Конструкција и анализа алгоритама, И смер, писмени испит - септембар 2018.

1. **(6 поена)** Нека су d_1, d_2,...,d_n природни бројеви, $n \geq 2$. Доказати да ако је $\sum_{i=1}^{n} d_i = 2n - 2$, онда постоји стабло са n чворова чији су степени бројеви d_1, d_2,...,d_n.

2. **(6 поена)** Формулисати инваријанту петље у основној петљи алгоритма *Quicksort* и доказати коректност алгоритма.

3. **(6 поена)** Задат је граф $G = (V, E)$. Конструисати алгоритам сложености $O(|E| + |V|)$ који у случају ацикличког графа G даје тополошко уређење чворова графа, а у супротном проналази неки циклус у графу G. Прецизно и детаљно образложити алгоритам и временску сложеност.

4. **(6 поена)** Претпоставимо да се израчунавање производа две квадратне матрице реда четири може свести на k производа елемената. (а) Колика би била асимптотска сложеност општег алгоритма за израчунавање производа две квадратне матрице реда n заснованог на овом свођењу. (б) За коју највећу вредност k би овакав алгоритам био још увек бржи од Штрасеновог алгоритма?

5. **(6 поена)** Размотрите следећи алгоритам за утврђивање да ли граф има клику (подграф графа G у ком су сви чворови међусобно повезани гранама из G) величине k. Најпре се генеришу скупови од по тачно k чворова (има их $O(n^k)$). Затим се за сваки подграф индукован тим подскупом проверава да ли је комплетан. Да ли је овај алгоритам полиномијалне временске сложености? Прецизно и детаљно образложити одговор.

6. **(6 поена)** Нека је дат низ целих бројева a са n чланова. Конструисати алгоритам који ће распоредити заграде у израз $a[1] - a[2] - ... - a[n]$ тако да вредност израза буде минимална. Прецизно и детаљно образложити временску и просторну сложеност алгоритма.

Решења

1. Постоји. Доказ се изводи индукцијом по броју чворова n. (1 поен)

База индукције: Ако $n = 2$, и ако је $d_1 + d_2 = 2*2-2=2$, онда може да постоји стабло тако да $d_1=1$, $d_2=1$. То је стабло које има 2 чвора и то оба степена 1. (2 поена)

Индуктивна хипотеза: Претпоставимо да је тврђење тачно за d_1, d_2,...,d_{n-1} и нека је дато $n \geq 3$ природних бројева d_1, d_2,...,d_n чији је збир $\sum_{i=1}^{n} d_i = 2n - 2$ (3 поена)

Тада постоји бар једно i $(1 \leq i \leq n)$ бар једно j $(1 \leq j \leq n)$ тако да је $d_i=1$, $d_1 > 1$ Зашто? ПРЕТПОСТАВИМО СУПРОТНО: $\forall i \leq n\ d_i \neq 1$. Тада не може да важи $\sum_{i=1}^{n} d_i = 2n - 2$ (инд.хипотеза) Слично за d_j се претпостави супротно: $\forall j \leq n\ d_j \leq 1$ Тада је $\sum_{i=1}^{n} d_i \leq n$, а опет за $n \geq 3$ је $n < 2n - 2$ Избацимо из скупа d_i и умањимо d_i за један и добиће се скуп за кога по индуктивној хипотези постоји стабло са тим степенима. Потом на то стабло додамо нови чвор, заправо лист степена $d_i = 1$ прикаченог за чвор степена $d_j - 1 \geq 1$ и добија се стабло из формулације задатка тј. стабло са n чворова степена d_1, d_2,...,d_n. Решење проблема у општем случају није јединствено.

2.

```
Algoritam Quicksort(X,n)
Ulaz: X (niz od n brojeva)
Izlaz: X (sortirani niz)
begin
  Q_sort(1,n)
end

procedure Q_sort(Levi, Desni)
begin
   if Levi<Desni then
      S=Razdvajanje(X,Levi,Desni)  {S je indeks pivota posle razdvajanja}
     Q_sort(Levi, S-1)
    Q_sort(S+1, Desni)
end
```

```
function Razdvajanje(X, Levi, Desni)
Ulaz: X (niz), Levi (leva granica niza), Desni (desna granica niza)
Izlaz: X, S=Razdvajanje, takav indeks da je X[i] <= X[S] za sve i<=S i X[j] > X[S] za sve j > S

begin
    pivot=X[Levi]
   L=Levi;  D=Desni;
   while L < D do
      while X[L] <= pivot and L<=Desni do L=L+1
      while X[D] > pivot and D>=Levi do D=D-1
      if L<D then swap(X[L], X[D])

 Razdvajanje=D
zameni X[Levi] sa X[S]
end
```

На крају сваког проласка кроз основну петљу **while L < D do** тачно је следеће тврђење за $X[i] \leq pivot$ за $1 \leq i \leq L$ и $X[i] > pivot$ за $D \leq i \leq n$ што се лако доказује индукцијом по броју пролазака кроз петљу. (2 поена)

Поред тога у самом проласки кроз петљу L се повећава, а D смањује најмање за 1, па после највише $n/2$ пролазака кроз петљу мора да наступи услов за искакање из петље $L \geq D$. Прецизније, у том тренутку је $D = L - 1$, јер се L зауставља на вредности индекса D затеченој из претходног проласка кроз петљу, а онда се D смањује само за 1. (2 поена)

Пошто је показивач L у последњем проласку прешао све чланове низа до индекса D, закључује се да је на крају $X[i] \leq pivot$ за $i \leq D$ и $x[i] > pivot$ за $i \geq D + 1$.(2 поена)

3. Основа алгоритма је алгоритам за тополошко сортирање (сложеност $O(|E| + |V|)$) који чува листу чворова улазног степена нулаи са ње скида један по један чвор графа (заједно са инцидентним гранама, смањујући за један улазне степене чворова на крајевима грана и додајући на листу евентулане нове чворове улазног степена нула).

