

STRUCTURI DE DATE

SEMESTRUL 1

$O(n^2)$: Insertion Sort, Bubble Sort,

$O(n \log n)$: Merge Sort, Tim Sort, Heap Sort, Quick, Ynlro Sort

$O(N + m \alpha)$: Counting Sort

1) Se dă un vector. Calculați nr. intersecții?

7 9 3 5

7 9 3 5 3 7 5 9
+ 3 9 5 3 5 2 9
3 + 9 5

Se parcurge vectorul. Se compare primele 2 elemente.

În casul în care nu sunt în ordine cresc, se interzolvă și se compară următorul. Atât timp ce nu mai depășește comparație între 2 sau 3. (Bubble Sort)

sol:

```

for (i = 0; i < n - 1; i++)
    for (j = i + 1; j < n)
        if (v[i] > v[j]) c++;
        cout << c;
    
```

$O(\frac{n(n-1)}{2}) \sim O(n^2)$

SOL II

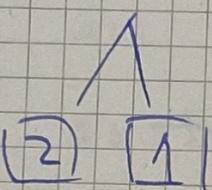
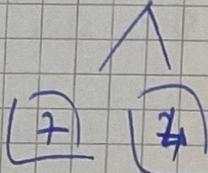
Merge Sort
 $O(n \log n)$

+	5	2	1	+	6	3
---	---	---	---	---	---	---

+	4	2	1
+	5	6	3

+	1	4
+	2	n

+	5	6	3
+	5	3	



+	4	7
+	1	2

+	5	6
+	3	

+	2	2		
+	1	2	5	7

+	2		
+	3	5	6

+	2	2	1	1	1		
+	1	2	3	4	5	6	7

2) Se dau n nr. reale sorteate. Căte permutări sunt S ?

$S = 20$

$U = 1 \ 3 \ + \ 9 \ 11 \ 13 \ 15 \ 17 \ 18 \ 21 \ 22$

$\Rightarrow 2$

Comparăm sume cu elem vectorului, să le obțină la stânga. Când sume este mai mica decât elem curent,

se verifică dacă patrucesc apartine multimii:

$$C: O(n^2)$$

$$\text{Complexitate: } O(1)$$

Sol I Se caută de n ori $s - \alpha[i]$. Se face o căutare binară în α → $C: O(\log n)$

Sol II Se face n de căutări. Se verifică pt. fiecare $\alpha[i]$ dacă $s - \alpha[i]$ e în α .

→ $C: O(n)$

$$\text{Complexitate: } O(n)$$

Sol III

nr-apariții

Termen: vint ~~do~~ (vector (int) values, vint value)

→ R-sortat

→ se fol. cb.

0 1 2 3 4 5 6 +
1 2 9 9 9 11 13 15

$$R: 9 \rightarrow R = 3$$

12 2 6 7 8

Sol III $i = 1$; $j = n$

Se verifică dacă $\alpha[i] + \alpha[j] > s$

→ $\alpha[i] \Rightarrow j--$

→ $\alpha[j] \Rightarrow i++$

→ egal $\Rightarrow ct++$; $i++$, $j--$

+ Se verifică dacă elementul din patrucesc

$$C: O(n)$$

3) Se dă un vector sortat. Se sătăcă un vector care să fie circular și apoi să fie sortat.

11 14 17 19 23 27 32 (36) 25 + 9

matrice circ (vector $\leftarrow \text{int} > n\right)$ } ...
vector sort

5 12. 3 1

Se calculează mijlocul vectorului. ~~Se verifică~~ da că el este mijlocul vectorului și că $\text{el}[mijl] < \text{el}[mijl+1]$.

Dacă da, se calculează mijlocul vectorului și să se repeta.

Să se compara cu primul element din vector.

TERMIN

4) Se dă un vector. Câtă per centaj de elemente sunt zero?

Hint: cauza unei lungi $\leq l$

\hookrightarrow pt. complexitate

\Rightarrow vector de cauza: setul de zero
nr de zero

\Rightarrow sets?

metri

metri

* era

* trea

erubie

* aer

cb (vector < int > & v, int value)

pow

{ pow = 1 << 20;

par = 0;

while (par <

pow

{ if (par + ~~par~~ < v.size() & v[par + par] <= value)

{ par += par * 2;

pow ~~par~~ / 2; }

return v[par] == value; }

4

SEMINAR 2

stiva: FIFO

vector <int> st;

push (int & val) } st.push_back (val); ↴

volt pop() } return st.pop(); ↴

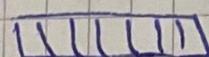
1. Cum implementăm 2 stive cu un vector?

