Algoritmi i strukture podataka
3. blic – zadaci za vježbu skupljeni iz bliceva od prijašnjih godina
ak. god. 2007/08

by majah

LISTE-01

Predložena je funkcija za pronalazak čvora sa zadanom cjelobrojnom šifrom u jednostruko povezanoj linearnoj listi:

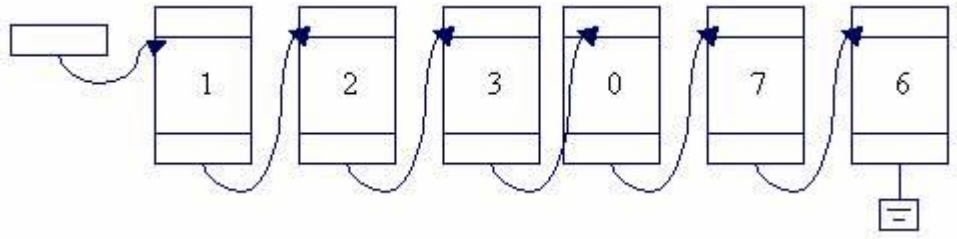
```
cvor *trazi(cvor *glava, int sifra){
  if (glava->sifra == sifra) return glava;
    if (glava->sljed)
      return trazi(glava->sljed, sifra);
  else
    return NULL;
}
```

Koja od slijedećih tvrdnji (koje se odnose na predloženu funkciju) je ispravna?

a) Funkcija je ispravna i vraća pokazivač na čvor sa zadanom šifrom ili NULL ako čvor sa zadanom šifrom ne postoji u listi.
b) Funkcija nije ispravna: pretražuje sve čvorove liste osim zadnjeg!
c) <u>Funkcija nije ispravna: ne radi za praznu listu (uzrokuje pogrešku kod poziva)!</u>
d) Funkcija nije ispravna: ukoliko čvor sa zadanom šifrom ne postoji u listi, funkcija neće "nikad" završiti
e) Funkcija nije ispravna: pretražuje sve čvorove liste osim prvog!

LISTE-02

Što će vratiti priložena funkcija za zadanu jednostruko povezanu linearnu listu:
Slika (unutar čvorova prikazana je vrijednost varijable `element`):




```
int f(cvor *glava){
  if (glava){
    if (glava->element > 3)
      return glava->element + f(glava->sljed);
    else
      return f(glava->sljed);
  }else{
    return 0;
  }
}
```

a) 6
b) 19
c) 0
d) 13
e) 16

LISTE-03

U dvostruko povezanu listu spremaju se zapisi slijedećeg tipa:

```
typedef struct s1{
  int mbr;              // matični broj studenta
  char ime[40+1];       // ime studenta
  float prosjek;        // prosjek ocjena
  struct s1 *sljed;
  struct s1 *preth;
} zapis;
```

Kako glasi funkcija koja izbacuje prvi element iz liste, oslobađa zauzetu memoriju te vraća 1 ako je operacija uspjela, a 0 ako operacija nije uspjela?

a)

```
int izbaci(zapis **glava, zapis **rep) {
      if (*glava == NULL) return 0;
      if ((*glava)->sljed == NULL) {
        free(*glava);
        *glava = *rep = NULL;
      }
      else {
        *glava = (*glava)->sljed;
        free((*glava)->preth);
        (*glava)->preth = NULL;
      }
      return 1;
    }
```

b)

```
int izbaci(zapis **glava) {
      if (*glava == NULL) return 0;

      *glava = (*glava)->sljed;
      free((*glava)->preth);
      (*glava)->preth = NULL;
      return 1;
    }
```

c)

```
int izbaci(zapis *glava, zapis *rep) {
      if (glava == NULL) return 0;
      if (glava->sljed == NULL) {
        free(glava);
```

```
        glava = rep = NULL;
      }
      else {
        glava = glava->sljed;
        free(glava->preth);
        glava->preth = NULL;
      }
      return 1;
    }
```

d)

```
int izbaci(zapis **glava) {
      zapis *pom = *glava;
      if (*glava == NULL) return 0;
      glava = (glava)->sljed;
      free(pom);
      return 1;
    }
```

e)

```
void izbaci(zapis **glava, zapis **rep) {
      if (*glava != NULL) {
        if ((*glava)->sljed == NULL) {
          free(*glava);
          *glava = *rep = NULL;
        }
        else {
          *glava = (*glava)->sljed;
          free((*glava)->preth);
          (*glava)->preth = NULL;
        }
      }
    }
```

STABLA-01

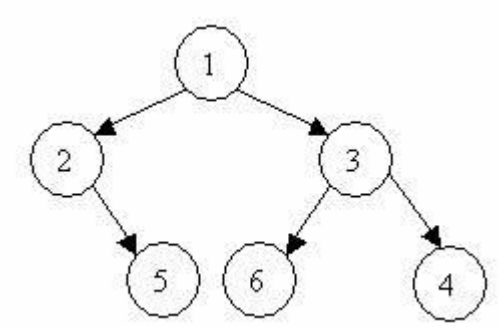
Što će se ispisati funkcijom:

```
void ispis(cvor *korijen){
  if (korijen) {
    ispis(korijen->desno);
    ispis(korijen->lijevo);
    printf("%c", korijen->element);
  }
}
```

za stablo na slici pozivom funkcije

```
ispis (korijen);
```

ako je `korijen` u trenutku poziva pokazivač na korijen stabla?



