

SUB 2

Legenda:

nn = teorie
??? = cum ar trebui să fie
 = - intuitiv

A. Subalgoritmu $\sigma(n)$

$i \leftarrow n$

căt timp $i \neq 0$ execută

$i \leftarrow [i/2]$

tipărește "gata"

$c_F = c_D = c_M$

$$T(n) = 1 + \sum_{i=1}^{\log_2 n} 1 + 1 = \alpha + \log_2 n \in \Theta(\log_2 n) \quad \left\{ \rightarrow \text{overall complexity } \Theta(\log_2 n) \right.$$

B. Înălțimea nodului va fi de 3, decarece înălțimea unui nod este definită ca fiind lungimea celui mai lung lanț de la nod la un nod frumos.

c. b) $\Theta(n)$

c_F : Elementul e în oglindă sau nu $x_1 \rightarrow$ nu este în vector $\rightarrow T(n) = 1 \in \Theta(1)$

c_D : Elementul nu este în oglindă și nu este pe poziția sa în vector $\rightarrow T(n) = p$ (poz. a găsit elementul) + $n-p$ (pt. a muta el) = $n \in \Theta(n)$

Sau el este sau nu mai devărt $x_n \rightarrow T(n) = n \in \Theta(n)$

c_M : Elementul nu este în oglindă și nu este pe poziția sa în vector

Elementul nu este în oglindă și nu este pe poziția la care ar trebui inserat elementul) + $\underbrace{1}_{\text{nu e o parz}}$

$$\Rightarrow T(n) = \frac{n + n + \dots + n + 1 + 2 + \dots + n+1}{n!} = \frac{n^2 + n(n+1)}{n!} = \frac{n+1}{\alpha^n} \in \Theta(n)$$

d)

Cele două variabile reprezentă numărul de liniile atunci când coada este vidă. După adăugarea unei el, ele trebuie să ne modifice o). ambele să pună pe el nou adăugat.

E.

STRGERE IN ANSAMBLU

cp:

 $\text{H: } \text{TELEMX} \times \text{TELEM} \{T, F\}$ ans: $\text{TELEM} [0, \text{cp.-1}]$ dim: mtrig

} sterge elementul cu prioritatea
cel mai mare din coada cu
prioritate

pre: cp - coada cu prioritati
nevidabilipost: e: TELEM , elementul
din coada cu prioritata
numarata, care a fost sterssterge (cp^i , e^i) $e \leftarrow \text{cp. ans}[0]$ $cp. ans[0] \leftarrow cp. ans[cp. dim-1]; dim \leftarrow dim - 1$ $el \leftarrow cp. ans[0]$ $i \leftarrow 0$ $j \leftarrow 4^k i + 1$ DE REVINIT LA TOT
SUB ASTA(Complexitate $O(\log n)$)

Reprezentare: tablou multidimensionala (leni'ori)

căt timp $j \leq \text{cp. dim}$ executăminim $\leftarrow j$ pentru $k \leftarrow j+1, j+3$ executădacă $j \leq \text{cp. dim}$ și $(\text{cp. ans}[k], \text{cp. ans}[\text{minim}])$ este unaminim $\leftarrow k$ minim $\leftarrow j$ dacă $\text{cp. H}(el, \text{cp. ans}[j])$ $j \leftarrow \text{cp. dim} + 1$

altfel

 $\text{cp. ans}[i] \leftarrow \text{cp. ans}[j]$ $i \leftarrow j$ $j \leftarrow 4^k i + 1$

SUB 2

A.

Pentru funcția g : $cH = cT = c\Delta$

$$g(n) = \sum_{i=1}^n i = \frac{n(n+1)}{2} = \Theta(n)$$

???

Pentru funcția f : $cH = cT = c\Delta$

$$f(n) = \sum_{i=1}^n \sqrt{i} = \sum_{i=1}^n \sqrt{i} = \sqrt{1} + \sqrt{2} + \dots + \sqrt{n}$$

B.

Adresare deschisă:

→ toate el. sunt memorate în interiorul tablei

→ se examinăsuccesiv locurile începând cu val. de dispersie

c	35	2	18	6	3	10	8	5
d(i)	5	2	1	6	3	0	8	5

initializare:

poz	0	1	2	3	4	5	6	7	8	9
elem.	nil									



- 1) cu verificare liniară
- 2) cu verificare patratică
- 3) cu dispersia dublă

Adăugarea 35: poziția 5 este libera, adăugăm acesta

poz	0	1	2	3	4	5	6	7	8	9
elem.	nil	nil	nil	nil	nil	35	nil	nil	nil	nil

Adăugarea 2: aplicăm principiul de inserie

poz	0	1	2	3	4	5	6	7	8	9
elem.	nil	nil	2	nil	nil	35	nil	nil	nil	nil

La fel 12, 6, 3, 10

poz	0	1	2	3	4	5	6	7	8	9
elem.	10	Nil	2	3	Nil	55	6	Nil	18	Nil

Adăugăru 8: Poziția 8 este ocupată, găsimu poziția următoare liberă și adăugăm acolo (în acest caz 9)

poz	0	1	2	3	4	5	6	7	8	9
elem.	10	Nil	2	3	Nil	55	6	Nil	18	8

Adăugăru 5: Conformu aceluiși principiu ca la pasul anterior, adăugăm la poz 7

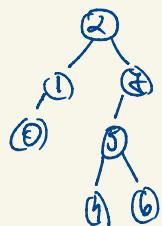
poz	0	1	2	3	4	5	6	7	8	9
elem.	10	Nil	2	3	Nil	55	6	5	18	8

c. b) fapt

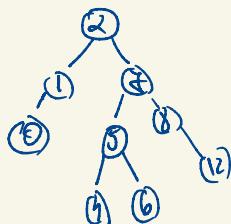
Acență a învaiatice nu este în totdeauna aderătoare.

Să considerăm cazul în care, adăugările - e pe x în arbore, să adăugarea următoare o facă y, x va deveni străinătate al lui y. Evident, dacă l-am inserat mai întâi pe y, apoi pe x, x nu ar mai putea fi străinătate al lui y, deoarece nu se produce același efect.

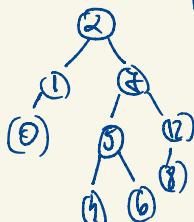
ABC initial



inserarea întâi a apel 12



inserarea întâi 12 apoi 8



$$10n^2 < 5 \cdot \alpha^{m-1} / :5 \Leftrightarrow n^2 < \alpha^{m-1} / : \alpha \Leftrightarrow n^2 < \alpha^{m-2}$$

incorectă

E.

AB: Mod:

Nod: Mod

e: Telenu

st: Nod

dr: Hed

Parcursarea în inordine

folosim algoritmul de parcursare în inordine și actualizăm conținutul pe parcurs

Subiectivă numără (ab, c, nr)

(Complexitate O(n))

{ pre: ab : arbore binar

e : Telenu, valoarea din ab pt. care urmărește să găsimu alt asc.

post: nr: întreg, numărul asociat valoii e în parcursarea în inordine

}

conținut \leftarrow 0; găsit \leftarrow false

crează (n) // nodul următor este adresa de noduri

p \leftarrow ab.nod // începeam cu nodul arborelui

cât timp (\neg ido (n) \vee p != NIL) \wedge \neg găsit executa

{ cât timp p != NIL execuță // parcursul descedentă
adaugă (n, p) // adăugă o nodului n-o adăugare
p \leftarrow [p].st // în stivă

sterge (n, p) // nodul stocat în p nu mai are descendenți
conținut \leftarrow conținut + 1 // înălță stângă de la părem conținut

dacă [p].e = e atunci // dacă suntem la element
nr \leftarrow conținut // în parcursul mai departe
găsit \leftarrow true

p \leftarrow [p].dr // ne apucăm de parcursul descedentă

„atârgă o jocuri drept al modului

dacă găsit = face atunci

$\text{nr} \leftarrow -$,

SUB 3

1. $T(1) = 1$

$$T(2) = 2T(1) = 2$$

$$T(3) = 2 \cdot T(2) = 2 \cdot 2 \cdot T(1) = 2^2$$

$$T(4) = 2 \cdot T(3) = 2 \cdot 2^2 = 2^3$$

— —

$$T(n) = 2T(n-1), n >= 2$$

$$T(n) = 2T(n-1) = 2 \cdot 2 \cdot T(n-2) = 2^2 \cdot T(n-2) = 2^3 \cdot 2T(n-3) = 2^3 \cdot T(n-3) = \dots = 2^{n-1} \cdot T(1)$$

$$CF = c\Delta = cM$$

2. $M = 5$

$$d = c \bmod 5$$

initial arborei sunt vii. Fiecare pas din tabelă conține un permutație sau o
permutare a pozițiilor inițial vii.

0	→
1	→
2	→
3	→
4	→

Adăugăm 25 în arborele 0:
 $(25)^o$

Adăugăm 4 în arborele 2:
 $(4)^o$

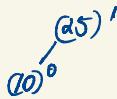
Adăugăm 18 în arborele 3:
 $(18)^o$

Adăugăm 6 în arborele 1:
 $(6)^o$

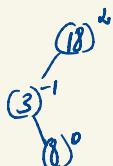
Adăugăm 3 în arborele 3:



Adăugăm 10 în arborele 0:



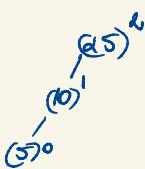
Adăugăm 8 în arborele 3:



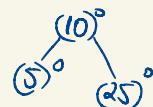
Arborele nu mai este echilibrat. Aplicăm o dublă rotație spre dreapta.



Adăugăm 5 în arborele 5



Arborele nu mai este echilibrat. Aplicăm o singură rotație spre dreapta.



+ Am mai vor. finisat" a arborelor

c. c) $O(n \log n)$

În următorul său parcurgut totă rețea de rețele pentru a le adăuga în heap face întotdeauna în pas. La fiecare pas, se face o adăugare în heap, care în caz de favorabilitate parcurge tot heapul. \Rightarrow complexitate în caz de favorabilitate $O(n \log n)$.

Arenumea, la stergerea din heap vom avea ocazi complexitate în caz de favorabilitate, $O(n \log n)$.

Deci, overall complexity în cazul de favorabilitate este $n \log n + O(n \log_2 n)$

c2. b)

Atât în adăugare, cât și în stergere, jocul să se facă pe resursele pe, pozitie sau implicat schimbarea el^p, de unde rezultă tipul de lucru

d.

