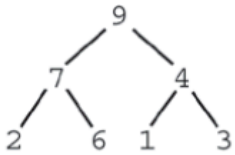
Capitolul IC.09. Autoevaluare

Observație

Există întrebări cu mai multe răspunsuri corecte

1. Se consideră arborele MaxHeap cu implementarea statică [9, 7, 4, 2, 6, 1, 3] şi reprezentarea



Care va fi conţinutul tabloului după ştergerea din heap a valorii maxime?

a) [7, 4, 2, 6, 1, 3]
b) [7, 4, 6, 2, 1, 3]
c) [7, 4, 6, 2, 3, 1]
d) [7, 6, 4, 2, 1, 3]
e) [7, 6, 4, 2, 3, 1]

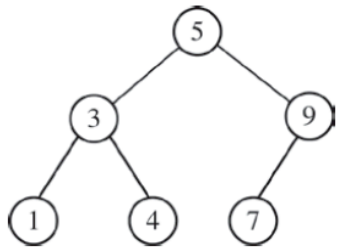
2. Într-un arbore binar plin, toate nodurile interne au exact doi descendenţi. Câte noduri interne sunt într-un arbore binar plin cu 512 frunze?

a) 256 b) 511 c) 512 d) 513 e) 1024

3. Care dintre următoarele structuri de date este cea mai potrivită pentru stocarea simbolurilor şi a cuvintelor rezervate specifice unui limbaj de programare?

a) tabelă de dispersie | b) listă înlănţuită | c) arbore binar | d) coadă de priorităţi | e) tablou unidimensional

4. Se consideră arborele binar de căutare de mai jos.



Pornind de la un arbore gol, inserarea cărei secvenţe de chei întregi conduce la BST de mai sus?

(A) 5, 9, 1, 7, 3, 4
(B) 5, 7, 4, 9, 3, 1
(C) 5, 4, 7, 3, 9, 1
(D) 5, 3, 4, 9, 1, 7
(E) 5, 3, 1, 7, 9, 4

5. Care dintre structurile de mai jos este cea mai potrivită pentru a implementa o colecţie de valori cu următoarele caracteristici:

- Elementele sunt inserate, regăsite şi şterse în mod FIFO;
- Numărul de elemente a colecţiei nu este cunoscut apriori;
- Este necesară alocarea unei zone de memorie relativ mare pentru fiecare element al colecţiei.

a) Listă liniară simplu înlănţuită, cu pointeri la primul şi ultimul element
b) Listă dublu înlănţuită, cu pointer la unul din capete
c) Tablou
d) Arbore binar
e) Tabelă de dispersie

6. Care este ordinul de complexitate pentru următoarea secvenţă?

```
while (i>0)
{       for (j = 0 ; j < n/2 ; j++)
                cout << i << j << endl;
        i--;
}
```

a) $O(n^2)$ b) O(nlogn) c) O(logn) d) O(n)

7. Se consideră tabloul V ce conţine reprezentarea implicită a unui arbore binar:

60	25	45	20	15	30	40	10	18	12	5	22

Arborele este:
a) AVL b) BST c) Heap d) nicio variantă nu este corectă

8. Fie un arbore heap cu N elemente şi k nivele complete, implementat static "fără goluri". Atunci:

a. Pe ultimul nivel sunt $N-2^k+1$ noduri
b. Pe ultimul nivel sunt $2^{k+1}-N+1$ goluri
c. Pe ultimul nivel sunt $N-2^k-1$ noduri
d. Pe ultimul nivel sunt $2^{k+1}-N-1$ goluri

9. Care din afirmaţiile de mai jos sunt corecte?

a) Dacă un arbore binar de căutare şi un arbore B conţin acelaşi numar de chei, atunci cei doi arbori au adâncimi egale.
b) Ordinul de complexitate pentru operaţia de inserare a unui nod într-o coada liniară alocată dinamic este O(1).
c) Căutarea binară este de complexitate mai mică decât căutarea secvenţială.
d) Dacă un arbore B este de ordin 3, atunci orice pagină cuprinde cel mult 3 chei.