a) 123456
b) 146352
c) 643521
d) 321
e) <u>463521</u>

STABLA-02

Koliko razina ima potpuno binarno stablo koje sadrži 100 čvorova i koliki je broj čvorova na posljednjoj razini ?

a) <u>broj razina=7 broj čvorova=37</u>
b) broj razina =6 broj čvorova=64
c) broj razina =7 broj čvorova=64
d) broj razina =7 broj čvorova=50
e) broj razina =6 broj čvorova=50

STABLA-03

Što radi slijedeća funkcija?

```
int f(cvor *glava) {
  int i = 0;
  if (glava) {
    if (glava->lijevo || glava->desno) i++;
    i += f(glava->lijevo);
    i += f(glava->desno);
  }
  return i;
}
```

a) Broji čvorove u stablu koji imaju oba djeteta.
b) Broji čvorove u stablu koji imaju lijevo dijete.
c) Broji čvorove u stablu koji imaju desno dijete.
d) <u>Broji čvorove u stablu koji nisu listovi (imaju bar jedno dijete).</u>
e) Ništa od navedenog.

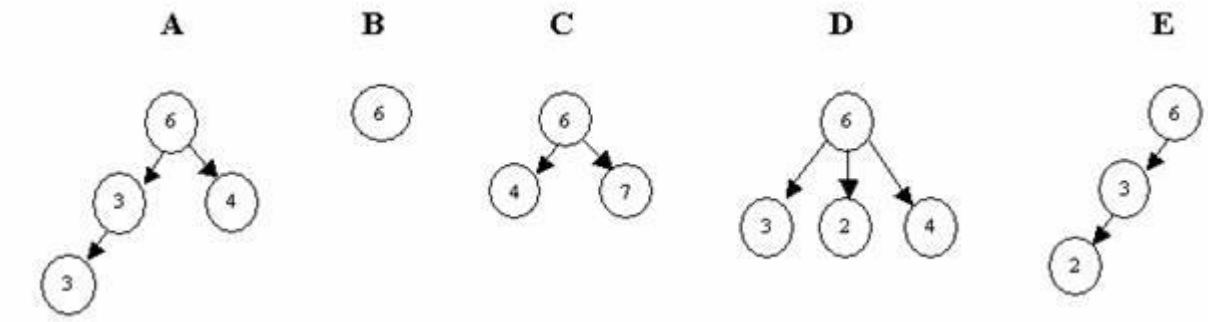
GOMILA-01

Najefikasniji algoritam stvaranja gomile od *n* elemenata za najgori slučaj ima složenost

a) O(1)
b) O(*n*)
c) $O(n \cdot \log_2 n)$
d) $O(n^2)$
e) $O(\log_2 n)$

GOMILA-02

Koja od prikazanih stabala su gomile:



a) A, B
b) B
c) A, B, E
d) A, B, C, E
e) A

GOMILA-03

Neka se u gomili koja je pohranjena u polje nalaze sljedeći podaci:

66	57	32	30	36	

Što će se desiti kada se u takvu gomilu doda podatak 35 (a nakon što se pozove funkcija *ubaci* koja dodaje elemente u gomilu)?

a) Izbaci se broj 66, a 35 se umetne na to mjesto
b) Taj će se broj dodat na kraj polja jer je time očuvano svojstvo gomile
c) Broj 35 će zamijeniti mjesto s brojem 32
d) Broj 35 će zamijeniti mjesto s brojem 36
e) Broj 35 će zamijeniti mjesto s brojem 30

OOP

OO-01

Zadan je program u jeziku C++:

```
class MojRazred {
        int _MojPodatak;
};
int main ( ) {
        MojRazred obj;
        obj._MojPodatak = 10;
         return 0;
}
```

Koja od sljedećih tvrdnji je točna?

a) Program se može prevesti, povezati i izvoditi bez greške.
b) Program se prevodi uz upozorenje (warning), no ipak se može povezati i izvoditi bez greške.
c) Program se može prevesti i povezati. Kod izvođenja, naredba za pridruživanje se ignorira, no to ne izaziva grešku.
d) Program se može prevesti i povezati. Naredba za pridruživanje se izvršava, no nakon nje se «baca» iznimka.
e) <u>Program se ne može prevesti jer prevodilac javlja grešku.</u>

OO-02

Zadan je razred u jeziku C++:

```
class MojRazred {
        int _MojPodatak;
};
```

U glavnoj funkciji deklariran je objekt:

```
MojRazred obj;
```

Kako izgleda naredba kojom glavna funkcije članskoj varijabli `_MojPodatak` unutar objekta obj pridružuje vrijednost 20?

a) `obj._MojPodatak = 20;`
b) `obj ->_MojPodatak = 20;`
c) `MojRazred *pobj = &obj;  (*pobj)._MojPodatak = 20;`
d) Takva naredba postoji, no nije ni jedna od gore navedenih.
e) <u>Takva naredba ne postoji.</u>

OO-03

Promatramo dvije deklaracije u jeziku C++:

```
struct MyStruct {
        int _Mydata;
};
```

```
class MyClass {
        int _MyData;
};
```

Koja od sljedećih tvrdnji je točna?

a) Dvije deklaracije su ekvivalentne u smislu da odgovarajući objekti imaju ista svojstva.
b) Članska varijabla u MyStruct je javna a članska varijabla u MyClass je privatna.
c) Prva deklaracija nije dozvoljena jer C++ ne poznaje struct.
d) Druga deklaracija nije dozvoljena jer razred mora imati barem jednu člansku funkciju.
e) Obje deklaracije su sintaktički neispravne, zato jer su im članske varijable privatne, a ne postoje članske funkcije koje bi mijenjale te varijable.