PARCURGEREA ÎN LÂTIME

Nod:

e: TECNI

st.: 1 Nod

dr.: 1 Nod

nivel: mutreg

AB:

Nod: ↑ Nod

idee de rezolvare: parcursuri în lâime
cu actualizare de nivel

Subiectivu vorj (ab, e, e', n₂) (Complexitate O(n))

↑ pre: ab: abele braun

e: TECNI, nod din ab

e': TECNI, nod din ab

post: nod = true dacă ne află pe același nivel
față altfel

↓

nivel₁ ← -1 // în aceste variabile vom păstra niveli argumantelor
nivel₂ ← -1

aceeași (c)

dacă ab. nod != NIL // dacă arăt nu este vid

adaugă (c, ab. Nod)

dacă e = ab. Nod \vee^{an} e' = ab. Nod atunci // verificăm dacă am
nivel₁ ← 0 // găsit unul dintre elemente

[ab. Nod]. nivel ← 0 // asignăm valoarea niv. corresp

↓

cât timp Vida(c) execută

sterge (c, p) // eliminăm el. următor al parcursului

dacă [p]. st != NIL atunci

adaugă (c, [p]. st) // adăugăm în coadă de caderul
nămăg

$[sp]. nt]. nivel \leftarrow [sp]. nivel + 1$ // actualizare niv. dex.

dacă $[p]. nt]. e = e$ sau $([p]. nt]. e = e'$ atunci // dacă
dacă nivel $1 != -1$ atunci

nivel $2 \leftarrow [sp]. nt]. nivel$ verificare dacă
rez \leftarrow nivel $1 =$ nivel 2 autorizația este
afirmativă

afirmativă în calea rez; dacă

do, compararea nivelurilor;

dacă rez, actualizarea

prin urmă nivel

este și în calea rez;

nu este ca rez să depășească nivelul

dacă rez, actualizarea

prin urmă rez este rezultatul

acestei operații de la final

atfel

nivel $1 \leftarrow ([p]. nt]. nivel$

dacă $[p]. dh != NIL$ atunci

adaugă(c , $[p]. dh$)

$[sp]. dh]. nivel \leftarrow [sp]. nivel + 1$

dacă $[p]. dh]. e = e$ sau $([p]. dh]. e = e'$ atunci

dacă nivel $1 != -1$ atunci

nivel $2 \leftarrow [sp]. dh]. nivel$

rez \leftarrow nivel $1 =$ nivel 2

afirmativă în calea rez;

nivel $1 \leftarrow ([p]. dh]. nivel$

atfel

dacă $\text{nivel}_1 = -1 \text{ și } \text{nivel}_2 = -1$ atunci
 $\text{next} \leftarrow \text{false}$

SUB 4

A.

Subalgoritmu Δ (liniu)

pentru $i \leftarrow 1, n$ execută

pentru $j \leftarrow 1, m$ execută
 @ op elementație

$N(N-1)$

M

$$T(1) = 1$$

$$T(2) = 4 + T(1) = 5$$

$$\begin{aligned} T(N) &= N^2 + T(N-1) = N^2 + (N-1)^2 + T(N-2) = N^2 + (N-1)^2 + (N-2)^2 + \dots + 2^2 + T(1) = \\ &= 1^2 + 2^2 + \dots + N^2 = \frac{N(N-1)(2N-1)}{6} \in \Theta(N^3) \quad C\Delta = Cf = CM \end{aligned}$$

B. Dacă este un vector numărătore care realizează condiția $v[i] < v[\alpha^k i]$ și $v[i] < v[\alpha^k i+1]$ (nu există în care $\alpha^k i \leq \alpha^k i+1$ nu depășește dimensiunea vectorului), deci următoare orice nod este numărătore decât fiind năoibă.

C. e) succinu

vidă: verifică dacă precum este $N/2$

aceeași: accesăm informație utilă din punctul precum

aferevoie: placu devine sprețuț. urmă

adăugare: ne creem să nu mai avem, se face legătură cu placu și se actualizează placu

c.

Stiva: $\left\{ \begin{array}{l} \text{vid} \\ \text{vid} \end{array} \right\}$ $\left\{ \begin{array}{l} 4 \\ \text{vid} \end{array} \right\}$ $\left\{ \begin{array}{l} 1 \\ + \end{array} \right\}$ $\left\{ \begin{array}{l} 3 \\ + \end{array} \right\}$ $\left\{ \begin{array}{l} 43 \\ + \end{array} \right\}$ $\left\{ \begin{array}{l} 43 \\ + \times \end{array} \right\}$

Coada: $\left\{ \begin{array}{l} 36 \\ + \times \end{array} \right\}$ $\left\{ \begin{array}{l} 363 \\ + \times \end{array} \right\}$ $\left\{ \begin{array}{l} 363 \\ + \times (-) \end{array} \right\}$ $\left\{ \begin{array}{l} 363 \\ + \times (-) \end{array} \right\}$ $\left\{ \begin{array}{l} 363 \\ + \times (-) \end{array} \right\}$

$$\left\{ \begin{array}{l} 4363 * 12 - (* + \\ \text{vid} \end{array} \right.$$

$$(6 * 3 - 12) * 3 + 4$$

$$\Rightarrow d) 4 ?$$

D.

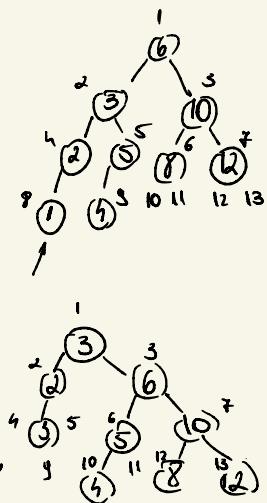
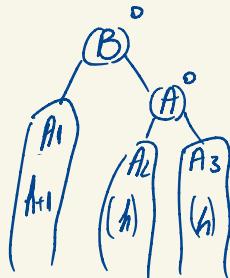
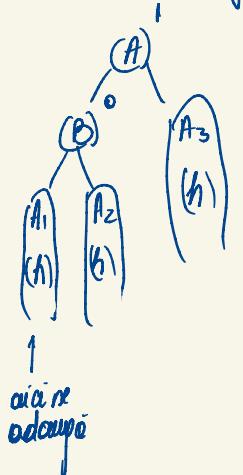
ABC:

ans - \uparrow mutare [1, cap]

dinu - \uparrow mutare [eg]

???

SIMPLA ROTATIE SPRE AREAPTA



Subalgoritmu SRA (af, p)

pre : p: mutare, năd unui subarbore

post : se modifică poziția elementelor în vector a.t. să reflecte noile legături după SRA

$p \Delta \leftarrow \alpha^* p$ // pΔ este poz. fizică stângă al poz. p

$\alpha \Delta \leftarrow cp. ans[\alpha^* p \Delta + 1]$ // αΔ este

SUB 5

A. $T(1) = 1$

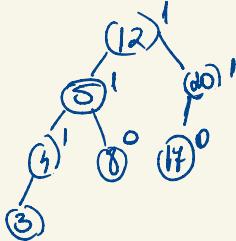
$$T(2) = 1 + T(1) = 2$$

$$T(3) = 1 + T(1) = 2$$

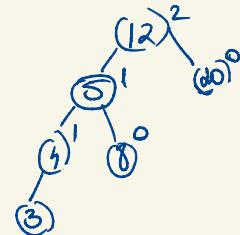
$$T(4) = 1 + T(2) = 1 + 2 = 3$$

$$\begin{aligned} T(n) &= 1 + T(n/2) = 1 + 1 \quad T(n/2) = 2 + T(n/2^2) = 3 + T(n/2^3) = \dots = \\ &= K + T(1) = K + 1 \quad \left. \begin{array}{l} \text{unde } 2^K = n \Rightarrow K = \log_2 n \\ \text{CM} = CF = CD \end{array} \right\} \Rightarrow T(n) = 1 + \log_2 n \in \Theta(\log_2 n) \end{aligned}$$

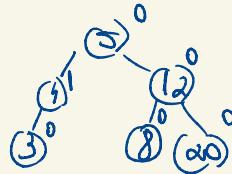
B.



vineau nă pătrageau modul 17 \Rightarrow



Arborele numarul este extins \Rightarrow SRA



C. a), c), d)

Mergesort și HeapSort nu pot face pe orice tip de el. comp.

RadixSort se poate face cu BucketSort pentru fiecare pez. din numere (ultima zecimă, penultima etc.)

c2. a)

Într-un array cu $n \geq 1$, dacă el este mai mare sau egal decât toti fișii lui. Deși urmăre, valoarea cea mai mare va fi data în prima poziție în rezultat.

A.

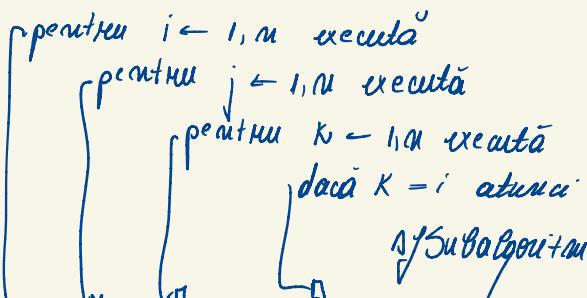
???

SUB 6

A. Subalgoritmu $n(m, i)$

pre: m - întreg, $m \geq 1$
 i - întreg

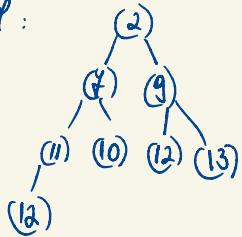
g



B

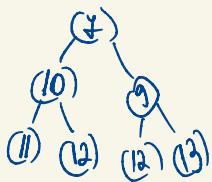
deducție: -

B. initial:



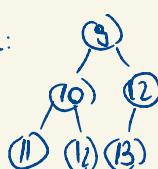
Stergerea ne face doar pe el din vîrful arcului său. Stergem 13. sun. Puteam să văd unde ești și tu din arcul său mi îl vorbesc până se restabilește propriu de arcul său.

Stergere 1:

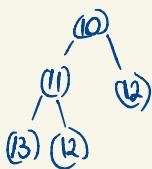


Analog următoarele.

Stergere 2:



Stergere 3:



c₁ b) m.c)

Dacă alegem o tabelă de dispersie cu n locuri, atunci complexitatea este $\Theta(n)$.