10. Care este ordinul de complexitate pentru următoarea secvenţă?

```
for (i = 1 ; i < n ; i=i*2)
        for (j = 0 ; j < n; j++)
                cout << i << j << endl;
```

a) $O(n^2)$ b) O(nlogn) c) O(logn) d) O(n)

11. Dată fiind Q o coadă circulară alocată static, Q.vect este vectorul ce conţine elementele cozii, Q.head şi Q.tail reprezintă indexul primului element respectiv indexul primului loc liber în care se poate insera un element şi funcţia următoare:

```
put(Q,a)
  if Q.tail=Nextpoz(Q.head) then
        print "Coada plina"
  elseif
    Q.vect[Q.tail] ←a
    Q.tail←Nextpoz(Q.tail)
  endif
end
```

Unde este o eroare în secvenţa dată:

a) Pe ramura **then**
b) Pe ramura **elseif**
c) Testul de la **if**
d) Secvenţa este corectă

12. Completaţi spaţiile libere:
Într-un arbore B inserarea se face întotdeauna (precizaţi locul) .....
Într-un arbore binar complet de adâncime h, numărul total de noduri este .....
Căutarea nodului de cheie minimă într-un BST, **în cazul cel mai defavorabil**, este de complexitate .....
Într-un arbore B, nodurile sunt grupate în ..... , cheile fiind ordonate strict crescător.
Algoritmul de sortare prin insertie este de complexitate .....
Într-un arbore binar complet de adâncime h, numărul total de noduri este.....
Căutarea nodului de cheie minimă într-un BST, **în cazul mediu**, este de complexitate .....

13. Care trebuie să fie ordinea introducerii elementelor 2,4,5,6,7,9 într-un arbore binar de căutare astfel încât, la parcurgerea acestuia în preordine să se obțină următoarea secvență 4 2 7 5 6 9
    a. 4 7 9 2 5 6
    b. 4 9 7 6 5 2
    c. 4 2 7 5 6 9
    d. 7 2 4 5 9 6

14. Precizaţi care din afirmaţiile următoare sunt adevărate:
a) Complexitatea inserării unui nod într-o coadă alocată dinamic este O(n);
b) Algoritmul de sortare QuickSort este de complexitate mai mică decât BubbleSort;
c) Un arbore BST cu n>2 nu poate fi HEAP.

15. Fie o listă circulară simplu înlănţuită alocată dinamic unde pointerul *l* conţine adresa primului element din listă iar pointerul *x* conţine adresa unui element din lista *l*. Secvenţa

```
y←succ(x)
succ(x) ←succ(y)
free_sp(y)
```

poate fi utilizată pentru:
a. ştergerea din listă a primului nod
b. ştergerea din listă a nodului indicat de x
c. ştergerea din listă a nodului ce urmează celui indicat de x

16. Fie S o stivă ordonată în care ***S.vect*** este vectorul de dimensiune N în care se reprezintă elementele stivei ***S*** iar ***S.virf*** este indicele vârfului stivei. Atomii sunt de tip **Atom** iar stiva de tip **Stack**. Funcţia **push**:

```
void push(Stack &S,Atom a)
{if (S.virf >= N)
    printf("\n Stiva plina");
else
    S.vect[S.virf++] =a;
}
```

a. Este corectă
b. Prototip eronat
c. Eroare pe ramura Else
d. Eroare condiţie if

17. Se consideră un arbore AVL. În urma efectuării operaţiilor de echilibrare, ordinea afişării nodurilor arborelui este aceeaşi pentru:
    a. Parcurgerea în preordine
    b. Parcurgerea în inordine
    c. Parcurgerea în postordine
    d. Toate tipurile de parcurgeri

18. Fie o listă simplu înlănţuită alocată dinamic, pointerul q conţine adresa unui element din listă, iar p adresa unui element ce se doreşte a fi inserat . Secvenţa următoare

succ(p)←succ(q)
succ(q) ←p
aux← data(q)
data(q)← data(p)
data(p)← aux

poate fi utilizată pentru:
a. Inserarea în listă a data(p) după data(q)
b. Inserarea în listă a data(p) după succesorul nodului q
c. Inserarea în listă a data(p) înainte de data(q)
d. secvenţă eronată pentru scopul prous

19. Ce rotaţii trebuiesc aplicate asupra arborelui binar de căutare pentru care s-au introdus elementele 4 2 8 6 9 7
    a. Rotaţie dublă dreapta
    b. Rotaţie dublă stânga
    c. Rotaţie simplă stânga
    d. Rotaţie simplă dreapta

20. Asupra căror structuri poate fi folosită următoarea funcţie de parcurgere în inordine?

```
void inordine(int i)
{
        if (i<=n && (A[i]!='-'){
                inordine(i*2);
                prelucrare(A[i]);
                inordine(i*2+1);
        }
}
```

    a. Reprezentarea implicită a arborilor binari de căutare
    b. Arbori heap
    c. Nu poate fi folosită, fiind greşită.

