \leq c \cdot n \quad (\forall)\, n \geq n_0$

$\underbrace{n}_{\infty} \leq \underbrace{c}_{\text{constantă}}$ ↯

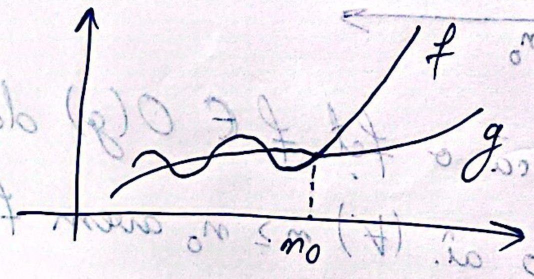
• $2^{n+1} \in O(2^n)$?

$c = 2$

$n_0 = 1$

$2^{2n} \in O(2^n)$? Temă

$\Omega(g)$ - informal reprezintă mulțimea funcțiilor care cresc mai repede sau la fel de repede ca g.



Exemple: $n^2 \in \Omega(n \log n)$

$\frac{n}{2} \in \Omega(n)$

$2^n \in \Omega(\log n)$

Def.: $f \in \Omega(g)$ dacă $(\exists)\ n_0, c > 0$ a.î. $(\forall)\ n > n_0$ avem $f(n) \geq c \cdot g(n)$.

$f \in \Theta(g)$ - informal f crește la fel de repede ca g

Def.: $f \in \Theta(g)$ dacă $(\exists)\ c_1, c_2, n_0 > 0$ a.î.

$(\forall)\ n \geq n_0$ avem $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$200n^2 \notin \Theta(n)$

$200n^2 \in \Theta(n^2)$

$\frac{n^2}{10} \in \Theta(n^2)$

$n \notin \Theta(n \log n)$

Teoremă: $f \in \Theta(g) \iff f \in O(g)$ și $f \in \Omega(g)$

$f \in o(g)$ - informal f crește strict mai mic decât g.

Exemple:

$n \in o(n \log n)$

$n^2 \in o(n^3)$

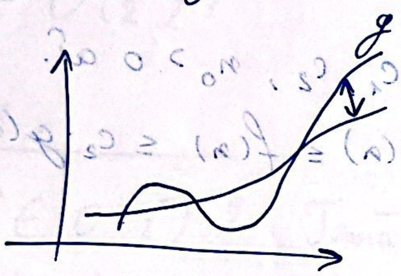
$n \notin o(n)$

$n \notin o(100n)$

$\frac{n}{2} \notin o(n)$

Def.: $f \in o(g)$ dacă $(\forall) c > 0\ (\exists) n_0 > 0$ a.î. $(\forall) n \geq n_0$,

$f(n) < c \cdot g(n)$



$f \in \omega(g)$ - informal f crește strict mai repede decât g

$n^3 \in \omega(n^2)$

$2^n \in \omega(n)$

Exercițiu: $n \in o(n^2)$

fie $c > 0$ fixat

Vrem să găsim un n_0 a.î. $(\forall) n > n_0$ să avem $n < c \cdot n^2$.

n_0 depinde de c. De exemplu putem alege $n_0 = \frac{1}{c}$, $(\forall) n > \frac{1}{c}$ $n < cn^2$

Seminar 1

! $O(g(n)) = \{ f(n) \mid \exists\, c > 0, n_0 > 0$ a.î. $f(n) \le c \cdot g(n), \forall n \ge n_0 \}$

$n^2 \in O(n^4)$ DA

$n^2 \le 4n^4, \forall n \ge 5$ (A)

$\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f(n) \in O(g(n))$

! $\Omega(g(n)) = \{ f(n) \mid \exists\, c > 0, n_0 > 0$ a.î. $c \cdot g(n) \le f(n), \forall n \ge n_0 \}$

$n^3 \in \Omega(n^2 \log n)$ DA

$n \log n \in \Omega(n \sqrt{n})$ NU

$\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0 \Rightarrow f(n) \in \Omega(g(n))$

! $\Theta(g(n)) = \{ f(n) \mid \exists\, c_1, c_2 > 0, n_0 > 0$ a.î. $c_1 g(n) \le f(n) \le c_2 g(n), \forall n \ge n_0 \}$

$n^2 \in \Theta(n^2)$ DA

$n \in \Theta(n^2)$ NU

$\lim_{n \to \infty} \frac{f(n)}{g(n)} \in R_+ \Rightarrow f(n) \in \Theta(g(n))$

! $o(g(n)) = \{ f(n) \mid \forall c>0\ \exists n_0 \geq 0$ a.î. $f(n) < cg(n), \forall n \geq n_0 \}$

$n^2 \in o(n^4)$ NU

$n^2 < cn^2$ (F)

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

! $\omega(g(n)) = \{ f(n) \mid \forall c>0\ \exists n_0 > 0$ a.î. $cg(n) < f(n), \forall n \geq n_0 \}$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

Ex. Cormen pg. 52-53:

3.1.1. fie f, g asimptotice poz. $\Rightarrow \exists n_1 \geq 0$ a.î. $f(n) \geq 0, \forall n \geq n_1$

$\exists n_2 > 0$ a.î. $g(n) \geq 0, \forall n \geq n_2$

aleg $n_0 = \max(n_1, n_2)$

~~mag~~ $\max(f, g) \in \Theta(f+g)$

$$\max(a, b) = \frac{a+b+(a-b)}{2}$$

$$\max(f(n), g(n)) = \frac{f(n)+g(n)+\overbrace{|f(n)-g(n)|}^{\geq 0}}{2} \geq \frac{f(n)+g(n)}{2} \quad (1)$$

$$\max(f(n), g(n)) \leq f(n)+g(n) \quad (2)$$

(1), (2) $\Rightarrow \max(f, g) \in \Theta(f+g)$

$n! \in O(n^n)$

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n \leq n^n, \forall n \geq 2$

$\log(n!) \in \Theta(n \log n)$

$$\boxed{\begin{aligned} f \in \Theta(g) \iff\ & f \in \Omega(g) \\ & f \in O(g) \end{aligned}}$$

$\log_2 n! = \log_2(1 \cdot 2 \cdot \ldots \cdot n) \leq \log_2 n^n = n \log_2 n$

$\underline{(n!)^2 \geq n^n, \forall n \geq 10} \Rightarrow \log_2(n!)^2 \geq \log_2 n^{\frac{n}{2}} \Rightarrow$

$\Rightarrow 2 \log(n!) \geq \frac{n}{2} \log_2 n$

$\Rightarrow \log_2 n! \geq \frac{1}{4} n \log_2 n$

$(1 \cdot 2 \cdot 3 \cdot \ldots \cdot n)^2 = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n \cdot 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n =$

$= (1 \cdot n)(2(n-1))(3 \cdot (n-2)) \ldots =$

$= \underbrace{(1 n^2)}_{\geq n}\underbrace{(2(n-1))^2}_{\geq n}\underbrace{(3(n-2))^2}_{\geq n} \ldots \geq n^{\frac{n}{2}}$

3.1.2. $\forall a, b,\ b>0$

$$(n+a)^b \in \Theta(n^b) \Leftrightarrow c_1 n^b \le (n+a)^b \le c_2 n^b$$

$$\lim_{n\to\infty} \frac{(n+a)^b}{n^b} = \lim_{n\to\infty} \left(\frac{n+a}{n}\right)^b = 1^b = 1 > 0 \text{ și } \in \mathbb{R}_+ \Rightarrow$$

$$\Rightarrow (n+a)^b \in \Theta(n^b)$$

SAU

$$(n+a)^b = \underbrace{C_b^0 n^b a^0}_{\le n^b a^b} + \underbrace{C_b^1 n^{b-1} a^1}_{\le n^b a^b} + \dots + \underbrace{C_b^b n^0 a^b}_{\le n^b a^b} \le$$

$$n^b a^b (C_b^0 + C_b^1 + \dots + C_b^b) =$$

$$= n^b a^b \cdot 2^b = n^b \cdot \underbrace{(2a)^b}_{c_2}$$

3.1.7. $\underbrace{o(g(n))}_{A_1} \cap \underbrace{\omega(g(n))}_{A_2} = \emptyset$

P.p. că $A_1 \cap A_2 \ne \emptyset \Rightarrow \exists f \in A_1 \cap A_2 \Rightarrow$

$f \in A_1 \Rightarrow \forall c_1 > 0\ \exists n_1 > 0$ a.î. $f(n) < c_1 g(n),\ \forall n \ge n_1$

$f \in A_2 \Rightarrow \forall c_2 > 0\ \exists n_2 > 0$ a.î. $f(n) > c_2 g(n),\ \forall n \ge n_2$

$n_0 = \max(n_1, n_2)$

$c = c_1 = c_2$

$\Rightarrow \left. \begin{array}{l} f(n) < c \cdot g(n) \\ f(n) > c \cdot g(n) \end{array} \right\}$ abs

---

3.1.8.

$O(g(n,m)) = \{ f(n,m) \mid \exists\, c_1 > 0, n_0 > 0, m_0 > 0$ a.î.
$f(n,m) \le c_1 g(n,m), \forall n \ge n_0, m \ge m_0 \}$

---

pag. 62:

3-4.

a. $f(n) = O(g(n)) \Rightarrow g(n) = O(f(n))$ F

$n^2 \in O(n^4) \Rightarrow n^4 \in O(n^2)$ fals

b. fals

c. adev.

d. fals

e. $f(n) \in O(f(n^2))$

$f(n) \le f^2(n)$?

$f(n) \ge 1$

f. $f(n) \in O(g(n)) \Rightarrow g(n) \in \Omega(f(n))$

$\Rightarrow f(n) \leq c\, g(n)$

$\frac{1}{c} f(n) \leq g(n)$

g. fals

$f(n) \in \Theta(f(n/2))$

$f(n) = 2^n \Rightarrow 2^n \in \Theta(2^{\frac{n}{2}})$

h) $f(n) + o(f(n)) \in \Theta(f(n))$

fie $g \in o(f(n)) \Rightarrow \exists\, c_1 > 0$ a.î. $g(n) < c_1 \cdot f(n)$

$\exists\, n_0 > 0$

$f(n) \leq f(n) + g(n) \leq f(n) + c_1 \cdot f(n)$

$\leq (c_1 + 1) f(n)$

Curs 2

Analiza timpului de rulare a algoritmilor recursivi.

exemplu: metode de sortare

- Quicksort
- Mergesort
- Heapsort

} $O(n \log n)$

- Radixsort
- Bubblesort
- Countsort
- Insertionsort

Nu sunt bazați pe comparații între chei.

$O(n^2)$

Insertionsort

39 5 7 33 200 96 70

I. 39 rămâne pe loc

II. căutăm poziția pe care să-l inserăm pe 5

5 39 7 33 200 96 70

III. căutăm poziția pe care să-l inserăm pe 7

5 7 39 33 200 96 70

IV. 5 7 33 39 200 96 70

V. 200 e deja unde trebuie

VI. 5 7 33 39 96 200 70

VII. 5 7 33 39 70 96 200

Complexitate: La pasul i facem (în cel mai nefavorabil caz) i operații.