Dacă tabela este mai puțin de n locuri, sortarea nu mai este garantată.

Dacă tabela are mai mult de n locuri, complexitatea este $\Theta(N)$, N - nr de el din tabelă.

Dacă complexitatea este $\Theta(n)$, atunci se apără în $O(n)$.

c₂ b) fals

Adăugarea într-o tabelă de dispersie nu face niciun lucru în timp constant.

D. PARCURGERE ÎN PREORDINE

Folosim parcurgerea în preordine pentru un α să construim codură unui word =

AB:

v : TPerche [f₁, q₀]

dinu : Tattreg

TPerche:

e : TEcam

cod : zîr de caractere

Sub algoritm preordine (ab, e, cod)

{ pre : ab : astore binar

e : TEcam, modul pt căre vă se nu să găsim codul

post : cod : zîr de caractere, codul el. e

aceeași (n) // creem rotiva

dacă dinu != 0 atunci

adăugă (a, 1) // adăugăm word în rotivă

v f₁] cod = '1'

cât timp > vînd (n) execută

Storage (Δ , p)

dacă $\alpha^* p + 1 \leq \dim \text{si ab. } v[\alpha^* p + 1]! = -1$ atunci

adaugă (Δ , $\alpha^* p + 1$)

$\text{ab. } v[\alpha^* p + 1]. \text{cod} = \text{concatenaza}(\text{ab. } v[p]. \text{cod},$

dacă $\text{ab. } v[\alpha^* p + 1]. e = e$ atunci $'0'$

$\text{cod} \leftarrow \text{ab. } v[\alpha^* p + 1]. \text{cod}$

SfSubaplicație

dacă $\alpha^* p \leq \dim \text{si ab. } v[\alpha^* p]! = -1$ atunci

adaugă (Δ , $\alpha^* p$)

$\text{ab. } v[\alpha^* p]. \text{cod} = \text{concatenaza}(\text{ab. } v[p]. \text{cod}, '2')$

dacă $\text{ab. } v[\alpha^* p]. e = e$ atunci

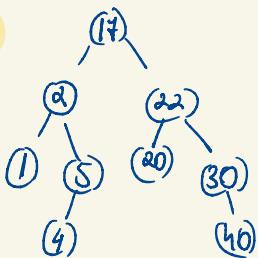
$\text{cod} \leftarrow \text{ab. } v[\alpha^* p]. \text{cod}$

SfSubaplicație

SUB 7

A. -

B.



Rădăcina are în descendență stâng și descendență dreptă, deci primul în locul ei cel mai mic element din arborele drept, apoi urmatorul nodul aferent acestui cel mai mic element.

C. a)

La lista implementată recursive avem acces la el de pe o anumită poziție în timp constant, pe cînd la listele implementate operația se face în timp linear

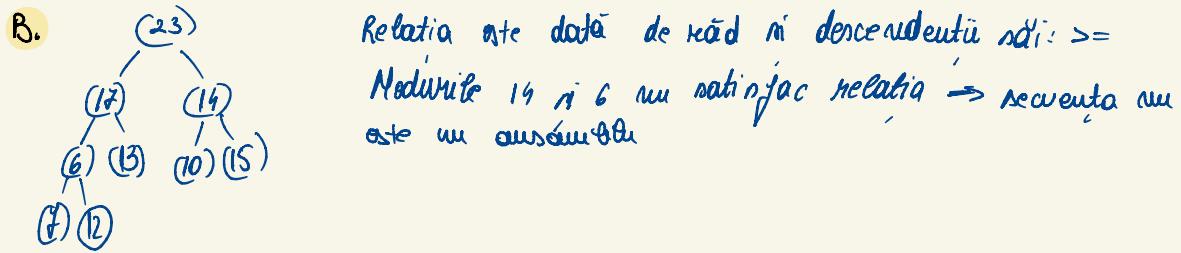
C. b)

D. fărăt

SUB 8

A. $T_{\text{rec}}(i) = 1 + T_{\text{rec}}(i/2) = 2 + T_{\text{rec}}(i/2^2) = 3 + T_{\text{rec}}(i/2^3) = \dots = K + T_{\text{rec}}(1) = K + 1$, unde $2^K = i \Leftrightarrow K = \log_2 i$
 $\Rightarrow T_{\text{rec}}(i) = \log_2 i + 1 \in \Theta(\log_2 i)$

$$\begin{aligned} T_{\text{op}}(1) &= 1 \\ T_{\text{op}}(n) &= \sum_{i=0}^{\log_2 n} T_{\text{rec}}(2^i) = \log_2 1 + \log_2 2 + \log_2 2^2 + \dots + \log_2 2^{\log_2 n} = \\ &= 1 + 2 + \dots + \log_2 n = \frac{\log_2 n (\log_2 n + 1)}{2} \in \Theta(\log_2^2 n) \end{aligned}$$



C. făcut

C₂. Voi reprez. nriva la fiecare pas

$$1: 6 \quad 2: 63 \quad 3: 632 \quad 4: 6324 \quad 5: 6326 \quad 6: 6-3 \quad 7: -18 \Rightarrow a)$$

Punem operanșii pe nrivă și în ceea ce urmăritul în care dăru de un operator, extragem o operanșă de pe nrivă, aplicăm operația răsăre al cărui rezultat va fi adăugat în nrivă, apoi punem rez. pe nrivă.

D. Parcurgerea în ordine de numărare nivelului

SUB 9

A. $T(1) = 1$

$$\begin{aligned}
 T(n) &= T(n/2) + n - 1 + T(n/2) = 2T(n/2) + (n-1) = 2(2T(n/4) + (n/2 - 1)) + \\
 &+ (n-1) = 4T(n/2^2) + n - 2 + n - 1 = 4T(n/2^2) + n - (2^2 - 1) + \\
 &= 4(2T(n/2^3) + (n/2^2 - 1)) + 2 \cdot n - 2^2 + 1 \\
 &= 2^3 T(n/2^3) + n - 2^2 + 2 \cdot n - 2^2 + 1 = 2^3 T(n/2^3) + 3n - 2^3 + 1 \\
 &= 2^K \cdot T(1) + K \cdot n - 2^K + 1 \text{ unde } n = 2^K e, K = \log_2 n \\
 &= n + \log_2 n \cdot n - n + 1 \\
 &= \log_2 n + 1 \in O(n \log_2 n)
 \end{aligned}$$

B. Adresare directă folosind verificare patratică

$$d(c_1, i) = (d'(c_1 + c_1 \cdot i + c_2 \cdot i^2) \bmod m)$$

cheie	10	22	31	4	15	28	17	88	59
$d(\text{cheie}, 0)$	10	0	9	4	4	6	6	0	4

inserăm 10 pe poz 10:

Pozitie	0	1	2	3	4	5	6	7	8	9	10
cheie											10

inserăm 22 pe poz 0:

Pozitie	0	1	2	3	4	5	6	7	8	9	10
cheie											22

inserăm, în aceeași manieră pe 31 și pe + la poz $d(31, 0)$ după $d(4, 0)$:

Pozitie	0	1	2	3	4	5	6	7	8	9	10
cheie										31	10

$d(15, 0) = 4$, poz 4 este ocupată, inserăm la poz $d(15, 1) = (4 + 1 + 3) \bmod 11 = 8$

Pozitie	0	1	2	3	4	5	6	7	8	9	10
Cheie	12				4				15	31	10

Inversăm la poz 6:

Pozitie	0	1	2	3	4	5	6	7	8	9	10
Cheie	22				4		28		15	31	10

În același numărătore, la 14, $d(14, 1) = (6+1+3) \text{ mod } 11 = 10$. Pozitia este ocupată. Încercăm la $d(14, 2) = (6+2+12) \text{ mod } 11 = 20 \text{ mod } 11 = 9$. Ocupată. $d(14, 3) = (6+3+24) \text{ mod } 11 = 36 \text{ mod } 11 = 3$. Neocupată, putem insera

Pozitie	0	1	2	3	4	5	6	7	8	9	10
Cheie	0			14	4		28		15	31	10

$$\frac{16}{48}$$

$$d(88, 1) = (0+1+3) \text{ mod } 11 = 4$$

$$d(88, 2) = (0+2+12) \text{ mod } 11 = 3$$

$$d(88, 3) = (0+3+24) \text{ mod } 11 = 30 \text{ mod } 11 = 8$$

$$d(88, 4) = (0+4+38) \text{ mod } 11 = 52 \text{ mod } 11 = 8$$

$$d(88, 5) = (0+5+45) \text{ mod } 11 = 80 \text{ mod } 11 = 3$$

$$\begin{array}{r} 16 \\ 3 \\ \hline 48 \end{array}$$

$$d(88, 6) = (0+6+108) \text{ mod } 11 = 114 \text{ mod } 11 = 4$$

$$d(88, 7) = (0+7+149) \text{ mod } 11 = 156 \text{ mod } 11 = 0$$

$$d(88, 8) = (0+8+192) \text{ mod } 11 = 200 \text{ mod } 11 = 2$$

$$\begin{array}{r} 36 \\ 3 \\ \hline 108 \end{array}$$

Inversăm la 2

Pozitie	0	1	2	3	4	5	6	7	8	9	10
Cheie	0		88	14	4		28		15	31	10

$$\begin{array}{r} 11 \\ 19 \\ \hline 154 \end{array}$$

$$d(59, 2) = (4+2+12) \text{ mod } 11 = 18 \text{ mod } 11 = 8$$

Inversăm la *

Pozitie	0	1	2	3	4	5	6	7	8	9	10
cheie	0		88	17	4		28	59	15	31	10

c1) $y(n) = \sum_{i=1}^n 2^i = 2 + 2^2 + \dots + 2^n = 2^{n+1} - 3$

$$2^{n+1} - 3 \in \Theta(2^n)$$

$$2^{n+1} - 3 \in \Theta(3^n) \rightarrow a), c)$$

$$2^{n+1} - 3 \in O(3^n)$$

c2) b) căutarea nu consideră locurile care au fost preuze

b. PARCURGERE IN LATIME

cu diferență între nr. maxim și nr. v. ep. căutat

SUB 10

A. Subproblem $\Delta(i, i)$
 { pre \cup nu întreg, $n \geq 1$; i - întreg $i \geq 1$

↓

tipărește m
 $m \leftarrow [m]$
 $\Delta(m)$

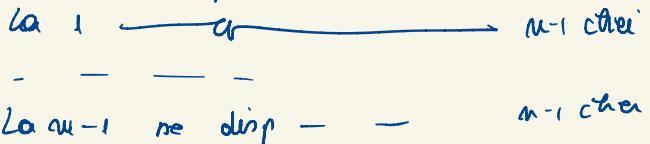
↓

$$T(n) = \dots = \log_2 n + 1 \in \Theta(\log_2 n + 1) \in O(\log_2 n + 1)$$

B. $|U| > m^k m$

Pp. PRA că mulțimea G este submultime a lui U de multimea m care conține chei ce ne dispunem să aibă mulțimea $m-1$ chei.