21. Pentru două structuri de date predefinite tip coadă şi stivă s-au implementat **numai** funcţiile pentru inserare, ştergere şi testare dacă structura este goală. Se implementează o aplicaţie care numără elementele din structura A (stivă sau coadă) aflată în stare iniţială utilizând o structură auxiliară B (de tip stivă sau coadă). Structurile A şi B pot fi utilizate în orice combinaţie stivă-coadă, dar după numărare, structura A trebuie să se fie în starea iniţială. Pentru care din combinaţiile I-III de mai jos, numărarea elementelor din A este posibilă în condiţiile enunţate mai sus:
    I. A este coadă, B este stivă
    II. A este stivă, B este stivă
    III. A este stivă, B este coadă

a) Nici una b) Numai II şi III c) Numai I şi II d) I,II şi III e) Numai II şi III

22. Într-un arbore binar plin, toate nodurile interne au exact doi descendenţi. Câte noduri interne sunt într-un arbore binar plin cu 128 frunze?

a) 256 b) 127 c) 128 d) 129 e) 64

23. Ce fel de rotații pot fi implementate utilizând următorul cod:

```
Nod* Rot(Nod* p)
{
        Nod* aux;
        aux = p->stg;
        p->stg = aux->drt;
        aux->stg = p;
        p->fact = 0;
        aux->fact= 0;
        return aux;
}
```

a. Rotație dublă dreapta
b. Rotație simplă stânga
c. Rotație simplă dreapta
d. Codul nu va efectua nici o rotație, fiind greșit.

24. Complexitatea timp a algoritmului de generare a unui heap (fără a utiliza memorie suplimentară) pornind de la un vector cu **n** elemente poate fi

a. Întotdeauna O(log*n*)
b. O(*n*log*n*) sau O(*n*)
c. Numai O(*n*)
d. Întotdeauna O(*n*log*n*)

25. Care din afirmaţiile de mai jos sunt corecte?

e) Dacă un arbore binar de căutare şi un arbore B conţin acelaşi numar de chei, atunci cei doi arbori au adâncimi egale.
f) Ordinul de complexitate pentru operaţia de inserare a unui nod într-o coada liniară alocată dinamic este O(1).
g) Căutarea binară este de complexitate mai mică decât căutarea secvenţială.
h) Dacă un arbore B este de ordin 3, atunci orice pagină cuprinde cel mult 3 chei.

26. Se consideră funcţia de dispersie *h(x) = x* ***modulo*** *m* care asociază unui întreg *x* o intrare într-o tabelă de dimensiune *m*. Pentru care din următoarele valori ale lui *m* funcţia *h* este o funcţie de dispersie perfectă peste setul de chei { 0, 6, 9, 12, 22, 31 }?

a) m = 11
b) m = 4
c) m = 7
d) m = 13

27. Precizaţi care din afirmaţiile următoare sunt adevărate:

a. Un arbore HEAP este sigur şi AVL;
b. Se consideră o listă alocată dinamic, identificată prin pointerul ***p***. Operaţia de inserare a unui nod în listă, înaintea elementului pointat de ***p*** este de complexitate O(n);
c. Complexitatea algoritmului de înmulţire a două matrici este $O(n^2)$;
d. Algoritmul QuickSort este de complexitate O(nlog(n)).

28. Fie un arbore binar de căutare cu structura **Nod**:

```
Nod* f(Nod* &nod)
{Nod* p;
if(nod->drt==0)
        {p=nod;
        nod=nod->stg;
        return p;
        }
else
        return      f(nod-
>drt);}
```

Funcţia f implementează operaţia prin care:

a. se şterge din arbore nodul de valoare minimă şi întoarce pointer spre acesta;
b. se caută nodul de valoare maximă şi întoarce pointer spre acesta ;
c. se detaşează din arbore nodul de valoare minimă şi întoarce pointer spre acesta;
d. se detaşează din arbore nodul de valoare maximă şi întoarce pointer spre acesta;
e. Nici una din variantele anterioare.