→ punem pe traiul elementele

→ se pun st de la cap și st2 de la coadă

coada: FIFO

2. Implementația coada folosind 2 stive.



Eleme din st1 și multe din st2.

Când ambele push, se pun eleme de st1 și se face pop în coada

dim st2 $\rightarrow O(n)$

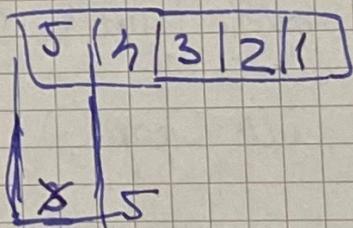
3. Temă: implementarea unei sticle cu 2 cări

3. Se dă un sir. Se să verifice dacă e parțialat
corect.

Se utilizează sticle. Push parțialatele deschise,
pop parțialatele închise. Aceste sticle nu e goală / nu mai
aveau ce să scoată / elan dim și sticlele dif de elan
la care vorbiște să dim pop \rightarrow nu e parțialat
corect.

4. [3] [5] [2] [1] [5]

4 5 2 1 3



5 4 3 1

1 2 5 3 4

hash table map



1 3 5
93%

5

isul de sp.

sp de at.

suprareducere << >>

Operări cu hash-wi

- inserare
- stergere
- căutare

$\downarrow - O(1)$

\hookrightarrow cu o implementare bună

În urma unei fct. de hash, un ob. ajunge într-un loc
învecinat, și pos. ce 2 ob să ajungă în același loc \Rightarrow

\rightarrow COLIZIUNI



Cum rezolvăm?

- 2 fct. de hash slifite - hash dublu
- înălțuire: metoda clasică
 - \hookrightarrow cat refluiești: lista are prea multe elemente
- adresație obiectelor

Fct de dispersie

\hookrightarrow op. inceară

\hookrightarrow number of slots

\rightarrow momentan, $t_c(t) = 4\%p$, p -prime

\rightarrow

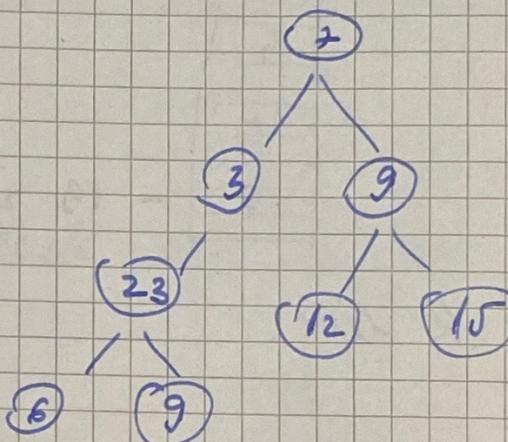
1. Kap: un arbore binar complet cu proprietăți co-tatăl și mai mult decât anumii fi (Kap de mijloc)

→ 1 banet pt.

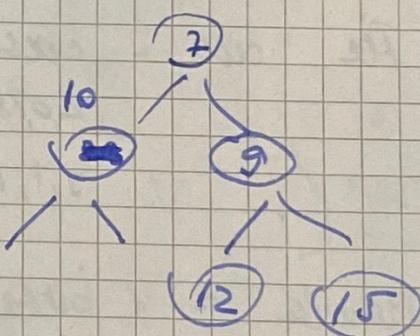
- inserare

- modificare naștere

- stergere



mkt: 1 modificare



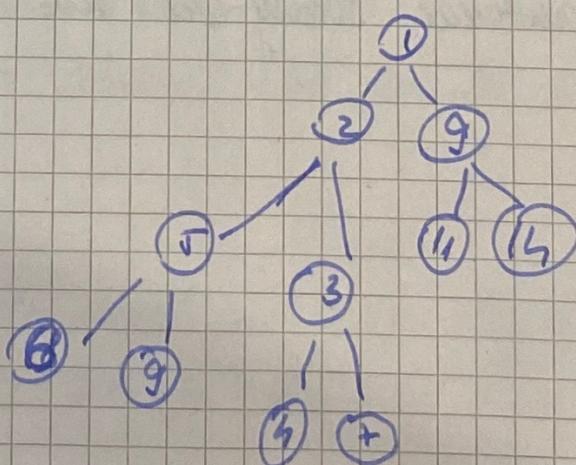
! Structure !