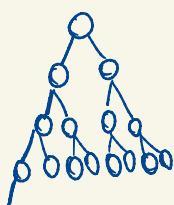
\Rightarrow G este o submultime a lui U de multimea $m-1$ chei.



} \rightarrow

$$\Rightarrow |U| \leq m \cdot (m-1) \rightarrow |U| < m \cdot m \text{ Contradictie!} \rightarrow |U| > m \cdot m$$

C1.



Arboarele are adâncimea 4 \Rightarrow arboarele are 5 nivele.
 Prin urmare 4 rânduri pline $\Rightarrow G: 2^0 + 2^1 + 2^2 + 2^3 = 16 - 1 = 15$ noduri
 iar ultimul nivel conține un nod $\Rightarrow 16$ noduri în total
 $\Rightarrow C)$

- C2. a) dateaza rezultatul nu pot fi reprezentate pe vector (ori adaugarea, ori sortarea nu ar face înțelegător), pe vînd rezultat pot fi reprez. pe vector
 c) dateaza oada adaugă numărul capăt și sterge din celelalte, iar astfel

face anulete op la uu ng. capăt

L. ALGORITM PREDECESSOR

ABC:

nod: ↑ Nod

Nod:

e: TELEM

st: ↑ sted

dh: ↑ Nod

Funcția predecessor (abc, p)

pre p: ↑ Nod, p ≠ NIL, nod din abc

abc: ABC

Complexitate $O(h^2)$

post: se returnează adresa nodului cu ceea cea imediat anteriorul decât cheia din nodul p

dacă $\lceil p \rceil. st \neq NIL$ atunci

predecessor ← maxim ($\lceil p \rceil. st$)

altfel

prec ← predecessor (abc, p)

căt timp prec != NIL \wedge p = [prec]. st execută

p ← prec

prec ← predecessor (abc, p)

predecessor ← prec

□

Funcția maxim (p)

pre: ↑ Nod, p ≠ NIL

Complexitate $O(h)$

căt timp $\lceil p \rceil. dr \neq NIL$ execută

p ← $\lceil p \rceil. dr$

markian = p

Funcția părinte (abc, p)

pre: abc : ABC nr. id

Complexitate O(n)

p : \uparrow Nod, nod din ABC, $p \neq \text{nil}$

post: returnează adresa părintelui nodului p

current = abc.nr.id

dacă current = p atunci
părinte \leftarrow nil

cât timp [current].nr != p și [current].dr != p execută

dacă [p].e \leq [current].e atunci
current \leftarrow [current].nr

altfel

current \leftarrow [current].dr

părinte \leftarrow current

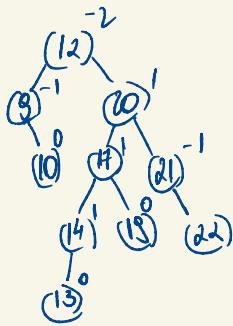
SUB 11

A.

$$T_f(n) = \sum_{i=1}^n 1 = n$$

$$T_{\text{pr}}(n) = \sum_{i=1}^n T_f(i) = T_f(1) + T_f(2) + \dots + T_f(n) = 1+2+\dots+n = \frac{n(n+1)}{2} \in \Theta(n^2)$$

CM = CD



Modul 12 are coș -2, este necesară o LRS

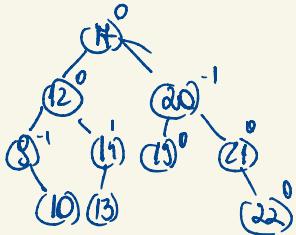
Aducem nodul 17 și nu reacționează.

Legătura nr la dreapta va fi cu poziție 20.

Legătura vecine la dreapta va fi legătura la stânga a modulu 20.

Legătura la stânga a modulu 17 va fi nodul 12.

Legătura vecine la st. a modulu 17 va fi legătura dreapta a modulu 12



c₁. a) c) d) Înălțimea și adâncimea unui arbore au aceeași valoare.
Într-un AVL, înălțimea unui arbore este legată (nu numai de
arbore)

c₂. a) adevărat

Indiferent de ordinul rădăcinii, legăturile se fac după același reguli;
deci următoarea rea. va fi același

d. făcut

SUB 12

A. Este un oraș de mai mult

B.

pe2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
e	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Wm	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

prioritate = 0

el	23	11	8	18	3	19	34	20	18						
d(c)	7	11	8	2	3	3	2	4	2						

pe2 + e liberă. Adăugăm 23

pe2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
e	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Wm	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Similare pt. 11 8 18 3

pe2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
e	-	-	18	3	-	-	-	-	-	-	-	-	-	-	-
Wm	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Adăugăm 19. Pe 3 este ocupată, adăugăm la prima pe2 liberă și facem legătură

pe2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
e	19	-	18	3	-	-	-	-	-	-	-	-	-	-	-
Wm	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-

prioritate = 1

Similare pt. restul el.

per	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
e	19	34	18	3	20	18		33	8			11			
Wron	-	5	1	0	-	-	-	-	-	-	-	-	-	-	-

C1. a) Adaugări la în capăt

Stergerea nu are presupunere iterarea până la tot

C2. c) initializare cu capătul S[1], apoi ne crește capacitatea

A.

CP3:

$$n : \text{TElem} \times \text{TElem} \rightarrow \{T, F\}$$

ans : ansamblu binar

Subalgoritm β terge (cp , e)

{ prel: cp - CP3, β nevidă

???

post: e - al treilea cel mai probabil element
 cp - nu mai conține el. e

y

Complexitate: $O(\log n)$

β terge (ans, e_1)

β terge (ans, e_2)

β terge (ans, e_3)

adăugă (ans, e_1)

adăugă (ans, e_2)

D

SUB 13

A. $T(1) = 1$

$$T(n) = \sum_{j=1}^n 1 + T(n/2) = n + T(n/2) = n + n/2 + T(n/2^2) = \dots$$

$$\Rightarrow T(2^k) = 2^k + 2^{k-1} + \dots + 1 = 2^{k+1} - 1 \quad \left. \begin{array}{l} \\ \\ \end{array} \right\}$$

$$n = 2^k \Rightarrow k = \log_2 n$$

$$\Rightarrow T(n) = 2^{\log_2 n + 1} - 1 = 2 \cdot 2^{\log_2 n} - 1 = 2n - 1 \in \Theta(n)$$

B. Adăugăm (1). Numarul de =

$\sqrt{1}$

Adăugăm (5)

(1)

Hu ne potriveste nici
unul nodul

(5)

(1)

(1)

(1)

(1)

(2)

(1)

(3)

(1)

(3)

(1)

(3)

(1)

(2)

(1)

-

(2)

(1)

(4)

(1)

(3)

(1)

(5)

(1)

(2)

(1)

(3)

(1)

-

C. $\frac{\text{faza}}{\text{naste}} = 2$

\Rightarrow se inseră la data [1] $\rightarrow a$)

c2 14 2 11 1 3 10 30 7 40 $\Rightarrow d)$

Vor ilustrație mode de lucrare PDS

14 {≤ 11 {≤ 13 {≤ 13 10 30 {≤ 3 10 30 {≤ 10 30 {≤ 30 7 {≤ 90 {≤ 10
1 -

D. Vom parcurge în ordine mărginită. cum se anunță nodurile, să nu se tăiem niveliile

AB :

v : TPerche [0, cap]

m : Antrep

T Perche :

niv : Antrep

e : TELEM

Sub algoritmul tipărește (ab, e, k)

{ pre : ab : arbore binar k : Antrep

 e : TELEM element din ab

post : tipărește — —

crează (c) // creare coada

dacă ab. n > 0 atunci

adaugă (1)

