Topološko sortiranje

Ulaz

G = (V, E) -- usmereni aciklički graf

Izlaz

Podatak o rednom broju rb za svaki čvor v takav da za svaki čvor w gde postoji veza v → w važi v.rb < w.rb (***)

Rešenje

Numerišemo čvor koji ima stepen 0 (koji uvek postoji u acikličkom grafu) sa rb = 1. Ostatak numerišemo { 2, 3, ... } što znamo po induktivnoj hipotezi (IH: umemo da numerišemo na zahtevani način (***) čvorove svih usmerenih acikličkih grafova sa manje od *n* čvorova)

Algoritam

Inicijalizujemo `ulazniStepen` za svaki čvor (DFS).

```
Grb = 0;  //globalni redni broj
lista = {};

// Dodajemo sve čvorove koji imaju ulazni stepen nula u listu

for (i ∈ 1 .. n) {
   if (v[i].ulazniStepen == 0) {
      stavi v[i] u listu;
   }
}

// Uzimamo jedan po jedan čvor, i smanjujemo
// ulazniStepen njegovim sledbenicima

while (lista nije prazna) {
   v = uzmi prvi čvor liste;

   Grb = Grb + 1;
   v.rb = Grb;

   for (sve grane (v, w)) {
      w.ulazniStepen = w.ulazniStepen - 1;
      if (w.ulazniStepen == 0) {
         stavi w u listu;
      }
   }
}
```

Složenost

DFS: O(|V| + |E|)

for petlja: jedna iteracija za svaki cvor

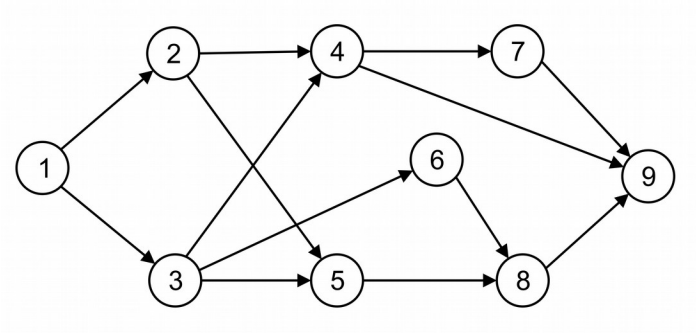
while petlja: po jednom za svaku granu

•Ukupno: O(|V| + |E|)

Zadaci

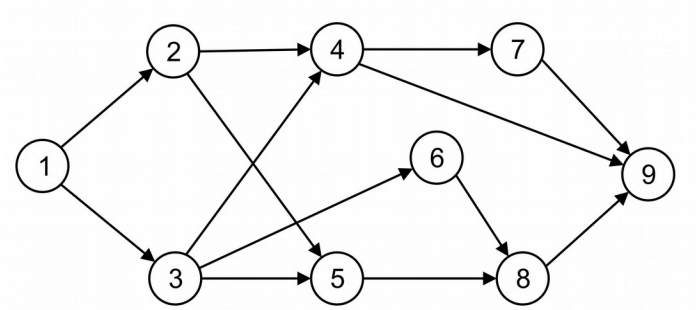
Zadatak 1

Za graf sa slike naći topološko sortiranje



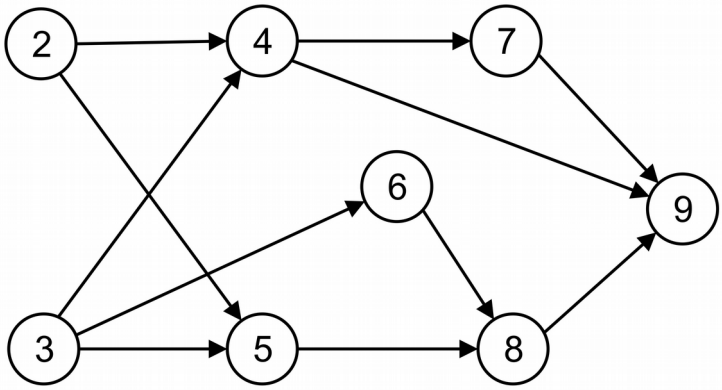
IDEJA: Algoritam topološkog sortiranja sastoji se u sledećem postupku. Pronađe se čvor bez ulaznih grana, iz sekvencijalnog reda čvorova u skupu V, i zapiše se čvor u niz. Potom se iz grafa ukloni taj čvor i obrišu sve grane koje polaze iz tog čvora. Za novodobijeni graf se ponovo nađe čvor bez ulaznih grana i zapiše u niz, zatim se iz grafa ukloni taj čvor i pobrišu sve grane koje polaze iz tog čvora itd.