În total vom avea: $1+2+\ldots+n = O(n^2)$

$T(n)$ (= timpul de rulare pt. a rezolva o problemă de dimensiune n)

$T(n) = T(n-1) + O(n) = O(n^2)$

Ex.: $T(n) = T(n/3) + T(2n/3) + O(n) = ?$

Merge sort

Idee: Împărțim problema în subprobleme mai mici pe care le rezolvăm și combinăm rezultatele.

49 33 17 63 43 24 7 136

17 33 49 63 | 7 24 43 136

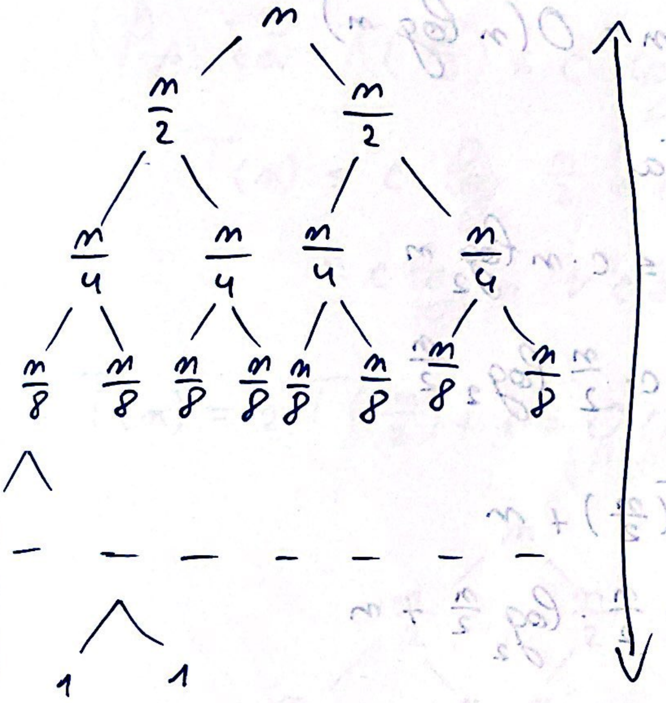
interclasare

33 49 | 17 63 | 24 43 | 7 136

49 | 33 | 17 | 63 | 43 | 24 | 7 | 136

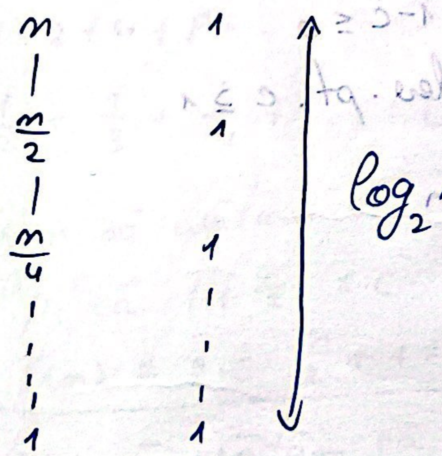
$$T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n = O(n \log n)$$

Nu scriem baza pt. că putem oricum să tranformăm



Cautare binară

$$T(n) = T\left(\frac{n}{2}\right) + 1 = O(\log n)$$



Metoda substituției pt. rezolvarea recurențelor

<u>Exemplu:</u> $T(n) = 2T(\frac{n}{2}) + n = O(n \log n)$

1. „Ghicim" soluția

2. Arătam că $T(n) \le c \cdot n \log_2 n$

$Sp.$ că $T(n/2) \le c \cdot \frac{n}{2} \log_2 \frac{n}{2}$

Știm că $T(n) = 2T(\frac{n}{2}) + n$

$$T(n) \le 2 \cdot c \cdot \frac{n}{2} \cdot \log_2 \frac{n}{2} + n$$

$$T(n) \le c \cdot n \cdot \log_2 \frac{n}{2} + n$$

$$T(n) \le c \cdot n \cdot \log_2 n - cn \log_2 2 + n$$

$$T(n) \le cn \log_2 n + \underbrace{n(1-c)}$$

dacă $1-c \le 0$

adev. pt. $c \ge 1$

Avem $T(n) \le c \cdot n \log_2 n$

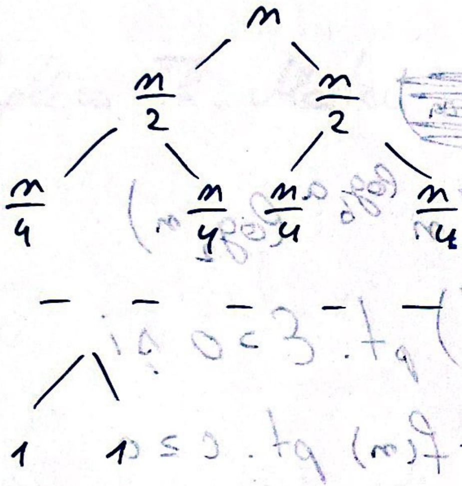
• $T(n) = T\left(\frac{n}{2}\right) + 1 = O(\log_2 n)$

Vrem să arătăm că $T(n) \leq c \cdot \log_2 n$

P.p. că $T\left(\frac{n}{2}\right) \leq c \cdot \log_2 \frac{n}{2}$

$$T(n) \leq c \cdot \log_2 \frac{n}{2} + 1 = c \cdot \log_2 n \underbrace{- c + 1}_{\leq 0,\ \forall c \geq 1}$$
$$\leq c \log_2 n \quad \forall c \geq 1$$

• $T(n) = 2T\left(\frac{n}{2}\right) + 1 = O(n)$



$2^{\log_2 n} = n$

$1 + 2 + 4 + 8 + \ldots = O(n)$ Formule Kinda ?

$\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots = O(\log n)$

Vrem să arătam că $T(n) \leq cn$

P.p. că $T\left(\frac{n}{2}\right) \leq c \cdot \frac{n}{2}$

$T(n) \leq 2 \cdot c \cdot \frac{n}{2} + 1 = c \cdot n + 1$

Vrem să arătam că $T(n) \leq c \cdot n - d$ → constantă

P.p. că $T\left(\frac{n}{2}\right) \leq c \cdot \frac{n}{2} - d$

$$T(n) \leq 2 \cdot \left(c \cdot \frac{n}{2} - d\right) + 1 = c \cdot n - 2d + 1 = c \cdot n - d \underbrace{- d + 1}_{\leq 0,\ \forall d \geq 1}$$

Teorema Master:

Dacă avem recurențe de forma $T(n) = aT(\frac{n}{b}) + f(n)$

Trei cazuri:

1) $f(n) \in O(n^{\log_b a - \varepsilon})$ pt. un $\varepsilon > 0$

atunci $T(n) \in \Theta(n^{\log_b a})$

2) $f(n) \in \Theta(n^{\log_b a})$ $\cancel{\log_2 n}$

atunci $T(n) = \Theta(n^{\log_b a} \log_2 n)$

3) $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ pt. $\varepsilon > 0$ și

$a \cdot f(\frac{n}{b}) \le c \cdot f(n)$ pt. $c \ge 0$.

atunci $T(n) \in \Theta(f(n))$

- $T(n) = T(\frac{n}{2}) + 1$

$f(n) = 1$

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$f(n) \in \Theta(1)$

n)

- $T(n) = 9T\left(\frac{n}{3}\right) + n$

  $f(n) = n$

  $a = 9$ $\qquad n^{\log_b a} = n^2$

  $b = 3$

  $f(n) \in O(n^{2-\varepsilon})$

  $T(n) = \Theta(n^2)$

- $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + 1$

  nu pot aplica Th. Master

Seminar 2

$T(n) = 2T(\frac{n}{2}) + n$

$T(n) \in O(n\log n)$

I. Metoda substituției:

$T(n) \le c \cdot n\log n$ (! de aici în jos e lg nu log)

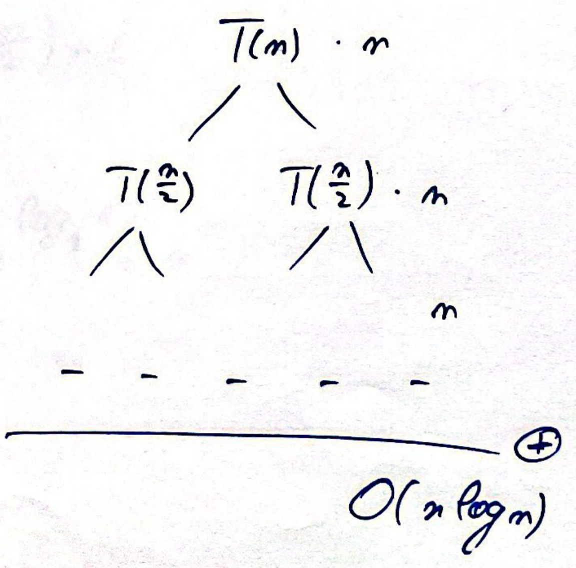
Pp. că $T(k) \le c \cdot k\log k, \forall k < n$

$T(n) = 2T(\frac{n}{2}) + n \le 2 \cdot (c \cdot \frac{n}{2} \cdot \log\frac{n}{2}) + n = c \cdot n\log\frac{n}{2} + n =$

$= c \cdot n \cdot \log n - c \cdot n \cdot \log 2 + n =$

$= c \cdot n \cdot \log n - n(c \cdot \log 2 - 1) \le c \cdot n \cdot \log n, \forall c \ge 10$

II. Arbore de recurență:



III. Teorema Master: $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

a) dacă $\exists\, \varepsilon > 0$ a.î. $f(n) \in O(n^{\log_b a - \varepsilon}) \Rightarrow T(n) \in \Theta(n^{\log_b a})$

b) dacă $f(n) \in \Theta(n^{\log_b a}) \Rightarrow T(n) \in \Theta(n^{\log_b a} \log_2 n)$

c) dacă $\exists\, \varepsilon > 0,\ c < 1,\ n_0 \in \mathbb{N}^*$ a.î. $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n),\ \forall n \geq n_0$, ~~at~~ și $f(n) \in \Omega(n^{\log_b a + \varepsilon}) \Rightarrow T(n) \in \Theta(f(n))$

Exerciții:

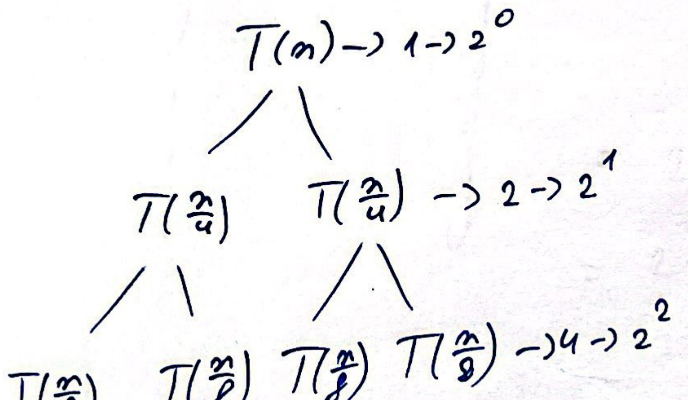
$T(n) = 2T\left(\frac{n}{4}\right) + 1$

$a = 2$
$b = 4$

$n^{\log_4 2} = \sqrt{n}$

a) $1 \in O(n^{\log_b a - \varepsilon})$,

$\varepsilon = \frac{1}{2} \xRightarrow{\text{T Master}} T(n) = \Theta(\sqrt{n})$



