

Curs 1

Report de curs.

- Cormen → Introduction to algorithms

Nota

$$\frac{2}{3}$$

ex.

$$\frac{1}{3}$$

lab.

Notatii asymptotice

$$\circ, O, \Theta, \Omega, \omega$$

Când proiectăm algoritmi avem în vedere

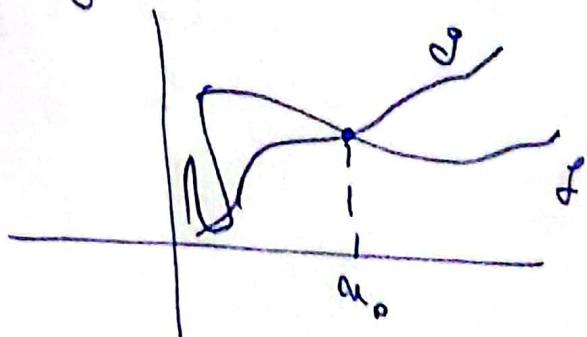
→ eficiență: spațiu de memorie  
                            temp de reacție

→ corectitudine

Temper de reacție = nr. de operații

- → o mic
- → o mare
- Ω → omaga
- ω → omega mic
- Θ → theta

$O(g)$  = (informal) mulțimea funcțiilor care cresc mai  
încet sau la fel decât crește ea funcție  $g$ .



Definiție:

Să punem că o funcție  $f \in O(g)$  dacă și numai dacă există  $m_0, c > 0$  astfel încât pentru  $n \geq m_0$  avem  $f(n) \leq c \cdot g(n)$ .

Exemplu:  $\alpha n^2 \in O(n^2)$

$$f(n) = \alpha n^2$$

$$g(n) = n^2$$

$$m_0 = 2$$

$$c = 1$$

$$\text{dorind: } \alpha n^2 \leq 1 \cdot n^2$$

•  $200n^2 \in O(n^2)$  Adevărat

$$m_0 = 1$$

$$\forall n \geq 1: 200n^2 \leq c \cdot n^2$$

$$c = 200$$

•  $n^2 \notin O(n)$

P.p. red. abs.  $n^2 \in O(n) \Rightarrow \exists m_0, c > 0$  ast.  $n^2 \leq c \cdot n$

$$\forall n \geq m_0.$$

$$\downarrow \\ n < c \text{ nu}$$

pt că  $c$  = constantă.

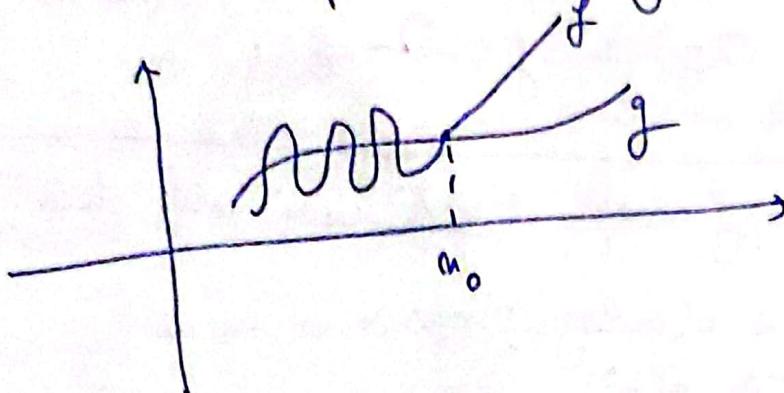
•  $2^{n+1} \in O(2^n)$  ✓

$$c = 2$$

$$m_0 = 1$$

•  $2^{2n} \in O(2^n)$  NU  
Tatăcăseu.

$\Omega$  = (informatică) reprezintă multimea funcțiilor care cresc mai repede decât la fel îde repede ca  $g$ .



Exemplu:  $n^2 \in \Omega(n \log n)$  |  $\frac{n}{\log n} \in \Omega(n)$

Definiție:

Să spunem că  $f \in \Omega(g)$  dacă și numai dacă există  $c > 0$  și  $n_0$  astfel încât pentru  $n > n_0$  avem  $f(n) \geq c \cdot g(n)$ .

$f \in \Theta(g)$  = (informatică) :  $f$  crește la fel de repede ca  $g$ .

Definiție:

Să spunem că  $f \in \Theta(g)$  dacă și numai dacă există  $c_1, c_2 > 0$  și  $n_0$  astfel încât pentru  $n > n_0$  avem:  $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ .

Exemplu:  $200n^2 \notin \Theta(n)$        $n \notin \Theta(n \log n)$

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

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

Teorema:

(fuzionare)

teoreme

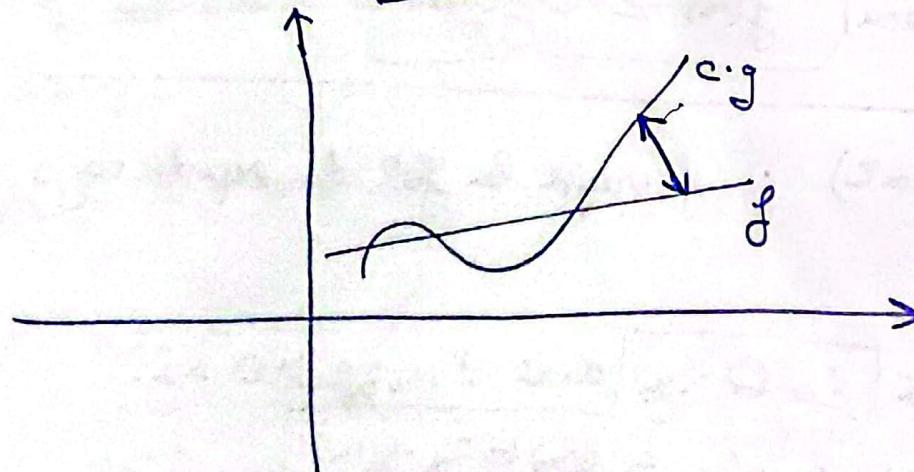
$f \in \Theta(g) \Leftrightarrow f \in O(g)$ , și  $f \in \Omega(g)$ .

$f \in o(g)$  = (informal)  $f$  crește strict mai închiș decât  $g$ .

Exemplu:  $n \in o(n \log n)$   
 $n^2 \in o(n^3)$

$$n \notin o(n) \quad \frac{n}{\infty} \notin o(n)$$
$$n \notin o(100n)$$

Definiție:  $f \in o(g)$  dacă  $\forall c > 0 : \exists n_0 > 0$  astfel încât  $\forall n \geq n_0$  avem  $f(n) < c \cdot g(n)$ .



$f \in w(g)$  = (informal)  $f$  crește strict mai repede decât  $g$ .

Exemplu:  $n^3 \in w(n^2)$

$2^n \in w(n)$

Definiție:  $f \in w(g)$  dacă  $\forall c > 0 : \exists n_0 > 0$  astfel încât  $\forall n \geq n_0$  avem  $f(n) > c \cdot g(n)$ .

### Exercițiu:

$$\rightarrow m \in o(n^2)$$

Fie  $c > 0$  fixat.

Vrem să găsim un  $n_0$  a.i.  $\forall n > n_0$  să avem  $m < c \cdot n^2$ .

\*  $n_0$  depinde de  $c$ ; de exemplu: putem alege  $n_0 = \frac{1}{c} \Rightarrow$

$$\Rightarrow \forall n > \frac{1}{c} \text{ avem } m < c \cdot n^2$$

$\rightarrow n! \in o(n^m)$  (exc.) + apărăm log înmulțirea părții =)

$$\log n! \in \Theta(n \log n) \text{ (exc.)}$$

Seminar +

I Căutare secvențială

II Căutare binară:

g<sub>0</sub>

3 5 7 8 10  
0 1 2 3 4

x=13  
13 19 20 30 55  
5 6 7 8 9

$$(g+0)/2 \rightarrow h$$

$$10 = 13 ?$$
$$13 > 10$$

$$(5+g)/2 \rightarrow 7$$

$$20 = 13 ?$$
$$13 < 20$$

$$(5+4)/2 \rightarrow 5$$

$$13 = 13 ?$$

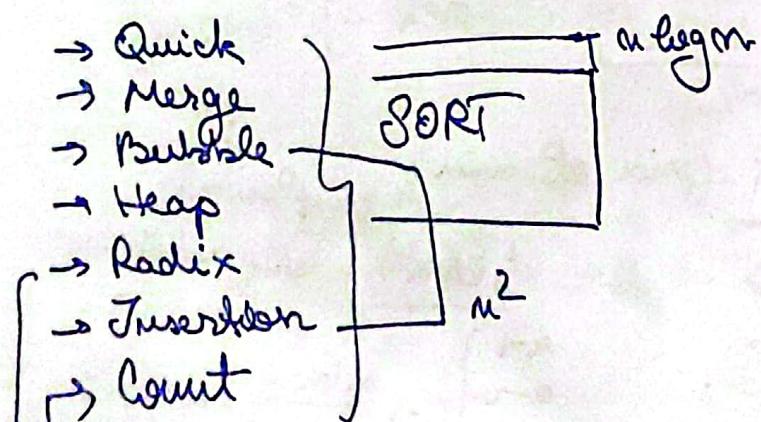
DA

23.02.2022.

## Curs 2

Analiza timpului de lucru a algoritmilor  
recursivi.