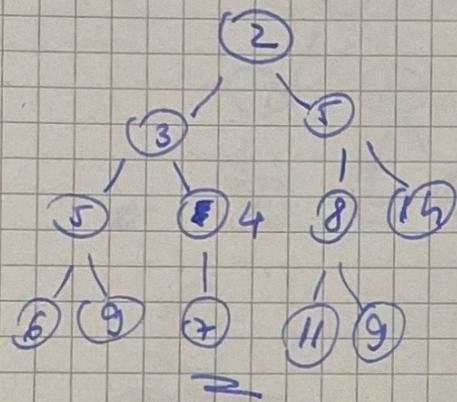
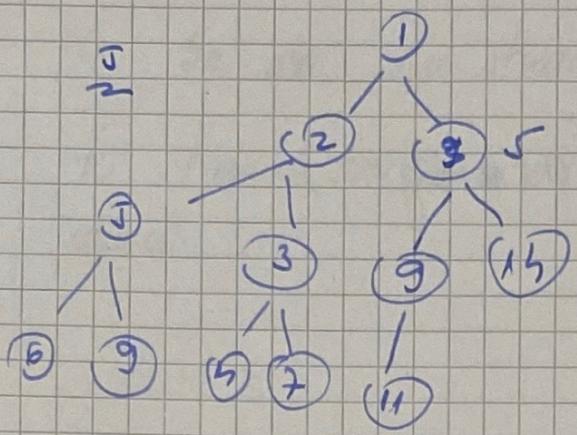
$$\begin{array}{c} \text{fie } \\ \boxed{i \cdot 2 + 1} \\ i \cdot 2 + 2 \end{array}$$

$$\text{raț} = \frac{(i-1)}{2}$$

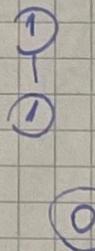
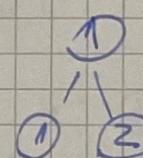
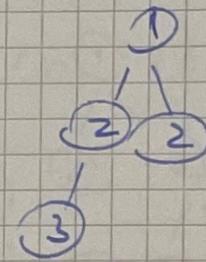
7-5-2

$\frac{7}{2}$





\times ~~max~~ \rightarrow heap



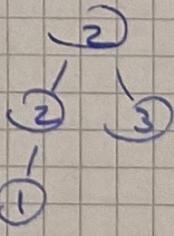
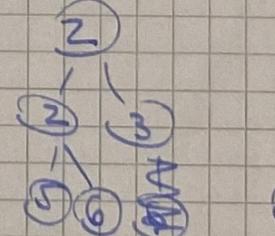
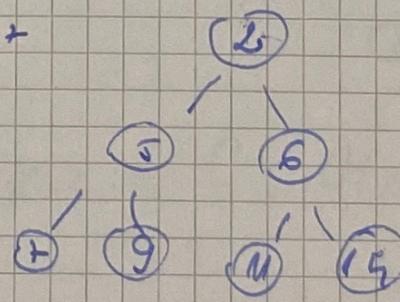
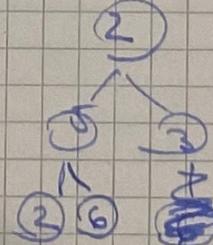
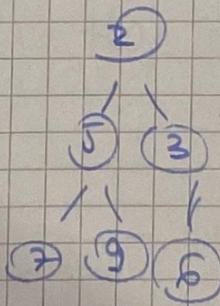
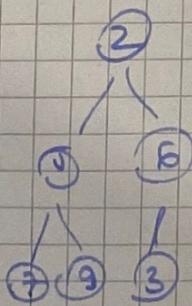
clone (int i):

heap [i] = heap [-n]

clonar(i);

wce(i);

2. 15 + 2 9 - 5 6 11 - +



Se $n-1$ doi:

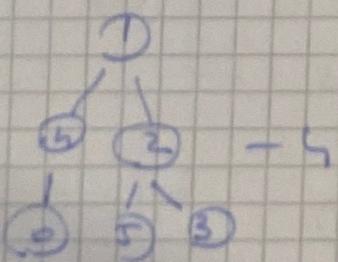
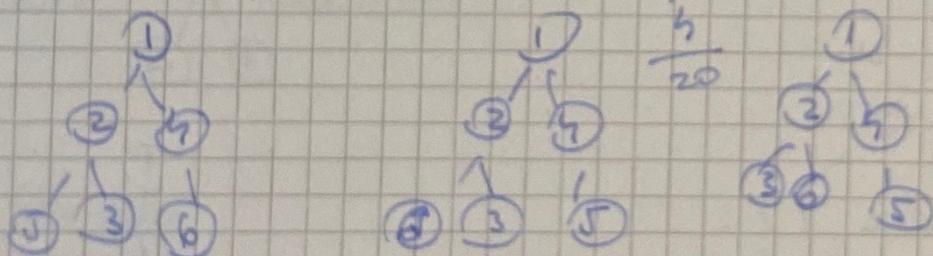
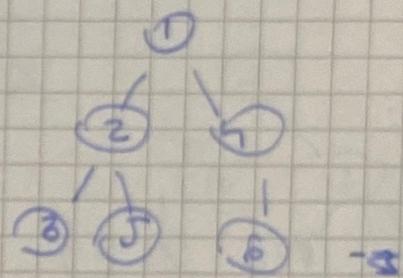
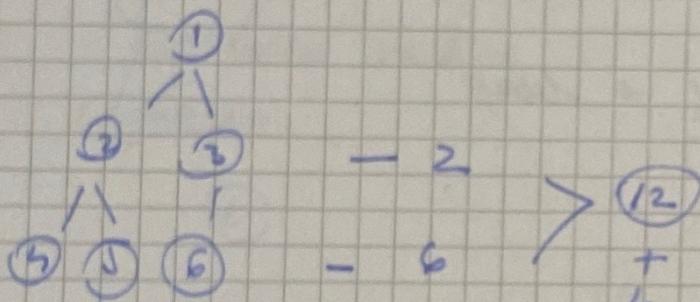
→ extrage valoare (max)

→ $-n -$

→ inserare difuzo

$O(\log n)$

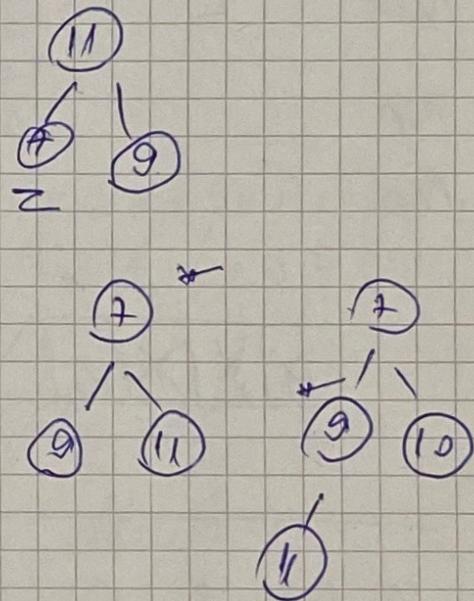
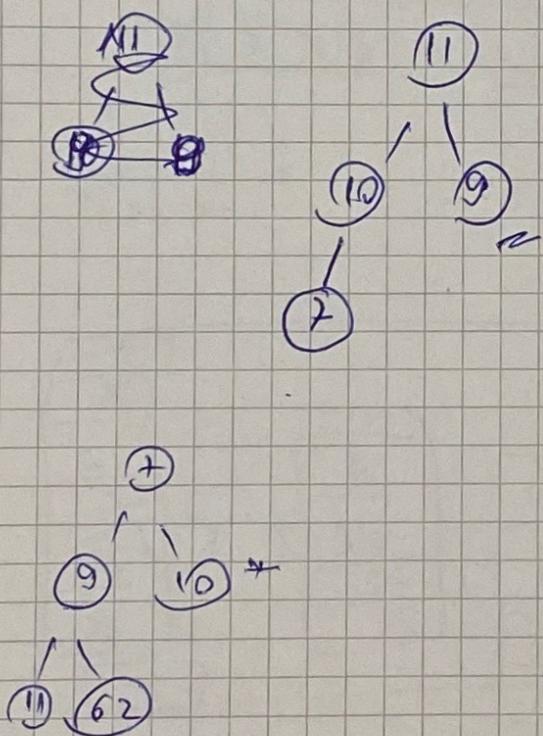
3. Cât heap-uri se pot forma cu $1, 2, 3, 4, 5, 6$?