$k = \log_4 n$

$1 + 2 + 2^2 + \dots + 2^k = \frac{2^{k+1} - 1}{2 - 1} = 2^{k+1} - 1 \Rightarrow 2^{\log_4 n + 1} - 1 =$

$= 2 \cdot 2^{\log_4 n} - 1 = 2n^{\log_4 2} - 1 = 2\sqrt{n}$

$$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$$

$$n^{\log_4 2} = \sqrt{n}$$

$$\sqrt{n} \in \Theta(\sqrt{n})$$

$$\Rightarrow T(n) = \Theta(\sqrt{n} \log n)$$

$$T(n) = 2T\left(\frac{n}{4}\right) + n$$

$$n^{\log_4 2} = \sqrt{n}$$

$$n \in \Omega\left(n^{\log_4 2 + \varepsilon}\right), \ \varepsilon = \frac{1}{2}$$

$$2 \cdot \frac{n}{4} \leq c \cdot n \iff \frac{n}{2} \leq c \cdot n \Rightarrow c = \frac{1}{2}$$

$$\Rightarrow T(n) = \Theta(n)$$

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

$T(n)$ 1

$T\left(\frac{n}{2}\right)$ 1

$T\left(\frac{n}{4}\right)$ 1

$$\log_2 n = \underbrace{1 + 1 + \ldots + 1}_{\log_2 n} = O(\log_2 n)$$

$T(n) = T(n-1) + n$

$T(n)$ $\quad n$

$T(n-1)$ $\quad n-1$

$\vdots$

$n \Rightarrow 1 + 2 + \ldots + n = \frac{n(n+1)}{2} \in O(n^2)$

$T(n) = 2T(\frac{n}{2}) + 1$

$T(n)$ $\quad 1$

$T(\frac{n}{2}) \quad T(\frac{n}{2})$ $\quad 2$

$\vdots$

$\log_2 n \Rightarrow 1 + 2 + \ldots + 2^k = 2^{\log_2 n + 1} - 1 =$

$= 2^{\log_2 n} \cdot 2 - 1 =$

$= 2n - 1 \in O(n)$

$x, y \rightarrow n$ cifre

$O(n^2) \rightarrow$ alg. clasic

$$x = x_H \cdot 10^{\frac{n}{2}} + x_L$$

$$y = y_H \cdot 10^{\frac{n}{2}} + y_L$$

$$xy = (x_H \cdot 10^{\frac{n}{2}} + x_L)(y_H \cdot 10^{\frac{n}{2}} + y_L) =$$

$$= x_H \cdot y_H \cdot 10^n + x_H 10^{\frac{n}{2}} \cdot y_L + x_L y_H 10^{\frac{n}{2}} + x_L y_L$$

$$= \underbrace{x_H y_H}_{a} 10^n + 10^{\frac{n}{2}} \underbrace{(x_H y_L + x_L y_H)}_{b} + \underbrace{x_L y_L}_{c}$$

$$x_H y_L + x_L y_H = (x_H + x_L)(y_H + y_L) - a - c$$

$$T(n) = 4\,T\left(\frac{n}{2}\right) + n$$

$$T(n) \in \Theta(n^{\log_2 4})$$

$$T(n) = 4\,T\left(\frac{n}{3}\right) + n$$

$\cancel{n^{\log_3 4}}$ $a = 4$

$b = 3$

$f(n) = n$

$n^{\log_b a} = n^{\log_3 4}$ $(> n) \Rightarrow$ caz I: $T(n) \in \Theta(n^{\log_3 4})$

$(f(n) \in O(n^{\log_b a - \varepsilon}),\ \varepsilon = 0{,}1)$

Scanned with CamScanner

$T(n) = 4T(\frac{n}{3}) + n$

$T(n) \le c \cdot n^{\log_3 4}$

$T(n) = 4T(\frac{n}{3}) + n \le 4\left(c\left(\frac{n}{3}\right)^{\log_3 4}\right) + n = \cancel{4} \cdot c \cdot \frac{n^{\log_3 4}}{\cancel{4}} + n$

$= c\, n^{\log_3 4} + n$ nu merge

$T(n) = 4T(\frac{n}{3}) + n$

$T(n) \le c \cdot n^{\log_3 4} - dn$

$T(n) = 4T(\frac{n}{3}) + n \le 4\left(c \cdot \left(\frac{n}{3}\right)^{\log_3 4} - d\left(\frac{n}{3}\right)\right) + n =$

$= c \cdot n^{\log_3 4} - 4d\frac{n}{3} + n =$

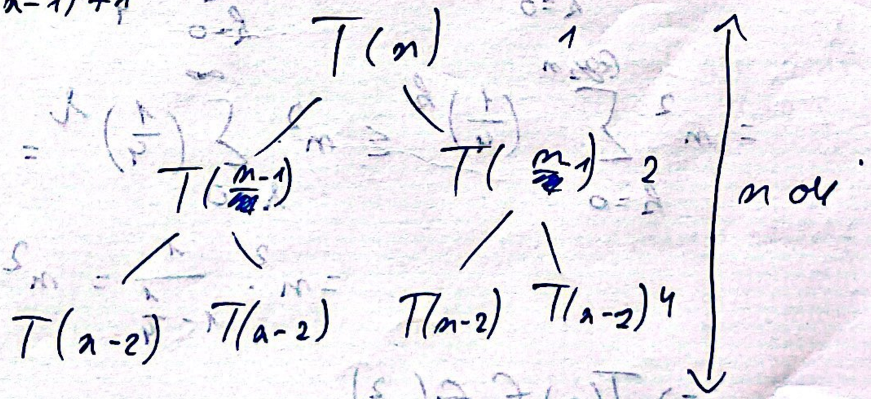
$= c n^{\log_3 4} - n\left(\frac{4d}{3} - 1\right) \le c n^{\log_4 3} - dn$

$n\left(\frac{4d}{3} - 1\right) \ge dn$

$\frac{4d}{3} - 3 \ge d \Rightarrow 4d - 3 \ge 3d$

$d \ge 3$

$T(n) = 2T(n-1) + 1$



$1 + 2 + \ldots + 2^n = 2^{n+1} - 1$

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$$

$$\sum_{k=0}^{\infty} x^k = \begin{cases} \infty, & x \geq 1 \\ \text{divergentă}, & x \leq -1 \\ \frac{1}{1-x}, & x \in (-1,1) \end{cases}$$

$$T(n) = T\left(\frac{n}{2}\right) + n^2$$

$$\begin{array}{lll} T(n) & n^2 & \\ T\left(\frac{n}{2}\right) & \left(\frac{n}{2}\right)^2 & \log_2 n \\ T\left(\frac{n}{4}\right) & \left(\frac{n}{4}\right)^2 & \end{array}$$

$$T(n) = \sum_{k=0}^{\log_2 n} \left(\frac{n}{2^k}\right)^2 = n^2 \sum_{k=0}^{\log_2 n} \frac{1}{4^k} =$$

$$= n^2 \sum_{k=0}^{\log_2 n} \left(\frac{1}{4}\right)^k \leq n^2 \cdot \sum_{l=0}^{\infty} \left(\frac{1}{4}\right)^l =$$

$$= n^2 \cdot \frac{1}{1-\frac{1}{4}} = n^2 \cdot \frac{4}{3}$$

$$\Rightarrow T(n) \in \Theta(n^2)$$

Scanned with CamScanner

Curs 3

Algoritmi probabiliști. Analiza Quicksort

Problema 1 Secretary problem

Avem n candidați la un job pe care îi intervievăm pe rând.

• Strategia 1:

Dacă c_i (candidatul i) este mai bun decât toți candidații anteriori atunci îl/o angajăm și plătim un cost x (generat de concedierea celui anterior)

Câți candidați vom angaja?

În cazul cel mai defavorabil vom angaja n candidați -> sortați crescător.

• Strategia 2:

Permutăm aleator candidații și aplicăm alg. anterior.

Care este nr. mediu de candidați angajați?

Vom considera variabile aleatoare cu două valori: 0 și 1

Ex.: dat cu banul

$$x = \begin{cases} 1 & \text{pt. pajură} \\ 0 & \text{altfel} \end{cases}$$

$$p[x=1] = \frac{1}{2} = p[x=0]$$

Media unei variabile aleatoare:

$$\mathbb{E}[x] = 1 \cdot p[x=1] + \cancel{0 \cdot p[x=0]} = p[x=1]$$

expectation

Ex.: Dăm de n ori cu banul. Probabilitatea ca moneda să pice „pajură" este p, altfel $1-p$.

Care este nr. mediu în care avem „pajură".

$$x_i = \begin{cases} 1 & \text{dacă la a } i\text{-a aruncare avem pajură} \\ 0 & \text{altfel} \end{cases}$$

$$x = x_1 + x_2 + \ldots + x_n$$

$$\mathbb{E}[x] = \mathbb{E}[x_1 + x_2 + \ldots + x_n] = \mathbb{E}[x_1] + \mathbb{E}[x_2] + \ldots + \mathbb{E}[x_n] =$$

$$= n \cdot \mathbb{E}[x_i] = n \cdot (1 \cdot p[x_i = 1] + \cancel{0 \cdot p[x_i = 0]}) = n \cdot p$$

Scanned with CamScanner

astfel

$$x_i = \begin{cases} 1 & \text{dacă angajăm candidatul } i \\ 0 & \text{altfel} \end{cases}$$

Vrem să calculăm $x = x_1 + x_2 + \ldots + x_n$

$$E[x] = E\left[\sum_{i=1}^{n} x_i\right] = \sum_{i=1}^{n} E[x_i] = \sum_{i=1}^{n} p[x_i = 1] =$$

$$= 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = O(\log n)$$

Experiment - Birthday Paradox - pers. născute în aceeași zi

$$x_{ij} = \begin{cases} 1 & \text{dacă } i \text{ și } j \text{ sunt născ. în ac. zi} \\ 0 & \text{altfel} \end{cases}$$

$$X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{ij}$$

$$E[x] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[x_{ij}] =$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{365} = \frac{n(n+1)}{2 \cdot 365} = \frac{50 \cdot 51}{2 \cdot 365}$$

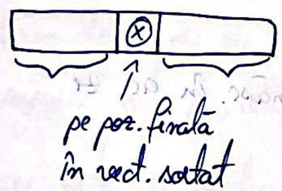
$$P[x_{ij}=1] = \sum_{z=1}^{365} P[\text{studentul } i \text{ să fie născut în ziua } z \text{ și } j \text{ la fel}]$$

$$= 365 \cdot P(\text{studentul } i \text{ să fie născ. în ziua } z) \cdot P(j \text{ să fie născ. în } z)$$

$$= 365 \cdot \frac{1}{365} \cdot \frac{1}{365} = \frac{1}{365}$$

Quicksort: Cap. 5 & 7 Cormen

Ideea de bază:



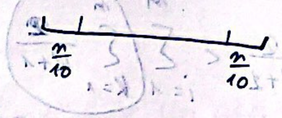
1. Alegem un pivot
2. Apelam funcția de partiție care aranjează pivotul pe poziția finală în vectorul sortat
3. Apelam recursive quicksort pe șirul din stânga, respectiv din dreapta pivotului.

