Konstrukcija i analiza algoritama Grupa 1

1. Koja reč iz skupa S (S={'jabuka','matematika','informatika','fizika','kruska','motika'}) je najbliža reči *nauka* ako se kao mera rastojanja koristi Levenštajnovo rastojanje? Uz odgovor priložiti vrednost rastojanja do svake niske iz skupa S.

Rešenje:

d(jabuka, nauka)=2 0 1 2 3 4 5 1 1 2 3 4 5 2 2 1 2 3 4 3 3 2 2 3 4 4 4 3 2 3 4 5 5 4 3 2 3 6 6 5 4 3 2	d(matematika, nauka)=7 0 1 2 3 4 5 1 1 2 3 4 5 2 2 1 2 3 4 3 3 2 2 3 4 4 4 3 3 3 4 5 5 4 4 4 4 6 6 5 5 5 4 7 7 6 6 6 5 8 8 7 7 7 6 9 9 8 8 7 7 10 10 9 9 8 7	d(informatika, nauka)=7 0 1 2 3 4 5 1 1 2 3 4 5 2 1 2 3 4 5 3 2 2 3 4 5 4 3 3 3 4 5 5 4 4 4 4 5 6 5 5 5 5 5 7 6 5 6 6 5 8 7 6 6 7 6 9 8 7 7 7 7 10 9 8 8 7 8 11 10 9 9 8 7
d(fizika, nauka)=4 0 1 2 3 4 5 1 1 2 3 4 5 2 2 2 3 4 5 3 3 3 3 4 5 4 4 4 4 4 5 5 5 5 5 4 5 6 6 5 6 5 4	d(kruska, nauka)=3 0 1 2 3 4 5 1 1 2 3 3 4 2 2 2 3 4 4 3 3 3 2 3 4 4 4 4 3 3 4 5 5 5 4 3 4 6 6 5 5 4 3	d(kruska, nauka)=4 0 1 2 3 4 5 1 1 2 3 4 5 2 2 2 3 4 5 3 3 3 3 4 5 4 4 4 4 4 5 5 5 5 5 4 5 6 6 5 6 5 4

2. Napisati C program koji metodom dinamičkog programiranja brojevnom nizu **A[0..N-1]** određuje maksimalnu dužinu strogo rastućeg podniza elemenata (koji ne moraju biti susedni, ali sa rastućim indeksima). Na primer, ako je dat niz od 9 elemenata: 2, 1, 5, 5, 2, 7, 1, 9, 11; traženi podniz je dužine 5 i obrazuju ga: 2, 5, 7, 9 i 11.

Rešenje:

Označimo sa **L[i]** vrednost funkcije, koja je jednaka dužini maksimalnog rastućeg podniza segmenta celobrojnog niza **A** od prvih **i** elemenata koji se završava sa **a[i]**. Da bi se rešio zadatak treba izračunati vrednosti **L[i]** za **i=0,...,n-1.**

Važi da je **L[0]=1**.

Prema uslovu zadatka da bi se izračunao **L[i]** (**0<i<n**) treba iskoristiti već izračunate vrednost **L[j], j=0, ..., i-1,** i odrediti najveću za koju važi **a[j] < a[i].** Dobijena vrednost plus 1 daje traženi **L[i]**. Dakle, izračunavanje se realizuje po formuli

L [i]=max L[j]+1 j=0,1,.., i-1 za koji je a[j]< a[i]

