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

1. Rekurzija

Prvi primjer

Što će se ispisati izvođenjem sljedećeg programskog odječka ?

```
void f( int n )  {
    if( n == 2 ) return;
    if( n == 5 ) return;
    f(n-2);
    printf("%d", n);
}
int main() {
    f(9);
    return 1;
}
```

Ispisuje se 7 9

Drugi primjer

Koja od sljedećih funkcija računa n-ti član aritmetičkog niza:

a)
```
long aniz(long a0, long d, long n) {
     if (n >= 0) return a0;
     else return d + aniz(a0, d, n-1);
   }
```

b)
```
long aniz(long a0, long d, long n) {
     if (n == 0) return a0;
     else return d + aniz(a0, d, n+1);
   }
```

c)
```
long aniz(long a0, long d, long n) {
     if (n == 0) return a0;
     else return d + aniz(a0, d, n);
   }
```

d)
```
long aniz(long a0, long d, long n) {
     return d + aniz(a0, d, n-1);
   }
```

e)
```
long aniz(long a0, long d, long n) {
```

```
    if (n == 0) return a0;
    else return d + aniz(a0, d, n-1);
  }
```

Treći primjer

Što će se ispisati sljedećom funkcijom:

```
void f(int x) {
  if (x > 3) return;
  f (x-1);
  printf ("%d ", x);
}
```

ako je poziv funkcije

```
  f(2);
```

a) `-3 -2 -1 0 1 2`
b) `1 0 -1 -2 -3 -4` ... sve dok se ne prepuni stog
c) prepunit će se stog prije bilo kakvog ispisa
d) ništa, funkcija se odmah vraća u pozivajući program
e) `2 0 -1 -2 -3 -4` .... sve dok se ne prepuni stog

SORT-01

A posteriori analiza pokazala je da na nekom računalu sortiranje polja od 10000 elemenata algoritmom Selection Sort traje otprilike 1 sekundu. Kolika je realna procjena za vrijeme sortiranja dvostruko duljeg polja od 20000 elemenata na istom računalu istim algoritmom?

a) Više od 1 sekunde, no manje od 2 sekunde.
b) Otprilike 2 sekunde.
c) Više od 2 sekunde no manje od 4 sekunde.
d) Otprilike 4 sekunde.
e) Više od 4 sekunde no manje od 8 sekundi.

SORT-02

A posteriori analiza pokazala je da na nekom računalu sortiranje polja od 100000 elemenata algoritmom Merge Sort traje u prosjeku 1 sekundu. Kolika je realna procjena za vrijeme sortiranja dvostruko duljeg polja od 200000 elemenata na istom računalu istim algoritmom?

a) Otprilike 1 sekundu.
b) Više od 1 sekunde no znatno manje od 2 sekunde.
c) Otprilike 2 sekunde.
d) Više od 2 sekunde no znatno manje od 4 sekunde.
e) Barem 4 sekunde.

SORT-03

Apriorne ocjene za vrijeme sortiranja polja od n elemenata algoritmom Quick Sort glase:

a) $\Theta(n \log n)$ u prosječnom i u najgorem slučaju.
b) $\Theta(n \log n)$ u prosječnom slučaju, $\Theta(n^2)$ u najgorem slučaju.
c) $\Theta(n^2)$ u prosječnom i u najgorem slučaju.
d) $\Theta(n \log n)$ u prosječnom slučaju, $\Theta(n^{3/2})$ u najgorem slučaju.
e) $\Theta(n^{3/2})$ u prosječnom i u najgorem slučaju.

SORT-04

Polje: 17 31 45 43 11 24 8 uzlazno sortiramo algoritmom Quick Sort. Stožer se bira kao medijan između tri elementa koji se nalaze na početku, kraju odnosno sredini polja. Odredite medijan koji će biti izabran u prvom koraku algoritma.

a) 8
b) 24
c) 17
d) 43
e) ni jedan od ponuđenih

SORT-05

Polje [17 31 3 43 11 24 8] uzlazno sortiramo algoritmom Quick Sort. Stožer je početni element polja. Odredite dva pod-polja koja će nakon prvog koraka algoritma biti podvrgnuta rekurzivnim pozivima istog algoritma.

a) Prazno polje , [31 43 11 24 8]
b) [31 3 43] , [11 24 8]
c) [3 11 8] , [31 43 24]
d) [3 8 11] , [24 31 43]
e) [8 3 11] , [43 24 31]