Ако се у току извршавања испразни листа, а нису елиминисани сви чворови графа, у преосталом графу сви чворови имају улазни степен бар један, што омогућује проналажење циклуса у временској сложености $O(|E|)$. (2 поена)

Поступак проналажења циклуса:

1. бира се произвољан чвор v, означава као посећен

2. пронађи чвор w суседан чвору v, чвор w се означава као посећен

Понавља се корак 1 са чвором w, и тако даље све до наиласка на већ означен чвор који затвара циклус. (2 поена)

Проналажење циклуса се одвија у временској сложености $O(|E|)$ зато што се пказивачи уназад формирају у $O(|E|)$ корака, јер се врши обрада једне по једне гране из листе повезаности графа . (2 поена)

4. Без смањења општости, ограничимо се на случај $n = 4^k$. Ако са $T(n)$ означимо сложеност израчунавања производа две матрице реда n , онда је $T(n) = kT(n/4) + cn^2$. По мастер теореми $T(n) = O(n^{log_4 k})$. Да би алгоритам био асимтотски бржи од Штрасеновог, мора да важи $n^{log_4 k} < n^{log_2 7}$ односно $log_4 k < log_2 7 = log_4 49$ односно $k < 49$

5. Не. Погледати сличан задатак са вежби и у уџбенику.

6**.** Neka je dat niz celih brojeva *a* sa *n* članova. Konstruisati algoritam koji će rasporediti zagrade u izraz a[1]-a[2]-...-a[n] tako da vrednost izraza bude minimalna.

Resenje:

Ako **k**-to oduzimanje sleva na desno je poslednje oduzimanje koje se izvrsava u resenju (tj. optimalnom zagradjivanju), onda su u izrazu levo od **k**-tog operatora minus zagrade postavljene tako da on ima mnimalnu vrednost,a u izraz desno tako da ima maksimalnu vrednost.
Neka je **l**-to oduzimanje sleva ono koje se poslednje izvrsava pri racunanju *maksimalne* vrednosti izraza. Tada su u prvom delu izraza zagrade postavljene tako da on ima maksimalnu vrednost, a u drugom delu tako da ima minimalnu vrednost.

Stoga formiramo dve matrice *Max* i *Min* tako da:

za 1 <=i <=n:	Min[i,i]=Max[i,i]=a[i]
za 1 <=i < j < = n:	Min[i,j] je najmanja vrednost izraza koja pocinje sa a[i] i zavrsava se sa a[j]
za 1 <=i < j <= n:	Min[j,i] je redni broj oduzimanja koje se poslednje izvrsava tako da izraz koji pocinje sa a[i] i zavrsava se sa a[j] ima minimalnu vrednost
za 1 <= i < j <= n:	Max[i,j] je najveca vrednost izraza koja pocinje sa a[i] i zavrsava se sa a[j]
za 1 <= i < j <= n:	Max[j,i] je redni broj oduzimanja koje se poslednje izvrsava tako da izraz koji pocinje sa a[i] i zavrsava se sa a[j] ima maksimalnu vrednost

Svaka od matrica *Max* i *Min* se koristi za pamcenje dva tipa podataka:

- informacija o ekstremnoj vrednosti izraza
- informacija o rednom broju poslednje izvrsene operacije oduzimanja, potrebne za rekonstrukciju izraza

dakle:

- ako je j-i=1, onda:
  Min[i,j]=Max[i,j]=a[i]-a[j]
  Min[j,i]=Max[j,i]=i (jer se izvrsava samo jedno oduzimanje)

- ako je j-i>1, onda:
  $Max[i,j]=\max_{i <= k < j} \{Max[i,k]-Min[k+1,j]\}$
  $Min[i,j]=\min_{i <= k < j} \{Min[i,k]-Max[k+1,j]\}$

Matrice se popunjavaju po dijagonalama, a pocev od glavne i vrsi se popunjavanje elemenata ciji indeksi su [s,t] gde s=i, t=j-i.

```
Algoritam OptIzrazOduzimanja (a,n)
ulaz: a /* niz celih brojeva, clanova izraza a[1]-a[2]-...-a[n] */
      n /* dimenzija niza a */
izlaz: optimalni izraz sa zagradama

for (k=1;k < = n; k++) Max[k,k]=Min[k,k]=a[k];

for (t=1;t < = n-1; t++)
  for(s=1;s < = n-t; s++)
   {
    Max[t+s,s]=s;
    Max[s,t+s]=Max[s,s]-Min[s+1,t+s];
    Min[t+s,s]=s;
    Min[s,t+s]=Min[s,s]-Max[s+1,t+s];
   for(k=s+1;k<=t+s-1;k++)
     {
       if (Max[s,k]-Min[k+1,t+s] > Max[s,t+s])
         {
           Max[s,t+s]=Max[s,k]-Min[k+1, t+s];
           Max[t+s,s]=k;
         }

         if (Min[s,k]-Max[k+1,t+s] < Min[s,t+s])
         {
           Min[s,t+s]=Min[s,k]-Max[k+1, t+s];
           Min[t+s,s]=k;
         }
     }

  }

OPT_ispis(1,n);
   }

Algoritam OPT_ispis (i,j);
if (i==j) print(a[i]);
else
   {
     print( "(" ); OPT_ispis(i, Min[j,i]);
```

```
    print( ") - (" ); OPT_ispis(Min[j,i]+1, j); print (" ) " );
  }
}
```