Exemplu: metode de sortare



aceste sunt bazati pe comparatii intre chei

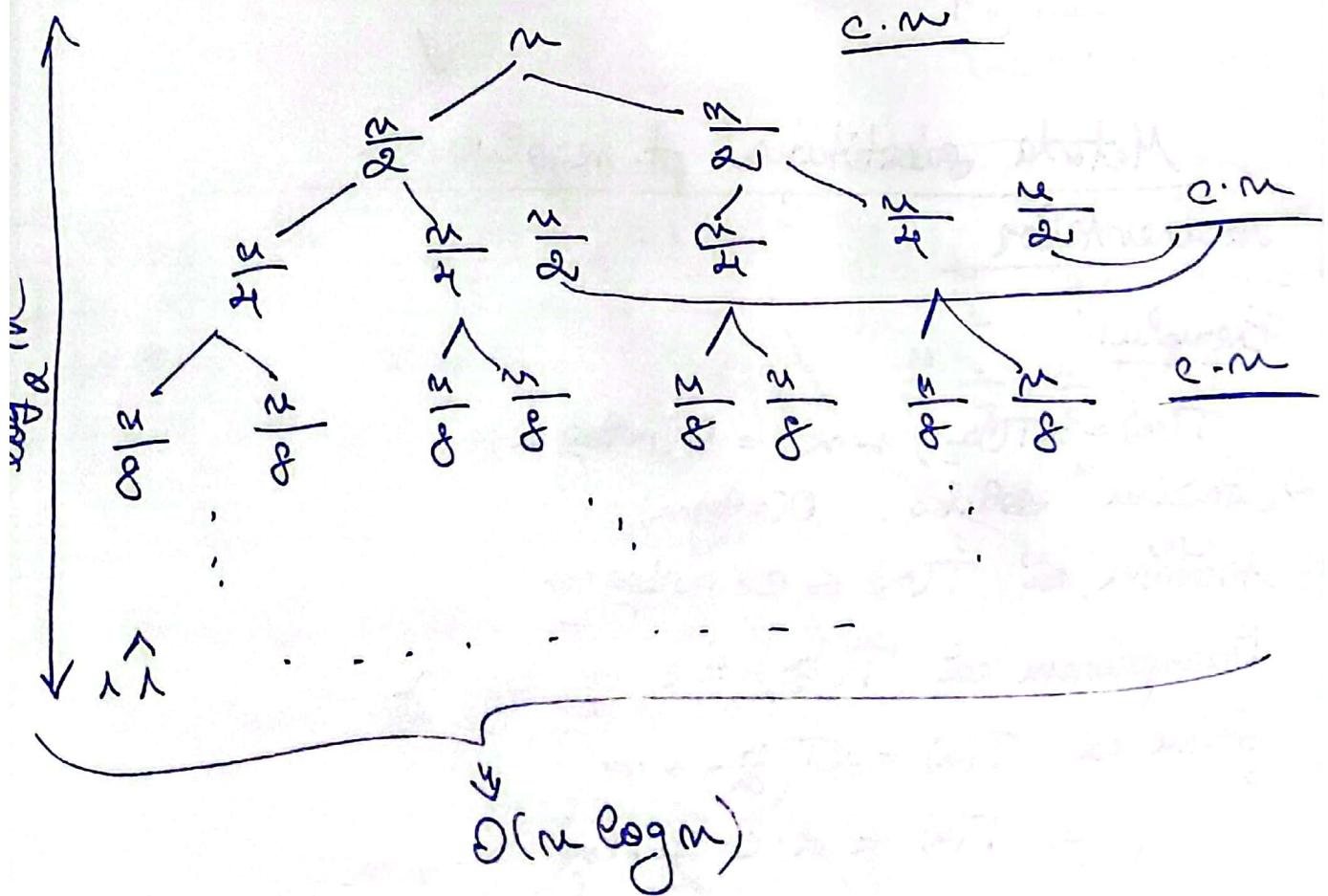
Insertion sort recursive

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

$$\begin{aligned}T(n) &= T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) \\&= \dots + T\left(\frac{n}{3^k}\right)\end{aligned}$$

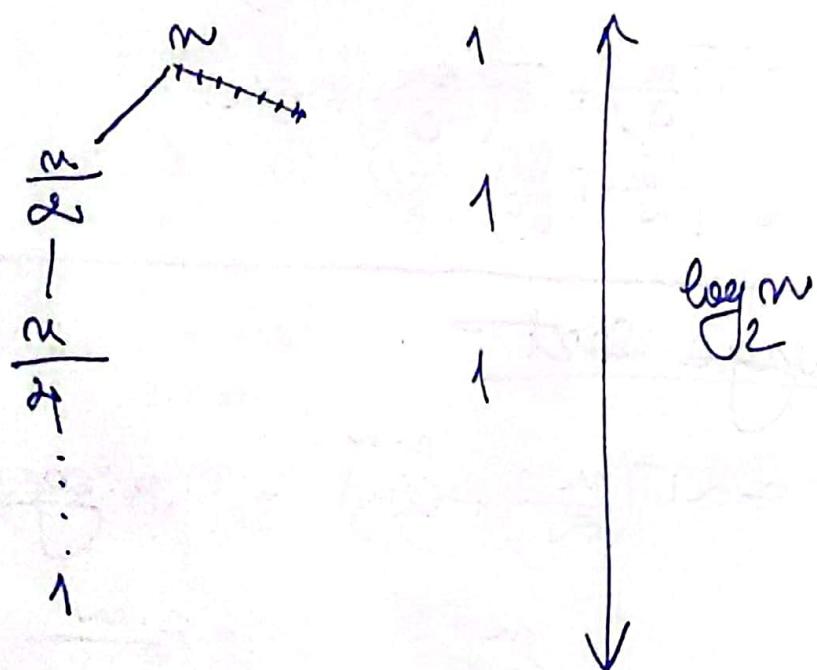
### Merge Sort

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



## Căutarea binară:

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



## Metoda substituției pt. rezolvarea

recursivelor

Exemplu:

$$T(n) = dT\left(\frac{n}{d}\right) + n = O(n \log n)$$

1) „Ghicim” soluție.  $O(n \log n)$

2) Arătăm că  $T(n) \leq c \cdot n \log_2 n$

Presupunem că  $T\left(\frac{n}{d}\right) \leq c \cdot \frac{n}{d} \cdot \log_2 \frac{n}{d}$

Stim că  $T(n) = dT\left(\frac{n}{d}\right) + n$

$$\Rightarrow T(n) \leq d \cdot c \cdot \frac{n}{d} \cdot \log_2 \frac{n}{d} + n$$

$$\Rightarrow T(n) \leq c \cdot n \log_2 \frac{n}{d} + n$$

$$\Leftrightarrow T(n) \leq c \cdot n \cdot (\log_2 n - 1) + n$$

$$\Leftrightarrow T(n) \leq c \cdot n \cdot \log_2 n - c \cdot n + n$$

$$\Leftrightarrow T(n) \leq cn \log_2 n + \underbrace{c(n-c)}_{\text{daca } 1-c \leq 0}$$

Vremă  $T(n) \leq cn \log_2 n$  dacă  $1-c \leq 0$   
Adică pt.  $c \geq 1$

---

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

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

$$\text{Presupunem că } T\left(\frac{n}{2}\right) \leq c \cdot \log_2 \frac{n}{2}$$

~~$\frac{n}{2} \leq \frac{n}{2}$~~   $\Rightarrow T(n) \leq c \cdot \log_2 \frac{n}{2} + 1$

$$\Leftrightarrow T(n) \leq c \log_2 n - c + 1$$

$$\Leftrightarrow T(n) \leq c \log_2 n + (1-c)$$

dacă

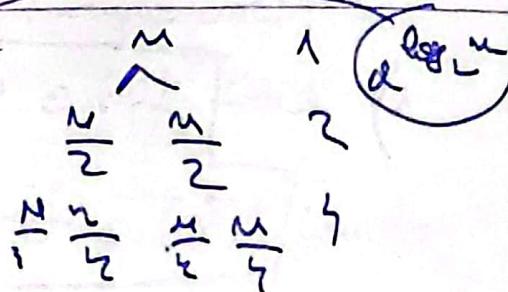
$$1-c \leq 0 \Rightarrow c \geq 1$$


---

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

$$1+2+4+\dots = O(n)$$

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



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

$$\text{Presupunem că } T\left(\frac{n}{2}\right) \leq c \frac{n}{2}$$

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

$$\Leftrightarrow T(n) \leq cn + 1$$


---

!

Vrem să arătăm că  $T(n) \leq cn - d$

$$\text{Presupunem că } T\left(\frac{n}{2}\right) \leq c \frac{n}{2} - d \quad O(n-1)$$

$$\Rightarrow T(n) \leq c \cdot \frac{n}{2} - d + 1$$

$$\Rightarrow T(n) \leq cn - 2d + 1$$

$$\Leftrightarrow T(n) \leq cn + (\underbrace{1-d}_{\leq 0}) - d$$

$\Rightarrow d \geq 1$

$$\Rightarrow T(n) \leq cn - d + (1-d)$$

## Teorema Master

Recurșiuni recurrente de formă:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

Trei cazuri:

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

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

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

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

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

si  $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$  pt.  $c \geq 0$  atunci

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

Example:

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

$$f(n) = 1$$

(case 2)

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

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

$$T(n) = 9T\left(\frac{n}{3}\right) + n \quad a=9, b=3$$

$$\Rightarrow f(n) = n$$

$$n^{\log_b a} = n^{\log_3 9} = n^2 \quad (n^{\log_b a - \epsilon})$$

$$\Rightarrow f(n) \in O(n^{2-\epsilon}) \text{ pt. } \epsilon > 0$$

(case 1)

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

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

$$O(n \log n)$$

use pot optiver  
formule Master

Curs 3

Algoritmi probabilisti:

Analiza QuickSort:

Problema: | Secretary Problem | Strategie 1

Tine se candidati la un job pe care îi intervieneam pe rând. ~~scrisi nu amea control arepră~~  
 Iată că  $c_i$  (candidatul  $i$ ) este mai bun decât toti candidatii anteriori atunci îl o angajăm.  
 și plătirea său cost  $\times$  (generat de concecherea  
 celui anterior).

Căti candidati vorne angaja?

\* În cazul cel mai nefavorabil vorne angaja  
 ai candidati (and sunt sortati crescator).