Pivot determinist $T(n) = T(n-1) + O(n) = O(n^2)$

cazul defavorabil

În cazul favorabil: $T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n)$

Ex.:



Dacă am avea $T(n) = T(\frac{n}{10}) + T(\frac{9n}{10}) + O(n) = O(n \log n)$

Și mai rău $T(n) = T(\frac{n}{1000}) + T(\frac{999n}{1000}) + O(n) = O(n \log n)$

Notăm elementele $z_1, z_2, \dots, z_n$ unde

$z_1 \le z_2 \le \dots \le z_n$

$$x_{ij} = \begin{cases} 1 & \text{dacă } z_i \text{ este comparat cu } z_j \\ 0 & \text{altfel} \end{cases}$$

x = nr. de comparații efectuat de algoritm $= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}$

$$E[x] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[x_{ij}] =$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P[x_{ij}=1]$$

$P[x_{ij}=1] = P[z_i$ sau z_j să fie aleși ca pivot din intervalul

$z_i, z_{i+1}, \dots, z_j] = \frac{2}{j-i+1}$

$z_i, z_{i+1}, \dots, z_j$

$$E[x] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n} \sum_{k=1}^{n-i} \frac{2}{k+1} < \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k+1}$$

$$\sum_{k=1}^{n} \frac{2}{k+1} \approx O(\log n)$$

$$= O(n \log n)$$

Curs 4

1. Heap-uri (HeapSort)
2. Arbori binari de căutare

Heap-uri

a) Ce operații suportă?

a.1) Inserția unui element $O(\log n)$

a.2) extragerea minimului/maximului: $O(\log n)$

a.3) returnarea minimului/maximului $O(1)$

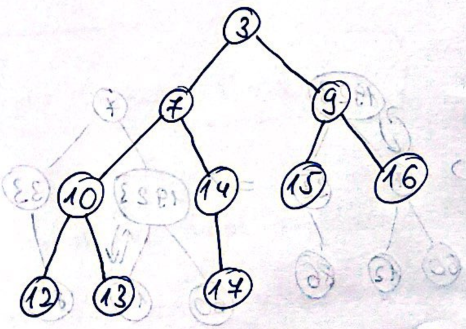
Def.:

Un heap este un arbore binar plin (fiecare nod are 2 fii, cu excepția, posibil, a penultimului nivel) cu proprietatea că valoarea din fiecare nod este mai mare (/mai mică) decât toate valorile din subarbore.

Min-heap => fiecare nod va fi mai mic decât toate nodurile din subarbore.

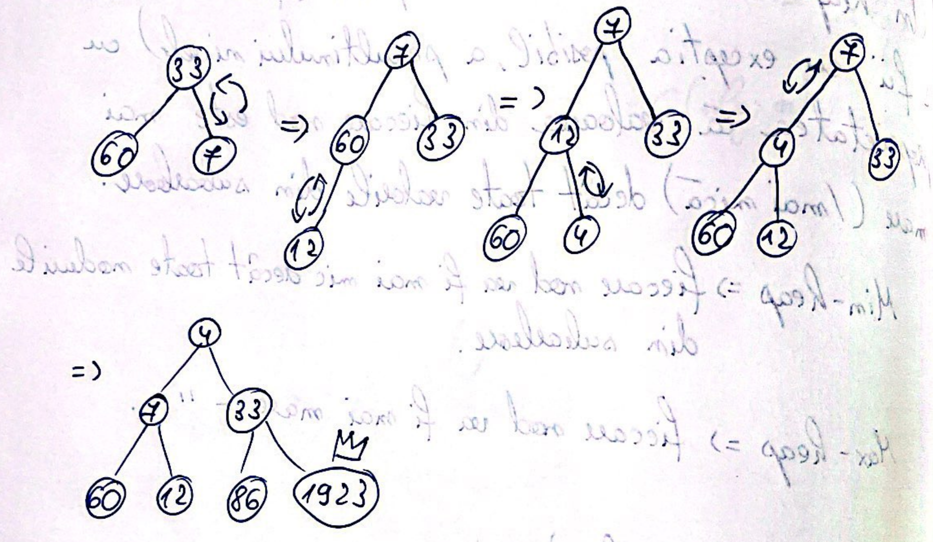
Max-heap => fiecare nod va fi mai mare - " -.

Exemplu: (min-heap)

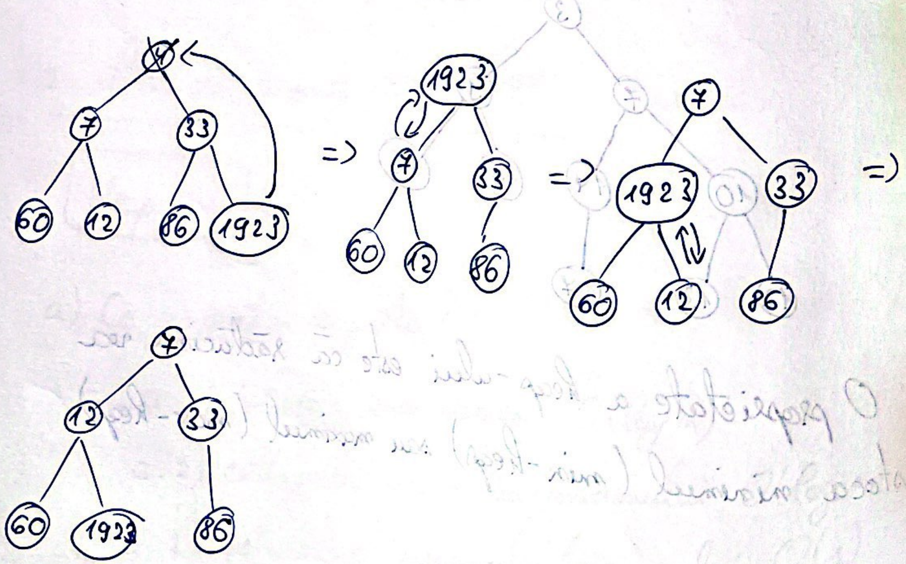


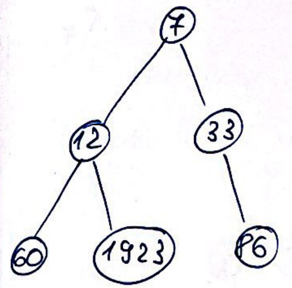
O proprietate a heap-ului este că rădăcina va stoca minimul (min-heap) sau maximul (max-heap).

Exemplu: 33, 60, 7, 12, 4, 86, 1923



Exemplu:





Sub formă de vector

	1	2	3	4	5	6	7	
V	7	12	33	60	1923	86	x	

i → fiu stâng 2 * i
i → fiu drept 2 * i + 1
i → tată [i/2]

Exemplu: 33, 60, 7

1	
33	

1	2
~~33~~	~~60~~

1	2	3	
33	60	7	

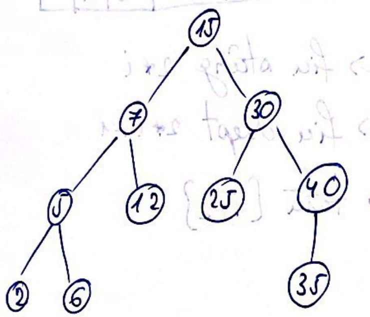
=>

7	60	33	

Arbori binari de căutare

Def.: Un arbore binar de căutare este un arbore binar cu prop. că valoarea asociată fiecărui nod este mai mare decât toate valorile din subarborele stâng și mai mică decât toate valorile din subarborele drept.

Exemplu:

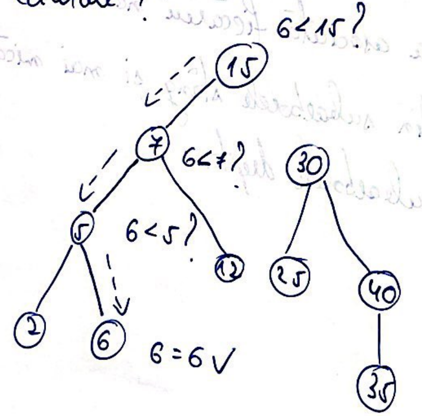


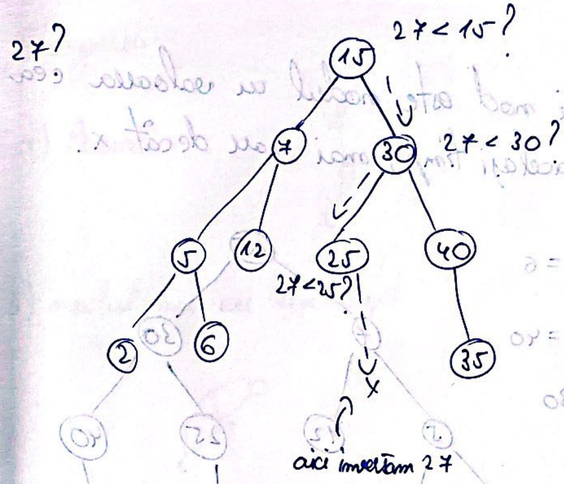
Operații:

1. Inserție $O(n)$
2. Căutare $O(n)$
3. Min/Max $O(n)$
4. Succesor/Predecesor $O(n)$
5. Ștergere

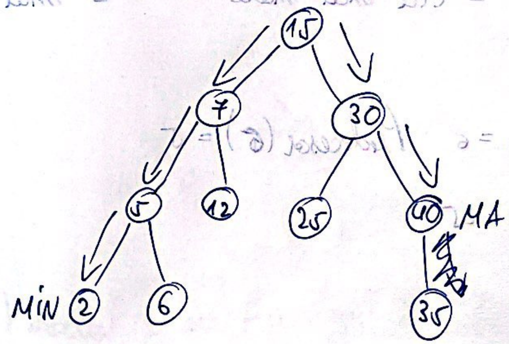
Ex.: Căutare?

6?

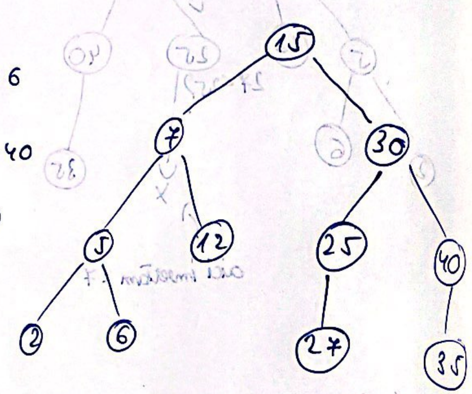




Ex Min/Max



Succesorul unui nod este nodul cu valoarea cea mai mică, dar, în același timp, mai mare decât x.