dacă ab. v[1]. e = e atunci

ab. v[1]. niv = 0

altfel

ab. v[1]. niv = -1

înțeț timp "vîdă(c)" execută

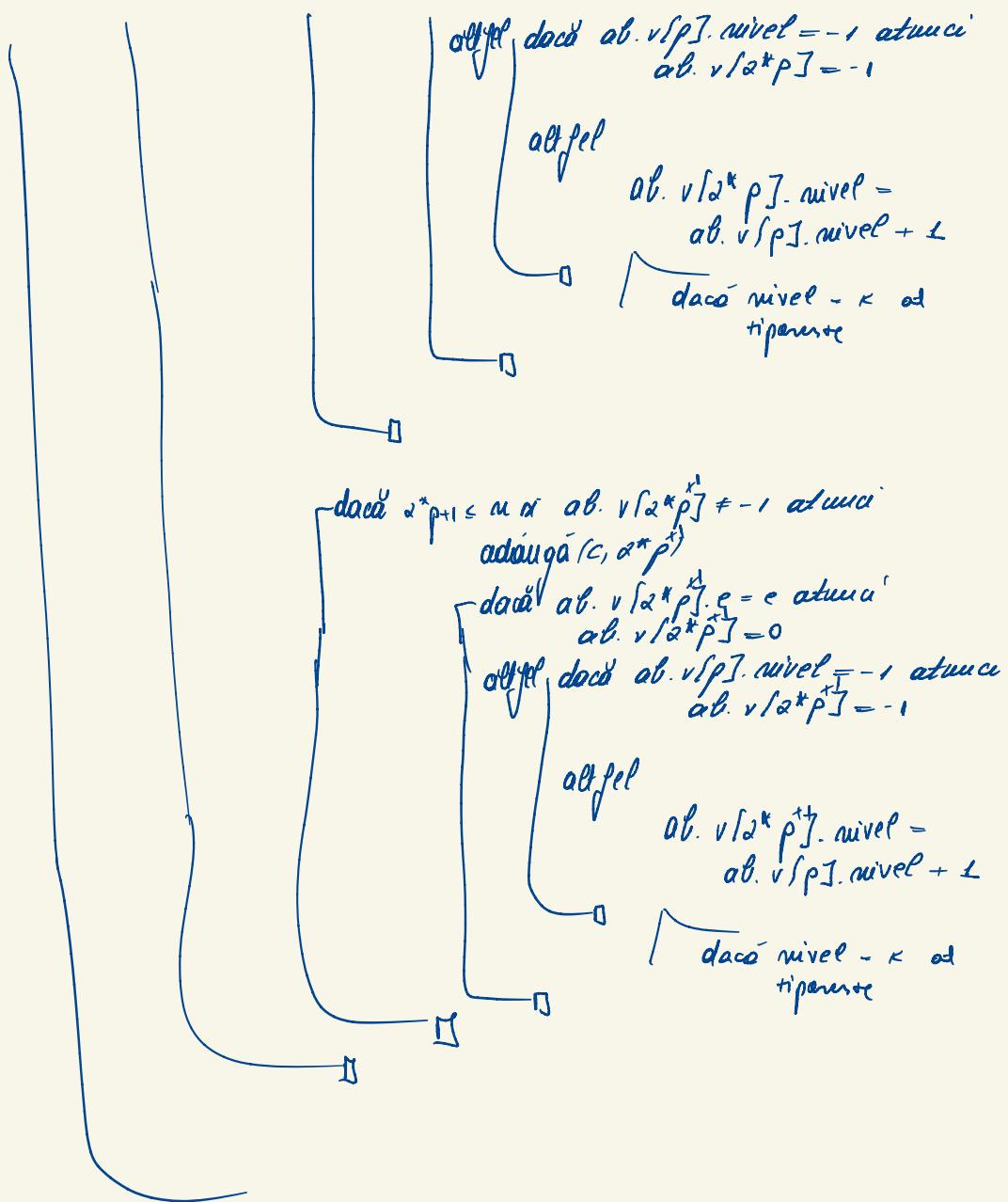
șterge (c, p)

dacă $\alpha^* p \leq n$ și ab. v[$\alpha^* p$] = -1 atunci

adaugă (c, $\alpha^* p$)

dacă ab. v[$\alpha^* p$]. e = e atunci

ab. v[$\alpha^* p$] = 0



SUB 14

A. $T(n) = T(n/2) + \sum_{j=1}^{j=n-1} \sum_{k=1}^{j+1} 1 + T(n/2) =$

$$\left\{ \sum_{j=1}^{n-1} j+1 = 2+3+\dots+n = \frac{n(n+1)}{2} - 1 = \frac{n^2+n-2}{2} = \frac{n^2+2n-n-2}{2} = \frac{(n+2)(n-1)}{2} = \frac{(n-1)(n+1)}{2} \right.$$

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

$$+ 2 \left(\frac{(2^{k-1}+1)(2^{k-1}+2)}{2} - 1 + 2T(2^{k-2}) \right) = \frac{(2^k-1)(2^k+2) + (2^{k-1}-2)(2^{k-1}-2)}{2} - 1 - 2 + 2^k T(2^{k-2}) =$$

$$- (2^k-1)(2^k+2) + (2^k-2)(2^{k-1}-2) - 1 - 2 + 2^2 \left/ \left(\frac{(2^{k-2}-1)(2^{k-2}+2)}{2} - 1 + 2^3 T(2^{k-3}) \right) \right. =$$

$$= \frac{(2^k-1)(2^k+2) + (2^k-2)(2^{k-1}-2) + (2^{k-2}-2)(2^{k-2}-2)}{2} - 2 - 2^2 + 2^3 T(2^{k-3}) = \dots$$

$$= \frac{(2^k-1)(2^k-1) + (2^k-1)(2^{k-1}-2)(2^{k-1}-2)(2^{k-2}-2) + 2^k(2^1+1)(2^1-2)(2^1-1)}{2} +$$

$$+ 2^k \cdot T(1) = \frac{2^k(2^{k+2} - 2^k + \dots + 2^1 + k)}{2} - (2^k - 1) + 2^k =$$

=

B. -

c. Ambutele vinute care adăunătoare → Are 4 niveuri

Ambutele este plin → are ocazii mai multe de mediu pe acelasi nivel

$$\rightarrow 2^0 + 2^1 + 2^2 + 2^3 = 2^4 - 1 = 15 \text{ mediu} \Rightarrow e)$$

c₂ b)

Împărtirea cantitățea de verificat în jumătate pentru găsirea elementului logul de favorabilitățile este când cantitatea ajunge la 1.

Pp. că n-ai făcut k pasi și ca cant. să ajunga la 1 $\Rightarrow n/2^k = 1$
 $\rightarrow 2^k = n \Rightarrow k = \log_2 n$, de unde rezultă raportul logarithmic

A. Vom folosi un curs de la pt. a ajunge la primele cele mai mari și cel din urmă

Funcție $\text{muua}(\text{listă}, k^i)$

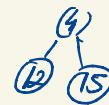
(12) (9) (15) 20 3 2 1
21

{ pre: listă : Intreg[], el mult >= 0, distincte

k : Intreg, $K > 0$

post: Mult muua

menajă (ans, $c = 1$)



it ← iterator (listă)

i ← 1

căt timp $i \leq K$ și valid(it) execută

element (it, e)

adăugă (ans, e)

iterator (it)

$$T(n) = \sum_{i=1}^K \log_2 K + \\ + \sum_{i=K+1}^n \log_2 K + \\ + \sum_{i=1}^K \log_2 K$$

$$= n \log_2 K + K \log_2 K \\ \in O(n \log_2 K)$$

căt timp valid(it) execută

prinu (ans, e)

element (it, e')

dacă $e' > e$ atunci

sterge (ans, e)

adăugă (ans, e')

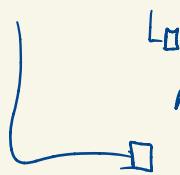
□

meniu ← 0

{ pentru i ← 1..K execută

sterge (ans, e)

meniu ← meniu + c



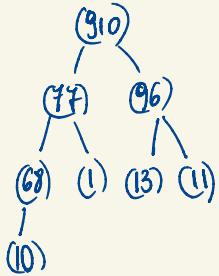
Area \leftarrow sum

SUBIE

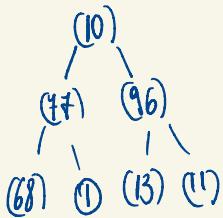
A. $T_f(n) = \sum_{i=1}^{\log_{10} n} 1 = \log_{10} n$

$$T_A(n) = \sum_{i=1}^n T_f(2^{i-1}) = \log_{10} 3 + \log_{10} 5 + \dots + \log_{10}(2^n+1)$$

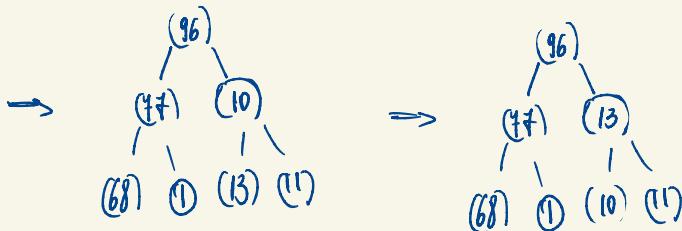
Se găsește 910.



Pas I: Aducem în locul lui 910 ultimul el din aranjament



Pas II: Înlocuim 10 pătră cu un număr care face propria de acesta și care este mai mare decât cel mai mare joi al său, numărul acesta; dacă nu, îl înlocuim cu cel mai mare joi al său.



c1. Căutăm binar pe către trebuire să inserăm + stăfămu el. \Rightarrow
 $\Rightarrow O(\log_2 n) + O(n) \Rightarrow O(n)$

Căutarea sequentială și costul găsirii pe către el \Rightarrow
 $\Rightarrow K$ pasi pt găsirea oprii $n - K$ pasi pt schimbare $\Rightarrow \Theta(n)$
 $\Rightarrow b) c)$

c2. a)

La adăugarea în sirul merged, se adaugă mai întâi se din primul cu ambele de la eg.

b. parcurgere în lățime

SUB 16

$$\text{A. } T(n, i) = T(n/2, i-1) + \lceil \log_2 n - 1 \rceil n^2 + T(n/2, i-1)$$

$$= 2T(n/2, i-1) + \lceil \log_2 n - 1 \rceil n^2$$

Re RG eä ne mnoepe cu i par

$$= 2T(n/2, i-1) + n^2$$

$$= 2(2T(n/2^k, i-2)) + n^2$$

$$= 2^2 T(n/2^k, i-1) + n^2$$

$$= 2^2 (2T(n/2^3, i-3) + n^2) + n^2$$

$$= 2^3 T(n/2^3, i-3) + (2^2 + 1)n^2$$

$$= 2^3 (2T(n/2^4, i-4)) + (2^2 + 1)n^2$$

$$= 2^4 T(n/2^4, i-4) + (2^2 + 1)n^2$$

$$= 2^4 (2T(n/2^5, i-5) + n^2) + (2^2 + 1)n^2$$

$$= 2^5 T(n/2^5, i-5) + (2^4 + 2^2 + 1)n^2$$

$$= 2^K T(1) + (2^K + 2^{K-2} + \dots + 2^4 + 2^2 + 1)n^2$$

$$S = 1 + 2^2 + \dots + 2^K$$

$$2^K S = \underbrace{2^2 + 2^4 + \dots + 2^K}_{S-1} + 2^{K+2}$$

$$\text{or } 4S = S - 1 + 2^{K+2} \Rightarrow 3S = 2^{K+2} - 1 \text{ or } S = \frac{2^{K+2}-1}{3}$$

$$\Rightarrow T(n) = 2^K + \frac{2^{K+2}-1}{3}n^2 = 2^{\log_2 n} + \frac{4 \cdot 2^{\log_2 n} - 1}{3}n^2 =$$

$$= n + \frac{4n-1}{3}n^2 \in \Theta(n^3)$$

B. -

c1. d) Adaugări pe unul sau mai mult loc liber

c2. c) Deplasare după stângă: $fata = i$, $nate = n$, $fata = npate + 1$
 deci $fata$ presupune $fata = npate$, deci adăugarea poate face
 ca npatele să ajungă la poziția n , sau că o poz. în fata fetei.
 Stergerea crește atributul $fata$ și îl adaptează de npate, numărul
 diferență este de 1, și apoi trebuie să fie $npate - fata + 1$
 Vida nu modifică atributele