SORT-06

Polje [17 31 3 43 11 24 8] uzlazno sortiramo algoritmom Shell Sort, uz primjenu inkrementalnog slijeda brojeva: 3 > 2 > 1. Odredite izgled polja nakon prve faze Shell Sort-a, dakle nakon provedenog 3-subsorta.

a) [8 11 3 17 31 24 43]
b) [3 8 11 17 24 31 43]
c) [11 24 3 43 17 31 8]
d) [3 24 8 31 11 43 17]
e) [8 31 3 17 11 24 43]

STOG-01

Koja od sljedećih nizova naredbi u pseudokodu će zamijeniti vrijednost varijabli A i B pomoću stoga:

a) stavi(A); stavi(B); skini(A); skini(B);

b) stavi(A); skini(B);

c) stavi(B); skini(A);

d) stavi(A); stavi(Pom); stavi(B); stavi(Pom); skini(A); stavi(Pom);skini(B);

e) stavi(A); skini(B); stavi(B); skini(A);

STOG-02

Na stog prikazan poljem pohranjuju se samo cijeli brojevi. Prototip funkcije za skidanje cijelog broja sa stoga je (funkcija vraća 0 ili 1, ovisno o tome da li se zapis uspio skinuti s vrha stoga):

a) `int skini(int stavka, int stog[], int *vrhStog);`
b) **`int skini(int *stavka, int stog[], int *vrhStog);`**
c) `int skini(float stavka, float stog[], int vrhStog);`
d) `int *skini(int *stavka, int stog[], int vrhStog);`
e) `void *skini(int *stavka, int stog[], int n, int *vrhStog);`

STOG-03

Složenost funkcije

```
int dodaj (zapis stavka, zapis stog[], int n, int *vrh) {
   if (*vrh >= n-1) return 0;
    (*vrh)++;
    stog [*vrh] = stavka;
    return 1;
}
```

je:

a) O(n)
b) O(log n)
c) složenost ovisi o veličini zapisa stavke, pa se ne može jednoznačno odrediti
d) O(1)
e) $O(\log_2 n)$

STOG-04

Ako je stog realiziran cjelobrojnim poljem od n elemenata, kolika je apriorna složenost skidanja SVIH elemenata sa stoga:

a) O(1)
b) <u>O(n)</u>
c) $O(n^2)$
d) $O(\log_2 n)$
e) ovisi o operacijskom sustavu

STOG-05

Ako fukcija stavljanja na stog vraća 1 u slučaju uspjeha a 0 u slučaju neuspjeha i ima prototip

```
int push (int element);
```

a funkcija skidanja sa stoga vraća vrijednost elementa s vrha ili -1 ako je stog prazan i ima prototip

```
int pop ();
```

što će biti na stogu nakon obavljanja sljedeće naredbe, uz pretpostavku da je stog bio prazan i da stog raste s lijeva na desno:

```
push(push(pop()));
```

a) 1 1
b) <u>-1 1</u>
c) 1
d) 0
e) stog će biti prazan

STOG-06

Funkcija za dodavanje elementa na stoga realiziran listom glasi:

<u>a)</u>

```
atom *dodaj (atom *vrh, int element) {
          atom *novi;
          if ((novi = (atom *) malloc(sizeof(atom))) != NULL) {
               novi->element = element;
               novi->sljed = vrh;
           }
       return novi;
     }
```

b)

```
atom *dodaj (atom *vrh, int element) {
```

```
          atom *novi;
          novi->element = element;
          novi->sljed = vrh;
          return novi;
     }
```

c)

```
int dodaj (atom *vrh, int element) {
          atom *novi;
          if ((novi = (atom *) malloc(sizeof(atom))) != NULL) {
               novi->element = element;
               novi->sljed = vrh;
               return 1
           }
          else
                 return 0;
     }
```

d)

```
atom *dodaj (atom *vrh, int element) {
          atom *novi;
          if ((novi = (atom *) malloc(sizeof(atom))) != NULL) {
               novi = element;
          }
       return novi;
     }
```

e)

```
int dodaj (atom *vrh, int element) {
          atom *novi;
          if ((novi = (atom *) malloc(sizeof(atom))) != NULL) {
               novi = element;
               return 1
          }
          else
                 return 0;
     }
```

RED-01

Ukoliko je ulaz = 1, a izlaz=4, koliko ima elemenata u redu realiziranom pomoću cirkularnog polja, ako je veličina polja 10 (pretpostavite da ulaz pokazuje na prvi prazan element, dok izlaz pokazuje na prvi stavljeni element)?