29. Se consideră operaţia de ştergere a elementului de valoare maximă dintr-o listă liniară alocată dinamic (complexitate O_{lista}), dintr-un arbore heap (complexitate O_{heap}) şi dintr-un arbore binar de căutare (complexitate O_{BST}) . Care din următoarele relaţii este corectă:

a. $O_{lista} = O_{BST} = O_{heap}$
b. $O_{lista} < O_{BST} < O_{heap}$
c. $O_{heap} = O_{BST} < O_{lista}$
d. $O_{heap} < O_{BST} < O_{lista}$

30. Fie T un arbore binar. Dacă notăm:

n_2 - numărul de noduri de grad 2 din arborele binar T,

n_1 - numărul de noduri de grad 1 din arborele binar T,

n_0 - numărul de noduri terminale (frunze) din arborele binar T,

care din următoarele afirmaţii este corectă?

a. $n_0 = n_2 + 1$.
b. $n_2 = n_0 + 1$.
c. $n_1 = n_2 + 1$.

31. Se consideră o tabelă de dispersie cu valori în { 0, 1, …, *m* - 1 }. Tabela poate conţine mai mult de *m* înregistrări atunci când

a) rezolvarea coliziunilor se face prin adresare deschisă
b) rezolvarea coliziunilor se face prin înlănţuire
c) rezolvarea coliziunilor se face prin dispersie dublă
d) funcţia de dispersie realizează o dispersie perfectă

31. Operaţiile care trebuiesc făcute în cazul unei rotaţii duble la dreapta într-un arbore AVL sunt următoarele:

a.
```
lchild(a) =
    rchild(c)
b = lchild(a)
rchild(b) = lchild(c)
c = rchild(b)
rchild(c) = a
lchild(c) = b
a = c // se schimbă
rădăcina arborelui.
```

b.
```
b = lchild(a)
c = rchild(b)
lchild(a) = rchild(c)
rchild(b) = lchild(c)
rchild(c) = a
lchild(c) = b
a = c // se schimbă rădăcina
arborelui.
```

c)
```
b = lchild(a)
c = rchild(b)
rchild(b) = lchild(c)
rchild(c) = a
lchild(c) = b
a = c// se schimbă rădăcina
arborelui
```

32. Într-un graf complet cu 6 noduri sunt maxim:
    a. 360 cicluri hamiltoniene;
    b. 720 cicluri hamiltoniene;
    c. 12 cicluri hamiltoniene;
    d. 120 cicluri hamiltoniene.

33. Fie f : N → N şi g : N → N două funcţii. Care din definiţiile următoare este corectă:
    e. Spunem că f ∈ O(g) şi notăm f = O(g) dacă şi numai dacă ∃ c ∈ R şi ∀ n_0 ∈ N astfel încât pentru ∀ n > n_0 : f(n) < c·g(n)
    f. Spunem că f ∈ O(g) şi notăm f = O(g) dacă şi numai dacă pentru ∀ c ∈ R şi ∀ n_0 ∈ N, ∃ n ∈ N astfel încât f(n) < c·g(n)
    g. Spunem că f ∈ O(g) şi notăm f = O(g) dacă şi numai dacă ∃ c ∈ R şi ∃ n_0 ∈ N astfel încât pentru ∀ n > n_0 : f(n) < c·g(n)

34. Fie S o stivă înlănţuită şi f,g,h implementări ale celor trei operaţii de bază (push, pop,top):

```
f(S,a)
    p← get_sp);
    data(p)←a;
    succ(p)←S
    S←p
Endf
```

```
g(S)
  if S=0 then
    eroare("Stiva vida")
  else
    return(data(S))
  endif
endg
```

```
h(S)
    if S=0 then
      eroare("Stiva vida")
    else
      p←S;
      S←succ(S)
      Free_sp(p)
    endif
end
```

Funcţiile f,g, h implementează respectiv operaţiile
    a. pop, top, push
    b. top, push, pop
    c. push, top, pop

35. Dată fiind Q o coadă circulară alocată static, Q.vect este vectorul ce conţine elementele cozii, Q.head şi Q.tail reprezintă indexul primului element respectiv indexul primului loc liber în care se poate insera un element şi funcţia următoare:

```
put(Q,a)
  if
Q.tail=Nextpoz(Q.head)
then
        print "Coada plina"
  elseif
    Q.vect[Q.tail] ←a
    Q.tail←Nextpoz(Q.tail)
  endif
end
```