Strategia 2: permutăm aleator candidati și aplicăm algoritmul anterior.  
 Care este nr. median de candidati angajați?

Vom considera variabila aleatoare cu 2 valori 0 și 1.

Exemplu: dat cu banuri

$$X = \begin{cases} 1, & \text{dacă pică „pejitură”} \\ 0, & \text{altfel} \end{cases}$$

$$P[X=1] = \frac{1}{2} = P[X=0] \quad \text{entropie}$$

Media unei variabile aleatoare:  $E[X] = 1 \cdot P[X=1] + 0 \cdot P[X=0]$   
expectation -  $P[X=1]$

Ex.: Băne de  $n$  ori cu banuri. Probabilitatea ca moneda să pică pejitură este  $p$ , altfel  $1-p$ .

Care este nr. mediu în care avem pejitură?

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

$$X = X_1 + X_2 + \dots + X_n$$

$$\begin{aligned} E[X] &= E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n] \\ &= n \cdot E[X_i] = \underbrace{\left(1 \cdot P[X_i=1] + 0 \cdot P[X_i=0]\right)}_{= p} = \underbrace{n \cdot p}_{= m} \end{aligned}$$

si la probabilitatea de pejitură  
aleatorii sunt independente

$$X_i = \begin{cases} 1, & \text{dacă alegem candidatul } i \\ 0, & \text{altfel.} \end{cases}$$

Vrem să calculăm  $X = X_1 + X_2 + \dots + X_m$ .

$$\begin{aligned} E[X] &= \cancel{\sum} E\left[\sum_{i=1}^m X_i\right] = \sum_{i=1}^m E[X_i] \\ &= \sum_{i=1}^m P[X_i=1] = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{m} = O(\log n) \end{aligned}$$

$$X_{ij} = \begin{cases} 1, & \text{daca studentul } i \text{ si studentul } j \text{ sunt} \\ & \text{nascuti in acelasi zi.} \\ 0, & \text{altfel} \end{cases}$$

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

$$\mathbb{E}[X] = \mathbb{E} \left[ \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij} \right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \mathbb{E}[X_{ij}] = !$$

$$\Pr[X_{ij} = 1] = \sum_{z=1}^{365} \Pr[\text{st. } i \text{ si st. } j \text{ sunt nascuti in zilele } z] \\ = 365 \cdot \Pr[\text{st. } i \text{ sunt nascut in zilele } z] \cdot \Pr[\text{st. } j \text{ sunt nascut in zilele } z]$$

(Birthday paradox)

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

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

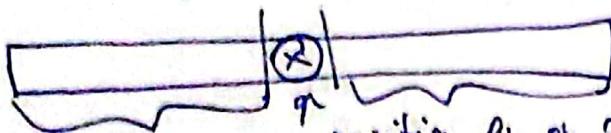
$$\text{pt. } \frac{50 \cdot 51}{2} \cdot \frac{1}{365} > 1$$

= in medie 4 perechi

## Capitolul 5 & 7 (Cormen)

## QuickSort:

Ideeă de bază:



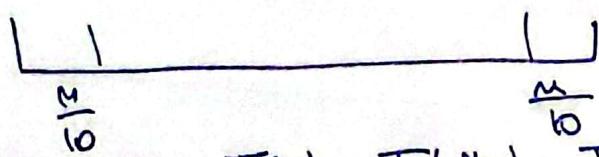
pe poziția finală în vectorul sortat

- I. Alegem un pivot.
  - II. Apelăm funcția de partitie care alege pivotul pe poziția finală în vectorul sortat.
  - III. Apelăm recursiv QuickSort pe simbolul din stânga, respectiv pe din dreapta pivotului.
- 

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

casă de favorabil

în casă de favorabil:  $T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n)$



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

similar  $T(n) = T\left(\frac{n}{1000}\right) + T\left(\frac{999n}{1000}\right) + O(n)$   
 $= O(n \log n)$

Notăm elementele  $z_1, z_2, \dots, z_n$ , unde  $z_i \leq z_{i+1} \leq \dots \leq z_n$

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

$X$  = nr. de comparații efectuate de algoritmul.

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

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

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

unde  $\Pr[x_{ij} = 1] = \Pr\{\text{ti sunt } z_j \text{ să fie alea ca pivot din } z_i, z_{i+1}, \dots, z_j \text{ intervalul } z_i, z_{i+1}, \dots, z_j\}$

$$= \frac{2}{j-i+1}$$

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

$$k = j-i$$

$$k \neq i$$

$$= O(n \cdot \log m) \quad \leftarrow \cdot m$$

$O(\log n)$

```

for(i=n/2-1; i>=0; i--) {
    // int.
    // heapify(2i+1 sau 2i+2)
}

```

$$\downarrow (n/2) \cdot \log n$$

decapitare  $\frac{n \cdot \log n}{3n/2} +$

$$(3n/2) \log n \Rightarrow O(n \log n)$$

09.03.2012.

## Curs 14

1) Heap-uri (HeapSort)

2) Arborei binari de căutare:

→ a) Ce operări suportă?

- a1) Însertia unui element  $O(\log n)$
- a2) Extragerea minimului / maximului  $O(\log n)$
- a3) Determinarea minimului / maximului  $O(1)$

Un heap este un arbore binar plin ( fiecare nod are alături, cu excepția, posibil, a penultimului nivel)

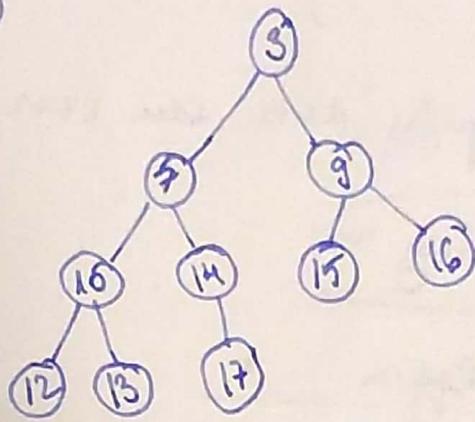
cu prop. că valoarea din fiecare nod este mai mare decât toate valoările din subarbore. (mici)

Min-Heap  $\Rightarrow$  fiecare nod va fi mai mic decât toate medurile din subarbore.

Max-Heap  $\Rightarrow$  — — — —

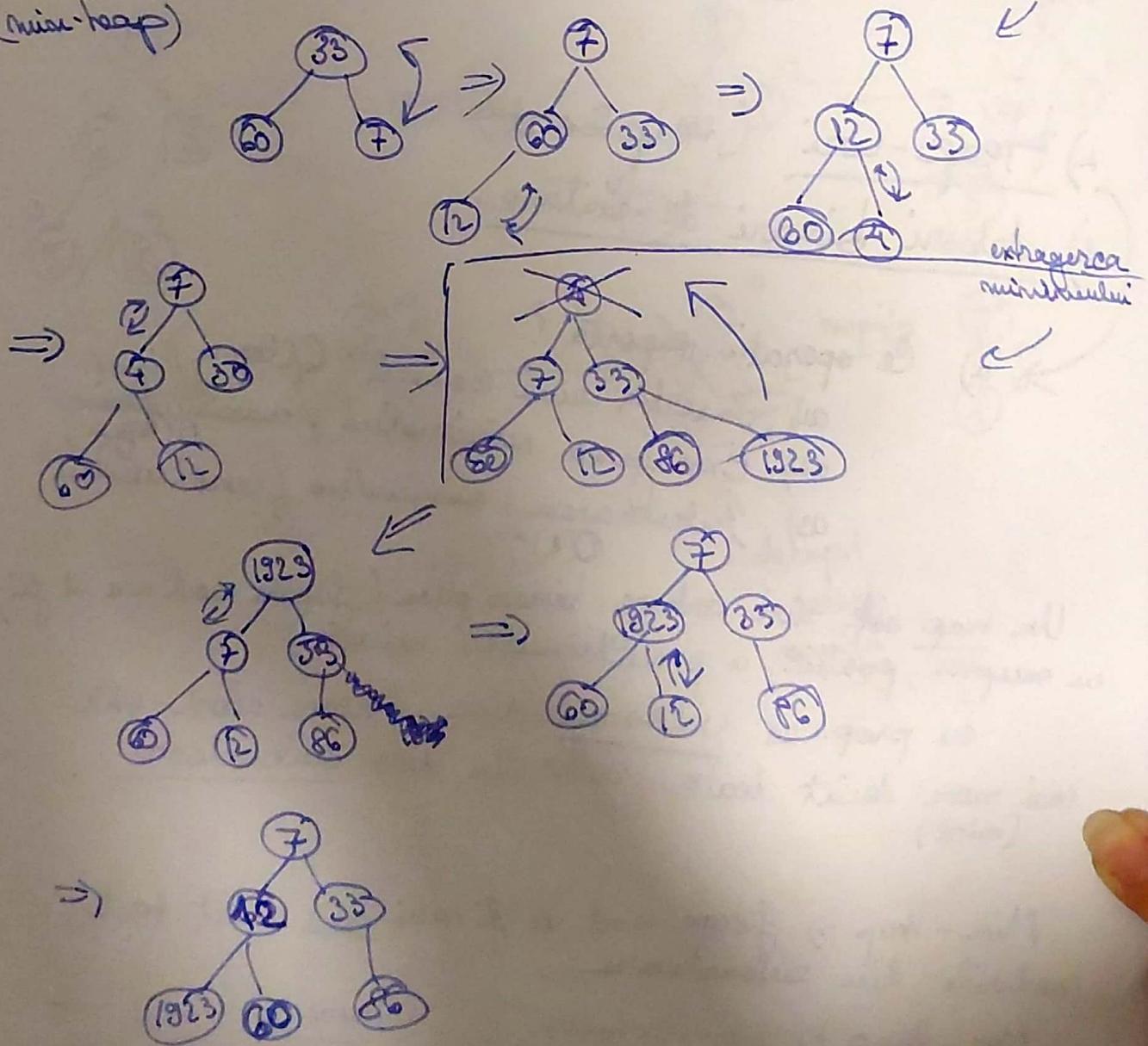
mare — — —

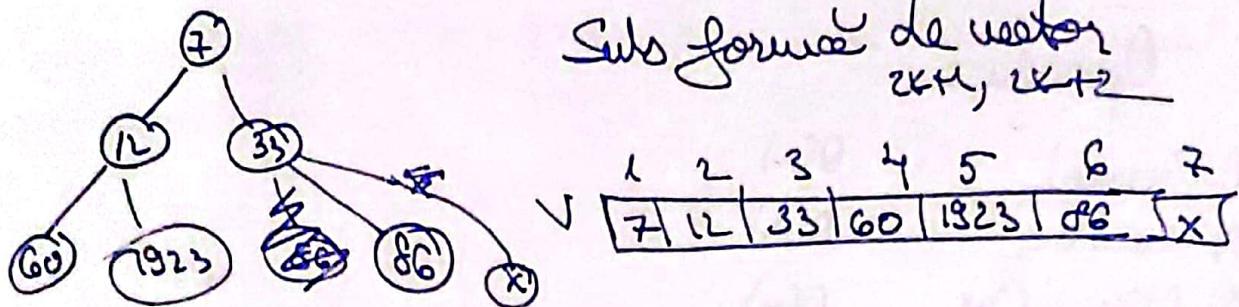
Exemplu: (min-heap)