4. Se dă un stivu de numere începând cu pasul k . Trebuie să se evidențieze al k -lea cel mai mare număr.

$$k = 3$$

11	7	9	10	62	5	+	5	22
		7	9	10	10	10	10	11



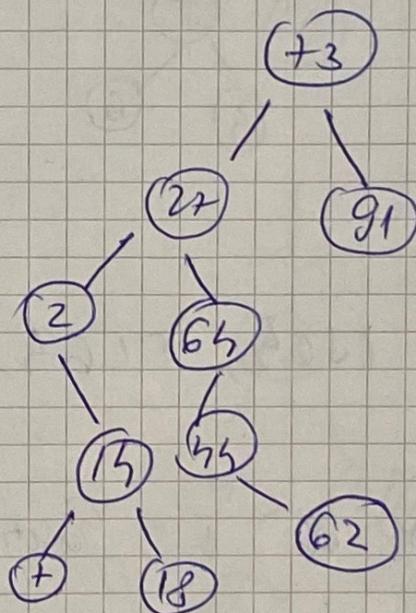
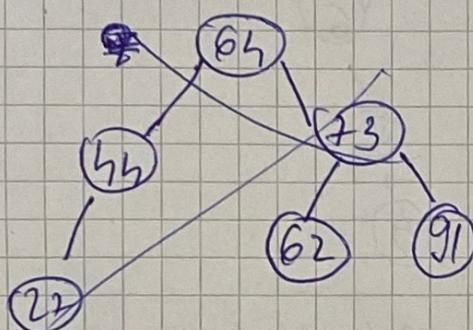
Sol: Heap cu stivu k elem $\Rightarrow C: O(n \log k)$

Arbore binar de căutare

Este un arbore binar unde fiecare nod

se poate sătăcește din subarborele stâng
 $L - n \rightarrow$ drept

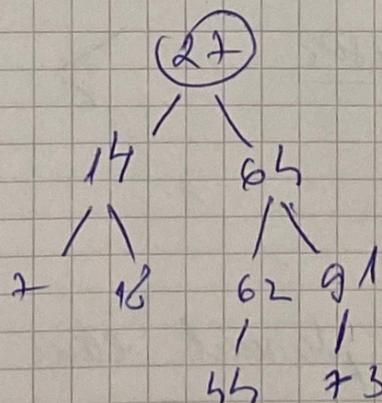
Exemplu : +3, 27, 2, 14, 91, 65, 18, 7, 55, 62



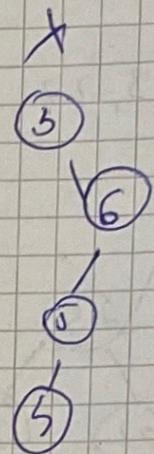
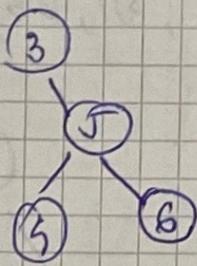
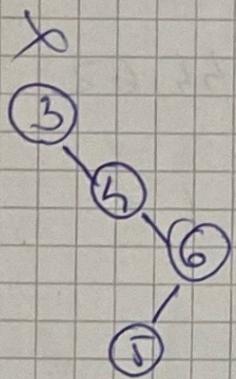
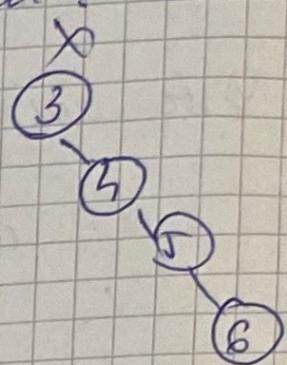
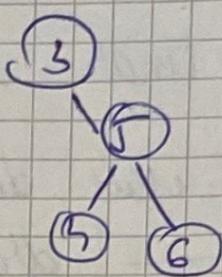
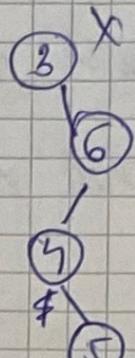
Înordine : stânga - rădăcine - dreapta (SRD)

U sortat: $\textcircled{2} \quad 2 + 14 18 27 55 62 65$

+3 91



① Acea nod 3, sau de mărime 3, 5, 6, veau
la numere. Care mărime distinție sunt?