Ex.: Succesor (5) = 6
Succesor (35) = 40
Succesor (27) = 30

Predecesorul - '' - cea mai mare - '' - mai mică decât x.

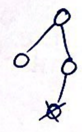
Ex.: Predecesor (7) = 6   Predecesor (6) = 5
Predecesor (27) = 25

Succesor x:

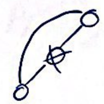
a) Min din subarborele drept dacă x are fiu drept.
b) dacă x nu are fiu drept, cel mai mic strămoș al cărui fiu stâng este tot strămoș al lui x.

Ștergerea:

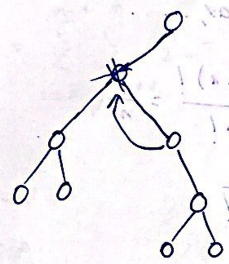
a) frunză



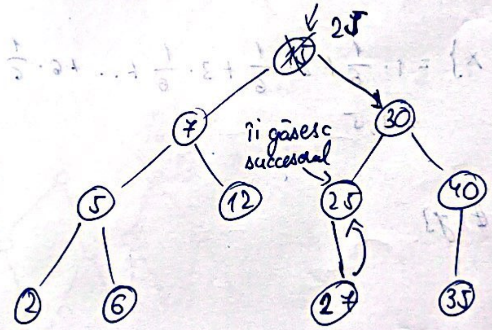
b) nodul nu are fiu drept



c) Vrem să ștergem un nod z care are fiu drept y, dar y nu are fiu stâng



d) nodul z are fiu drept y care are fiu stâng



Sem 3

- Probabilități

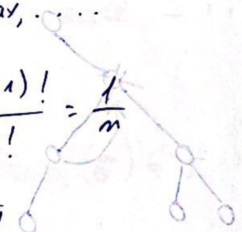
$C_1 \dots C_n$

$C_{(i)} \to f(i) \quad f(i) \neq f(j)$

best = 0 $\quad f_0 = 0$

for $i = 1 \dots n$
  if $f(i) > f$ best
    best = i

max, ...



$\frac{(n-1)!}{n!} = \frac{1}{n}$

$\frac{1}{n!}$

$X = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$

$E[x] = \sum x \, Pr\{x\} = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + \dots + 6 \cdot \frac{1}{6}$

$= 3,5$

$E[x+y] = E[x] + E[y]$

$x_i = \frac{1}{n}$

$E[x_1 + \dots x_n] = E[x_1] + \dots + E[x_n] = n \cdot \frac{1}{n} = 1$

$x_i = \begin{pmatrix} 1 & 0 \\ \frac{1}{n} & 1 - \frac{1}{n} \end{pmatrix}$

$\to P_1 \dots P_n$

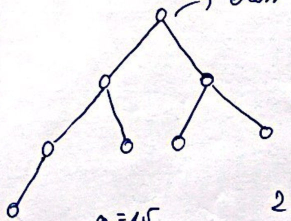
$(i, j) \quad i < j$
$P_i > P_j$

$x_{ij} = \frac{1}{2}$

$E[x] = \left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} \right] = \sum \sum x_{ij} =$

$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} = \frac{n(n-1)}{4}$

- Heap-uri
Max-heap



$\to$ elem max

$i \begin{cases} 2i \\ 2i+1 \end{cases}$

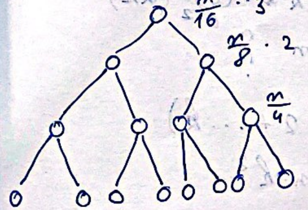
2 9 4 7 1 5

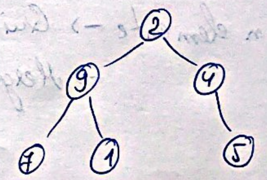
$n = 15$

$\frac{n}{16} \cdot 3$

$\frac{n}{8} \cdot 2$

$\frac{n}{4}$





$$\cancel{\sum_{n=0}^{\infty} n x^n = \frac{x}{(1-x)^2}}$$

$$T(n) = \frac{n}{4}\cdot 1 + \frac{n}{8}\cdot 2 + \frac{n}{16}\cdot 3 + \dots + 1$$

$0 \cdot \frac{n}{2}$

$$\sum_{i=1}^{\log_2(n)} \frac{n}{2^i}(i-1) \le \sum_{i=1}^{\log_2 n} \frac{n}{2^i}\cdot i$$

$$\sum_{n=0}^{\infty} n x^n = \frac{x}{(1-x)^2} \le n\cdot \sum_{i=1}^{\log_2 n} \frac{i}{2^i} \cdot \left(\frac{1}{2}\right)^i$$

$$= n \sum_{i=0}^{\infty} i\left(\frac{1}{2}\right)^i \le n \cdot \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} \le 2n$$

<u>Ex. tip examen:</u>

$$\boxed{\begin{matrix} x_{11} \\ \vdots \\ x_{k1} \end{matrix}} \begin{matrix} \le x_{12} \le \dots \le x_{1n} \\ \ddots \\ \le x_{k2} \le \dots \le x_{kn} \end{matrix}$$

$\hookrightarrow \log_2 k \cdot nk$

$k \cdot n$ elemente $\to$ Quicksort: $nk \cdot \log_2(nk)$

$\to$ Mergesort: $2n$?

$= n(2 + \dots k)$

$= n \cdot k^2$

V: 2 1 7 4 9 3 $\to n \log n$

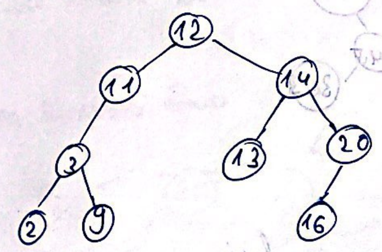
$\hookrightarrow$ Heap: $n \log k + n$

$O(n)$

$$n + \frac{n}{2} + \frac{n}{4} + \dots + \frac{n}{2^k} = n\underbrace{\left(\frac{1}{2} + \dots + \frac{1}{2^k}\right)}_{\le 2}$$

- Arbori binari de căutare

12 11 14 3 9 2



Fie T un arbore binar cât și un nod x cu 2 copii.

Ar. că succ. lui x nu are fiu stâng, iar predecesorul lui nu are fiu drept.

P.p. că $\exists k = succ(x)$ și are fiu stâng (y).

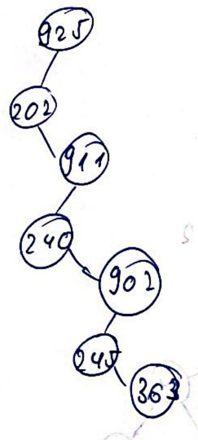
$k > x, \; x < y$

$k > y \quad x < y < k$

Este asta arbore binar bun?

ABC: 1-1000 $\rightarrow$ 363

925, 202, 911, 240, 902, 245, 363



NU

h = înălțimea

$\hookrightarrow 2^{h+1}-1$ = max nr. noduri

$1+\ldots+2^{h-1}+1 = 2^{h}-1+1 = 2^{h}$ nr. min noduri

23, 17, 14, 6, 13, 10, 1, 5, 7, 12 $\Rightarrow$ verif rel. pt. fii

c ..... n^3-1 $\rightarrow$ cum le sortăm $d(n+k)$

~~$x_2 x_1 \ldots 2n \ldots x_3$~~ $X = x_1 + x_2 n + x_3 n^2 \hookrightarrow 3(2n) = 6n$

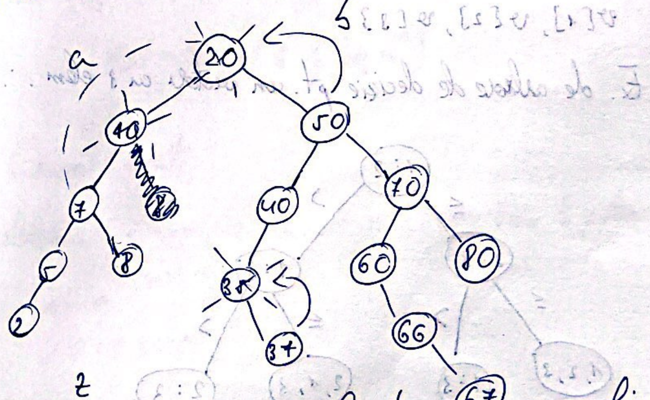
Curs 5

① $\rightarrow$ Recap lecția trecută: - Ștergere în arbori bin. de căutare

② Limite inferioare pt. sortare

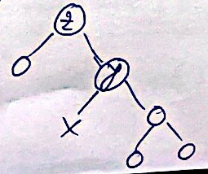
Cap. 8 Corman ③ Count Sort / Radix Sort

1) Ștergerea în arbori binari



a) nodul z pe care dorim să-l ștergem nu are fiu stâng sau fiu drept

b) nodul z pe care dorim să-l ștergem are un fiu drept y, iar y nu are fiu stâng.



c) Nodul z pe care vrem să-l ștergem are un fiu drept y care are fiu stâng

- înlocuiesc z cu succ(z)
- șterg succ de(z)

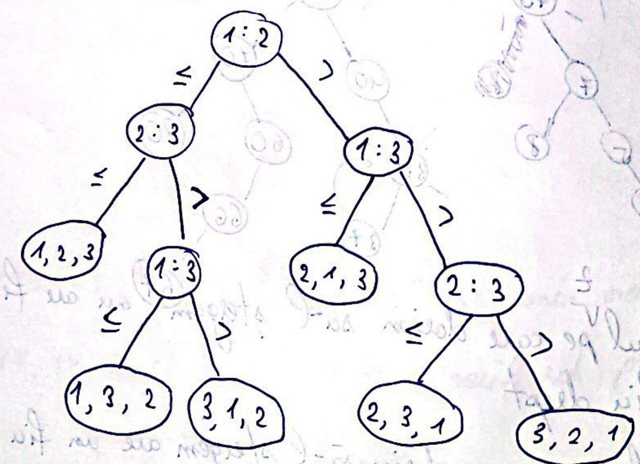
2) Teorema: Orice algoritm de sortare bazat pe comparația între chei efectuează $\Omega(n \log n)$ comparații.

Demonstrație:

1. Un alg. de sortare poate fi modelat ca un arbore de decizie.

$v[1], v[2], v[3]$

Ex. de arbore de decizie pt. un vector cu 3 elem.:



2. Arborele nostru are $n!$ frunze ($3! = 6$)
↳ avem $n!$ frunze => adâncime e $\log n_{frunze}$

3. Adâncimea minimă a unui arbore binar cu $n!$ frunze este $\log_2 n! = \Omega(n \log n)$

4. Timpul de rulare al unui alg. de sortare bazat pe comp. între chei este cel mai lung drum de la rădăcină la frunză în arborele de decizie coresp..

Temă: $\log_2 n! = \Theta(n \log n)$ (Am făcut la sem I, dar s-a înțeles)

$\log_2 n! \in O(n \log n)$

$\log_2 n! \in \Omega(n \log n)$

(Hint: $\log_2 n! \geq \frac{1}{2} \log_2 n^n$)

3) 1. Count Sort

Problemă: Avem n nr. de 0 și 1. Cum le putem sorta?