Čvor bez ulaznih grana: 1 (postoji u usmerenom acikličkom grafu, zašto?)

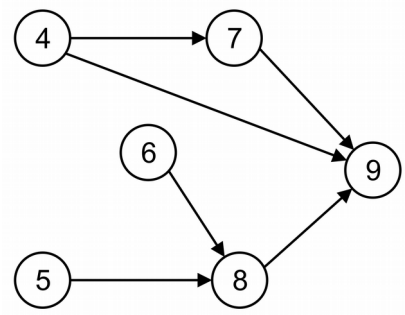


1								

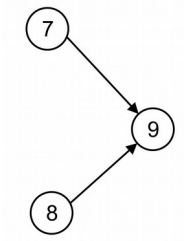
Čvor bez ulaznih grana u novom grafu: 2, 3



Čvor bez ulaznih grana u novom grafu: 4, 5, 6

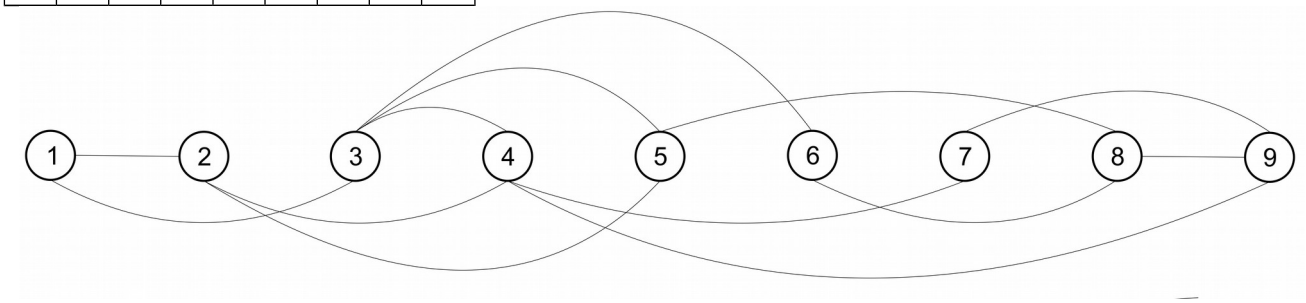


1	2	3	4	5	6			



1	2	3	4	5	6	7	8	

1	2	3	4	5	6	7	8	9



Da li su povezani 1,2? Jesu.
Da li su povezani 2,3? Nisu.

Zadatak 2

Naći topološko sortiranje za usmeren graf G=(V,E), gde V={a,b,c,d,e},
E= { (a, b), (a, c), (d, b), (e, c), (a, e), (a, d) }

Zadatak 3

Zadat je aciklički usmeren graf *G=(V,E).* Konstruisati algoritam linearne složenosti koji utvrđuje da li u G postoji neki prost put koji sadrži sve čvorove grafa.

REŠENJE:

Put je prost, ako se svaki čvor pojavljuje u njemu samo jednom.

Hamiltonov put je prosti put u kom se svaki čvor grafa pojavljuje tačno jednom.

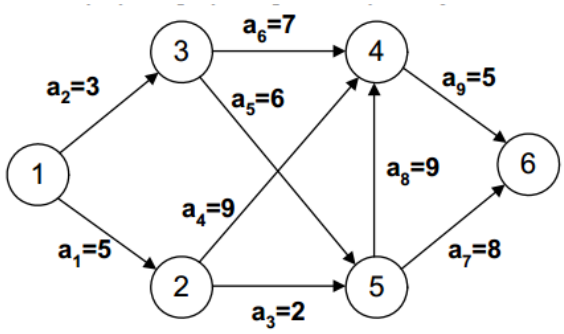
U ovom zadataku zahtev je da se konstruiše algoritam za pronalazak Hamiltonovog puta u grafu G.

Ideja je da se konstruiše topološki redosled čvorova grafa G, a potom se proveri da li postoji put tako što se proveri da li susedni čvorovi u topološkom redosledu jesu povezani. Ukoliko su povezani, onda Hamiltonov put čine topološki uređeni čvorovi.

Algoritam topološkog sortiranja acikličkog usmerenog grafa G je složenosti $O(|E| + |V|)$, tj. linearne složenosti kod grafova.

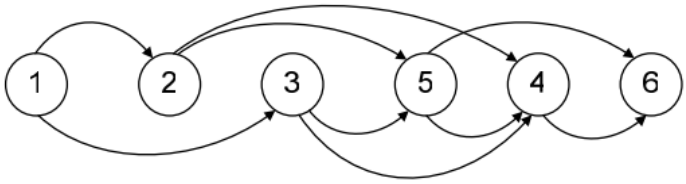
S druge strane, ako postoji Hamiltonov put, onda topološko sortiranje mora da dovede upravo do onog redosleda čvorova kojim se oni nižu u putu!!!