Unde lipseşte ceva pentru ca funcţia **put** să lucreze corect în orice situaţie:
    a. Pe ramura **then**
    b. Pe ramura **elseif**
    c. Testul de la **if**

36. Fie G=(V,E) un graf reprezentat prin matricea de adiacenţă A:

$$A = \begin{matrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}$$

a) Graful G este arbore b) Graful G nu este arbore c) Graful este neorientat d) Graful este conex e) Graful este complet

37. Funcţia **căutare-nod** ar trebui să implementeze operaţia de căutare a unui nod cu cheia k într-un arbore binar de căutare. Funcţia este incompletă:

```
Căutare_nod(rădăcină,k)        // rădăcină — pointer la rădăcină arborelui
  if (rădăcină = 0) then                  // primul if
    return NULL
  else
    if key(data(rădăcină)) > k then              // al doilea if
     return căutare_nod (fiu_stânga(rădăcină))
    else
     if key(data(rădăcină)) < k then             // al treilea if
      return căutare_nod (fiu_dreapta(rădăcină))
     endif
    endif
  endif
end{Căutare_nod}
```

Unde lipseşte ceva:

a. pe ramura **then** a primului **if**
b. pe ramura **then** a celui de-al doilea **if**
c. pe ramura **else** a celui de-al treilea **if** (lipseşte dar ar trebui să fie prezentă şi să conţină ceva)

38. Fie **T=(V,E)** un arbore heap incomplet eventual doar pe ultimul nivel şi în care valoarea de nod este chiar cheia acestuia. Care este varianta corectă pentru operaţia de **inserare a unui nod în heap-ul T** ?

a. **Se inserează nodul pe nivelul incomplet;**
   Se reorganizeaza structura arborelui : se detaşează din structura arborelui radăcina, se interschimba nodul detaşat cu nodul de pe nivelul incomplet şi apoi valoarea rădăcinii este retrogradată pîna când structura heap-ului este realizată.
b. **Nodul se inserează pe nivelul incomplet, după ultima frunză;**
   Se reorganizeaza structura arborelui : se promovează nodul inserat de la stânga la dreapta, prin interschimbare cu părintele acestuia până când structura heap-ului este realizată.
c. **Se inserează nodul pe nivelul incomplet, după ultima frunză;**
   Se reorganizeaza structura arborelui : se interschimbă nodul cu rădăcina, după care valoarea rădăcinii este retrogradată pîna când structura heap-ului este realizată.

39. Fie matricea de adiacenţă A:

0	1	0	1	0
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
0	1	0	1	0

Această matrice corespunde:
a. unui graf b. unui digraf c. unui graf conex d. unui graf ce nu este arbore

40. Se consideră un arbore AVL. În urma efectuării operaţiilor de echilibrare, ordinea afişării nodurilor arborelui este aceeaşi pentru:
a) Parcurgerea în preordine b) Parcurgerea în inordine c) Parcurgerea în postordine d) Toate tipurile de parcurgeri

41. Ce tip de rotaţie trebuie aplicată pentru un arbora AVL echilibrat dacă s-au introdus elementele {4 2 8 6 9 7} exact în ordinea dată?

a) Rotație dublă dreapta b) Rotație dublă stânga c) Rotație simplă stânga d) Rotație simplă dreapta

42. Complexitatea timp, în cazul cel mai defavorabil, pentru căutarea nodului de cheie minimă într-un arbore binar de căutare este:
a. $O(n^2)$
b. $O(n)$
c. $O(n\log n)$
d. $O(\log n)$

43. Complexitatea operaţiilor de inserare şi ştergere într-un arbore heap cu n noduri este:
a) $O(n\log n)$
b) $O(n)$
c) $O(\log n)$
d) $O(n^2)$

44. Fie T=(V,E) un arbore binar cu reprezentarea implicită:

40	25	70	10	30	60	80	5	20	28	35	50	65	75

Arborele T este :
a. Arbore binar de căutare
b. Arbore heap
c. Arbore binar complet

45. Nodurile unui arbore binar de căutare pot fi afişate în ordinea crescătoare a cheilor prin:
a) Parcurgere pe linie
b) Parcurgere în inordine
c) Parcurgere în preordine
d) Parcurgere în postordine