OO-04

Promatramo sljedeći C++ program:

```
#include <stdio.h>

class MojRazred {
        public:
                MojRazred( );
                int * _Podaci;
};
MojRazred :: MojRazred( ) {
        _Podaci = new int [10];
        for ( int i=0; i<10; i++ )
                _Podaci[i] = 0;
}
void Promijeni (MojRazred obj) {
        obj._Podaci[0] = 1;
}
int main ( ) {
        MojRazred ob;
        Promijeni(ob);
        printf("%d \n", ob._Podaci[0]);
         return 0;
}
```

Odaberite točnu tvrdnju:

a) Program ispisuje 0 .
b) Program ispisuje 1 .
c) Program ispisuje cjelobrojnu vrijednost koja nije definirana.
d) Program se ne može prevesti zato jer u razredu MojRazred nije definiran copy konstruktor.
e) Program neće ništa ispisati zato jer će doživjeti grešku tijekom izvođenja pri pokušaju mijenjanja članske varijable unutar objekta.

OO-05

U C++ programu nalazi se sljedeća deklaracija:

```
class MyClass {
        public:
                MyFunc( );
        private:
                int _MyData;
};
```

U glavnoj funkciji stvoren je objekt iz razreda MyClass na sljedeći način:

```
MyClass *p = new MyClass( );
```

Koja od sljedećih naredbi u nastavku glavne funkcije predstavlja ispravan poziv članske funkcije MyFunc() ?

a) p-> MyFunc( );
b) p.MyFunc( );
c) MyFunc(*p);
d) MyFunc(p);
e) (*p)->MyFunc( );

OO-06

Zadan je sljedeći C++ program:

```
#include <stdio.h>

class MyClass {
        public:
                MyClass( );
                MyClass(int a);
                ~MyClass( );
        private:
                int _data;
};
MyClass :: MyClass ( ) {
        _data = 0;
        printf("%d ", _data);
}
MyClass :: MyClass (int a ) {
        _data = a;
        printf("%d ", _data);
}
MyClass :: ~MyClass ( ) {
        printf("destroy %d ", _data);
}
int main ( ) {
        MyClass x;
        MyClass *y = new MyClass(5);
        delete y;
}
```

Što će ispisati ovaj program?

a) 0 5 destroy 5 destroy 0
b) 0 5 destroy 0 destroy 5
c) 0 5 destroy 5
d) 5 destroy 5
e) 0 0 destroy 0 destroy 0

1. Razlika između deep/shallow kopiranja objekta...

6. Koliko puta se zovu konstruktor i destruktor objekta koji je stvoren jednom u main()-u naredbom
objekt ime;
i onda opet u funkciji koja se poziva iz main()-a a koja sadržava naredbu:
*objekt *ime=new objekt;*
Konstruktor se zove dva puta, a destruktor jednom, jer se ne koristi naredba delete za drugi objekt (prvi se pobriše automatski kad se napusti main())

10. Imamo klasu student koja ima konstruktor koji prima integer. Taj integer se pohrani u člansku varijablu _X i nakon toga se ispise slovo "A" _X puta. U destruktoru se ispise slovo "B" isto _X puta. Što će ispisati program:
main() { ...
student novi(2);
student *novi2=new student(3);
... }
Rješenje: Ispisat će 5 slova "A" i 2 slova "B" jer se ne poziva destruktor za ovaj drugi objekt (ne koristi se delete)

11. Imamo klasu koja nema destruktor, a u konstruktoru se stvara polje (članska varijabla) pomoću naredbe **new**. Ispravna tvrdnja o toj klasi je **da se prostor koji zauzima polje neće osloboditi nakon brisanja objekta.**

14. Ako klasa "K" ima metodu "*metoda*" koja nije statična i ako imamo naredbu
K novi;
kako pozivamo metodu "metoda"?
Rješenje: novi.metoda();

15. Kako se incijalizira referenca?
Rješenje: int a; &ref=a;

17. Klasa trokut u kojoj su definirana dva konstruktora (s parametrom i bez-ovaj ima a=1 b=1 c=1) i funkcija *opseg*. U main()-u se definira *trokut t* i poziva se funkcija *opsegt=t.opseg();* uglavnom, **to poziva default konstruktor i opseg je 3**

18. Definiran je neki copy konstruktor za kompleksne brojeve i pita se sto radi this->re...
treba kliknuti na ono gdje piše da to ovisi o tome iz otkud je pozvan i za koje parametre...

34. Imamo destruktor ~Stog(); u nekoj definiranoj klasi Stog. Ako u glavnom programu instanciramo objekt na način Stog obj; kako će se taj objekt uništiti ili tako nešto!
Rješenje: AUTOMATSKI će se destruktor pozvati kada se izađe iz glavne funkcije i na taj način će se uništiti objekt.