- O proprietate a heap-ului este că rădăcina va stoca minimum în Min-Heap și maximum în Max-Heap.

Exemplu: 33, 60, 7, 12, 4, 86, 1923 insertie  
un element  
(min-heap)

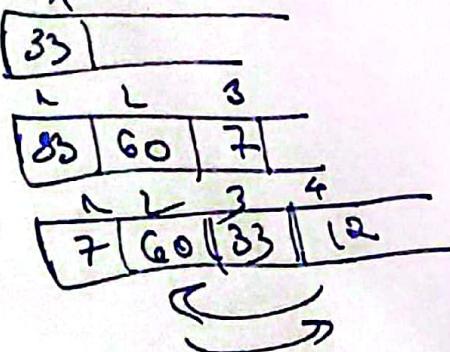




notă:  $i \rightarrow$

- ↑ fin stâng =  $2i$
- ↓ fin drept =  $2i + 1$
- data  $\{i/2\}$

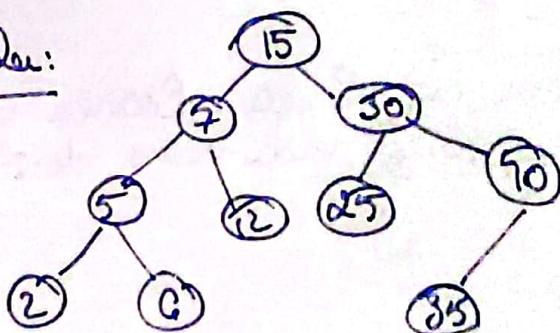
Exemplu: 33 60 7 12 1923 86



Arbore binar de căutare:

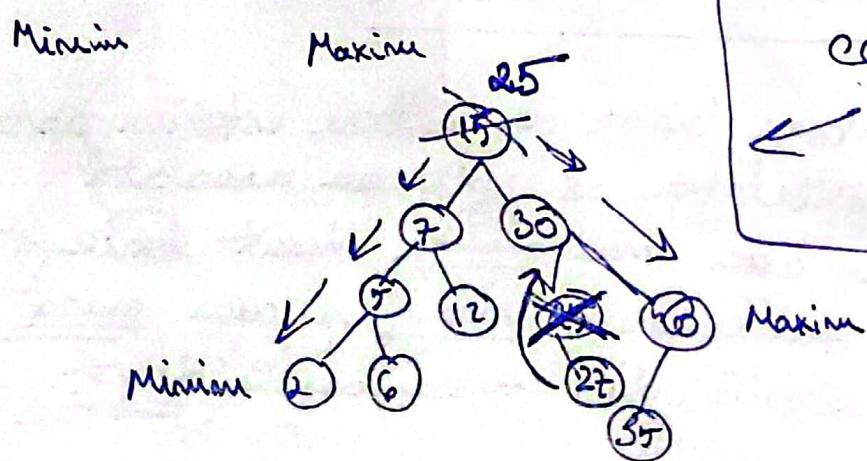
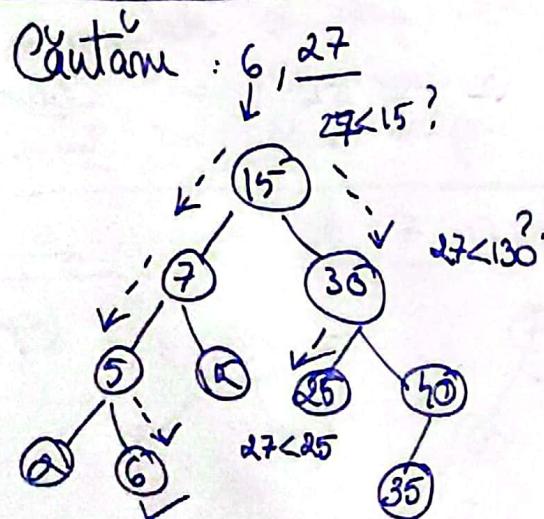
Def.: Un arbore binar de căutare este un arbore binar cu proprietatea că valoarea asociată fiecărui nod este mai mare decât toate valorile din subarborele stâng și mai mică decât toate valorile din subarborele drept.

Exemplu:



## Operări:

- 1) Insertie  $O(n)$
- 2) Căutare  $O(n)$
- 3) Min / Max  $O(n)$
- 4) Succesor / Predecesor  $O(n)$
- 5) Stergere



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

$$\text{succ}(5) = 6$$

$$\text{succ}(35) = 40$$

$$\text{succ}(27) = 30$$

Predecesor : — 11 — val. cea mai mare, mai mică decât x.

$$\text{Pred}(7) = 6$$

$$\text{Pred}(6) = 5$$

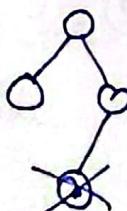
$$\text{Pred}(27) = 25$$

$\text{succ}(x) = \begin{cases} \text{mai din subarborele drept, dc. } x \text{ are fiu drept.} \\ \text{dc. } x \text{ nu are fiu drept., cel mai mic strămoș al căruia fiu stâng este tot strămoș,} \\ \text{al lui } > x. \end{cases}$

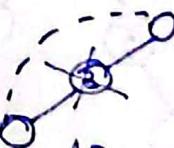
$$\text{Pred}(x)$$

Stergere:

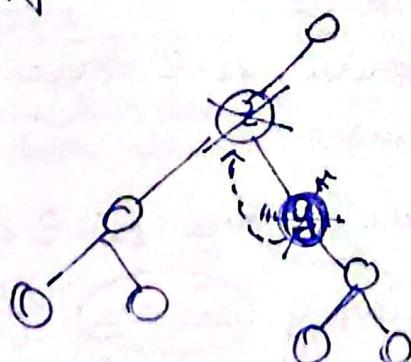
a) ~~Erase~~



b) Nodul care are fiu drept.



c) În se stergere următoare care  
are fiu drept, dar fiu drept nu are fiu stâng.



d) Nodul z care este fiu drept y care are fiu stang.

16.03.2022.

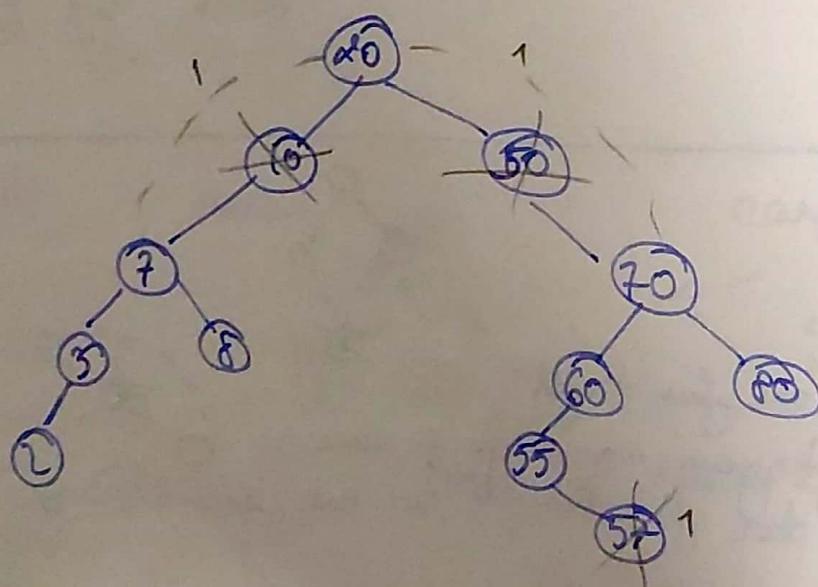
## Cursul 5

1) Recuperare lista trunchiată - stergere în  
arbori binari de căutare

2) Limite inferioare pt. sortare:

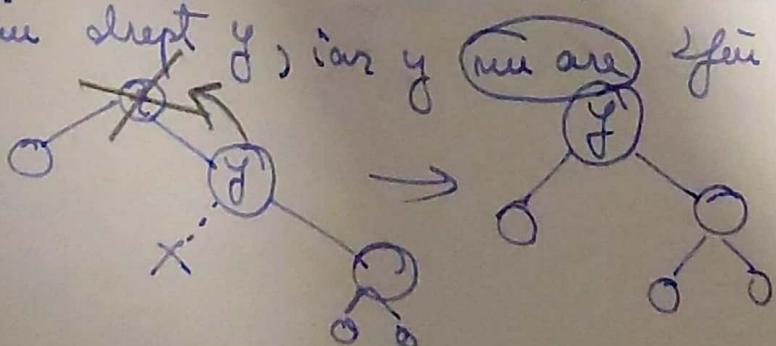
3) Count-Sort ; Radix-Sort

1)