1556, 1565, 5165, ~~5165~~ 5156

② 3' și de 4'. Găsiți cea mai apropiată
valoare de 4 din arbore.

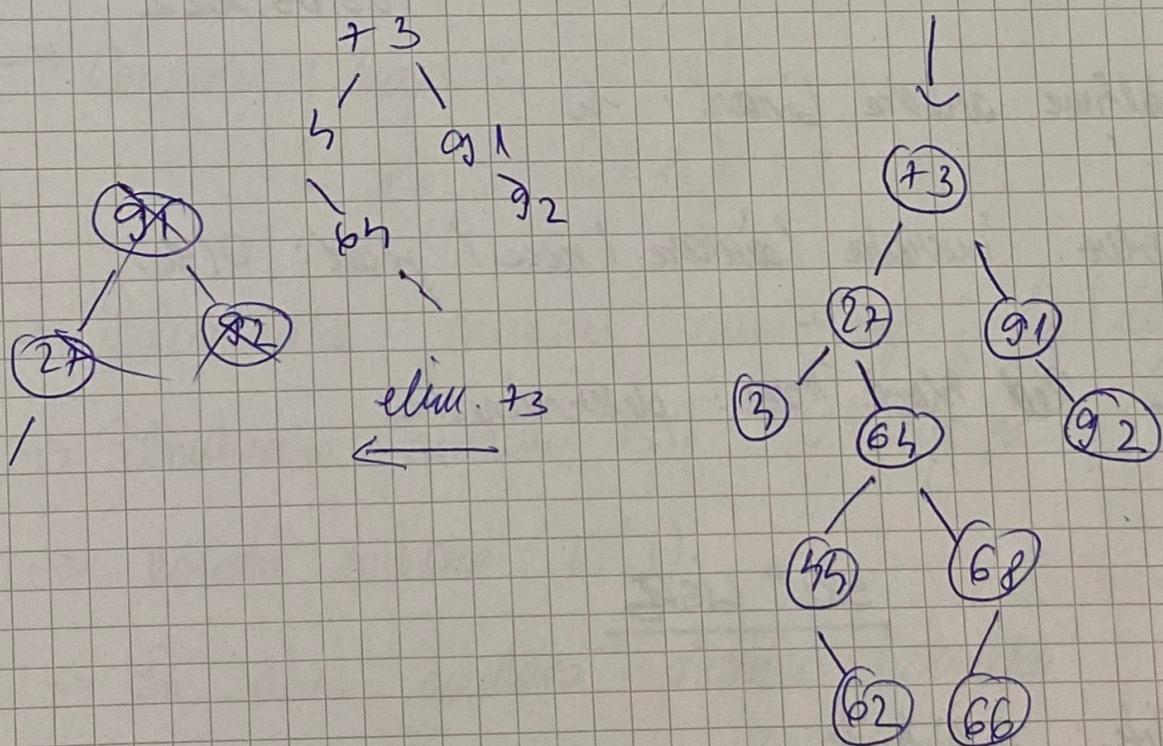
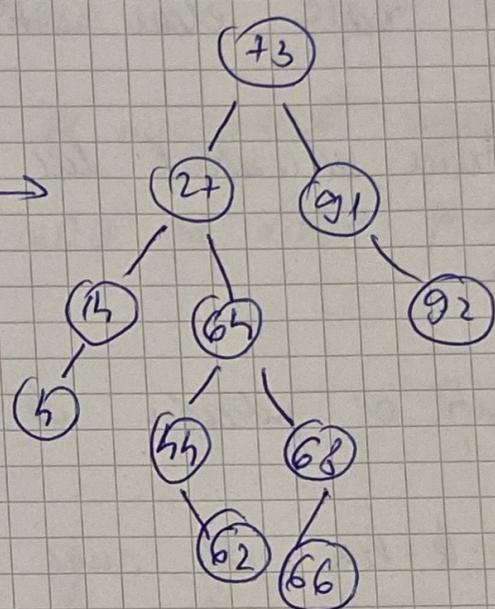
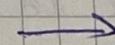
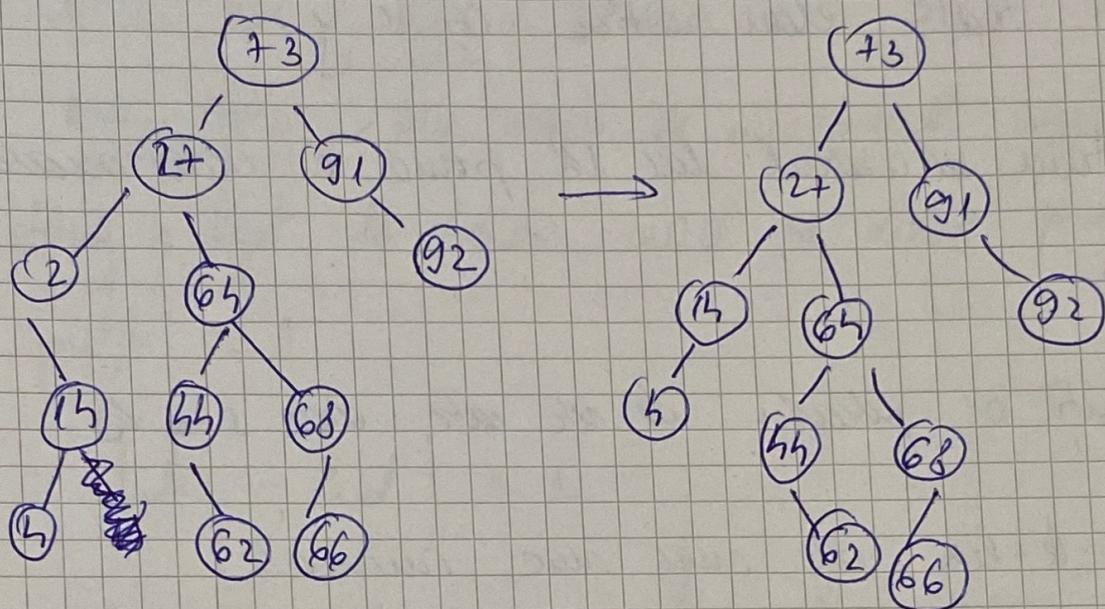
Căutare: $O(n)$