Zadatak 4 Pronaći topološki redosled čvorova za graf sa slike. Da li postoji neki Hamiltonov put u grafu?



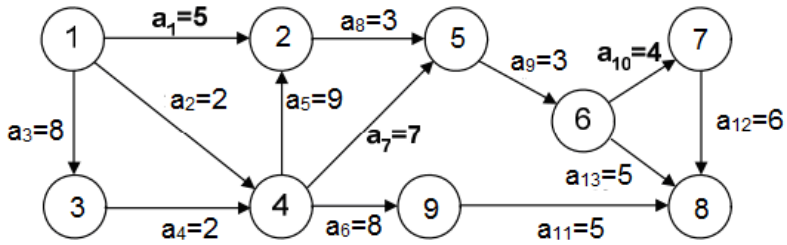
Rešenje:

Topološki redosled čvorova grafa je

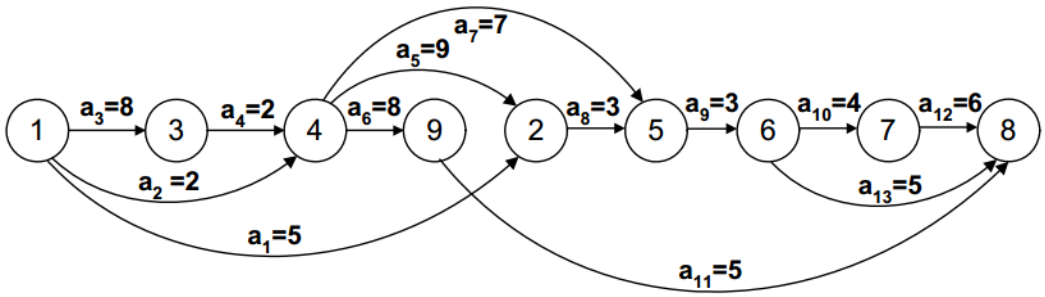


U ovom grafu topološki redosled čvorova nam ne daje Hamiltonov put, jer nisu svaka dva uzastopna čvora u topološkom redosledu povezani.

Zadatak 5 Pronaći topološki redosled čvorova za graf sa slike.



Rešenje:



Zadatak 6

Modifikovati algoritam za topološko sortiranje tako da, u slučaju da je ulazni graf ipak cikličan graf, onda algoritam nalazi bar jedan ciklus. Složenost algoritma treba da bude i dalje linearna. (Dakle, izvršiti modifikaciju bez povećanja složenosti).

Zadatak 7

Da bi student mogao izaći na ispite, mora da obezbedi potpise profesora da je pratio njihova predavanja. Ali, neki profesori daju potpis za svoj predmet SAMO AKO je student prethodno obezbedio potpise drugih kolega koji drže predavanja iz predmeta koji su važni za razumevanje predmeta koje on drži.

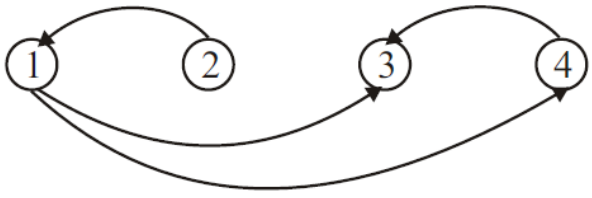
ULAZ: U prvoj liniji standardnog ulaza dat je broj profesora N od kojih student traži poptpise ($1<=N<=100$). U narednih N redova opisane su potrebne informacije. U svakoj i-toj liniji, prvi broj K predstavlja broj profesora čiji su potpisi preduslov za potpis (i-1). profesora. Narednih K brojeva u liniji sačinjavaju spisak tih preduslovnih profesora.

IZLAZ: Ako se ne mogu prikupiti potpisi svih profesora, onda na standardni izlaz ispisati poruku **NE.** Ako se mogu prikupiti svi potpisi, onda u prvom redu standardnog ulaza ispisati DA, a u sledećih N redova ispisati redosled prikupljanja potpisa.

(Objašnjenje: linija 1 2 ukazuje da je potpis profesora 1 preduslov za profesora 2, linija 0 ukazuje da za profesora 2 nema preduslova, linija 2 1 4 ukazuje da za profesora 3 postoje 2 preduslova: potpisi profesora 1 4, linija 1 1 ukazuje da za profesor 4 je uslov 1 potpis i to od profesora 1).