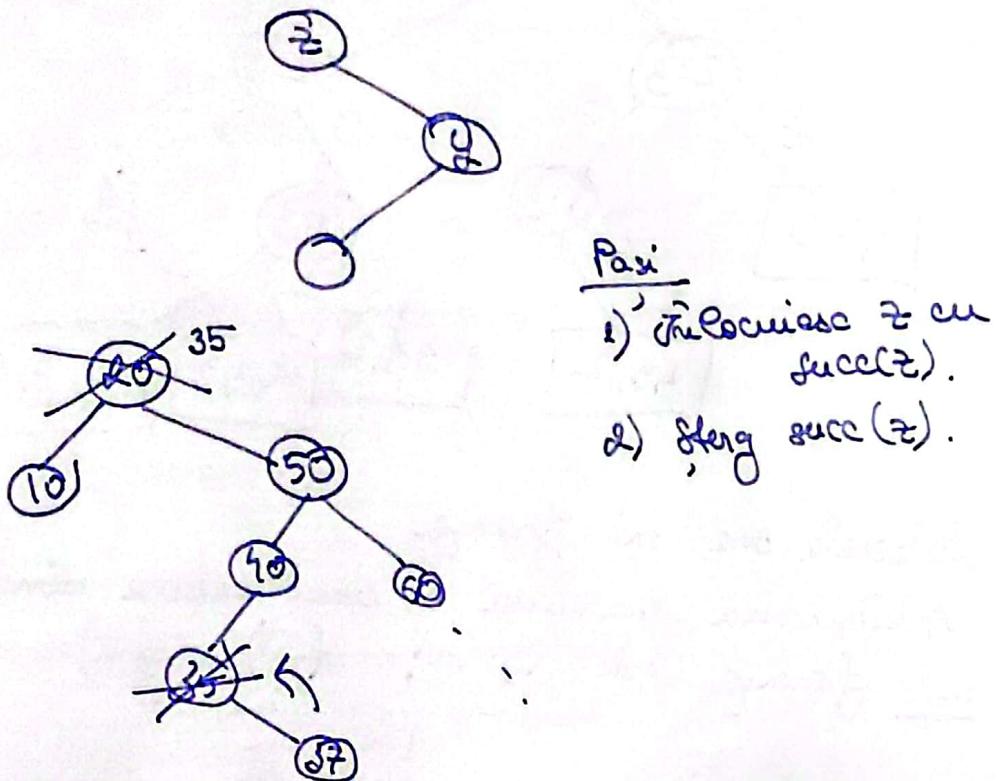


Caz 1: Nodul z pe care elorin să-l stergem nu are  
fiu stang sau fiu drept.

Caz 2: Nodul z pe care elorin să-l stergem  
are un fiu drept y, iar y nu are fiu stang.



Caz. 3: Nodul z pe care doarun să-l stergem are un fiu drept y, iar y are un fiu stâng.



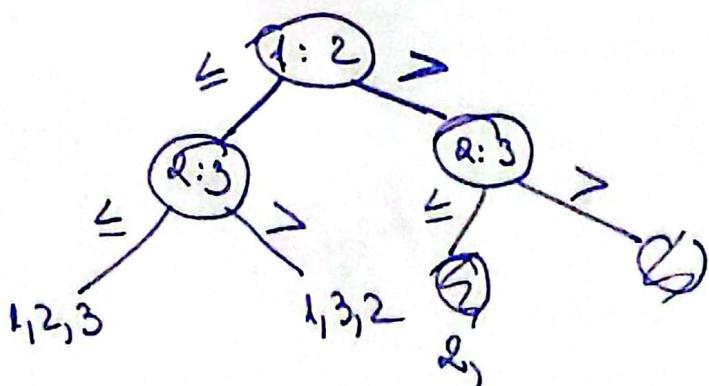
Evidență:

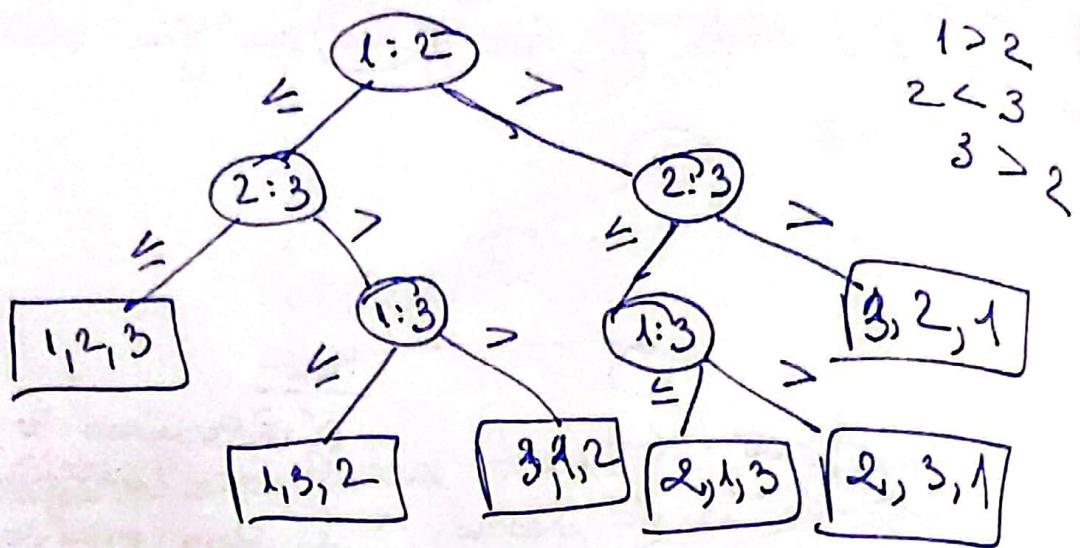
2) Teorema: Orice algoritm de sortare bazat pe comparații într-o cheie efectuează  $\Omega(n \log n)$  comparații.

Demonstratie:

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

Ex de arbore de decizie pt. un vector cu 3 elemente:  $V[1], V[2], V[3]$





- 2) Arboarele are  $m!$  frunze.
- 3) Adâncimea minimă a unei arboare binar este  $\log m! = \Theta(n \log n)$
- $\begin{matrix} 1 \cdot 2 \cdots n \\ n \cdot n \cdots n \end{matrix}$

4) Timpul de reținere al unui algoritm bazat de sortare bazat pe cheie este cel mai lung drum de la rădăcina la o frundă în arbore de decizie corespunzător.

$$n - l$$

Teme  
 $\log_2 m! = \Theta(n \log n)$

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

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

$$\log_2 m! \geq \frac{1}{2} \cdot \log_{10} m^n$$

## Count Sort:

Problema: Avem  $n$  nr. de 0 si 1.

Cum le putem sorta?

$n = 7; 0100011$

Solutie: numărăm căsi de 0 și căsi de 1 avem.

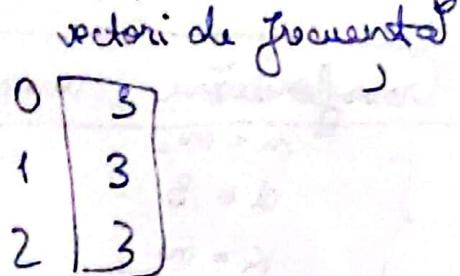
4 de 0     $\Rightarrow 0000111$   
3 de 1

Generalizare: Avem  $n$  numere din multimea  $\{0, 1, 2, \dots, k-1\}$ . Cum le sortăm?

$k = 3$

$m = 9$

$001221012 ; \rightarrow 000111222$



$O(n+k)$

## Radix Sort

Exemplu:

936	420
123	101
514	123
305	783
777	514
793	997
994	365
420	986
101	106
104	777

! Un algoritm de sorțare stabile:  
Alex 10, Mihaela 0, Iosu 9, Mihai 7, Alex 5  
Alex 5, Mihai 7, Iosu 9, Ana 0, Mihaela 0

101	Pass I	101	Pass II	101	Pass III
106		106		106	
514		514		514	
420		420		420	
123		123		123	
365		365		365	
777		777		777	
986		986		986	
793		793		793	
997		997		997	

## Timp de reune Radix Sort recursiv

- 1)  $n$  numere
- 2) Fiecare număr are cel mult  $d$  cifre
- 3) Numerele sunt în baza  $10$

$$O(d \cdot n)$$

1) Se dau  $n$  numere, între  $0$  și  $m^3 - 1$ . Cât de repede le putem sorta?

$$\left\{ \begin{array}{l} n = n \\ d \text{ cifre} = \log_{10} m \\ k = 10 \end{array} \right. \quad \begin{array}{l} \text{Radix Sort} \\ \text{brut} \end{array} \quad O(n \log n)$$

Transformă numerele în baza  $n$ . !

$$\Rightarrow \left\{ \begin{array}{l} n = n \\ d = 3 \\ k = n \end{array} \right.$$

$$\Rightarrow \text{OK} \quad O(3(n \cdot m)) = O(n)$$

.....

$$n > 100$$

$$0, \dots, 999999$$

Un nr. în baza 100.

$$100^1 \cdot 2 + 0 = 20$$

## Alegría media

randomize-select( $A, p, r, i$ )

if  $p = r$   
return  $A[p]$

$q = \text{randomized-partition}(A, p, r)$

$k = q - p + 1$

if  $i = k$   
return  $A[q]$

else if  $i < k$

return rand-select( $A, p, 2^{-k}, i$ )