36. Zadan je kod:

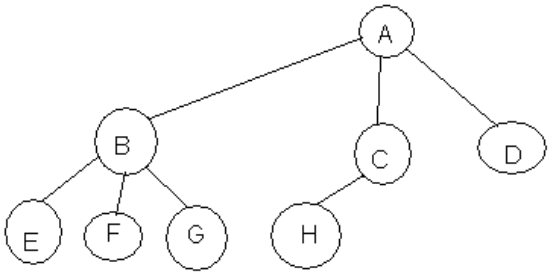
```
int a;
int &referenca = a;
a=1;

printf("%d %d ", a, referenca);
referenca=2;
printf("%d %d", referenca, a);
```

Što će se ispisati?
Rješenje: 1 1 2 2

---

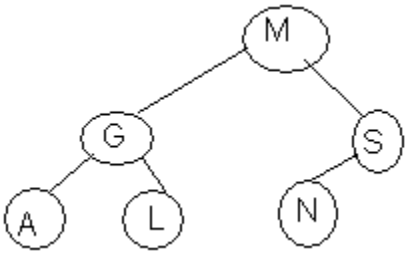
1. Stablo ima sljedeće atribute:



RJ: stupanj = 3, dubina = 3, potpuno

---

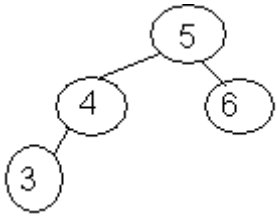
2. Zadana je funkcija (ne sjećam se koja), što se ispisuje?



RJ: G M

---

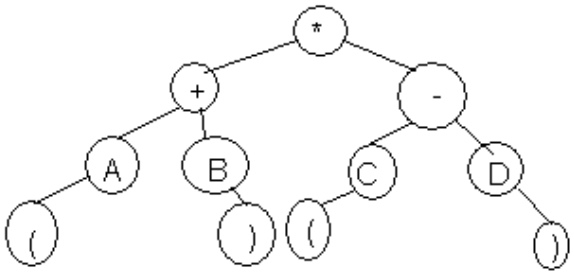
3. Zadano je 5, 4, 6, 3 i napravi se stablo i ima funkcija i treba znat što ispisuje:



RJ: 3 4 5 6

---

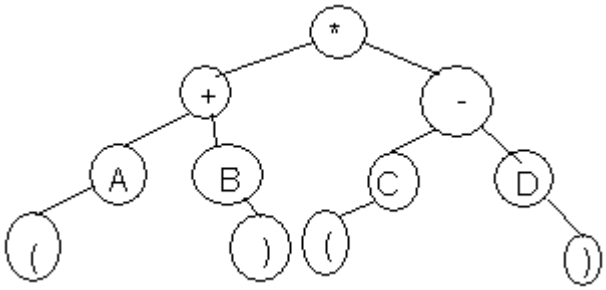
4. Inorder obilazak stabla na slici daje izraz:



RJ: (A + B) * (C – D)

---

5. Postorder obilazak stabla na slici daje izraz:



RJ: (A) B + (C) D - *

---

6. Koliko razina ima potpuno binarno stablo koje sadrži 100 čvorova i koliki je broj čvorova na posljednjoj razini ?

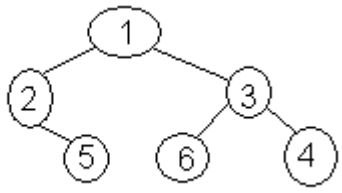
a) broj razina =6 broj čvorova=64
b) broj razina=7 broj čvorova=37 ←
c) broj razina =7 broj čvorova=50
d) broj razina =7 broj čvorova=64
e) broj razina =6 broj čvorova=50

---

7. Što će se ispisati funkcijom:

```
void ispis (cvor *korijen) {
      printf ("%c", korijen->element);
      if (korijen->lijevo && korijen->desno) {
            ispis (korijen->desno);
      ispis (korijen->lijevo);
      }
}
```

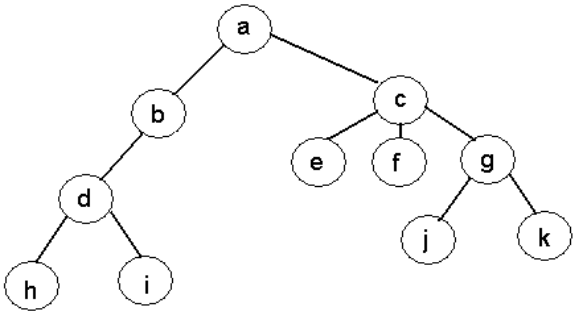
za stablo na slici pozivom funkcije?



RJ: 1 3 4 6 2

---

8. Koja od sljedećih tvrdnji nije istinita:



RJ:

a) Svi čvorovi sa stabla na slici su istog stupnja ←

b) U stablu sa slike listovi su:{h,i,e,f,j,k}

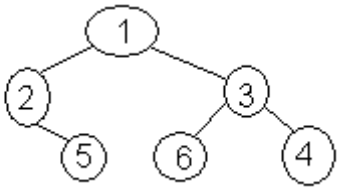
c) Maksimalni broj čvorova binarnog stabla na k-toj razini jednak je 2k-1

d) Maksimalni broj čvorova binarnog stabla dubine k jednak je 2k -1 za k>0

e) Binarno stablo koje je visine k i ima 2k -1 elemenata naziva se puno (full) binarno stablo

---

9. Zadana je neka funkcija (mislim postorder) i treba znat što ispisuje:



RJ: 4 6 3 5 2 1

---

10. Stupanj stabla (koji ima **n** razina) je:

RJ:

a) najmanji stupanj nekog čvora u stablu

b) n

c) najveći stupanj nekog čvora u stablu ←

d) broj čvorova u stablu

e) broj čvorova u potpunom stablu s *n* razina

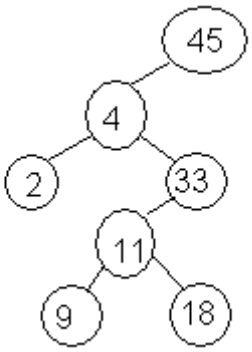
---

11. Koliko čvorova ima koso stablo sa **n** razina:

RJ:

a) 2*n-1
b) n+1
c) 2^n-1
d) n ←
e) 2^n

12.