Predecessor : / - - / - - / - -
Succesor : \ / - / - - / - -

Dacă nu se poate, cauță primul elem
parinte mai mare / mai deșert elem. cauță.

③

Stergrau: 18, +, 2, 14, 23



map, set
→ AVL > heap → prior queue
→ microcosm map/set : hash

B arbori

→ cel puțin $t-1$, cel mult $2t-1$ elevi

11.05.2022

SEMINAR 6

1) Adevăram că vectorul cu n elemente

~~+~~ 7 9 1 6 3 5 7 9 11 14
7 16 17 23 26 31 37 38 42
și am operații de 2 tipuri

a) suma intre i și j : $O(n)$

b) adaugare de elem ale pe poziția pot deosebita

val : $O(1)$

- i) + multe op de tip 1, 2
- ii) aversecate

Sol Σ : Vector

Sol $\bar{\Sigma}$: Vector de sume parțiale \oplus

Sol \bar{w} : Sumele lui Boty / (SQRT decomposition)
(sume de cost 3)

$v[\text{par}] \leftarrow \text{val}$

$\rightarrow O(1)$ pt. op de tip 2

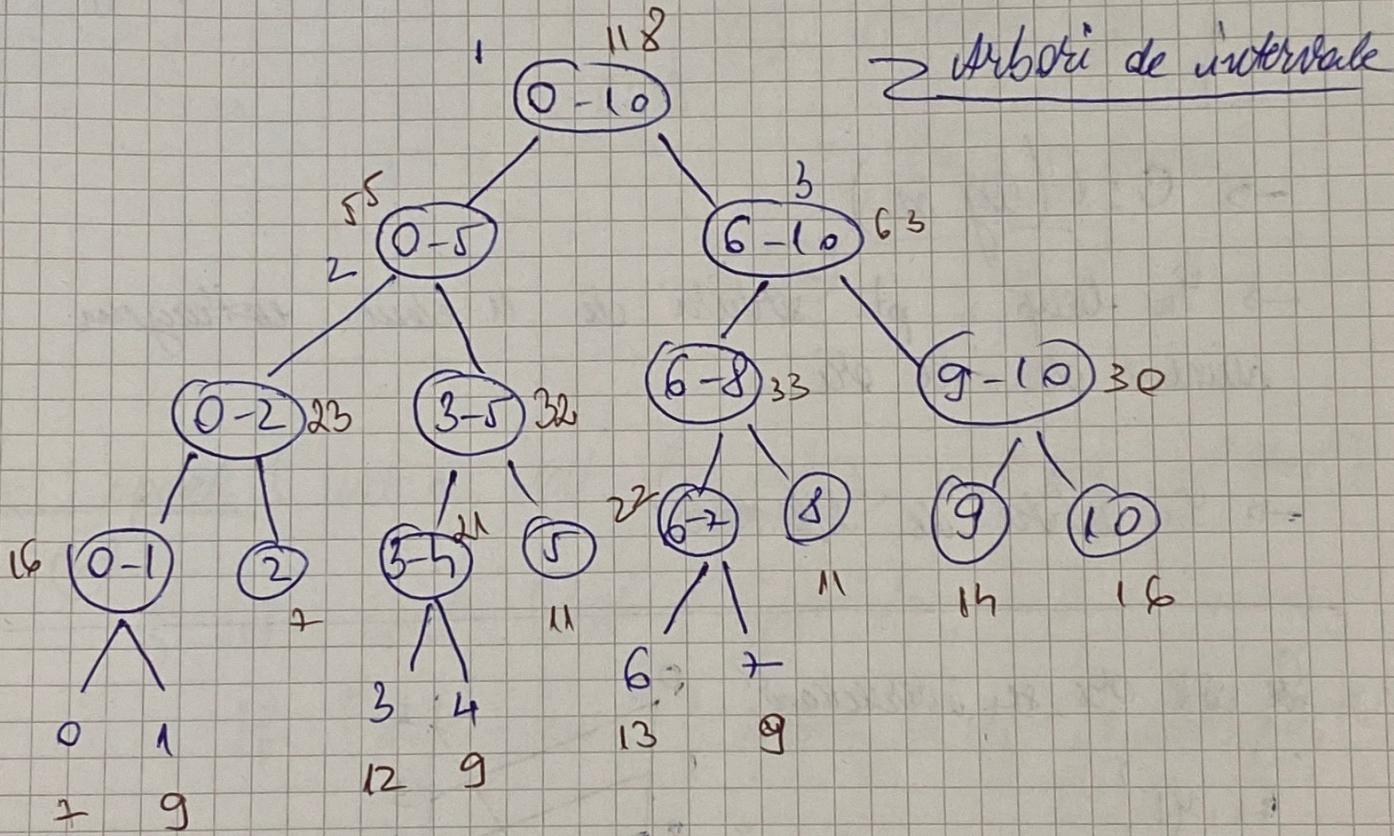
$B[\text{par}[1..L]] \leftarrow \text{val}$

$\rightarrow O(\ln n)$ pt. op - 1