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

$$T(n) = T(\max(n-k, k-1)) + O(n)$$

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

$$E(T(n)) \leq E\left(\sum_{k=1}^n p_k \cdot T(\max(k-1, n-k)) + O(n)\right)$$

$$= E\left(\frac{1}{n} \sum_{k=1}^n T(\max(n-k, k-1)) + O(n)\right)$$

$$= \frac{1}{n} \cdot 2 \sum_{k=\frac{n}{2}}^{n-1} E(T(k)) + O(n)$$

se repite max

$$E(T(n)) = O(n) \leq c \cdot n$$

$$\text{Vraam } E(T(n)) \leq c \cdot n$$

$$\begin{aligned}
 E(T(n)) &\leq \frac{2}{n} \sum_{k=1}^{\lfloor n/2 \rfloor} c \cdot k + a \cdot n \\
 &= \frac{2}{n} \left( \sum_{k=1}^{\lfloor n/2 \rfloor} c \cdot k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} c \cdot k \right) + a \cdot n \\
 &= \frac{2}{n} \cdot c \left( \frac{n(n-1)}{2} - \frac{\binom{n}{2}(\binom{n}{2}-1)}{2} \right) + a \cdot n
 \end{aligned}$$

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

$$\leq cn - \left( \frac{cn}{2} - \frac{c}{2} - an \right)$$

Add. pt.  $\frac{cn}{2} - \frac{c}{2} - an > 0$

---

Mediana  $O(n)$

$$\begin{aligned}
 V: 2 & 1 & 7 & 5 & 10 & 13 \rightarrow [1 \text{ } 2 \text{ } \boxed{7} \text{ } 5 \text{ } 10 \text{ } 13] \\
 |V| = n & \\
 k = \left\lceil \frac{n}{2} \right\rceil &
 \end{aligned}$$

$$\frac{n}{5} \cdot \frac{1}{2} \cdot 3 \text{ sp. numai numici} \leq \text{pivot}$$

$\frac{7n}{10}$  el. mai mari pivot

0	0	0		0	0
0	0	0		0	0
0	0	0		0	0
0	0	0		0	0
0	0	0		0	0

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{2n}{10}\right) + O(n)$$

$$T(n) \leq C \cdot n$$

$\Rightarrow$  Să se caștige că  $T(n) \leq cn + k \cdot m$

$$\begin{aligned} T(n) &= T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + m \\ &\leq c \cdot \frac{n}{5} + c \cdot \frac{7n}{10} + m \\ &\leq n \left( \frac{c}{5} + \frac{7c}{10} + 1 \right) \leq n \left( 1 + \frac{9c}{10} \right) \leq c_m \end{aligned}$$

$n^{th}$ -element

$$n \left( \frac{1c}{3} + \frac{2c}{3} + 1 \right) \leq c$$

~~c + 1 = c~~    $T(n) \leq c \cdot n \log n$

~~false~~

$$\begin{aligned} 1 + \frac{9c}{10} - c &\leq 0 \\ -\frac{c}{10} + 1 &\leq 0 \\ c &\geq 10 \end{aligned}$$

$$\begin{aligned} \frac{1}{x} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{5}{x} + \frac{1}{x} &+ \frac{3c+1}{x} + c \\ \frac{7c+1}{x} &\leq 0 \\ c &\leq -\frac{7}{2} \end{aligned}$$

x	5	4	93	8
	2	1	100	7
	1	1	30	15
x = 2				

1) Aplicație aleg. de găsire al celui mai mic mediană

$M\left(\frac{m}{2}\right)$  el în timp liniar.

mediana

2)  $d_i = |x_i - \text{Med}|$  sir

- 3) găsiu al k-lea cel mai mic el. din d.  
 4) iterare pînă d și alegem și elementele mai mici sau  
 - cu x.

$$x : 1 \ 4 \ 6 \ 8 \ 10 \ 12$$

$$y : 1 \ 2 \ 3 \ 4 \ 5 \ 6$$

$$x_{mid} = x \left\{ \frac{1+n}{2} \right\}$$

între ele căutăm  
căutarea

$$y_{mid} = y \left\{ \frac{1+n}{2} \right\}$$

$$\underbrace{x_1 \ x_2 \ x_3 \ x_4 \ x_5}_{\text{sun}} \quad \underbrace{y_1 \ y_2 \ y_3 \ y_4 \ y_5}_{\text{sun}}$$

$$a) x_{mid} = y_{mid} \Rightarrow \frac{x_{mid} + y_{mid}}{2}$$

$$b) x_{mid} < y_{mid}$$

$$x_{mid} \dots a, b \quad y_{mid}$$

$$\Rightarrow \begin{cases} x \{ mid \dots n \} \\ y \{ 1 \dots mid \} \end{cases}$$

reciclare a 2  
vectori sortati

c)  $x_{mid} > y_{mid}$  (Analoga invers)

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

TRIE  
Sortitudine

Lectura  
lectură  
lective

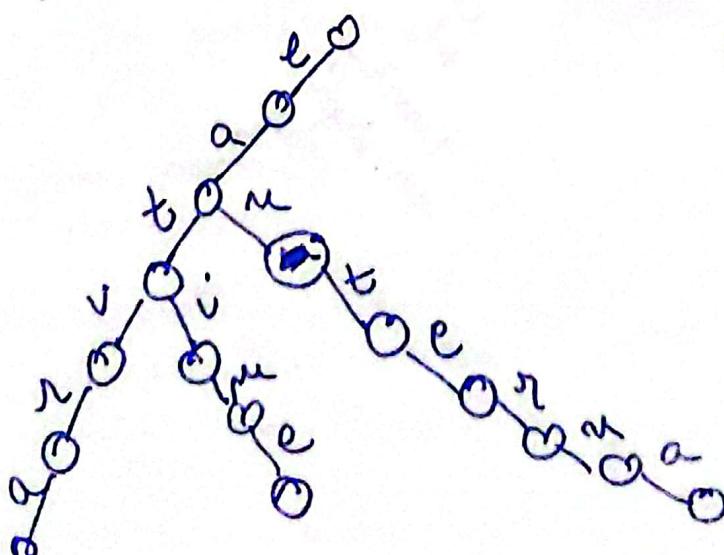
→ prefix comun

$$\begin{cases} 2 \\ 3 \\ 4 \end{cases}$$

Sortitudine

$$O(n \cdot |S|)$$

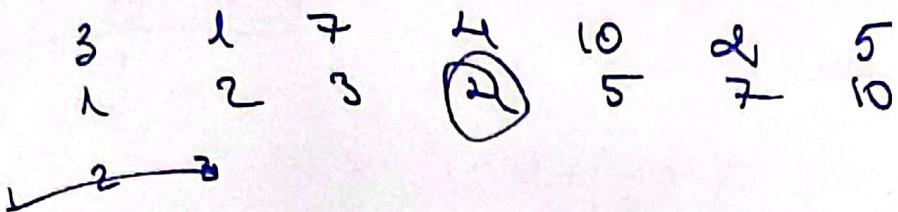
eng. cuv.



# Algoritmul de găsire a medianei

-0011-

## în $O(n)$ determinist



La algor. abator, în cazul favorabil :

$$T(n) = T(n/2) + O(n)$$

$$= O(n)$$

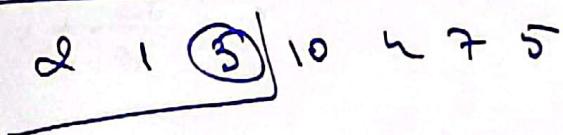
În cazul cel mai nefavorabil :

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

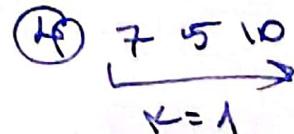
## Algoritmul determinist

$$k=5$$

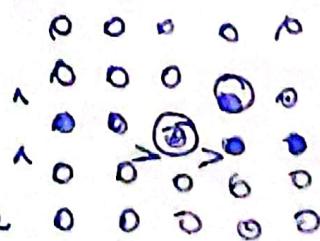
pivot = 2



$\rightarrow$   
k = 2  
pivot = 4



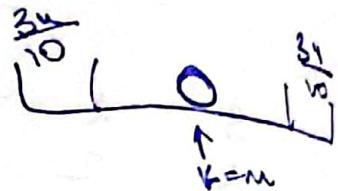
- 1) Împărțire sirul în grupe de căte 5 ('de ce nu 3, 7?')
- 2) Găsire mediana din fiecare grupă.
- 3) Găsire mediana celor  $n/5$  mediane (recursiv).
- 4) Alegere mediana "gușă" ca pivot și urmări același strategie ca la algor. recursiv.



$\frac{m}{5}$  grupe cu cîte 5 elemente

$\frac{m}{5} \cdot 1$  grupe care au fiecare cîte 3 el. mai  
mici decât medianele medianelor

$\hookrightarrow \frac{3m}{10}$  elemente



$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{3n}{10}\right) + O(n)$$

$$\text{Iară } T(n) \leq c \cdot n$$

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

$$T\left(\frac{3n}{10}\right) \leq c \cdot \frac{3n}{10}$$

$$\begin{aligned} \Rightarrow T(n) &\leq c \cdot \frac{n}{5} + c \cdot \frac{3n}{10} + n \\ &= n \left( \frac{c}{5} + \frac{3c}{10} + 1 \right) \\ &= n \cdot \left( \frac{gc}{10} + 1 \right) \leq c \cdot n \end{aligned}$$

$$\frac{gc}{10} - c + 1 \leq 0$$

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



$$\frac{1}{2} \cdot 2 \cdot \frac{n}{3} \Rightarrow \frac{2n}{3}$$

grupe 3:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) = O(n \log n)$$

grupe 7:

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

$$\frac{n}{7} \cdot 4 \cdot \frac{2n}{7} \Rightarrow \frac{8n^2}{49}$$

Tatal  $\Rightarrow$

## Coduri Huffman:

Problema: Se dă un text și vrem să îl codificăm ( fiecare caracter să fie reprezentat printr-o secvență de biti ) astfel încât numărul total de biti să fie minim.

car.	a	b	c	d	e	f
nr.	45	13	12	16	9	5
*	000	001	010	011	100	101

cod de lungime fixă.