Sortirano binarno stablo na slici (lijevo manji element, desno veći) generirano je sljedećim ulaznim nizom brojeva:

RJ: 45, 4, 33, 11, 2, 9, 18

---

13. U prazno binarno stablo uneseni su elementi 20, 15, 1, 3, 7, 48, 12, 19, 35. Kolika je dubina stabla?

RJ: 6

---

14. Što radi sljedeća funkcija:

```
void func (cvor *k, int *br){
        if (k != NULL){
              func(k->d, br);
              if(k->1 == NULL && k-> d == NULL) (*br)++;
                  func(k->d, br);
```

RJ: **a) broji listove u stablu ←**

b) broji elemente u stablu
c) računa zbroj elemenata u stablu
d) broji parne elemente u stablu
e) ništa od navedenog

15. Koja funkcija vraća najveći element u stablu:

```
RJ: int func(cvor *k){
            if (k->1 != NULL) return func(k->1);
                else return k->element;
```

---

16. Koja od sljedećih tvrdnji je istinita?

RJ: **a) inorder i preorder obilaskom bit će obrađeni svi čvorovi u stablu ←**

b) obilazak ppreorder moguće je jedino primijeniti na IpotpunimI stablima

c) postorder obilazak uvijek obrađuje samo listove stabla

d) preoder obilazak uvijek obrađuje samo listove stabla

e) inorder obilazak stabla obrađuje dvostruko više elemanata nego postorder obilazak

17. Uz prethodno ispravno deklarirane sve podatke i već formirano binarno stablo, što radi sljedeća funkcija:

```
void ispis (cvor *glava) {
      if (glava != NULL) {
            ispis (glava->lijevo);
            ispis (glava->desno);
            printf ("%s \n", glava->element);
      }
}
```

RJ:

a) uvijek ispisuje samo vrijednost elementa na koji pokazuje glava

b) funkcija ne radi ništa jer je tipa *void*

c) inorder ispisuje vrijednosti elemanata stabla

d) preorder ispisuje vrijednosti elemenata stabla

e) postorder ispisuje vrijednosti elemenata stabla ←

---

18. Što radi slijedeća funkcija?

```
int f(cvor *glava) {
      int i = 0;
      if (glava) {
            if (glava->lijevo || glava->desno) i++;
            i += f(glava->lijevo);
            i += f(glava->desno);
      }
      return i;
}
```

RJ:

a) Broji čvorove u stablu koji imaju lijevo dijete.

b) Broji čvorove u stablu koji imaju oba djeteta.

c) Broji čvorove u stablu koji imaju desno dijete.

d) Niša od navedenog.

e) Broji čvorove u stablu koji nisu listovi (imaju bar jedno dijete). ←

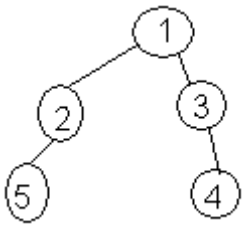
---

19. Što radi sljedeća funkcija:

```
int fx (cvor *glava) {
      if (glava) {
            return fx(glava->l) + fx(glava->d) + 1;
      } else return 0;
}
```

RJ:
a) broji razine stabla
b) računa zbroj elemenata u stablu
c) vraća vrijednost >=1 ako je stablo potpuno, 0 inače
d) broji listove stabla
e) broji čvorove stabla ←

20.



RJ: Stablo nije moguće ispisati sortirano inorder, preorder i postorder obilaskom

---

21. Pretraživanje binarnog stabla najbrže je ukoliko se radi o:

RJ: sortiranom potpunom stablu

---

22. Najefikasniji algoritam stvaranja gomile od *n* elemenata za najgori slučaj ima složenost:
RJ: a) $O(n*\log_2 n)$
b) $O(\log_2 n)$
c) $O(1)$
d) $O(n^2)$
e) O(n) ←

---

23. Koja je od sljedećih tvrdnji za gomilu točna?

RJ: Gomila se koristi kada je potrebno do najvećeg ili najmanjeg podatka doći algoritmom složenosti **O(1)**. Reorganizacija gomile nakon uklanjanja najvećeg ili najmanjeg podatka je složenosti **$O(\log_2 n)$**. Novi element u gomilu će se dodati algoritmom **$O(\log_2 n)$**.

---

24. Koji od ponuđenih ispisa gomile po razinama je ispravan ako je gomila formirana za ulazni niz 5, 10, 7, 3, 1, 90 algoritmom čija je složenost za najgori slučaj $O(n\log_2 n)$?