L [i]=1 ako a[i] nije vece ni od jednog prethodnika

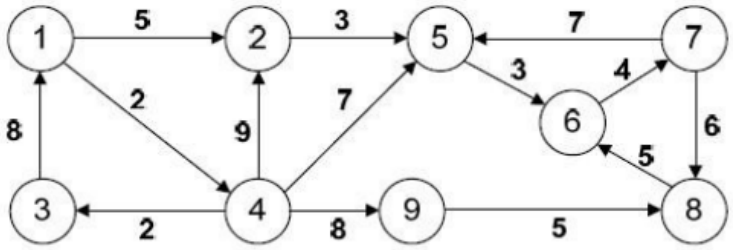
Da bi se mogao rekonstruisati maksimalni podniz može se uvesti podniz u oznaci **last[]** koji se inicijalizuje sa -1. Pri određivanju **L[i]** njemu se dodeljuje indeks onog **a[j], j=0, ... , i-1** koje je manje od **a[i]** i čije je **L[j]** najveće.

```
#include <iostream.h>
const int maxN=20;
const int n=9;
int a[maxN]={2,1,5,5,2,7,1,9,11};
int L[maxN],last[maxN];
void pisi(int i)
{
if (i>=0)
 {
   pisi(last[i]);
   cout << a[i] << " ";
 }
}

int main()
{
int Ind,max,i;
for (i=0; i<n; i++) // inicijalizacija pokazivaca na prethodni
last[i]=-1;
L[0]=1; // duzina podniza u koji ulazi samo prvi(nulti)
```

```
for (i=1; i<n; i++)
{
 max=0;
   for (int j=0; j<i; j++)
     if (a[j]<a[i] && L[j]>max)
     {
       max=L[j];
       last[i]=j;
    }
   L[i]=max+1; // max=0 ako nijedan a[j] nije manji od a[i]
}

Ind=0;
cout << "Trazeni podniz je ->";

for (i=1; i<n; i++)
  if (L[i]>L[Ind]) Ind=i;

pisi(Ind);
cout << endl;
return 0;
}
```

3. Konstruisati algoritam za konverziju oktalnog zapisa nekog broja u heksadecimalni. Dokazati korektnost napisanog algoritma.

4. Dat je na slici usmereni graf G = (V, E). Odrediti najkraće puteve od čvora 4 do svih ostalih čvorova u grafu.



Rešenje: Primenom Dijkstra algoritma dobija se konačno rešenje

S		d							
		1	2	3	5	6	7	8	9
4	-	∞	9	2	7	∞	∞	∞	8
4,3	3	10	9	2	7	∞	∞	∞	8
4,3,5	5	10	9	2	7	10	∞	∞	8
4,3,5,9	9	10	9	2	7	10	∞	13	8
4,3,5,9,2	2	10	9	2	7	10	∞	13	8
4,3,5,9,2,1	1	10	9	2	7	10	∞	13	8
4,3,5,9,2,1,6	6	10	9	2	7	10	14	13	8
4,3,5,9,2,1,6,8	8	10	9	2	7	10	14	13	8
4,3,5,9,2,1,6,8,7	7	10	9	2	7	10	14	13	8

5. Da li su sledeća tvrđenja tačna? Obrazložiti netačna tvrđenja kontraprimerom ili tačnim tvrđenjem.
a) Topološko sortiranje daje takav redosled čvorova da je uvek moguće konstruisati Hamiltonov put od redom sortiranih čvorova.
Ne. Ako susedni cvorovi topoloskog redosleda nisu putno povezani, onda ne mozemo konstruisati Hamiltonov put.
b) Topološko sortiranje radi efikasnije od DFS algoritma na neusmerenim grafovima koji imaju neparan broj ciklusa.
Ne. Topološko sortiranje se primenjuje na aciklicnim grafovima.
c) Levenštajnovo rastojanje d za niske X, Y, Z ima svojstvo d(X,Y)+d(Y,Z) <= d(X,Z)
Ne. Vazi nejednakost trougla, >=
d) Vremenska složenost Dijkstra algoritma jednaka je vremenskoj složenosti algoritma topološkog sortiranja.
Ne. Vidi poglavlje 6.5.2 udzbenika.
e) Za isti graf ne može postojati više obuhvatnih stabala minimalne cene.
Ne. Moze postojati vise obuhvatnih stable iste minimalne cene.