Ex.: $n = 7$

0100011

Sol.: numărăm câți 0 și câți 1 avem

4 de 0
3 de 1 } → 0000111

Generalizare: Avem n numere din mulțimea $\{0, 1, 2 \dots k-1\}$

Cum le sortăm? $k = 3, n = 9$ $O(n+k)$

0 0 1 2 2 1 0 1 2

0 $\boxed{3}$ 2 $\boxed{3}$

1 $\boxed{3}$

2. Radix Sort

Exemplu:

986
123
514
365
777
793
994
420
101
106

Ana: 10; Mihaela: 10; Ion: 9; Mihai: 7; Alex...

-> Un alg. de sortare stabil:

Alex: 5, Mihai: 7, Ion: 9, Ana: 10, M...

păstrează p...

Pas 1: sortare după cifra unit. ⇐

420
101
123
793
514
994
365
986
106
777

Pas 2: cifra zecilor ⇒

101
106
514
420
123
365
777
986
793
994

Pas 3: cifra sutelor ⇒

101
106
113
365
420
514
777
793
986
994

Timp de rulare Radix Sort:

- n numere:
- fiecare nr. are cel mult $\underline{d}$ cifre
- numerele sunt în baza k

$O(d(n+k))$

exerciții:

Se dau n numere între 0 și n^3-1 (cât de repede le puteți sorta?)

$\hookrightarrow \lg(n^3-1)(n+10) \rightarrow \begin{pmatrix} n=n \\ d=\lg(n^3-1) \\ k=10 \end{pmatrix}$ $\frac{\text{Radix Sort}}{\text{Best}}$ $n\log n$

$\hookrightarrow O(n\log n)$ (Standard)

- transf. nr. în baza n

$\left.\begin{matrix} n=n \\ d=3 \\ k=n \end{matrix}\right\} \rightarrow \begin{matrix} O(d(n+k)) \\ \log_n(n^3) \end{matrix} = O(3(n+n)) = O(n)$

ex.:

$n=100$ $\quad 0, \dots, 999999$

Un nr. în baza 100

200 în baza $100 = 2\cdot 100^1 + 0\cdot 100^0 = 20_{100}$

Sem 4

RMQ

$v = 3\,1\,6\,9\,4\,2\,5\,7$

$|v| = n$

$M(i,j) \rightarrow \min(v_i, v_{i+1}, \dots, v_j)$

$(1,4) \rightarrow 1$

Brute cu for $\rightarrow O(M n)$

$D[i][j] = \min(v_i, v_{i+1}, \dots, v_j)$

```
For i in [1,n]
    x = INF
    For j in [i...n]
        x = min(x, v_j)
        D[i][j] = x
```

$O(n^2 + M)$

$RMQ[\underset{\le \log N}{i}][\underset{\le M}{j}] = \min(v_j, v_{j+1}, \dots v_{j+2^i-1})$

$RMQ[0][j] = v_j$

$RMQ[1][j] = \min(v_j, v_{j+1})$

$RMQ[i][j] = [j, j+2^i-1] = \{j, j+2^{i-1}-1\} \cup \{j+2^{i-1}, j+2^i-1\}$

$\min(RMQ[i-1][j],\ RMQ[i-1][j+2^{i-1}])$

$[3,9] = [3,6] \cup [7,8] \cup [9,9]$

$O(N \log N + M \log N)$

$[3,9] = [3, 3+4-1] \cup [9-4+1, 9]$

$[a,b] = \underbrace{[a, a+2^p-1]}_{RMQ[p][a]} \cup [b-2^p+1, b]$

p max a.î. $2^p \le b-a+1$, adică $p = \log_2(b-a+1)$

Curs 9

$A \in \mathbb{N}^* \Rightarrow \underline{|a_1|a_2|a_3|\dots|a_n|a_{n-1}|}$

$a_i \le a_k$; $i \le K$

index i

$i=0$ $a_0 \leftarrow$ minim

$i=n-1$ $a_{n-1} \leftarrow$ maximul

$V \leftarrow$ rezultat

```
minimum (V):
    min = V[0]
    for i = 1,n-1 :
        if v[i] < min
            min = V[i]
    return min
```

complexitate $O(n)$

maximum (V):

```
minmax_barbar (V):                O(n)
    min = V[0]
    max = V[0]
    for i = 1, n-1
        if v[i] < min :
            min = v[i]
        if v[i] > max :
            max = v[i]
    return (min, max)
```

```
for i = 1,n-1 , i+=2
    if a_i < a_{i+1}
        if a_{i+1} > max
            max = a_{i+1}
    else if a_{i+1} < min
        min = a_{i+1}
```

$\frac{3n}{2} - 2$

$\underline{|\ |\ |p|\ |}$ pivot p $\xrightarrow{\text{PARTITION}}$ $\underline{|<p|p|>p|}$

```
RANDOMIZE-SELECT (A, p, r, i)
    if p == r :
        return A[p]
    q = RANDOMIZED_PARTITION(A, p, r)      -> O(n)
    K = q-p+1
    if i == k
        return A[q]
    else if i < k
        return RANDOMIZED_SELECT (A, p, q-1, i)
```

else
return RANDOMIZED_SELECT $(A, q+1, r, i-k)$

$$T(n) \simeq T(k-1), T(n-k) + O(n)$$
$$\simeq max(T(k-1), T(n-k)) + O(n)$$
$$\simeq T(max\{(k-1), (n-k)\}) + O(n)$$

$P_r(k) = \frac{1}{n}$

$E(T(n))$

$X = v_1, v_2 \dots v_n$

$E[x] = p_1 v_1 + p_2 v_2 + \dots + p_n v_n$

$E(x) = \sum_{i=1}^{n} p_i \cdot v_i$

$$E[T(n)] \le E\left[\sum_{k=1}^{n} p_k T(max(k-1, n-k)) + O(n)\right]$$
$$= E\left[\frac{1}{n}\sum_{k=1}^{n} T(max(k-1, n-k))\right] + O(n)$$
$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n)$$

$E[T(n)] = O(n) \le C \cdot n$

$E[T(n)] \le c n$

$$E[T(n)] \le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-2} ck + an$$

$$\Rightarrow \le \frac{2}{n}\left(\sum_{k=1}^{n-1} ck - \sum_{k=1}^{\lfloor n/2 \rfloor} ck\right) + a_n$$
$$= \frac{2c}{n}\left(\frac{n(n-1)}{2} - \frac{\lfloor n/2 \rfloor(\lfloor n/2 \rfloor - 1)}{2}\right) + a_n$$
$$\le \frac{2c}{n}\left(\frac{n^2-n}{2} - \frac{(n/2-2)(n/2-1)}{2}\right) + a_n$$

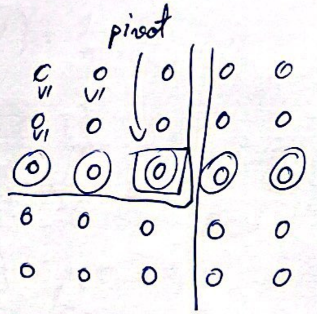
$$E[T(n)] = C_n - \left(\frac{cn}{4} - \frac{c}{2} - a_n\right)$$

$$\frac{cn}{4} - \frac{c}{2} - an > 0$$

~~Mediana~~

Sem I

Mediana $O(n)$



Cel puțin $\frac{n}{5} \cdot \frac{1}{2} \cdot 3$ elem $\leq$ pivot

Cel mult $\frac{7n}{10}$ elemente $\geq$ pivot

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + n$$

$$\underline{T(n) \leq c \cdot n}$$

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + n \leq \frac{cn}{5} + \frac{c7n}{10} + n \leq$$

$$\leq n\left(\frac{c}{5} + \frac{7c}{10} + 1\right) \leq n\left(1 + \frac{9c}{10}\right) \leq cn, \forall c \geq 10$$

$$1 + \frac{9c}{10} \leq c$$

$$1 \leq \frac{c}{10} \Rightarrow 10 \leq c$$

1. Determinăm al $\frac{n}{2}$-lea cel mai mic element în timp liniar din vector.
2. $d_i = |v_i - M|$
3. determinăm al k-lea cel mai mic elem. din $d(x)$
4. Afișăm elem din $d \leq x$

$X: 1\ 2\ \boxed{4}\ 6\ 8\ 10$

$Y: 1\ 3\ \boxed{6}\ 7\ 9\ 10$

$X+Y: 1\ 1\ 2\ 3\ 4\ \underbrace{6\ 6}_{(a,b)}\ 7\ 8\ 9\ 10\ 10$

$$X_{mid} = X_{\frac{n+1}{2}}$$

$$Y_{mid} = Y_{\frac{n+1}{2}}$$

a) $X_{mid} = Y_{mid}$

b) $X_{mid} < Y_{mid}$

$\ldots X_{mid} \ldots (a, b) \ldots Y_{mid}$

c) $X_{mid} > Y_{mid}$

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

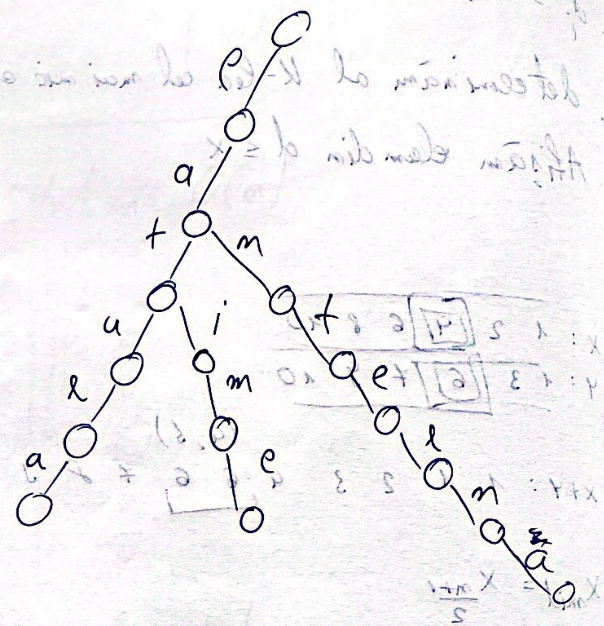
Scanned with CamScanner

TRIE

Catuea
Catime
Canteena

S = Catitudine

n / S /



Curs 10

Algoritmul de găsire a medianei în timp $O(n)$ determinat

3 1 7 4 10 2 5

1 2 3 4 5 7 10

La alg. aleator, în cazul favorabil:
$T(n) = T(n/2) + O(n) = O(n)$

În cazul defavorabil: $k=2$
$T(n) = T(n-1) + O(n) = O(n^2)$

$k = 5$

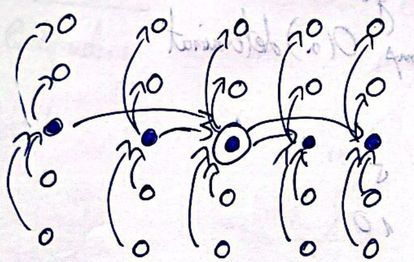
2 1 3 $\boxed{10 \quad 4 \quad 7 \quad 5}$

pivot = 4

4 $\underbrace{10 \quad 7 \quad 5}_{k=1}$

Algoritmul determinist:

1. Împărțim șirul în grupe de câte 5
2. Găsim mediana ~~f~~ din f grupă

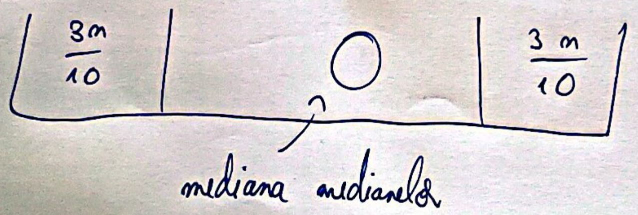


$T(n) = T(n/5) + T\left(\frac{7n}{10}\right) + O(n)$

3. găsim mediana celor n/5 mediane folosind același alg. (rec.)
4. Alegem mediana găsită ca pivot și urmăm aceeași strategie ca la alg. recursiv.