$$abfa = 000 \ 001 \ 000 \quad \downarrow \\ 300 \text{ biti}$$

*	0	101	100	111	1101	1100
	0	101	100	111	1101	1100

cod de lungime variabilă

$$45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4 = 224 \text{ biti}$$

$$abac = 0 \ 101 \ 0 \ 010$$

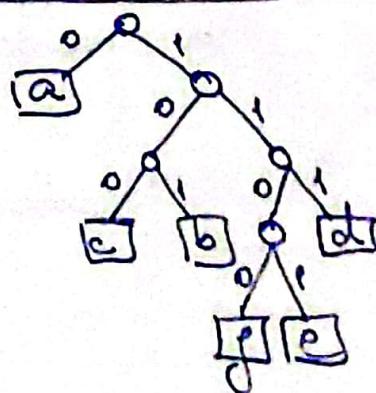
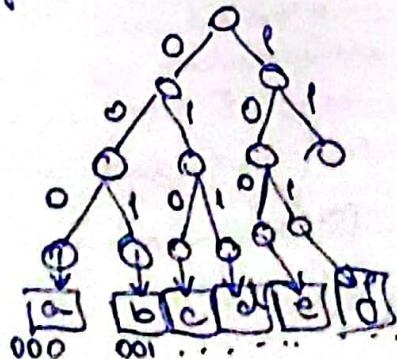
$$\Delta c \cdot f = 10 \Rightarrow 100 \xrightarrow{\text{fa}} c$$

$$\begin{array}{r} 35 \\ 25 \\ 48 \\ \hline 123 \\ 55 \\ \hline 68 \\ 45 \\ \hline 224 \end{array}$$

⇒ pt. a putea decodifica: niciun cod nu trebuie să fie prefix al altui cod.

Oricare cod poate fi reprezentat ca un arbore binar.

Exemplu: Cod fix vs. variabil



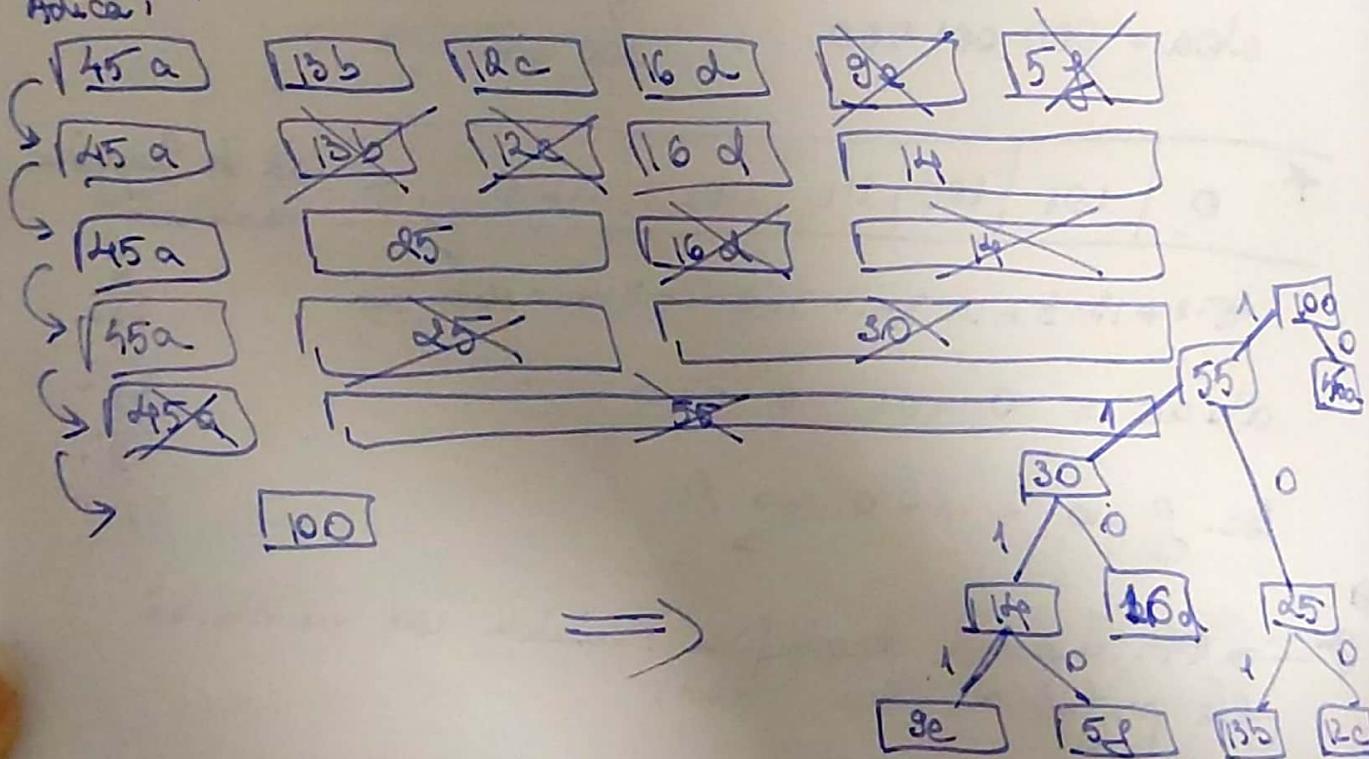
Dându-se o multime de caractere împreună cu frecvențele asociate, vrem să găsim codul optim.  
(lungimea totală a textului codificat să fie minimă).

### Algoritm:

- 1) Toate caracterele vor fi fuzionate în arbore.
- 2) Alegem obiectele cu frecv. minima și adăugăm  
acestea în frecvența însurată a celor 2 obiecte.  
→ nou nod = părintele obiectelor ale căror frecvență  
minimă.

3) Repetăm pasul de while (obiecte  $\geq 2$ )

Adică,



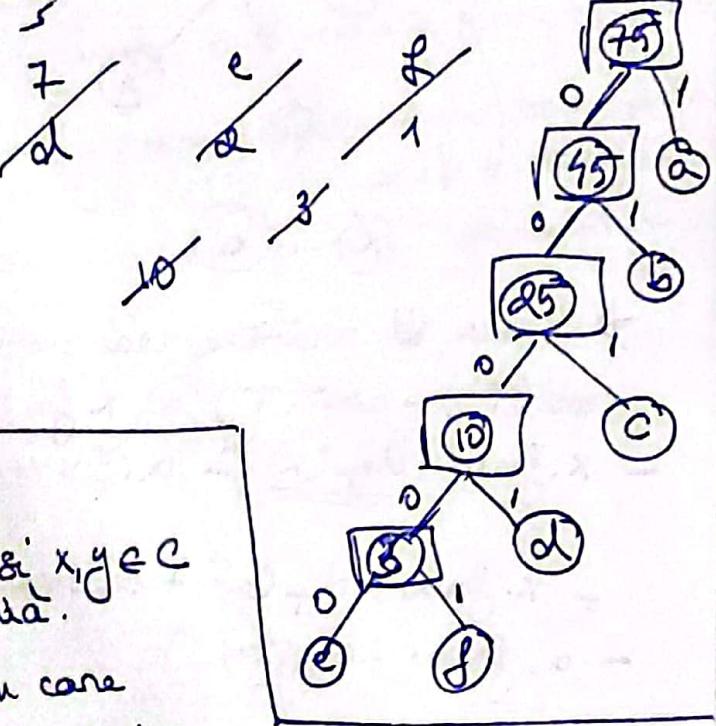
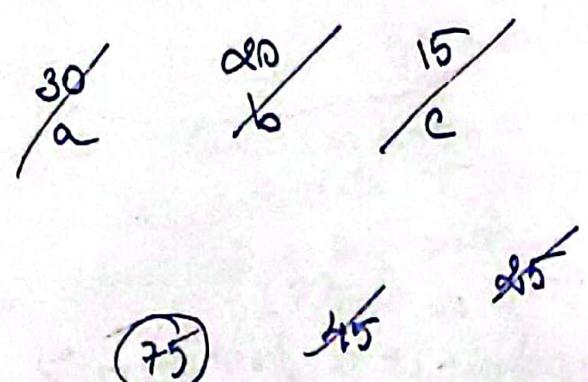
→ Aceea din extrageri din min heap și  
m inserții  
↓  
 $O(n \lg n)$

abs  
1 > 0

# Coduri Huffman

(4 mai)

Să dăm litere cu frecvențe asociate.



## Lemă

Fie  $C$  un set de caractere și  $x, y \in C$  dă numerice cu frecvență minimă.

Atenția! (F) un cod optim în care

cuvintele asociate caracterelor  $x$  și  $y$  au aceeași lungime și diferență primul ultimul bit.

## Demonstrare

(presupunerea abs)

Fie un arbore  $T$  corespunzător unui cod optim în care  $x, y$  nu sunt frunze de adâncime maximă.

## Notări

Fie  $x, y$  - caractere cu frecvență minimă

$a, b$  - caractere corespunzătoare nodurilor de adâncime maximă

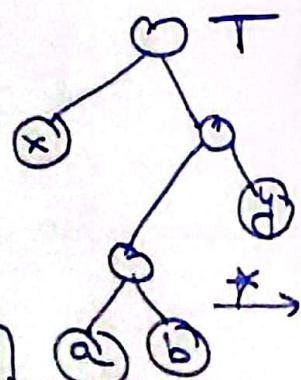
a. frecv. = frecv. caract. a

$d_T(x) =$  adâncimea cui  $x$  în arborele  $T$

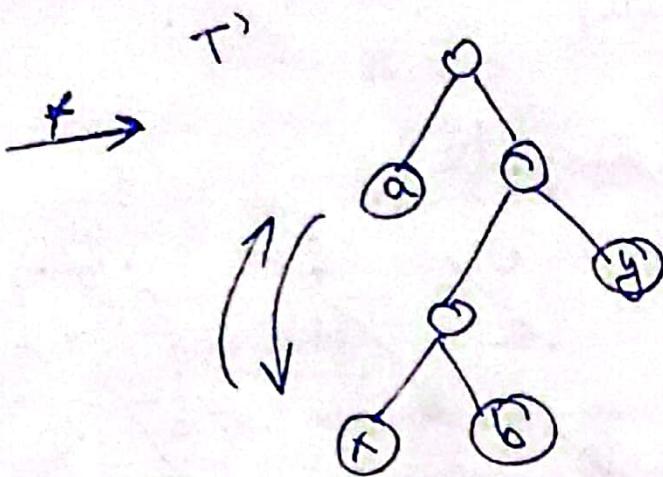
## Idea principală

dant.