Konstrukcija i analiza algoritama **Grupa 2**

1. Koja reč iz skupa S(S={'MAKEDONIJA','AMERIKA','RUSIJA','FRANCUSKA','HRVATSKA','SLOVENIJA'})je najbliža reči *SRBIJA* ako se kao mera rastojanja koristi Levenštajnovo rastojanje? Uz odgovor priložiti vrednost rastojanja do svake niske iz skupa S.

d(MAKEDONIJA,SRBIJA)=7 0 1 2 3 4 5 6 1 1 2 3 4 5 6 2 2 2 3 4 5 5 3 3 3 3 4 5 6 4 4 4 4 4 5 6 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 8 8 8 8 7 8 8 9 9 9 9 8 7 8 10 10 10 10 9 8 7	d(AMERIKA,SRBIJA)=5 0 1 2 3 4 5 6 1 1 2 3 4 5 5 2 2 2 3 4 5 6 3 3 3 3 4 5 6 4 4 3 4 4 5 6 5 5 4 4 4 5 6 6 6 5 5 5 5 6 7 7 6 6 6 6 5	d(RUSIJA, SRBIJA)=3 0 1 2 3 4 5 6 1 1 1 2 3 4 5 2 2 2 2 3 4 5 3 2 3 3 3 4 5 4 3 3 4 3 4 5 5 4 4 4 4 3 4 6 5 5 5 5 4 3
d(FRANCUSKA, SRBIJA)=7 0 1 2 3 4 5 6 1 1 2 3 4 5 6 2 2 1 2 3 4 5 3 3 2 2 3 4 4 4 4 3 3 3 4 5 5 5 4 4 4 4 5 6 6 5 5 5 5 5 7 6 6 6 6 6 6 8 7 7 7 7 7 7 9 8 8 8 8 8 7	d(HRVATSKA, SRBIJA)=6 0 1 2 3 4 5 6 1 1 2 3 4 5 6 2 2 1 2 3 4 5 3 3 2 2 3 4 5 4 4 3 3 3 4 4 5 5 4 4 4 4 5 6 5 5 5 5 5 5 7 6 6 6 6 6 6 8 7 7 7 7 7 6	d(SLOVENIJA, SRBIJA)=5 0 1 2 3 4 5 6 1 0 1 2 3 4 5 2 1 1 2 3 4 5 3 2 2 2 3 4 5 4 3 3 3 3 4 5 5 4 4 4 4 4 5 6 5 5 5 5 5 5 7 6 6 6 5 6 6 8 7 7 7 6 5 6 9 8 8 8 7 6 5

2. Napisati C program koji metodom dinamičkog programiranja određuje minimalni broj simbola koje treba umetnuti u string da bi se dobio palindrom. Na primer, ako je dat string "Ab3bd" od njega se mogu formirati palindromi umetanjem ne manje od dva simbola ("dAb3bAd" ili "Adb3bdA").

Za string s od n simbola formiraćemo matricu $r_{i,j}$ $(i, j = 0..n-1)$ gde je $r_{i,j}$ minimalni broj simbola koje treba umetnuti u segment stringa s od $i-tog$ do $j-tog$ simbola da bi se dobio palindrom. Elementi tražene matrice se mogu računati po formuli:

$$r_{i,j} = \begin{cases} r_{i+1,j-1}, & ako\ je\ s_i = s_j \\ \min(r_{i+1,j}, r_{i,j-1}) + 1, & ako\ je\ s_i \neq s_j \end{cases} \qquad (1)$$

za $i \leq j$. To znači da se popunjava glavna dijagonala ($r_{i,i} = 0$) i elementi iznad nje. Ali pošto se pri njihovom računanju koriste elementi prethodne paralele, to se računanje realizuje po paralelama (idući odozgo na dole) počev od glavne dijagonale. Ako u matrici **R** posmatramo $j-tu$ paralelu:

$r_{0,0}$ ... $r_{0,j}$ ... $r_{0,n-1}$

$r_{1,j+1}$ $r_{1,n-1}$

...

$r_{i,i+j}$ $r_{i,n-1}$

...

$r_{n-1-j,n-1}$

Elementima na j-toj paraleli pristupamo u for petlji

for (i=0; i<=n-1-j; i++) // gde je n=strlen(s)

{ //Izracunavanja: r[i][i+j] }

Izračunavanja na paralelama glavne dijagonale u redosledu od prve do poslednje realizujemo ciklusom koji obuhvata prethodni u kome se *j* menja od 1 do *n-1*:

```
for (j=1; j<=n-1; j++)
  for (i=0; i<=n-1-j; i++)
    { //Izracunavanja: r[i][i+j] }
```

Rezultat je element r[0][n-1]