a) 7
b) 6
c) 3
d) 5
e) ne može se odrediti

RED-02

Neka je na sljedeći način napisana funkcija koja skida element tipa `tip` iz reda realiziranog cikličkim poljem:

```
int SkiniIzReda (tip *element, tip red[], int n,
        int *izlaz, int ulaz) {
  if (ulaz == *izlaz) return 0;
  (*izlaz) ++;
  *izlaz %= n;
  *element = red[*izlaz];
  return 1;
}
```

Koja je od sljedećih tvrdnji **lažna**?

a) Složenost funkcije je O(1).
b) Funkcija vraća 0, ako se iz reda može skinuti točno jedan element.
c) Za poziv funkcije, kada u redu postoji barem jedan zapis koji se može skinuti iz reda, funkcija vraća 1.
d) Funkcija vraća 1, ako je zapis uspješno skinut iz reda.
e) Za poziv funkcije, kada je red prazan, funkcija vraća 0.

RED-03

U red realiziran jednostruko povezanom listom pohranjuju se zapisi koji sadrže cijele brojeve. Prototip funkcije za skidanje zapisa iz tako realiziranog reda je (funkcija vraća 1 ili 0, ovisno o tome je li zapis uspješno skinut iz reda):

a) `int skini (cvor **ulaz, cvor **izlaz, int element);`
b) `int skini (cvor **ulaz, cvor **izlaz, int *element);`
c) `void skini (cvor **ulaz, cvor **izlaz, int *element);`
d) `int skini (cvor *ulaz, cvor **izlaz, int element);`
e) `int skini (cvor *ulaz, cvor **izlaz, int *element);`

PITANJA S FORUMA:

STOG:
-naći točnu tvrdnju-4 su jako glupe, točna je: Da bi se pristupilo elementu na dnu stoga, treba se maknuti sve sa vrha
-dva programčića sa stogom (treba samo znati kako rade funkcije dodaj i skini)
- Imamo funkcije int push(int elem) i int pull() koje rade sa stogom. Na stogu postoje već neki elementi (nije prazan). Funkcija push vraća 1 ako je uspjela, 0 inače, a funkcija pull vraća element kojeg skine sa stoga i ne vraća ništa za grešku.
Što će biti na stogu nakon naredbe push(push(push(5)) + pull());
Rješenje : 5 2
- Stog punimo sa for petljom i=1 do 10, onda ga praznimo i=1 do 5, sto je ostalo?
Rjesenje: 1,2,3,4,5
- Na stog prikazan poljem pohranjuju se samo cijeli brojevi. Prototip funkcije za skidanje cijelog broja sa stoga je:
-Ponuđene su cijele funkcije za stavljanje elementa na stog...odabrati točnu
-Stog, koji ima barem dva elementa, treba zamijeniti vrijednosti dva elementa s vrha
a) int a, b a= pop(), b=pop(), push(a), push(b)<---
- Stog ostvarem statičkim poljem od *n* elemenata moze: primiti *n* elemenata
-. Netočna tvrdnja: Prazan stog je greška u programu
- Funkcije *dodaj* i *skini* uvijek imaju istu složenost bez obzira na broj članova**.**
-Bio je jedan koji stavlja na stog od 0 do 10 kao, al' veličina stoga je 5 i onda printf-a ono što popa...
Rješenje: 4 3 2 1 0
- Napisan neki, sa stogom i sad, MAXSTOG je 10, on stavlja elemente na stog i pitanje je koliko najviše može stavitI? A caka je u tome što mu je vrh definiran kao 0 a ne -1 što znači da je gore već jedan element. Rj: Može se stavit 9
- Je li u funkciji `dodajured)` potrebno primati pokazivač na izlaz iz reda po referenci i zašto?
RJ: Da, zato što u slučaju NULL vrijednosti glave podaci bivaju izbrisani (il neš u tom stilu)
- koja je složenost dohvaćanja zadnjeg elementa reda (red je realiziran listom)?**RJ:** O(n)
- Koji je prototip funkcije za skidanje elemenata iz reda listom (funkcija vraća 1 ako je uspjela skinuti element, inače vraća 0)?
- U red jednostruko povezanom listom pohranjuju se cjelobrojni zapisi. Prototip f-je za skidanje iz reda (1 za uspjesno, 0 neuspjesno obavljeno)
- Koja je tvrdnja za jednostruko povezanu linearnu listu je istinita: **RJ:** može ostvariti statičkom strukturom polje