I.c. (F) este arbore  $T$  în care  $x$  și  $y$  nu sunt noduri de adâncime maximă, pot construi un arbore  $T'$  în care  $x$  și  $y$  sunt noduri de adâncime maximă și  $\text{cost}(T') \leq \text{cost}(T)$ .

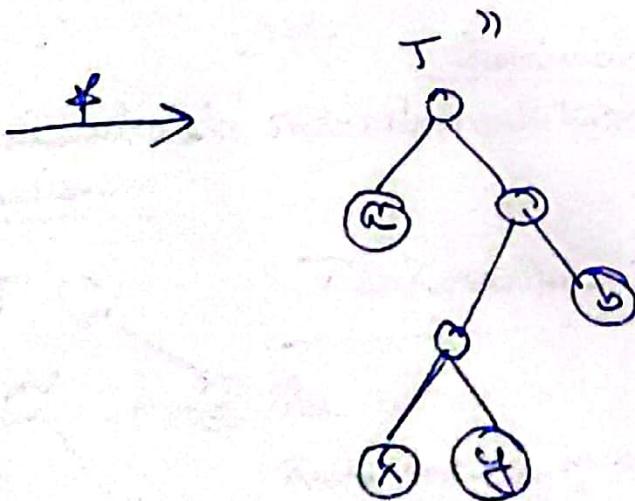


$$\text{unde } \text{cost}(T) = \sum_{v \in V} v \cdot \text{frecv} \cdot d_T(v)$$



pune x cu frev. < acă  
adăuc în arbore  
↓  
mai eficient

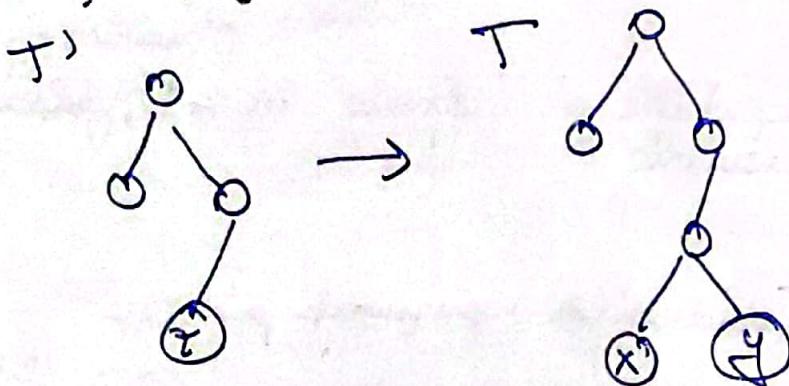
$$\begin{aligned}
 \rightarrow \text{Vom să arătăm că } \text{cost}(T') \leq \text{cost}(T) \\
 \text{cost}(T') - \text{cost}(T) &= x \cdot \text{frecv. } d_{T'}(x) + a \cdot \text{frecv. } d_{T'}(a) \\
 &\quad - x \cdot \text{frecv. } d_T(x) - a \cdot \text{frecv. } d_T(a) \\
 &= x \cdot \text{frecv. } d_T(a) + a \cdot \text{frecv. } d_T(x) - x \cdot \text{frecv. } d_T(x) \\
 &\quad - a \cdot \text{frecv. } d_T(a) \\
 &= (x \cdot \text{frecv} - a \cdot \text{frecv.}) d_T(a) \\
 &= x \cdot \text{frecv. } (d_T(a) - d_T(x)) + a \cdot \text{frecv. } (d_T(x) - d_T(a)) \quad (?) \\
 &= (d_T(a) - d_T(x))(x \cdot \text{frecv.} - a \cdot \text{frecv.}) \stackrel{\leq 0}{\leq 0}
 \end{aligned}$$



## Teorema

Fie  $C = \{x, y\} \cup \{z\}$  unde  $z \cdot \text{freq} = x \cdot \text{freq} + y \cdot \text{freq}$ .  
 Fie  $T' = \text{c}\backslash \{x, y\} \cup \{z\}$  unde  $z \cdot \text{freq} = x \cdot \text{freq} + y \cdot \text{freq}$ .

Ac.  $T'$  este un arbore optim corespondator unui cod optim pt.  $C$  at. arborele  $T$  format din  $T'$  în care  $z$  devine nod intern și  $x, y$  sunt fiu săi și este arbore optim pt.  $C$ .



$$\begin{aligned}
 (\text{cost}(T) - \text{cost}(T')) &= z \cdot \text{freq} \cdot d_{T'}(z) - x \cdot \text{freq} (d_T(z) + 1) \\
 - y \cdot \text{freq} (d_T(z) + 1) &= z \cdot \text{freq} \cdot d_{T'}(z) - (d_{T'}(z) + 1)(x \cdot \text{freq} + y \cdot \text{freq}) \\
 &= -(x \cdot \text{freq} + y \cdot \text{freq})
 \end{aligned}$$

$$\Rightarrow \text{cost}(T) = \text{cost}(T') + x \cdot \text{freq} + y \cdot \text{freq}.$$

P. red. abs. că  $T''$  cu  $\text{cost}(T'') < \text{cost}(T)$

Așa că, dacă  $T''$  este un arbore optim, conform Lemii de la punctul I a cursului  $x, y$  (caracterile cu freq. minină vor avea adâncime maximă și vor fi frăți)

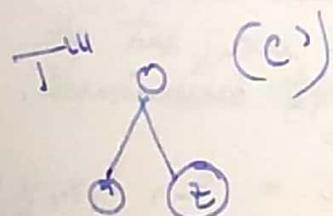
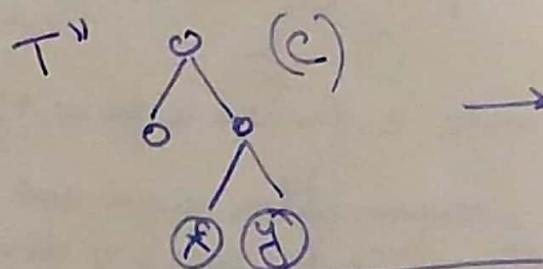
Construim un arbore  $T'''$  pentru setul  $C$  care are  $\text{cost}(T''') = \text{cost}(T'') + x \cdot \text{freq} + y \cdot \text{freq}$ .

$$\Rightarrow \text{cost}(T''') = \text{cost}(T'') - x \cdot \text{freq} - y \cdot \text{freq}.$$

Ac. prin absurd că  $T'''$  nu este arbore optim și  $\text{cost}(T'') < \text{cost}(T)$   
 așa că  $T'''$  cu  $\text{cost}(T''') < \text{cost}(T)$

$$\text{cost}(T''') = \text{cost}(T'') - x \cdot \text{freq} - y \cdot \text{freq}$$

$\Rightarrow \text{cost}(T'') < \text{cost}(T)$



## Hashing (tabelă de disperzie)

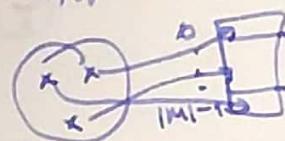
### Problema

Vom căuta reprezentare a multimii  $M \subseteq U$ , folosind  
căt mai puțină memorie.

$$|M| \leq |U|$$

Ex.: multimea studenților reprezentați prin CNP

$O(1)$  inserție/ căutare



Af. fiecare element  $x \in M$

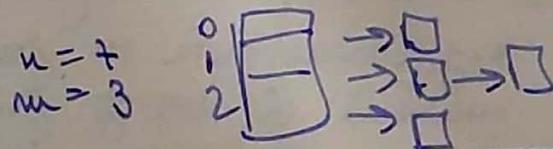
Aveam o funcție  $h: U \rightarrow \{0, \dots, |M|-1\}$

$$\text{Ex.: } h = (3x+2) \bmod 7$$

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

0	200
1	230
2	9
3	5
4	2
5	43
6	

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



Cum proiectăm o funcție hash, bine?

Ce înseamnă o funcție hash?

D.c. ~~hash~~ o funcție hash ar fi de dorit să  
respecte  $\Pr[h(x_i) = h(x_j)] = \frac{1}{m}$  pt. orice 2 el.  $x_i, x_j$ .

În modice lungimea listei i ar fi

$$\sum_{j \neq i} \Pr(h_j(x_j) = h(x_i)) = \frac{\text{căk el. am}}{\text{mărimea tabelei}}$$

În loc de o funcție, definiște o familie de funcții cu proprietatea că pt. o funcție  $h \in F$ , așa că aleator și fără elem.  $x_i$  și  $x_j$  au  $\Pr(h(x_i) = h(x_j)) \leq \frac{1}{m}$

Ex. de familie de funcții:

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod m^{n-1}$$

cu:  $m =$  mărimea tabelei

$p =$  nr. prim  $> m$

9, 8, 7, 4

9 8 7 4

8 9 7

7 8

4 9

$$\sqrt{\{0\}} = 8/8/7 \quad \sqrt{\{0\}} > \sqrt{\{1\}}$$

index = 1

$$\sqrt{\{1\}} = 8/9/7/1$$

index = 2

$$\sqrt{\{2\}} = 7/8$$

index = 3

$$\sqrt{\{3\}} = 1/9$$

7 4 8 9