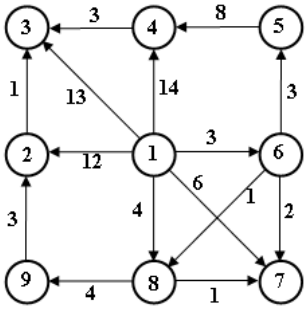
Sada samo treba izračunavati elemente gornjeg trougla matrice korišćenjem formule dobijene modifikovanjem formule (1):

$$r_{i,i+j} = \begin{cases} r_{i+1,i+j-1}, & ako\ je\ s_i = s_{i+j} \\ \min(r_{i+1,i+j}, r_{i,i+j-1}) + 1, & ako\ je\ s_i \neq s_{i+j} \end{cases}$$

```
#include <iostream.h>
#include <string.h>
const int maxN=100;
char s[maxN]="Ab3bd";
int r[maxN][maxN];
int min(int a, int b)
{ return (a<b)? a:b; }

int main()
{ int i,j;
  for (j=1; j<strlen(s); j++)
  for (i=0; i<=strlen(s)-1-j; i++)
  if (s[i]==s[i+j])  r[i][i+j]=r[i+1][i+j-1];
  else r[i][i+j]=min(r[i+1][i+j],r[i][i+j-1])+1;

cout << r[0][strlen(s)-1] << endl;
return 0;
}
```

3. Konstruisati algoritam koji konvertuje binarni zapis broja (niz bita b dužine k) u sam taj broj. Dokazati korektnost napisanog algoritma.
Videti zadatak 1.23 iz udžbenika

4. Dat je na slici usmereni graf G = (V, E). Odrediti najkraće puteve od čvora 1 do svih ostalih čvorova u grafu.



5. Da li su sledeća tvrđenja tačna? Obrazložiti netačna tvrđenja kontraprimerom ili tačnim tvrđenjem.
a) Pohlepni metod uvek vodi globalno optimalnom rešenju.
Ne. Rasitnjavanje 80 dinara na novcanice od po 50, 40, 20, 10 dinara. Pohlepni algoritam daje rasitnjavanje sa tri novcanice 50, 20, 10. Optimalno rasitnjavanje je 80=2*40 din
b) Za stringove iste veličine važi da Hemingovo rastojanje je gornja granica Levenštajnovog rastojanja.
Da
c) Ako se je T razapinjuće stablo minimalne cene u neusmerenom težinskom grafu G i ako se težine grana u grafu povećavaju za jednu istu konstantu c, onda T i dalje ostaje razapinjuće stablo minimalne cene.
Ne. Vidi primer sa vezbi
d) Vremenska složenost Dijkstra algoritma jednaka je vremenskoj složenosti algoritma Svi najkraci putevi.
Ne. Vidi odgovarajuca poglavlja udzbenika
e) Za isti graf može postojati više obuhvatnih stabala iste minimalne cene.
Da.

Konstrukcija i analiza algoritama - jun

1. Algoritmi za nalaženje najkraćih puteva biraju proizvoljan od nekoliko najkraćih puteva. Kako treba izmeniti algoritam da, ako ima više različitih puteva iste dužine, onda se bira onaj sa najmanjim brojem grana? U slučaju da ima više najkraćih puteva sa najmanjim brojem grana, dozvoljeno je izabrati proizvoljan među njima. Pretpostavka je da su dužine grana celi brojevi.

Rešenje: Videti zadatak 6.34 na strani 175 knjige *Algoritmi*, M. Živković

2. Neka je dat skup prirodnih brojeva $\{x_1, x_2, \ldots, x_n\}$ i neka je s njihova suma. Konstruisati algoritam složenosti $O(nS)$ koji razlaže ovaj skup u dva disjunktna podskupa sa jednakim sumama elemenata, ili utvrdjuje da takvo razlaganje nije moguće.d

Rešenje: Videti zadatak 4.18 na strani 78 knjige *Algoritmi*, M. Živković

3. Razmotriti algoritam Bijekcija. Da li je moguće da skup S postane prazan na kraju izvršavanja algoritma?

Rešenje: Videti zadatak 4.2 na strani 77 knjige *Algoritmi*, M. Živković

4. Fibonačijevi brojevi su definisani sledećom diferencnom jednačinom: $F(n) = F(n-1) + F(n-2),\ (n > 2)$
$F(1) = F(2) = 1$
Dokazati da se svaki prirodan broj $n > 2$ može predstaviti u obliku zbira najviše $log_2 n$ različitih Fibonačijevih brojeva.