ABC:

e: TELEAU [0, dimu - 1]

At: TUTREG [0, dimu - 1]

dh: TUTREG [0, dimu - 1]

priorLibere: TUTREG

năd: TUTREG

Subalgoritm adaugă⁽ⁱ⁾ (abc, e)

↑ pre: abc : ABC

e : TELEAU

post: se adaugă pe modul c

 $p \leftarrow abc \cdot nad$ $pozitate \leftarrow -1$ dacă $time(p + -1) < executa$ $pozitate \leftarrow p$ dacă $e \leq abc.e[p]$ atunci $p \leftarrow abc.e[n[p]]$

altfel

 $p \leftarrow abc.dh[p]$

e [prinuLiber] = e

dacă parinte = -1 atunci

abc. st ← prinuLiber

altfel

dacă e ∈ abc. e [parinte]

abc. st [parinte] ← prinuLiber

altfel

abc. dr [parinte] ← prinuLiber

prinuLiber ← abc. st [prinuLiber]

(permitem că este de loc
altfel de laq și prin să rupă st)

SUB 17

A. $Tg(n, i) = 1 + T(n-1, i+1) = 2 + T(n-2, i+2) = \dots = k + T(n-k, i+k)$

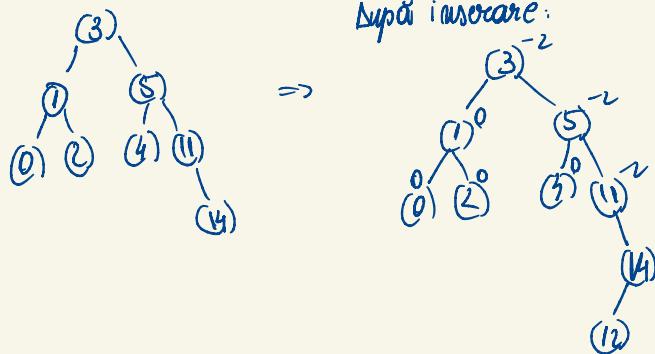
unde $n-k = i+k \Rightarrow 2k = n-i \Rightarrow k = \frac{n-i}{2}$

$$\Rightarrow Tg(n, i) = \frac{n-i}{2}$$

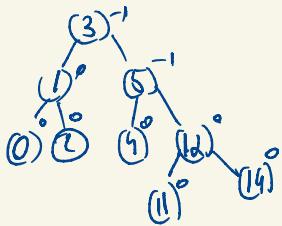
$$Th(n, m) = 1 + Th(n-1, m+1) = 2 + Th(n-2, m+2) = \dots = n + Th(0, m+n)$$

$$= n + Tg(n+m, 1) = \frac{n+n-1}{2} = \frac{3n+nm-1}{2} \in \Theta(n+m)$$

B.



Se aplică DRS în modul 11



- C. b) Pentru un arbore binar plin, fiecare nivel nu este complet, numărul de fiecare nod interior este exact 2 și

C2 facut

D. facut

A.

Subalgoritmu $s(n)$
 ș păre: n întreg, $n \geq 1$

 $i = 1$ căt timp $i \leq n$ execută $j = 1$ căt timp $j \leq n$ execută
 $j \leftarrow \alpha^k j$

D

 $\leftarrow \alpha^k i$

n

$$T(n) = \sum_{i=1}^{\log_2 n} \sum_{j=1}^{\log_2 n} 1 = \sum_{i=1}^{\log_2 n} \log_2 n = (\log_2 n)^2 \in \Theta((\log_2 n)^2)$$

B. Adresare deschisă cu dispersie directă

$$d(c, i) = (d_1(c) + i \cdot d_2(c)) \bmod m$$

Val	10	22	31	4	15	28	17	88	59
$d_1(c)$	10	0	3	4	4	6	6	0	1
$d_2(c)$	1	3	2	5	6	9	8	9	10
initial									

Poz	0	1	2	3	4	5	6	7	8	9	10
e											

 $p_{mijlociu} = 0$

Poz	0	1	2	3	4	5	6	7	8	9	10
e	22			4		5		7		8	31 10

$$d(15,1) = (4 + 1 \cdot 6) \bmod 11 = 10$$

$$d(15,2) = (4 + 2 \cdot 6) \bmod 11 = 16 \bmod 11 = 5$$

Poz	0	1	2	3	4	5	6	7	8	9	10
e	11				4	15				31	10

Poz	0	1	2	3	4	5	6	7	8	9	10
e	11				4	15	d8			31	10

$$d(17,1) = (6 + 1 \cdot 8) \bmod 11 = 3$$

Poz	0	1	2	3	4	5	6	7	8	9	10
e	11			14	4	15	d8			31	10

$$d(88,1) = (0 + 1 \cdot 8) \bmod 11 = 9$$

$$d(88,2) = (2 \cdot 8) \bmod 11 = 8$$

Poz	0	1	2	3	4	5	6	7	8	9	10
e	11			14	4	15	d8	88		31	10

$$d(59,1) = (4 + 10) \bmod 11 = 3$$

$$d(59,2) = (4 + 20) \bmod 11 = 2$$

Poz	0	1	2	3	4	5	6	7	8	9	10
e	11		54	14	4	15	d8	88		31	10

c₁ a)

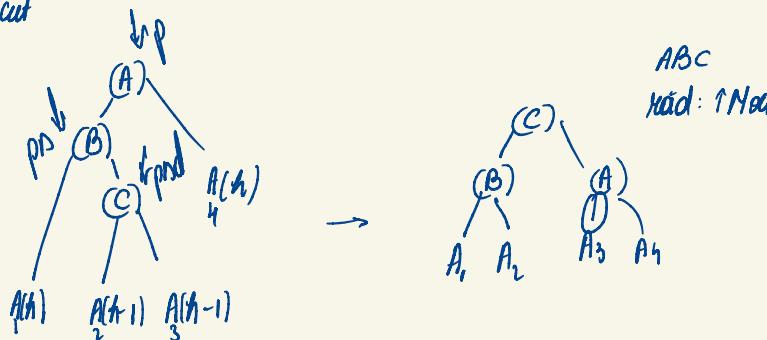
Toate cel mult nœuduri deosebiti sunt în juc

Toate nœurile sunt pline și nu sunt de ultimul

că? ??

c₂ făcut

Δ.



Nœd:
 e: 1 elem
 nt: 1 Nœd
 dr: 1 Nœd
 h: Indreg

Funcția $\Delta RA(p)$

{ prie: p - nœduri nerecalculați

$$p\Delta \leftarrow [p]. \text{nt}$$

$$pnd \leftarrow [p\Delta]. \text{dr}$$

 $\theta(1)$

$$[p]. \text{nt} \leftarrow [pnd]. \text{dr}$$

$$[p\Delta]. \text{dr} \leftarrow [pnd]. \text{nt}$$

$$[pnd]. \text{dr} \leftarrow p$$

$$[pnd]. \text{nt} \leftarrow p\Delta$$

ne recalculează nouă înalțime

$$\Delta RA \leftarrow pnd$$

A -

B. 3.

C. a) există arbore binar vizu

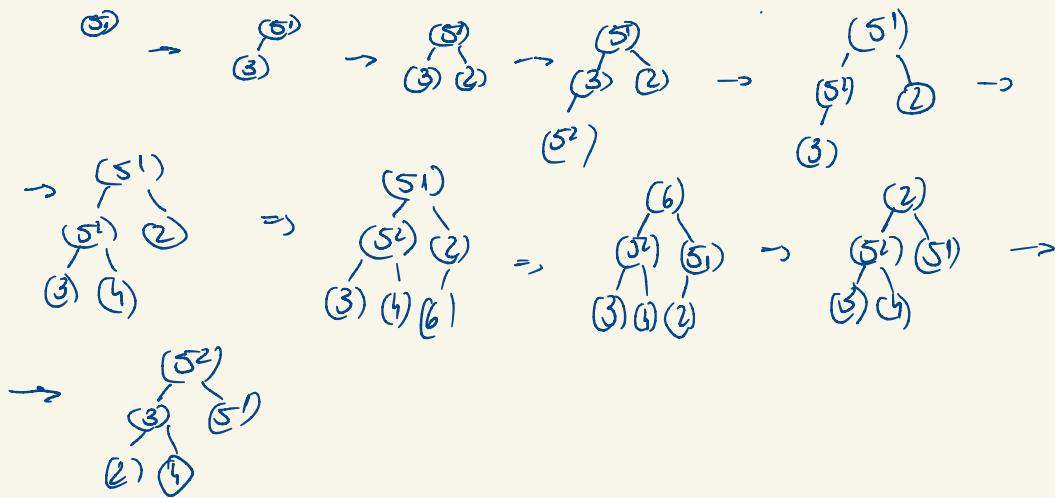
c) Nu neapărat, depinde de ce fel de arbore e verba

d) Nu neapărat, depinde de ce fel de graf e verba

c.) b) fals

Operatiile din cadrul bălu pot schimba ordinea unor el. egale

5, 3, 2, 1, 5, 4, 6



A. fals

A.