$\boxed{\frac{n}{5} \text{ grupe a câte 5 elemente}}$

$\frac{n}{5} \cdot \frac{1}{2}$ grupe care au câte 3 elem. mai mici decât mediana medianelor $\frac{n}{5} \cdot \frac{1}{2} \cdot 3 = \frac{3n}{10}$



Vrem să arătăm că $T(n) \leq c \cdot n$

P.p. că $T(n/5) \leq \frac{c \cdot n}{5}$

$T\left(\frac{7n}{10}\right) \leq \frac{7n \cdot c}{10}$

$T(n) \leq \frac{{}^{2)}c \cdot n}{5} + \frac{7n \cdot c}{10} + {}^{10)}n = \frac{9cn}{10} + n =$

$= cn - \frac{cn}{10} + n \leq cn$ pt. $c \geq 10$

Pt. 3: $T(n) \leq T(n/3) + T\left(\frac{2n}{3}\right) + O(n) = O(n \log n)$

Pt. 7: $T(n) = T(n/7) + T\left(\frac{5n}{7}\right) + O(n)$

__Problema__: Se dă un text și vrem să îl codificăm (fiecare caracter să fie repr. printr-o secvență de biți) a.î. nr. total de biți să fie minim.

a	b	c	d	e	f	
45	13	12	16	9	5	
000	001	010	011	100	101	
0	101	100	111	1101	1100	← cod de lung. fixă

Cod de lungime variabilă:

$45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4$

$abac = \frac{0}{a} \cdot \frac{101}{b} \cdot \frac{0}{a} \cdot \frac{100}{c}$

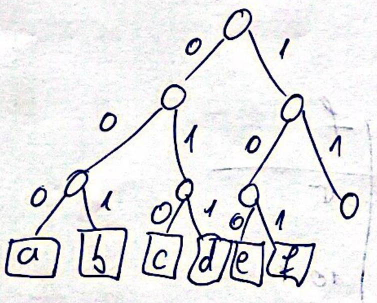
dacă aș fi avut $f = 10$

100 poate fi fa sau c

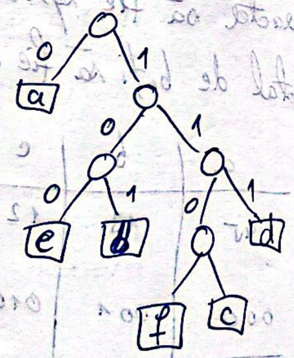
Pt. a putea decodifica, niciun cod nu trb. să fie prefix al altui cod.

Orice cod poate fi repr. ca un arbore binar.

Ex.: cod fix



cod variabil



Dându-se o mulțime de caractere împreună cu frecvențele asociate, vrem să găsim codul optim (lung totală a textului codificat să fie minimă).

Algoritm:

1. Toate caracterele vor fi frunze în arbore.
2. Alegem 2 noduri cu frecvența minimă și adăugăm un nod cu frecvența însumată a celor două noduri. Noul nod va fi părintele nodurilor de frecv. minimă
3. Repetăm pasul 2 cât timp avem ≥ 2 noduri

45a 13b 12c 16d ~~9e~~ ~~5f~~

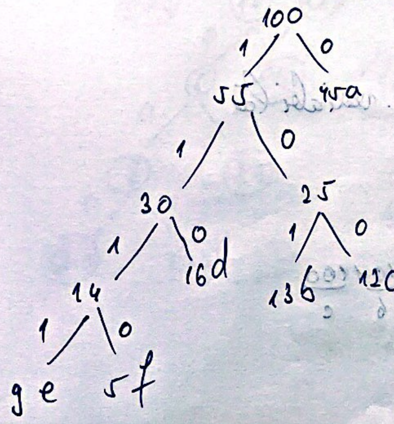
45a ~~13b~~ ~~12c~~ 16d 14

45a 25 ~~16d~~ ~~14~~

45a ~~25~~ ~~30~~

45a 55

$\boxed{100}$



Avem 2n extrageri din min-heap și n inserții. În total avem $O(n \log n)$

Sem 6

Algoritmul lui Huffman
Tabele Hash

$S = $ "aaab"

a = 0
b = 1

$f: V \to \{0,1\}^*$

I. Coduri de lung. fixă

a = 000
b = 001
c = 010
d = 011
e = 100
f = 101

$\underbrace{000}_{a}\underbrace{000}_{a}\underbrace{001}_{b}\underbrace{100}_{e}$

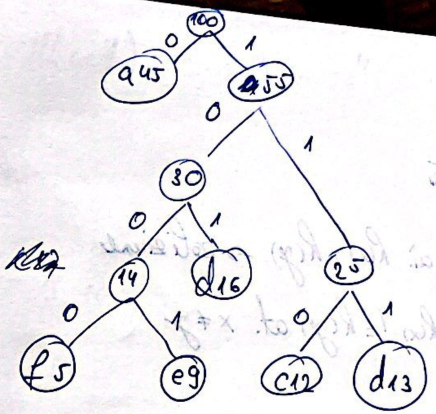
II. Codificări cu lung. variabilă

a = 0
b = 101
c = 100
d = 111
e = 1101
f = 1100

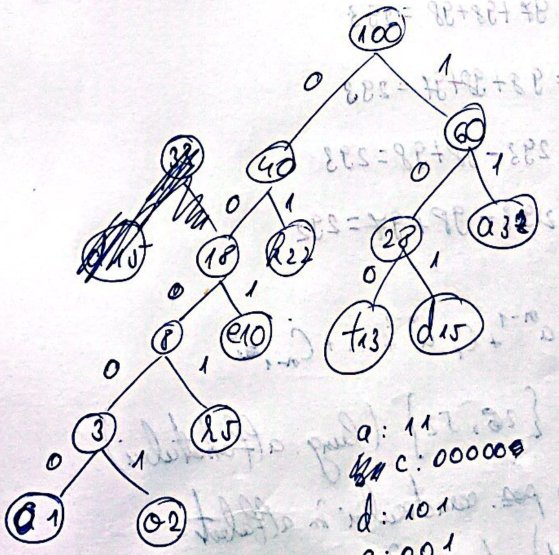
$\underbrace{1101}_{e}\underbrace{0}_{a}\underbrace{101}_{b}\underbrace{100}_{c}$

frecvente:

a = 45
b = 13
c = 12
d = 16
e = 9
f = 5



a 0
b 111
c 110
d 101
e 1001
f 1000



a: 11
c: 000000
d: 1010
e: 001
k: 01
o: 00001
r: 0001
f: 100

$h: A \to B$

B finită

$\exists x, y$ ai. $h(x) = h(y) \to$ coliziune

dacă $h(x) \,!= h(y)$ at. $x \neq y$

T = ~~baa~~ bbabaac

P = bab

brute $(O(|T| \cdot |P|))$

Rabim Karp:

$H(bab) = 97 + 98 + 98 = 293$

$H(bba) = 98 + 98 + 97 = 293$

$H(bab) = 293 - 98 + 98 = 293$

$H(aba) = 293 - 98 + 97 = 292$

$H(C) = C_0 \cdot a^{n-1} + C_1 \cdot a^{n-2} + \dots + C_{n-1} a^0$

$n \in \{26, 52\}$ (lung. alfabetului)

$C_0 \to$ poz. caracterului în alfabet

$H(acd) = (0 \cdot 26^2 + 2 \cdot 26^1 + 3 \cdot 26^0)$

$H(cda) = (2 \cdot 26^2 + 3 \cdot 26^1 + 0 \cdot 26^0)$

$H(cda) = 26 \cdot H(acd) + 0 \cdot 26^0 - 0 \cdot 26^4$

a) $h(x) = x \bmod m$

$m = 5$

12, 11, 7, 14, 5, 9, 100

0	→ 0, 100
1	→ 11
2	→ 12, 7
3	
4	→ 4, 9

b) $h(x) = [M \cdot \{x \cdot A\}]$

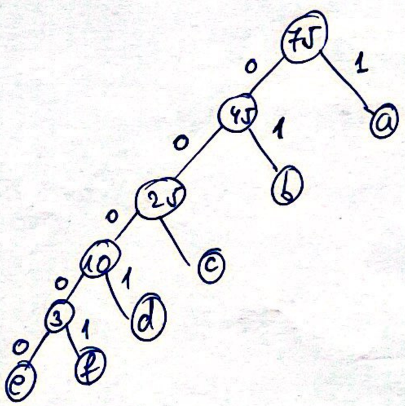
$0 < A < 1$ $\quad M, A$ alese de noi

Curs 11

Codări Huffman

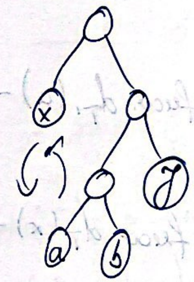
Se dau litere cu frecv. asociate

a	b	c	d	e	f
30	20	15	7	2	1



Lemă: Fie C un set de caractere și $x, y \in C$ două caractere cu frecvența minimă. Atunci există un cod optim în care cuvintele asociate caracterelor x și y au aceeași lungime și diferă doar în ultimul bit.

Dem.: Fie un arbore T corespunzător unui cod optim p.t. C în care x și y nu sunt frunze de adâncime maximă.