Rešenje: Videti zadatak 8.12 na strani 220 knjige *Algoritmi*, M. Živković

Konstrukcija i analiza algoritama GRUPA 1

1. 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 (" ) " );
   }
 }
```

2. Konstruisati algoritam koji prirodni broj n preslikava u prirodni broj sa istim ciframa, ali u obrnutom poretku (npr: 1238 se slika u 8321). Dokazati korektnost napisanog algoritma pomoću invarijante petlje.

Rešenje:

```
Algoritam INVERZIJA (n);
input: n;
output: m;
begin
m:=0;
k:=n;
i:=0;

while k>0 do
begin
  m:=m*10+k mod 10;
  k:=k div 10;
   i:=i+1;
end

end.
```

Dokaz izvodimo za brojeve n koji se ne završavaju nulom.

Invarijanta petlje je relacija izmedju promenljivih koja važi nakon svakog izvršenja bloka naredbi u okviru petlje.

Dokažimo primenom matematičke indukcije da je relacija $n = k \cdot 10^i + P(m)$ invarijanta petlje, tj. dokažimo da ova relacija važi pre ulaska u petlju i nakon svakog izvršenja bloka naredbi u okviru petlje. Takođe dokažimo i da je iskaz *"i je broj cifara broja m"* invarijanta petlje.

Baza: Za $i = 0$ (dakle pre prvog izvršenja petlje) imamo:

$$n = k \cdot 10^0 + P(m);$$
$$n = k + P(0);$$
$$n = n + 0 ,$$

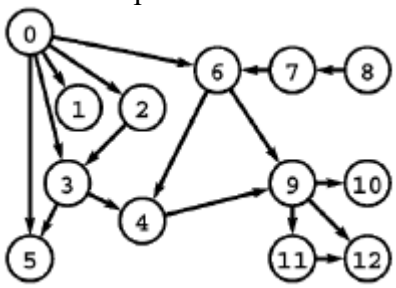
i m ima nula cifara (smatramo da broj 0 ima nula cifara) pa tvrđenje važi.

IH: Pretpostavimo da tvrđenje važi za i (tj. da važi nakon i izvršenja bloka naredbi u okviru petlje) i dokažimo da važi i za $i + 1$. Dakle, na osnovu pretpostavke važi $n = k \cdot 10^i + P(m)$.
Naredna vrednost promenljive k je k div 10, pa treba dokazati
$n = k' \cdot 10^{i'} + P(m')$, $\Leftrightarrow$
$n = (k \text{ div } 10) \cdot 10^{i+1} + P(m \cdot 10 + (k \text{ mod } 10))$, $\Leftrightarrow$

$n = (k \text{ div } 10) \cdot 10^{i+1} + P(m) + (k \text{ mod } 10) \cdot 10^{i}$, $\Leftrightarrow$
$n = ((k \text{ div } 10) \cdot 10 + (k \text{ mod } 10)) \cdot 10^{i} + P(m)$, $\Leftrightarrow$
$n = k \cdot 10^{i} + P(m)$
što je tačno na osnovu induktivne pretpostavke.
Pri prelasku na treći red koristili smo induktivnu pretpostavku da je i broj cifara broja m.
Naravno, i ovo tvrđenje treba dokazati za nove vrednosti m' i i'.
Tvrđenje važi, jer je $i' = i + 1$, $m' = m \cdot 10 + k \text{ mod } 10$.

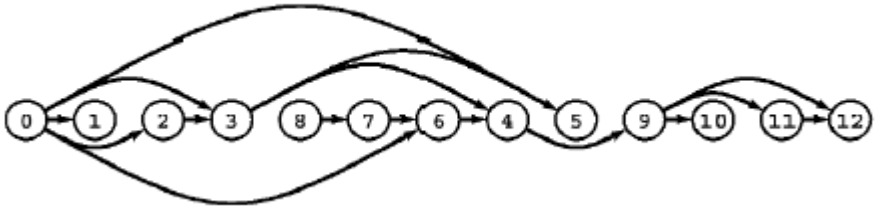
Dokažimo još i da algoritam završava rad. Na početku je $k > 0$ i nakon svakog prolaska kroz petlju k se smanjuje, tako da će u konačnom broju koraka k dostići vrednost 0.
Tada algoritam završava sa radom i za $k = 0$ imamo da je $n = P(m)$, pa je $P(n) = m$
(jer za svako $x \in N$, $P(P(x)) = x$), gde je $P(m)$ funkcija koja prirodan broj n preslikava u prirodan broj sa istim ciframa, ali u obrnutom poretku).
Napomena: Ako bismo želeli da obuhvatimo i brojeve koji se završavaju jednom ili sa više nula, algoritam bi bio isti, ali bi dokaz morao da se izmeni. Umesto pomenute invarijante petlje, trebalo bi uzeti relaciju
$n = k \cdot 10^{i} + P(m) \cdot 10^{l}$, gde je l broj nula kojima se završava broj n.

3. Pronaći topološki redosled čvorova za graf sa slike. Da li postoji neki Hamiltonov put u grafu?



Resenje

Topoloski redosled



Nema Hamiltonovog puta, jer, na primer, cvorovi 1 i 2 nisu putno povezani.

4. Dat je niz prirodnih brojeva a sa n članova. Konstruisati (u programskom jeziku C) algoritam složenosti O(n) koji određuje broj podnizova niza a (sastavljenih od uzastopnih elemenata niza a), čija je suma paran broj?