Subalgoritmul $\sigma(m, m)$

$\left\{ \begin{array}{l} m: \text{intreg } i=1, m: \text{intreg}, m \geq 1 \end{array} \right.$

daca $m > 1$ atuncipentru $i=1, m$ executăpentru $j=1, m$ executădaca $m = j$ atunci $j \leftarrow m+1$ 

■

 $\Delta(m-1, m)$

$$\text{cb: } T(m) = \sum_{i=1}^m \sum_{j=1}^m 1 + T(m-1) = \sum_{i=1}^m m + T(m-1) = m^2 + T(m-1) =$$

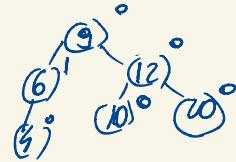
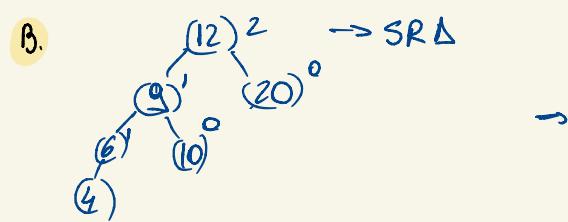
$$= m^2 + (m-1)^2 + (m-2) = \dots = m^2 + (m-1)^2 + \dots + 2 + T(1) =$$

$$= \frac{m(m-1)(2m-1)}{6} \in \Theta(m^3)$$

$$\text{CF: } \sum_{i=1}^m 1 + T(m-1) = m + T(m-1) = m + m-1 + T(m-2) = \dots =$$

$$= m + (m-1) + \dots + 2 + T(1) = \frac{m(m+1)}{2} \in \Theta(m^2)$$

$$\Rightarrow \Theta(m^3)$$



Axatorele este dezchisă brat și
probabilitatea nu se pe cînd absolut
12. Vom aplica acela SLD
deosebere obiectivă să modul
nu plus este în decr. stările a
obiectivă lui răzng

Refacem legăturile

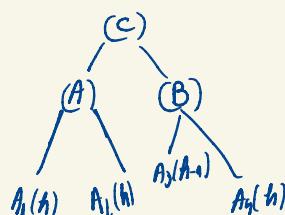
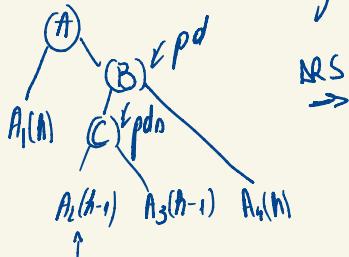
C₁. b)

Este clar, deosebere iteratorul are metode precum prim, element, următor,
ce fac lîsăză parcurgerea conținutului lui el cu e.p.

C₂. facăt

D.

ABC:
Hd: ↑ Hd
Md: ↑ Md
Nd: ↑ Nd
At: ↑ At
dt: ↑ dt
h: ↑ h



Functie DRS(p)

' pre: $p: \text{Nod}$, $p \neq \text{Nil}$

$pd \leftarrow [p].dr$

$pdn \leftarrow [pd].nt$

$[p].dr \leftarrow [pd].nt$

$[pd].nt \leftarrow [pdn].dr$

$[pdn].nt \leftarrow p$

$[pdn].dr \leftarrow pd$

$[p].h \leftarrow \text{măritime}(p)$

$[pd].h \leftarrow \text{măritime}(pd)$

$[pdn].h \leftarrow \text{măritime}(pdn)$

$DRS \leftarrow pdn$

Functie măritime(p)

' pre: $p: \text{Nod}$

dacă $p = \text{Nil}$ atunci

măritime = -1

altfel

$\text{măritime} \leftarrow \max \{ h([p].nt), h([p].dr) \} + 1$

Functie $h(p)$

' pre: $p: \text{Nod}$

dacă $p = \text{Nil}$ atunci

$h \leftarrow -1$

altfel

$h \leftarrow [p].h$

SUB 100

A. $T(n) = 4T(n/2) = 4 \cdot 4T(n/2^2) = 4^2 \cdot T(n/2^2) = \dots = 4^k T(1) = 4^k$
 unde $n = 2^k \Rightarrow k = \log_2 n$
 $\Rightarrow T(n) = 4^{\log_2 n} = (2^2)^{\log_2 n} = 2^{2\log_2 n} = 2^{\log_2 n^2} = n^2 \in \Theta(n^2)$

B. După inserări, tabloul va arăta astfel:

Poz	0	1	2	3	4	5	6	7	8
elem	18	29	19	20	9	5	15	33	14

Vine să inserăm 28, adică el. de la poz 4.

$i=1$

Pozitie de la $j=2$
 $c = 19 \Rightarrow d(19) = 1 \rightarrow$ mutarea datele

Poz	0	1	2	3	4	5	6	7	8
elem	18	19	19	20	9	5	15	33	14

Continuăm cu $i=2$

Pozitie de la $j=3$
 $c = 20 \Rightarrow d(20) = 2 \rightarrow$ mutarea datele

Poz	0	1	2	3	4	5	6	7	8
elem	18	19	20	20	9	5	15	33	14

Continuăm cu $i=3$

Pozitie de la $j=4$
 $c = 9 \Rightarrow d(9) = 0 \Rightarrow$ mutarea datele

Poz	0	1	2	3	4	5	6	7	8
elem	18	19	20	9	9	5	15	33	14

Continuăm cu $i=4$

Pozitie de la $j=5$
 $c = 5 \Rightarrow d(5) = 5 \rightarrow$ ok

$$\begin{cases} j=6 \\ c=15 \end{cases} \rightarrow d(15) = 6 \rightarrow \text{OK}$$

$$\begin{cases} j=7 \\ c=33 \end{cases} \rightarrow d(33) = 6 \rightarrow \text{OK}$$

$$\begin{cases} j=8 \\ c=17 \end{cases} \rightarrow d(17) = 1 \rightarrow \text{OK}$$

$$\begin{cases} j=9 \\ c=18 \end{cases} \rightarrow d(18) = 0 \rightarrow \text{OK}$$

$$\begin{cases} j=10 \\ c=19 \end{cases} \rightarrow d(19) = 1 \rightarrow \text{OK}$$

$$\begin{cases} j=11 \\ c=20 \end{cases} \rightarrow d(20) = 2 \rightarrow \text{OK}$$

$$\begin{cases} j=12 \\ c=9 \end{cases} \rightarrow d(9) = 0 \rightarrow \text{OK}$$

$j=13 \rightarrow$ am ajuns de unde sun plecat, niciu cbrei e cum este
afectat de păreașă pe j

Rez	0	1	2	3	4	5	6	7	8
etruu	18	19	20	9	-	5	15	33	14

c1) Jocuri

c2) e) În cazul în care nu mai sunt locuri, se poate redimensiona

D)

AB:
V: TElevii [0, cap]
m: Numar

inordine: SRL

Funcție inordine (ab, e)
{ pte:
post:
contor = 1
creoaza (n) // creare ntrive
 $p \leftarrow 1$

căt timp γ vidă (n) v($p \leq ab.m \wedge ab.v[p]! = -1$) execută

căt timp $(p \leq ab.m \wedge ab.v[p]! = -1)$ execută
adaugă (n, p)
 $p \leftarrow \alpha^k p$

nterge (n, p)

dacă $ab.v[p]! = c$ at

inordine ← counter

counter ← counter + 1

$p \leftarrow 2^k p + 1$

inordine ← -1



A. Pp FRG să funcția ne aperează cu i patră

$$T(n, \text{par}) = 2T(n/2, \text{impar}) + n$$

Notăm $a = 2^k$

$$\begin{aligned}
 &= 2^k T(2^{k-1}, \text{impar}) + 2^k \\
 &= 2^k T(2^{k-2}, \text{par}) + 2^k \\
 &= 2^3 T(2^{k-3}, \text{impar}) + 2^k \cdot 2^{k-2} + 2^k \\
 &= 2^5 T(2^{k-3}, \text{impar}) + 2^{k+1} \\
 &= 2^4 T(2^{k-4}, \text{par}) + 2^{k+1} \\
 &= 2^5 T(2^{k-5}, \text{impar}) + 2^k + 2^{k+1} \\
 &= 2^6 \cdot T(2^{k-6}, \text{par}) + 2^k + 2^{k+1} \\
 &= 2^7 T(2^{k-7}, \text{impar}) + 2^k + 2^k + 2^{k+1} \\
 &= 2^8 T(2^{k-8}, \text{par}) + 2^{k+2} \\
 &= 2^9 T(2^{k-9}, \text{impar}) + 2^k + 2^{k+2} \\
 &= 2^{10} T(2^{k-10}, \text{par}) + 2^k + 2^{k+2} \\
 &= 2^{11} T(2^{k-11}, \text{impar}) + 2^k + 2^k + 2^{k+2} \\
 &= 2^{11} T(2^{k-11}, \text{impar}) + 2^{k+3} \\
 &= 2^k \cdot T\left(\frac{1}{2}\right) + 2^{k+k/4} \\
 &= 2^k + 2^k \cdot 2^{k/4} = 2^k + 2^{k(1+\frac{1}{4})} = 2^k + 2^{\frac{5k}{4}} = n + n^{\frac{5}{4}} \in O(n^{\omega})
 \end{aligned}$$

$$2^{2k} = (2^k)^2 = n^2$$

8.

Tabela:

0
1
2
3
4

35 nm AVL 0:

(35)

27 nm AVL 2:

(2)

18 nm AVL 3:

(18)

6 nm AVL 1:

(6)

3 nm AVL 3:

(18)

(3)

10 nm AVL 0:

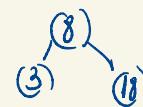
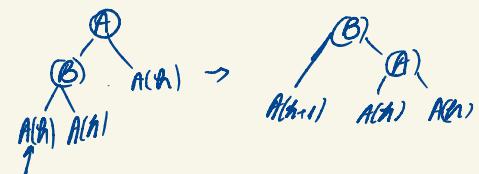
(35)

(10)

8 nm AVL 3:

(18)⁻²(3)¹

(8)

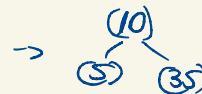
 \rightarrow desechar b/n

5 nm AVL 5:

(35)

(10)

(5)



c₁ facut

c₂ m/n

fatorul de transformare al tabelei este raportul dintre număr de valori m
și de locuri m în rep. numărul median de valori menționate la o tbc.

d.

A. $T(m, impar) = 2T(m/2, par) + m \quad \dots \quad \text{făcut}$

B. Folosim verificarea binară.

Tablă după ce inserăm el

Poz	0	1	2	3	4	5	6	7	8	9
cheie	20	30	22	13	Nic	5	15	25	18	Nic

Vine nă interogăm 5.

Incepem cu $i=5$ și $j=4$

$c = 15 \rightarrow d(c) = 15 \Rightarrow$ mutăm datele

Poz	0	1	2	3	4	5	6	7	8	9
cheie	20	30	22	13	Nic	15	15	25	18	Nic

continuăm cu $i=6$ și $j=7$

$c = 25 \rightarrow d(25) = 5 \Rightarrow$ mutăm datele

Poz	0	1	2	3	4	5	6	7	8	9
cheie	20	30	22	13	Nic	15	25	25	18	Nic

$i=7$ și $j=8$

$c = 18 \rightarrow d = 8 \rightarrow \text{OK}$

$j = 9$

$c = Nic \rightarrow$ putem merge $i = 4$

Poz	0	1	2	3	4	5	6	7	8	9
cheie	20	30	22	13	Nic	15	25	Nic	18	Nic

c₁ e)

Cazul de favorabil este atunci cand elementul nu se află în vector și trebuie să itereăm prin toate el. \rightarrow singur simbol

c₂ e)

Război din structura de ansamblu care ne spune că toate nivelele sunt pline cu excepția ultimului.