RJ:

a) 90
5 10
3 1 5

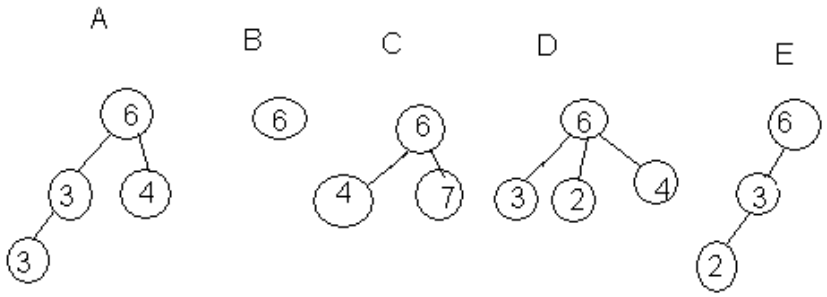
b) 90
10 7
5 1 3

c) 90 ←
5 10
3 1 7

d) 90
10 7
3 1 5

---

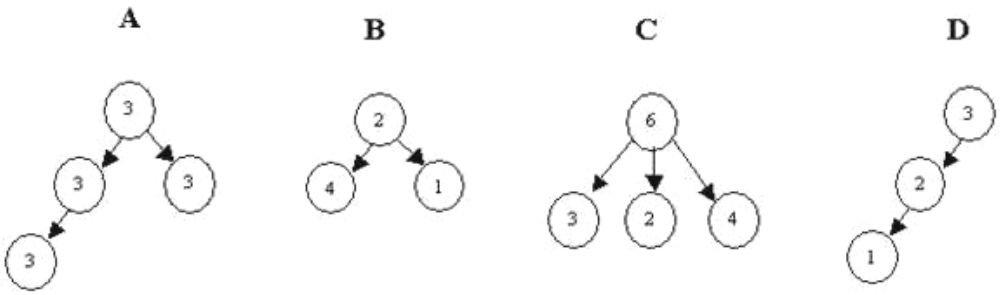
25. Koja od prikazanih stabla su gomile:



RJ: A, B

---

26. Koja od prikazanih stabala su gomile:



RJ:

a) A ←

b) B

c) C

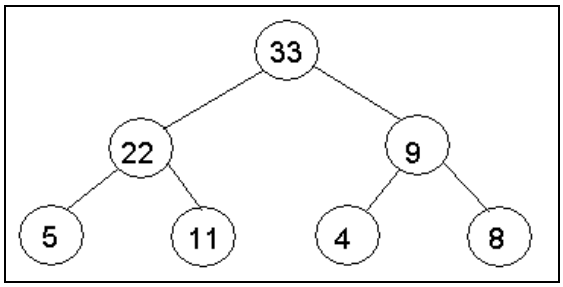
d) niti jedno od pokazanih atabala nije gomila

e) A, B, C, D

---

27. Koja slika prikazuje gomilu oblikovanu ulaznim nizom (11, 33, 4, 5, 22, 59) tako da za najgori slučaj složenost bude $O(n\log_2 n)$?

Rj:



---

28. Kada se gomila oblikuje dodavanjem jednog po jednog elementa u stablo uz očuvanje strukture gomile, tada je vrijeme izvođenja oblikovanja gomile za najgori slučaj (n je broj ulaznih elemenata):

RJ: a) O(log2n)
b) O(n)
c) O(n2 * log2n)
d) O(n * log2n) ←
e) O(n2)

29.

20 3 15 2 1 14 10

Ako je gomila realizirana u polju, u kojem od sljedećih elementi zapisani u polju:

RJ: 20
3 15
2 1 14 10

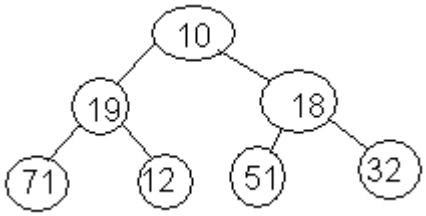
---

30. Stvori gomilu – složenost u najgorem slučaju poboljšane funkcije:

RJ: O(n)

---

31. Podaci za koje je potrebno stvoriti gomilu smješteni su u stablo na sljedeći način – koji će se prvi element zamijeniti (tu mi fali dio pitanja):

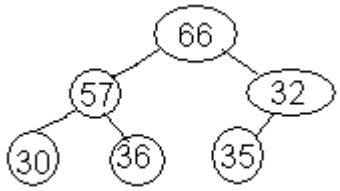


RJ: 51 i 18

---

32. 66 57 32 30 36 - zadano polje, ako se doda 35 na kraj sa kime će zamijeniti mjesto (tako nekako ide pitanje):

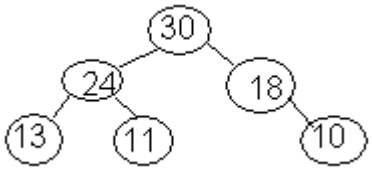
RJ: Broj 35 će zamijeniti mjesto s brojem 32



---

33. Koji od sljedećih ne zadovoljava gomilu:

RJ: 10 je na krivoj strani



---

34. Koje ne zadovoljava svojstvo gomile (ne sjećam se detalja):

RJ: 10 7 8 9 6 5 4 3 2 1

---

35. 22 33 44 99 66 77 88 55 – zadano je polje i koji element će se zamijeniti (ne sjećam se detalja):

RJ: 44 i 88

36.

RJ: Gomila je potpuno binarno stablo takvo da je podatak u nekom čvoru **<u>veći ili jednak</u>** podacima u čvorovima svoje djece.

37.

99 88 66 22 77 55 33 11 – zadana je gomila (poljem), nakon prvog podešavanja pri uzlaznom heap sortu što se dogodi (ne sjećam se točno pitanja):

RJ: 88 77 66 22 11 55 33 99

---

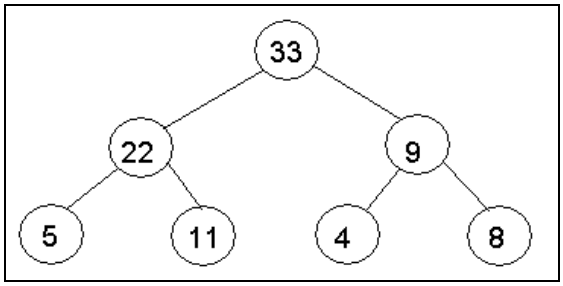
38. Struktura stog se dinamički najčešće predstavlja?