Na primer:

in.txt	out.txt
4 1 2 0 2 1 4 1 1	DA 2 1 4 3



Problem se može predstaviti usmerenim grafom u kom ako postoji grana od čvora A do čvora B, to onda znači da je potpis profesora A uslov za potpis profesora B.

Čvor koji nema dolazne grane predstavlja profesora koji svoj potpis daje bezuslovno. Ako to I-ti čvor, onda I-ta kolona ima sve vrednosti jedanke nula. Dakle, I-ti profesor daje potpis bezuslovno, te je potrebno ukloniti i sve grane koje vode od njega ka drugim profesorima (tj.postaviti nule u I-tu vrstu matrice). Dok god ima čvorova čiji je ulazni stepen 0 (tj. dok god ima profesora koji svoj potpis daju bezuslovno), ponavljati ovaj postupak.

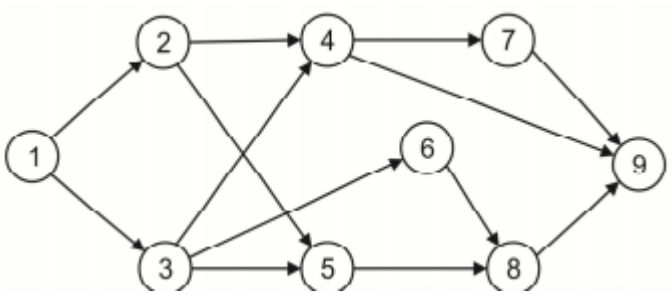
Ako je uklonjeno svih N čvorova, onda je student dobio sve potpise. Inače, ne može se nastaviti uklanjanje i to ukazuje da postoji ciklus. Tada ispisujemo poruku NE (jer se ne mogu dobiti svi potpisi).

```
#include <iostream>
using namespace std;
int a[101][101];

bool KolonaNula(int a[][101], int n, int j)
{
  for (int i=1; i<=n; i++)
        if (a[i][j]) return false;
  return true;
}

int main()
{
  int n, i,j, k, p, b[101];
  bool Dalje, mark[101];

//ucitavanje grafa
cin >>n;
for(i=1; i<=n; i++)
{
     cin >> k; mark[i]=false;
     for (j=1; j<=k; j++)
     {
         cin >> p;
        a[p][i]=1;
    }
}

Dalje=true; p=0;
while (Dalje) // dok ima cvorova za uklanjanje
{
   Dalje=false;
   for (int i=1; i<=n; i++)
       if (!mark[i] && KolonaNula(a,n,i))
       {
         b[++p]=i;
         mark[i]=true;
         for (j=1; j<=n; j++) a[i][j]=0;// i-ti profesor daje potpise
       Dalje=true;
     }
}

if (p==n)
{
   cout << "DA" << endl;
    for (i=1; i<=n; i++) cout << b[i] << endl;
}
else cout << "NE";

return 0;
}
```

Pitalice - topSort, Dijkstra, MCST

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.
b) Topološko sortiranje radi efikasnije od BFS algoritma na neusmerenim grafovima koji imaju neparan broj čvorova čiji ulazni stepen je nula.
c) Vremenska složenost Dijkstra algoritma jednaka je vremenskoj složenosti algoritma topološkog sortiranja.
d) Za isti graf ne može postojati više obuhvatnih stabala minimalne cene.
e) Ako je S razapinjuće stablo minimalne cene u neusmerenom težinskom grafu G i ako se težine grana u grafu povećavaju za jednu istu konstantu x, onda S i dalje ostaje razapinjuće stablo minimalne cene.
f) Vremenska složenost Dijkstra algoritma jednaka je vremenskoj složenosti Floyd-Warshal algoritma .

Resenja



a) Ne. Primer je graf
Topoloski uredjeni cvorovi 1,2,3,4,5,6,7,8,9 ne obrazuju Hamiltonov put, jer cvorovi 2 i 3 nisu povezani.

b) Ne.
Topološko sortiranje je algoritma koji se primenjuje na usmerene grafove, a sporno je i postojanje neusmerenih grafova koji imaju neparan broj čvorova čiji ulazni stepen je nula.

c) Ne. Vremenska složenost Dijkstra algoritma je O(|V|*|V|) u najgorem slucaju, a inace O(|E|+ |V| log |V|). Asimptotska vremenska složenot algoritma topološkog sortiranja je O(|V|+|E|).
d) Ne. Za jedan graf moze postojati vise razlicitih MCST iste minimalne cene.
e)
f) Ne. Vremenska složenost Dijkstra algoritma je O(|V|*|V|) u najgorem slucaju, a inace O(|E|+ |V| log |V|). Asimptotska vremenska složenost algoritma Floyd-Warshal je O(|V|*|V|*|V|).