5. a) Dat je neusmereni graf G = (V, E) sa skupom čvorova V = {1,2,3,4,5,6,7} i skupom grana E = { (1,2), (1,3), (1,4), (1,5), (1,7), (2,3), (2,4), (2,5), (2,7), (3,4), (3,5), (3,6), (4,5), (5,6), (5,7), (6,7) } (svaka grana navedena je samo jednom). Dokazati da postoji klika veličine 5 svođenjem na problem pokrivač grana.
b) Dokazati da je problem klika NP kompletan.

Konstrukcija i analiza algoritama septembar 2014.

1. Konstruisati algoritam za konverziju binarnog zapisa nekog broja u heksadekadni i dokazati korektnost algoritma.

Rešenje:
Pogledati (sličan) zadatak 3 iz materijala na adresi
http://poincare.matf.bg.ac.rs/~jelenagr/AIDA/cas1.pdf

2. Ako je dat algoritam za množenje dve n * n donje trougaone matrice čije vreme izvrsavanja je O(T(n)), dokazati da postoji algoritam za množenje dve proizvoljne n *n matrice čije vreme izvršavanja je $O(T(n)+n^2)$. (Može se pretpostaviti da je T(cn)=O(T(n)) za svaku konstantu c)

Rešenje:
Neka su A i B dve proizvoljne kvadratne matrice reda n .
Svaka od njih se moze predstaviti kao zbir po jedne gornje i po donje trougaone matrice (T_A, B_A, T_B, B_B, redom):
$A= T_A + B_A$
$B= T_B + B_B$
Dalje je: $AB=(T_A + B_A) (T_B + B_B)=T_AT_B + B_AB_B + B_AT_B + T_AB_B$
Ako se upotrebi algoritam iz formulacije zadataka moguće je izračunati proizvod:

$$\begin{vmatrix} B_A & 0 \\ T_A & B_A \end{vmatrix} \times \begin{vmatrix} B_B & 0 \\ T_B & B_B \end{vmatrix} = \begin{vmatrix} B_AB_B & 0 \\ B_AT_B + T_AB_B & B_AB_B \end{vmatrix}$$

Kao rezultat dobija se matrica koja sadrzi blokove: B_AB_B , T_AT_B , $B_AT_B + T_AB_B$
Ovi blokovi učestvuju u izračunavanju proizvoda A × B.
Na taj način se problem izračunavanja proizvoda dve proizvoljne matrice svodi na problem izracunavanja proizvoda kvadratne donje trougaone matrice kvadratnom gornjom trougaonom matricom.
Ukupno vreme izvršavanja : $O(T(2n)+n^2)=O(T(n)+n^2)$

3. Dat je usmereni aciklički graf *G = (V, E)*. Konstruisati algoritam linearne vremenske složenosti za nalaženje najdužeg usmerenog prostog puta u *G* (bilo kog među njima ako ih ima više).

Rešenje:
3.1. Neka se topološkim sortiranjem G dobija redosled čvorova $v_1, v_2, \ldots, v_n$.
3.2. Ako znamo najduži usmereni put čiji je poslednji čvor v_i za $i < m \leq n$, onda se najduži među putevima koji se završavaju u čvoru v_m dobija produžavanjem puta do nekog čvora vi, i < m, granom (vi, vm) - ako ona postoji.

Dakle, prilikom prelaska na naredni čvor po topološkoj numeraciji, treba proveriti sve grane koje vode u njega.
Tako se posle O(|E| + |V |) koraka dobijaju dužine najdužih puteva do čvorova v_i, i = 1, 2, . . . , n.

3.3. Od tih dužina bira se najveća.

3.4. Da bi se rekonstruisao najduži put, pri dodavanju novog čvora v_m treba pamtiti ne samo dužinu najdužeg puta do v_m, nego i poslednju granu na najdužem (ili jednom od najdužih) putu.

Složenost algoritma je O(|E| + |V |).

4. Dat je neusmereni graf *G = (V, E)* sa skupom čvorova *V = {1,2,3,4}* i skupom grana *E ={(1,2), (1,3), (1,4), (2,3), (3,4)}* (svaka grana navedena je samo jednom). Utvrditi da li postoji pokrivač grana veličine 2 svođenjem na problem dominirajući skup.

Rešenje:
Pogledati konstruktivan dokaz teoreme 11.5 iz udžbenika.

5. Fibonačijevi brojevi su definisani sledećom diferencnom jednačinom: F(n) = F(n-1) + F(n-2), (n > 2)
F(1) = F(2) = 1
Dokazati da se svaki prirodan broj n > 2 može predstaviti u obliku zbira najviše $log_2\, n$ različitih Fibonačijevih brojeva.

Rešenje:
Na primer: 18 = 13 + 5 (2 sabirka, dok je $\log_2 18 > 2$)