Rj: Jednostruko povezanom linearnom listom

---

39. Koja slika prikazuje gomilu oblikovanu ulaznim nizom (11, 33, 4, 5, 22, 59) tako da za najgori slučaj složenost bude $O(n\log_2 n)$?

Rj:



---

40. Pretraživanje binarnog stabla najbrže je ako se radi o:

Rj: Sortiranom potpunom stablu

---

41. Što radi:

```
void ispisi(cvor *glava) {
      cvor *p;
      for (p=glava; p!=NULL;p=p->sljed)
            printf("%d\n", p->element);
}
```

Rj: Ispisuje sve vrijednosti elemenata jednostruko povezane linearne liste.

---

42. U prazno binarno stablo uneseni su elementi 20, 15, 1, 3, 7, 48, 12, 19, 35. Kolika je dubina stabla?

Rj: 6

---

43. Koja je od slijedećih tvrdnji za gomilu točna?

Rj: Gomila se koristi kada je do najvećeg/najmanjeg potrebno doći sa složenošću $O(1)$. Složenost reorganizacije nakon uklanjanja prvog člana je $O(\log_2 n)$. Složenost dodavanja novog člana u gomilu je $O(\log_2 n)$.

---

44. Koji od ponuđenih ispisa gomile po razinama je ispravan ako je gomila formirana za ulazni niz 50 5 7 10 13 1 8 algoritmom čija je složenost za najgori slučaj O(nlog2n)?

a) 50
   13 10
   8 7 5 1
b) 50
   5 7
   10 13 1 8
c) 50
   13 7
   5 10 1 8
d) 50 ←
   13 8
   5 10 1 7
e) 50
   13 8
   10 7 5 1

---

45. Što ispisuje funkcija

```
void ispisi( struct cvor *glava ) {
      if( glava != NULL  && glava->elem % 2) {
            printf(" %d ", glava->elem);
            ispisi(glava->sljed);
       }
}
```

ako se u jednostruko povezanoj listi na koju pokazuje parametar glava nalaze sljedeći cijeli brojevi :
1 57 43 13 8 11 20 10 56 53

a) 1 57 43 13 ←
b) ne ispisuje ništa
c) 1 57 43 13 8 11
d) 1 57 43 13 8 11 20 10 56 53
e) 1 57 43 13 11 53

---

46. Ispravna deklaracija dvostruko povezane liste u memoriji glasi:

```
a) struct s1 {
      int mbr;
      char ime_pr[50];
      int spol;
      long pret;
      long sljed;
      }

b) struct s1 {
      int mbr;
```

```
      char ime_pr[50];
      int spol;
      zapis *pret;
      zapis *sljed;
       }
```

```
c) struct s1 {
      int mbr;
      char ime_pr[50];
      int spol;
      struct s1 *sljed;
      }
```

```
d) typedef struct s1{      ←
      int mbr;
      char ime_pr[50];
      int spol;
      } zapis1;

typedef struct s2{
      zapis1 element;
      struct s2 *pred;
      struct s2 *sljed;
      } zapis;
```

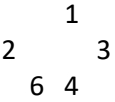
```
e) struct s1 {
      int mbr;
      char ime_pr[50];
      int spol;
      long *pret;
      long *sljed;
      }
```

---

47. Što će se ispisati funkcijom:

```
void ispis (cvor *korijen) {
      printf ("%c", korijen->element);
      if (korijen->lijevo && korijen->desno) {
            ispis (korijen->desno);
      ispis (korijen->lijevo);
      }
}
```

za stablo na slici pozivom funkcije



ispis (korijen);
ako je korijen u trenutku poziva pokazivač na korijen stabla?

a) 134625
b) 123456
c) 521634
d) 13462 ←
e) 12364

---

48. Koji od ponuđenih ispisa gomile po razinama je ispravan ako je gomila formirana za ulazni niz 5, 10, 7, 3, 1, 90 algoritmom čija je složenost za najgori slučaj O(n)?

a) 90
   7 10
   3 1 5
b) 90 ←
   10 7
   3 1 5
c) 90
   5 10
   3 1 5
d) 90
   10 7
   5 1 3
e) 5
   10 7
   3 1 5

---

49. Koliko čvorova ima *koso* stablo s **n** razina?

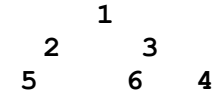
a) 2*n-1
b) n+1
c) 2^n-1
d) n ←
e) 2^n

---

50. Što će ispisati funkcija:

```
void ispis (cvor *korijen) {
     printf ("%c", korijen->element);
     if (korijen->lijevo && korijen->desno) {
          ispis (korijen->desno);
          ispis (korijen->lijevo);
     }
}
```

za stablo



RJ: 1 3 4 6 2

---

51. Kada se gomila oblikuje dodavanjem jednog po jednog elementa u stablo uz očuvanje strukture gomile, tada je vrijeme izvođenja oblikovanja gomile za najgori slučaj (n je broj ulaznih elemenata):

a) O(log2n)
b) O(n)
c) O(n2·log2n)
d) O(n·log2n) ←
e) O(n2)