P. ca multă în adveleme $\rightarrow u = \omega^0 + \omega^1 + \dots + \omega^h \dots$



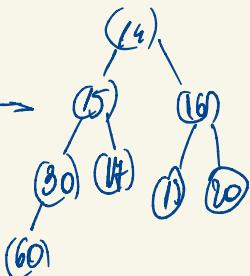
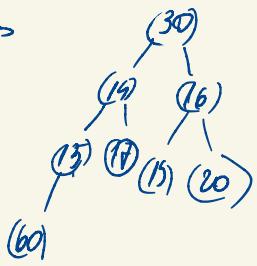
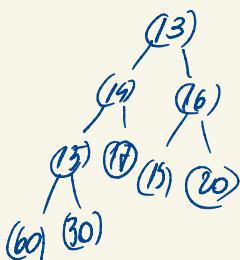
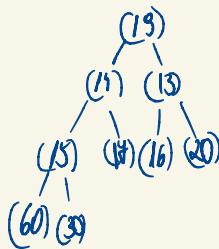
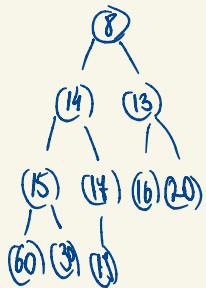
ABC:
bcd: întreg
e: întreg [0..cp-1]
st: ——n
de: ——n
h: ——n

funcția SRA($\overset{abc}{p}$)
 { p: întreg }
 $p \leftarrow abc. st[p]$
 $abc. nt[p] \leftarrow abc. nt[ps]$
 $abc. de[ps] \leftarrow p$
 $abc. h[p] \leftarrow \text{multime}(p)$
 $abc. h[ps] \leftarrow \text{multime}(p)$

SUB 123

A. $T(n) = 4T(n-1) = 4^n T(0) = \dots = 4^{n-1} T(1) = 4^{n-1} \in \Theta(4^n)$

B.



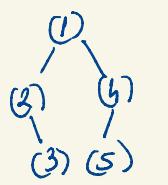
C)

- a) la adaugarea unui element trebuie sa contină oade ar priorității coresp.

c2. d / 10

Lăci fiecare nod cu care nu se poate face nici unul de noduri

4.



$RS\Delta: 1 \ 2 \ 3 \ 4 \ 5$
 $SR\Delta: 2 \ 3 \ 1 \ 5 \ 4$

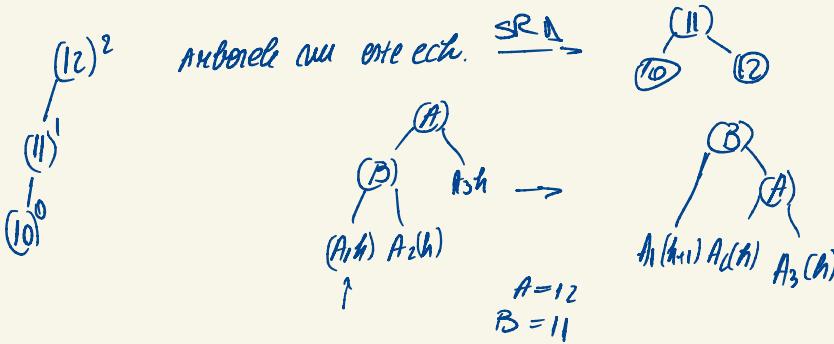
SUB 124

A. $T(n) = n + T(n/2) = 2^k + T(2^{k-1}) = 2^k + 2^{k-1} + T(2^{k-2}) =$

$\downarrow \quad 2^k = n$

$$\begin{aligned} &= 2^k + 2^{k-1} + \dots + 2^1 + T(1) \\ &= 1 + 2 + \dots + 2^k \\ &= 2^{k+1} - 1 = 2^{\log_2 n + 1} - 1 = 2n - 1 \in \Theta(n) \end{aligned}$$

B.



c1 e) Se poate redimensiona

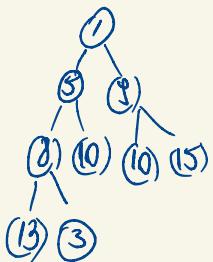
c2 facut

Δ facut

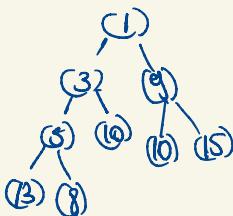
SUB 125

A. -

B.



Wort am 3



C. SRD:

Ver hepti ntiva la jocare pas

$$\begin{cases} \{14 \leq 1\} \\ \{30 \leq 0\} \end{cases} \quad \begin{cases} 14 \geq 1 \\ 30 \geq 0 \end{cases} \quad \begin{cases} 14 \geq 3 \\ 30 \geq 3 \end{cases} \quad \begin{cases} 14 \leq 11 \\ 30 \leq 11 \end{cases} \quad \begin{cases} 14 \geq 10 \\ 30 \geq 10 \end{cases} \quad \begin{cases} 14 \leq 15 \\ 30 \leq 15 \end{cases}$$

$$1 | 2 | 3 | 14 | \# | 10 | 11 | 30 \quad \Rightarrow \text{a)}$$

$$1 | 2 | 3 | 14 | \# | 10 | 11 | 10 | 30$$

c2. Adâncimusea minimă a lui T este adâncimina maximă a lui T este apreciată plin

$$\rightarrow 2^0 + 2^1 + \dots + 2^h = 14$$

$$\Leftrightarrow 2^{h+1} - 1 = 14 \Rightarrow 2^{h+1} = 15 \Rightarrow h+1 = 4 \Rightarrow h = 3$$

D.

CP2:

a: Anumitele trei numere

Anumitele trei numere:

v: TElem [1, cap]

u: Intreg

x: TElem \times TElem $\rightarrow \{T, F\}$

Subalgoritmu $\eta_{tore} (cp, e)$
 1 pre: cp : EP3V neviciu
 e: TElem, el η_{tore}^1
 $\eta_{tore} (cp, e_1)$

$\text{sterge} (\alpha, e)$
 $\text{adaugă} (\alpha, e)$

Subalgoritmu $\text{adaugă} (\overset{10}{\alpha}, \overset{1}{e})$

} pre:

post:

$$\alpha.m \leftarrow \alpha.m + 1$$

$$\alpha.v[m] \leftarrow e$$

$$\text{watt} (\alpha, m)$$

}

Subalgoritmu $\text{wacă} (\overset{10}{\alpha}, \overset{1}{p})$

} pre:

post:

$$e \leftarrow \alpha.v[p]$$

$$p \leftarrow [(p \times 3) + 1] / 3$$

în timp $p \geq 1$ și $\alpha.v[p] = e$ = false execută

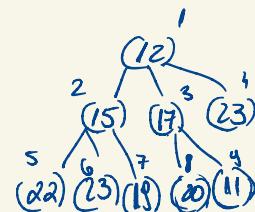
$$\alpha.v[p] \leftarrow \alpha.v[p]$$

$$p \leftarrow p$$

$$p \leftarrow [(p \times 3) + 1] / 3$$

$$\alpha.v[p] \leftarrow e$$

M



Subalgoritmu $\text{sterge} (\overset{10}{\alpha}, \overset{0}{e})$

} pre:

post:

$$e \leftarrow \alpha.v[1]$$

$$\alpha.v[1] \leftarrow \alpha.v[m]$$

$$\alpha.m \leftarrow \alpha.m - 1$$

celoara ($\alpha, 1$)

Subalgoritmul căreia (a, pos)

proces:

post:

$$e \leftarrow a.v[pos]$$

$$p \leftarrow pos^*3 - 1$$

cât timp $p \leq a.n$ execută

$$aux \leftarrow p$$

pentru $K \leftarrow 1, 2$ execută

dacă $a.v(a.v[aux], a.v[p+K]) = false$ atunci

$$aux \leftarrow p+K$$

$$p \leftarrow aux$$

dacă $\neg(a.v[pos], a.v[p]) = false$ at.

$$a.v[pos] \leftarrow a.v[p]$$

$$p \leftarrow a.n + 1$$

$$pos \leftarrow p$$

$$p \leftarrow 3^*pos + 1$$

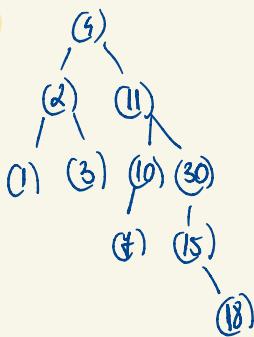
II



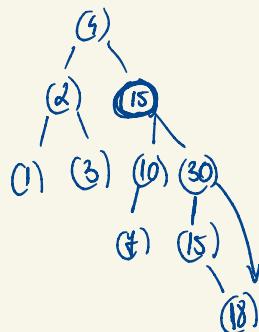
SUB 126

$$\begin{aligned}
 A. \quad T(n) &= T(n/2) + n - 1 + T(n/2) = 2^k - 1 + 2T(2^{k-1}) = \\
 &= 2^k - 1 + 2^{k-1} - 1 + 2T(2^{k-2}) = 2^k + 2^{k-1} + 2^{k-2} - 3 + 2T(2^{k-3}) = \\
 &= \dots = 2^k + 2^{k-1} + \dots + 2^1 - k + 2T(1) \\
 &= 2^0 + 2^1 + \dots + 2^k - (k-1) \\
 &= 2^{k+1} - 1 - (k-1) \\
 &= 2^{k+1} - k = 2^{\log_2 n + 1} - k = 2n - 1 \in \Theta(n)
 \end{aligned}$$

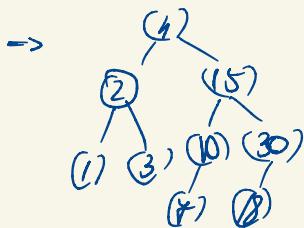
B.



Intervenirea II cu cea mai mare valoare din arbore
rest drept = 15



Faceam legătură 30, 18



C1. a) Se aplică principiul coziu de FIFO.

C2. a), c), d)

D. Subalgoritmu interclasare ($\ell_1, \ell_2, \dots, \ell_m$)