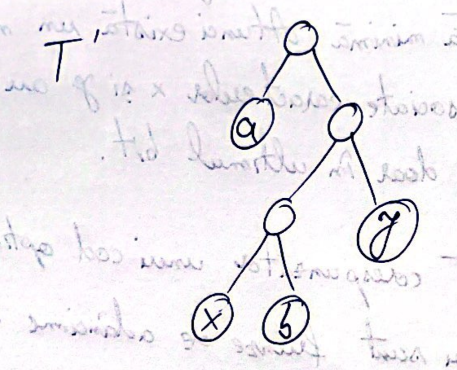
Fie x, y - caracterele cu frecv. minimă

a, b - caracterele coresp. nodurilor de adâncime maximă

a frecv - ~~adâncimea lui x~~ frecv. caract. a

$d_T(x)$ - adâncimea lui x

Ideea principală: dacă $(\exists)$ un arbore T în care x și y nu sunt nodurile de adâncime maximă, pot construi un arbore T' în care x și y sunt noduri de adâncime maximă și $cost(T') \le cost(T)$



$$cost(T) = \sum_{v \in V} v.frecv. \cdot d_T(v)$$

Vrem să arătăm că $cost(T') \le cost(T)$

$cost(T') - cost(T) =$

$= x.\,frecv.\, d_{T'}(x) + a.\,frecv.\, d_{T'}(a) - x.\,frecv.\, d_T(x) -$

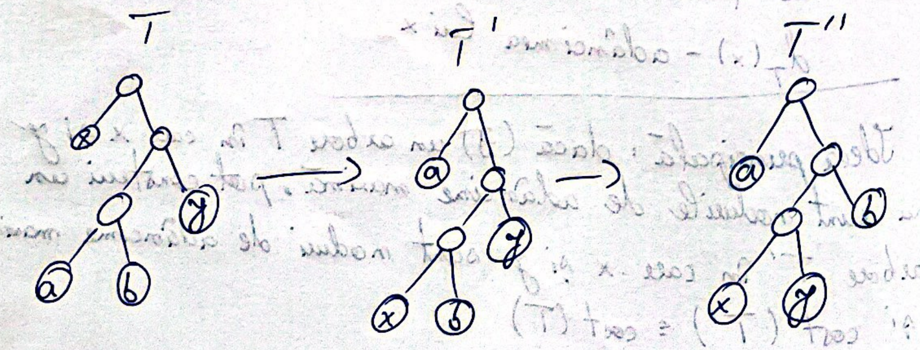
$- a.\,frecv.\, d_T(a)$

$= x.\,frecv.\, d_T(a) + a.\,frecv.\, d_T(x) - x.\,frecv.\, d_T(x)$

$- a.\,frecv.\, d_T(a)$

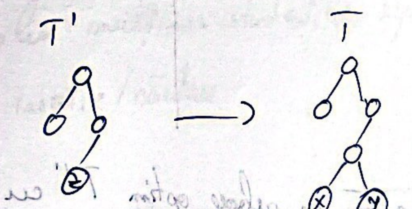
$= x.\,frecv.\,(d_T(a) - d_T(x)) + a.\,frecv.\,(d_T(x) - d_T(a))$

$= \underbrace{(d_T(a) - d_T(x))}_{\ge 0}\,(\underbrace{x.frecv. - a.frecv.}_{\le 0}) \le 0$



Teoremă: Fie c un set de caractere și x și y caracterele de frecvență minimă. Fie $c' = c \setminus \{x, y\} \cup \{z\}$ unde $z.\,frecv. = x.\,frecv. + y.\,frecv.$

Dacă T' e un arbore corespunzător unui cod optim pt. c' atunci arborele T format din T' în care z devine nod intern și x și y sunt fii lui z este arbore optim pt. c.



$cost(T') - cost(T) = z.\,frecv.\, d_{T'}(z) - x.\,frecv.\,(d_{T'}(z)+1)$

$- y.\,frecv.\,(d_{T'}(z)+1)$

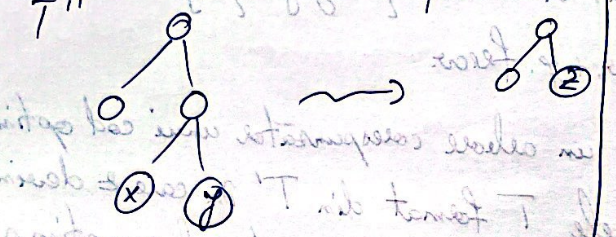
$= z.\,frecv.\, d_{T'}(z) - (d_{T'}(z)+1)(\underbrace{x.\,frecv. + y.\,frecv.}_{= z.\,frecv.})$

$= -(x.\,frecv. + y.\,frecv.)$

$cost(T) = cost(T') + x.\,frecv. + y.\,frecv.$

Presupunem prin R.A. că $(\exists)\, T''$ cu $cost(T'') < cost(T)$.

Atunci, deoarece T'' este un arbore optim, conform lemei de la partea I a cazului x, y (caract. cu frecv. min. vor avea adâncime maximă și vor fi frați.)



Construim un arbore T''' pt. setul C' care are

$cost(T''') = cost(T'') - x.frecv. - y.frecv.$

Dacă prin absurd există un arbore optim T'' cu

$cost(T'') < cost(T) \Rightarrow (\exists)\ T'''$ cu

$cost(T''') = cost(T'') - x.frecv. - y.frecv. \Rightarrow$

$cost(T''') < cost(T')$ (contradicție)

Hashing

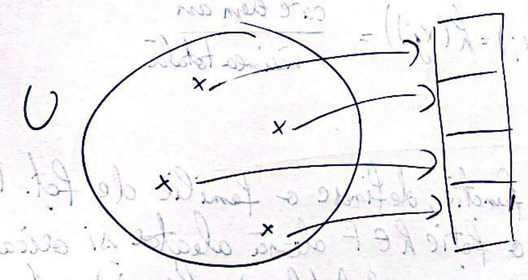
Problemă: Vrem să repr. o mulțime

$M \subseteq U$ folosind cât mai puțină memorie

$|M| << |U|$

Exemplu: mulțimea studenților reprezentanți prin CNP

$O(1)$ inserție/căutare



Avem o funcție $h: U \to \{0, \dots, |M|-1\}$

Exemplu: $h = (3x+2) \bmod 7$

$M = \{200, 13, 2, 30, 5, 9\}$

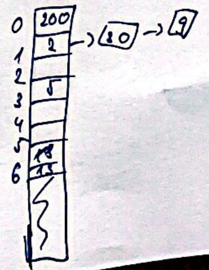
$h(200) = 3 \cdot 200 + 2 = 602 \bmod 4 = 0$

$h(13) = 6$

$h(2) = 1$

$h(5) = 3$

$h(9) = 8$



Dacă mărimea tabelului este m și cardinalul lui M este n cea mai lungă listă are cel puțin $\frac{n}{m}$ elemente.

Cum proiectăm o fcție hash „bună"?

Ce înseamnă o fcție hash „bună"?

Dacă $|\mathcal{U}| = n$ o fct. hash ar fi dorit să respecte $P[h(x_i) = h(x_j)] = \frac{1}{m}$ pt. orice două elem x_i și x_j

În medie lungimea listei i ar fi:

$$\sum_{j \neq i} P_r(h(x_i) = h(x_j)) = \frac{\text{câte elem am}}{\text{mărimea totală}}$$

În loc de o funcție, definesc o familie de fct. F cu prop. că pt. o fcție $h \in F$ aleasă aleator și oricare două elem. x_i și x_j avem $P_r(h(x_i) = h(x_j)) \leq \frac{1}{m}$

Exemplu de familie de fct.: $h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$, unde m e mărimea tabelului și p este un nr. prim $> n$

Sem 7

Recap - Model examen

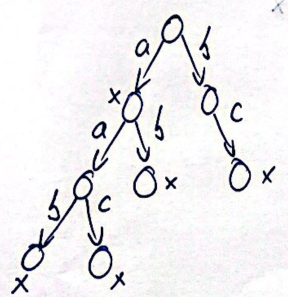
- $S = \{a, ab, bc, aab, aac, bd\}$

$|S| = N$ $\rightarrow a, aab, aac, ab, bc, bd$

$\max_{i=1}^{n} |S_i| = Lg\,Max$ $O(N \log N \cdot Lg\,Max + M)$

$V = \{1, 2, 5\}$ $O(\sum_{i=1}^{m} |S_i| + M)$

$|V| = M$



AHO - CORASICK

• T $\boxed{abaabbabab}$

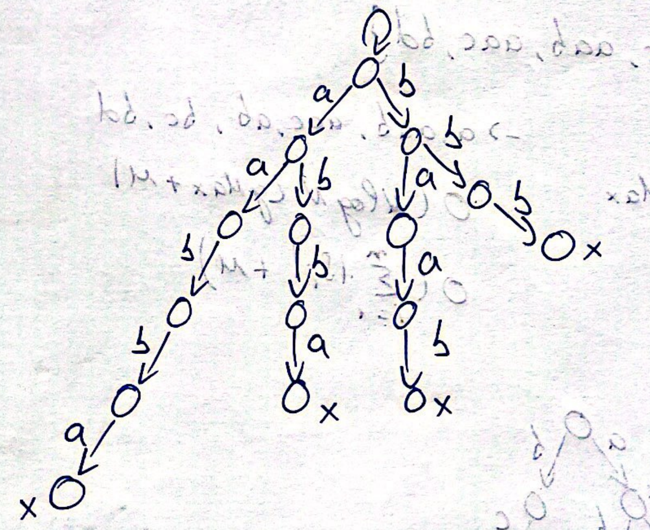
P_1 $\square$ aaba

P_2 $\square$ abba

P_3 $\square$ baab

P_4 $\square$ bbb

1) Suffix links



2) Output links

. . . . .

a) $\lg(n!) \in \Theta(n \log n)$

$\lg(n!) \le \lg(n^n) = n \lg n \Rightarrow \lg(n!) \in O(n \lg n)$

$\lg(n!) = \lg(1 \cdot 2 \cdot \ldots \cdot n) = \lg 1 + \lg 2 + \ldots + \lg n \le n \lg n$

$n! \ge n^{\lfloor \frac{n}{2} \rfloor}$ $(\forall)\, n \ge 5$

$\lg(n!) \ge \lg(n^{\frac{n}{2}}) = \frac{n}{2} \lg n \Rightarrow \lg(n!) \in \Omega(n \lg n)$

b) $(n+2^5)^3 \in \Theta(n^3)$

c) $\lg n^{10} + n^2 \in \Theta(n^2)$

d) $\lg n^5 \in \Theta(\lg n)$

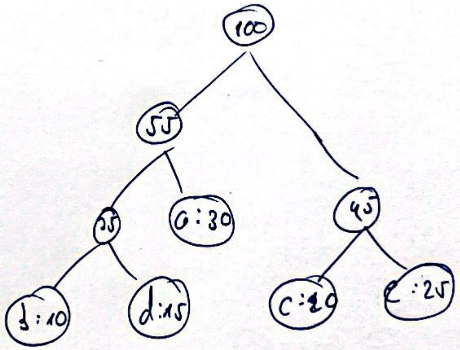
• a : 30

b : 10

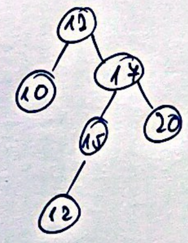
c : 20

d : 15

e : 25



12, 17, 10, 15, 13, 20



- $n^2 \log n \in o(n^3)$

1) $n^2 \log n < c \cdot n^3$ $(\forall)$ $c > 0$

2) $\lim\limits_{n \to \infty} \dfrac{n^2 \log n}{n^3} = \lim\limits_{n \to \infty} \dfrac{\log n}{n} = 0$

$T(n) = 2T(n-1) + 2$

$= 2(T(n-2)+2)+2$

$= 2(T$ [illegible] $\dots$