Red ostvaren ciklickim poljem, koja je tvrdnja neistinita?

```
int SkiniIzReda(tip *element, tip red[], int n, int *izlaz, int ulaz){
      if(ulaz==*izlaz) return 0;
      (*izlaz)++;
      *izlaz %=n;
      *element=red[*izlaz];
      return 1;
}
```

RJ: F-ja vraca 0, ako se iz reda moze skinuti tocno 1 element

Zadatak:
skini, stavi
1 ako je uspješno obavljeno, 0 ako nije uspjelo
pretpostavka da na stogovima ima dovoljno mjesta
Zadana je funkcija:
void prepis(int stog1[], int stog2[]) {
int element;
if skini(stog1, &element)
prepis(stog1, stog2)
stavi(stog2, &element)
}
Rješenje je:
premještamo elemente sa stog1 na stog2, i redoslijed elemenata stog1 je ISTI! kao redoslijed elemenata stog2

zadatak:
Prototipi skini i dodaj su zadani
Što će ispisati?
Kod je isto tu bio zadan...
int a=1, b=2, c=3;
dodaj a
dodaj b
dodaj c
skini a
skini c
skini b
// nisam sigurna da je takav redoslijed, ali slično je bilo ako ne isto
printf("%d %d %d", a, b, c);
Rješenje je: 3 1 2
16. Stog iz starih blitzeva gdje je rješenje 3 1 2

SORT:
-zadana tri koraka bubble sorta, što će se dogoditi u četvrtom
-odrediti stvarni medijan od zadanog niza (ima to pitanje u onim blicevima od lani, odgovor je 5,tj. treba naći broj koji bi stajao u sredini polja da je ono sortirano)
-koji sort treba najviše memorije (merge sort)
-koji sort nije poželjno koristiti s velikim poljima (bubble sort)
-Najmanje je uputno korisiti BUBBLE SORT kod velikih polja
-zadan je kod i treba vidjeti koji je to sort. u mom slučaju bio je insertion.
-slozenost bubble sorta: O(n2)
- Koja je apriorna složenost shell sorta Rješenje: ovisi o hk
-koji od sortova nema najlosiju slozenost O(n2):merge sort
-ako stog ima najvise n podataka,kolika je slozenost prilikom skidanja SVIH podataka s njega:O(n)
-zadan je niz brojeva ,kako izgleda nakon sortiranja shell sortom s hk=3
-zadana je nesortiran niz,i 4-5 koraka sortiranja.treba prepoznat koji sort se koristi ,meni bilo
34251
34251
23451
23451
12345
,ovo je insertion sort

3x Zadan kod - koji je to sort?

- Rečeno je da se treba sortirati selection sortom polje 3 5 1 4 2 i onda se pita koji koraci odgovaraju tom sortu:
malo je zbunjeno napisano, ali točan je odgovor (*ovaj prvi redak su ponovo napisali*):
Rješenje: 3 5 1 4 2
1 5 3 4 2
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
13. Pronaći stvarni medijan u sljedećem nizu
4 6 3 9 11 15 17
Rješenje: 9, samo se pronađe broj koji bi bio na polovici niza da je on sortiran

20. Shell sort s razmakom 3 za niz 66 88 99 22 77 55 33 11.
Rješenje: 22 11 99 33 77 55 33 88

- Kad vam daju niz brojeva 2,4,6,15,9,5,1, MEDIJAN (NE STOŽER!!!) je 5, a ne 2, razlika je u onoj funkciji sa predavanja gdje ako po njoj radite, NE RADITE ZADNJI KORAK, Zamijeni(&polje[sredina], &polje[desno-1]), dakle, još jednom, to se ne radi, već se **samo medijan klikne, u ovom slučaju 5**
-Koja je apriorna složenost Bubble sorta? e) O(n^2) <---
-koji sort ima najveće mem zahtjeve: Najveće memorijske zahtjeve ima merge sort
-zadan algoritam pita koji je to sort
-zadan niz pita koji je od ponuđenih pravilan selection sort
- Insertion sort na nekom čudnom primjeru gdje se sve ponavlja po dva puta...
-Quick sort, medijan tri elementa
zadano je polje 15 18 7 17 9 4
rješenje je: 4 18 9 17 7 15
-Zadan je prototip sorta, pitanje je:
Koji je to sort?
Rješenje: Bubble sort
-Što je od sljedećega istinito? b) najgore kod Shell sorta je O(n^2) <---