---

52. Od elemenata {20, 15, 1, 3, 7, 48, 12, 19, 35} formirano je sortirano binarno stablo. Kolika je dubina stabla?
RJ: 6

---

53. Koja procedura pronalazi zadani element u jednostruko povezanoj listi?

```
a)
cvor *trazi1(cvor *glava,tip element) {
      cvor *p;
      for (p=glava;p!=NULL;p++)
            if (p->element==element)
                  return p;
      return NULL;
}

b)
cvor *trazi1(cvor *glava,tip element) {
      cvor *p;
      if (p->element==element) return p;
      return NULL;
}

c)
cvor *trazi1(cvor *glava,tip element) {
      cvor *p;
      for (p=glava;p!=NULL;p=p->sljed)
            if (p->element!=element)
                  return p;
      return NULL;
}

d)
cvor *trazi1(cvor *glava,tip element){      ←
      cvor *p;
      for (p=glava;p!=NULL;p=p->sljed)
            if (p->element==element)
                  return p;
      return NULL;
}

e)
```

```
cvor *trazi1(cvor *glava,tip element){
      cvor *p;
      for (p=glava;p!=NULL;p++)
            if (p->element!=element)
                  return p;
      return NULL;
}
```

---

54. Koji je stupanj stabla s **n** razina?

a) najmanji stupanj nekog čvora u stablu
b) n
c) najveći stupanj nekog čvora u stablu ←
d) broj čvorova u stablu
e) broj čvorova u potpunom stablu s **n** razina

---

55. Što radi zadana funkcija za poziv `f(glava,7,&br)` ako je glava pokazivač na početak jednostruko povezane liste?

```
cvor *f(cvor *p,int broj,int *br) {
      if (p) {
            ++(*br);
            if (p->broj==broj)
                  return p;
            else
                  return f(p->sljed,broj,br);
      } else return NULL;
}
```

RJ: vraća pokazivač na čvor koji sadrži `broj` i njegov redni broj u listi (preko `br`)

---

57. Dinamička struktura za jednostruko povezanu listu sadrži:

RJ: Pokazivac na prvi element liste i proizvoljan broj čvorova

---

58. Ovdje je napisana funkcija za dodavanje elemenata u stog ostvaren jednostruko povezanom listom. Na kraju funkcije je izostavljena jedna naredba. Treba izabrati naredbu koja ide na to mjesto da funkcija radi.
Funkcija otprilike izgleda ovako:

```
dodajelement(element **vrh,int vrijednost) {
      element novi = new element;
      novi->vrijednost=vrijednost;
      novi->slijed=*vrh;
      ##### <- Što nedostaje?
}
```

RJ: Nedostaje naredba `*vrh=novi;`

---

61. Sto ce se dogoditi nakon sto u ovu gomilu ubacimo broj 35 i pozovemo funkciju "ubaci"?

66 57 32 30 36 ← ubacujemo broj 35

RJ: Zamijeniti ce se 35 i 32

---

62. Koja procedura trazi element u listi?

RJ:

```
cvor *trazi (cvor *g, tip elem){
      cvor *p;
      for (p=g;p!=NULL;p=p->slijed)
            if (p->vrijednost=elem) return p;
            return NULL;
}
```

---

65. Zadano je:

```
        1
      2   3
   5   9 4
```

Što će se ispisati s:

```
void ispis (cvor *korijen){
      if korijen{
            ispis (korijen->desno);
            ispis (korijen->lijevo);
            printf("%c",korijen->element);
      }
}
```

RJ: 4 9 3 5 2 1

RJ: Da, zato što u slučaju NULL vrijednosti glave podaci bivaju izbrisani (il neš u tom stilu)

---

67. Dinamička podatkovna struktura za real. jednostruko povezane liste sastoji se od:

RJ: pokazivača na prvi element liste i proizvoljnog broja povezanih čvorova

---

71. Koja tvrdnja nije ispravna za zadanu funkciju?

```
cvor *trazi(cvor *glava, int sifra){
      if (glava->sifra==sifra) return glava;
      if (glava->slijed)
            return trazi(glava->slijed, sifra);
```

```
      else
            return NULL;
}
```

RJ: F-ja nije ispravna, ne radi za praznu listu.

---

72. Sto ce se dogoditi pozivom `f(glava),f(glava)` f-je:

```
void f(cvor *glava){
      if (glava->sljed){
            printf ("%s\n", glava->ime);
            f(flava->sljed);
      }
}
```

RJ: Dva puta ispisuje imena osim od posljednjeg cvora.

---

75. Koja je tvrdnja za jednostruko povezanu linearnu listu je istinita:

RJ: može ostvariti statičkom strukturom polje

---

76. Koja od ponuđenih funkcija ispravno implementira Heap sort?

```
a) void HeapSort(tip A[],int n) {
       int i;
       StvoriGomilu(A,n/2);
       for (i=n/2;i<=0;i--) {
             Zamijeni(&A[1],&A[i]);
             Podesi(A,1,i-1);
       }
}
```

```
b) void HeapSort(tip A[],int n) {        ←
       int i;
       StvoriGomilu(A,n);
       for (i=n;i>=2;i--) {
             Zamijeni(&A[1],&A[i]);
             Podesi(A,1,i-1);
       }
}
```

```
c) void HeapSort(tip A[],int n) {
       int i;
       StvoriGomilu(A,1);
       for (i=1;i<=n/2;i++) {
             Zamijeni(&A[n],&A[1]);
             Podesi(A,1,i+1);
       }
}
```

```
d) void HeapSort(tip A[],int n) {
       int i;
       StvoriGomilu(A,1);
       for (i=1;i<=n;i++) {
             Zamijeni(&A[n],&A[1]);
             Podesi(A,1,i+1);
       }
}
```