```
Algoritam Gramziva_Predstava (n)
input:  n /* n>2 */
output: reprezentacija broja n kao sume Fibonačijevihbrojeva*/
 {
    f1=1;  f2=1;
    while (f2 <= n) {d=f1+f2; f1=f2; f2=d;}   /* naci k: f(k)<=n < f(k+1) */
    while (n>0)  {  if (n-f1>=0) { stampaj f1; n=n-f1;} d=f2-f1; f2=f1; f1=d;}
 }
```

Dokaz principom matematičke indukcije:

baza: n=3 F(1)=F(2)=1, F(3)=2, F(4)=3 => n=F(4) && $1<\log_2 3$

induktivna hipoteza: Za sve brojeve k manje od n važi

(IH) prirodan broj k > 2 može se predstaviti u obliku zbira najviše $\log_2 k-1$ Fibonacijevih brojeva

Dokaz:

Dakle,postoji k: F(k)<=n<F(k+1) =>F(k) > n/2

/* supr. pretp. F(k)<=n/2. Kako je F(k-1) < F(k) za k>2, onda je F(k-1) < n/2

Odatle, F(k+1)=F(k)+F(k-1) < n/2 + n/2 =n što je kontradikcija sa n< F(k+1) */

Po (IH) za broj (n-F(k))potrebno je <= $\log_2$ (n-F(k)) razl. Fibonač. brojeva

Kako je n-F(k)< F(k), /* jer iz gore dokaznog svojstva F(k) > n/2 => 2*F(k)>n => F(k) > n-F(k)) */

onda F(k) ne učestvuje u reprezentaciji broja n-F(k) Fibonačijevim brojevima, te se za broj n= F(k) + (n-F(k)) koristi <= 1 + $\log_2$ (n-F(k)) razl. Fibonač. brojeva.

Kako $1+ \log_2 (n-F(k))=\log_2 2 + \log_2(n-F(k))=\log_2(2n - 2 F(k)) = \log_2(n + (n- 2F(k))< \log_2 n$ (jer n-2F(k) <0 po gore dokazanom)