c) Adaugam la toate elevii din intervalul (i, j) valoarea real.

\rightarrow date update; trebuie mutat sunte elevii si valoarea

care trebuie adaugata la fiecare element.



update (mod, L, R, i, k) [op 2]

} if ($i > R$ || $i < L$) return;

if ($L == R$) { $A[\text{mod}] += k$; return; }

update (mod * 2, L, (L + R) / 2, i, k);

update (mod * 2 + 1, (L + R) / 2 + 1, R, i, k);

$A[\text{mod}] = A[\text{mod} * 2] + A[\text{mod} * 2 + 1];$

$\text{sum}(\text{med}, L, R, i, j)$ [sp 1]
 } if ($i > R \text{ || } j < L$) return 0;
 if ($i \leq L \text{ & } R \leq j$) return $A[\text{med}]$;
 return $\text{sum}(\text{med} + 1, L, (L+R)/2, i, j) +$
 $+ \text{sum}(\text{med} * 2 + 1, (L+R)/2 + 1, R, i, j')$;
 }

$\rightarrow C: O(\log n)$

\rightarrow în heap: pt. sortarea de n elem, extragem minimul de n ori.

\rightarrow în arbore de interval!

De căte ori se intersectează?

t_1, y_1 ?

$t_2 \geq t_1 \quad y_2$?

$y_2 \leq y_1 \quad || \quad t_1 = t_2$

