ALGORITMI I STRUKTURE PODATAKA

STUDIJSKI PROGRAMI:

SOFTVERSKO INŽENJERSTVO, RAČUNARSKA TEHNIKA, INFORMATIKA I MATEMATIKA

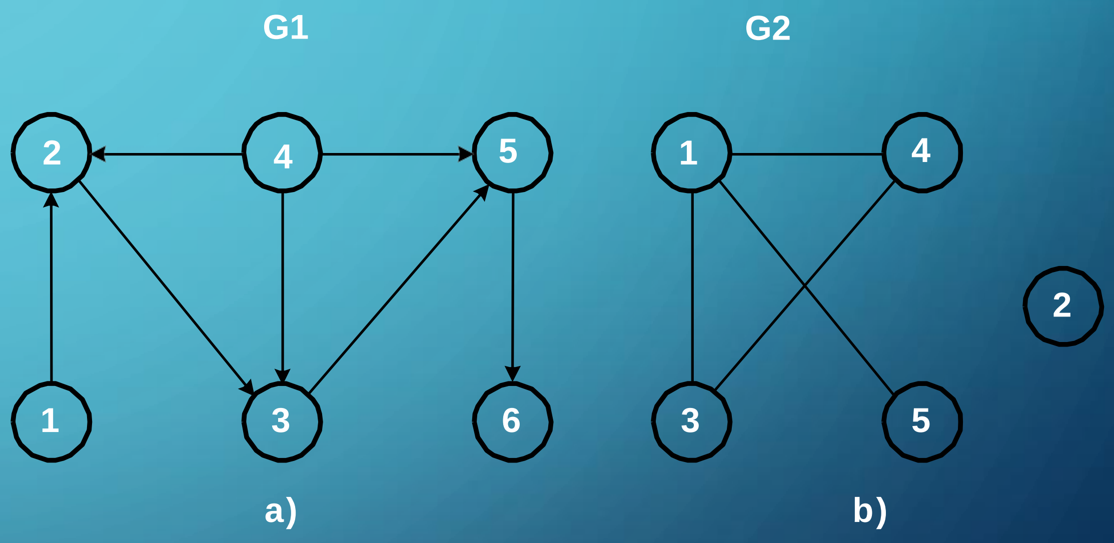
NASTAVNIK: DOC. DR ULFETA MAROVAC, UMAROVAC@NP.AC.RS

GRAFOVI

ALGORITMI I STRUKTURE PODATAKA

GRAFOVI

- Modeliranje proizvoljnih nelinearnih relacija
- Graf G je par skupova (V, E)
- V (čvorovi) - konačan neprazan skup
- E (grane) - binarne relacije između čvorova
- grana (u, v) incidentna na čvorovima



TERMINOLOGIJA

- Usmereni, neusmereni i mešoviti grafovi
- Susednost čvorova
- Ulazni i izlazni stepen čvora
- Petlje i paralelne grane
- Prosti graf, multigraf, hipergraf, podgraf
- Težinski graf
- Put, prost put, dostižnost

TERMINOLOGIJA

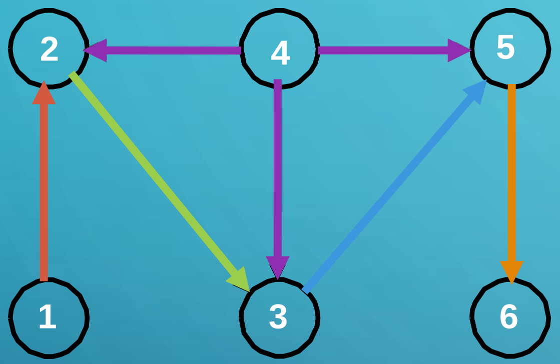
- Ciklus, ciklični i aciklični grafovi
- Kompletni, gusti i retki grafovi
- Bipartitni graf
- Povezani graf, povezane komponente
- Slobodno stablo
  - acikličan, povezan, neusmeren graf

MATRIČNA REPREZENTACIJA

- Matrica susednosti A
  - $a[i, j] = 1$ ako je ako $(i, j) \in E$
  - $a[i, j] = 0$ ako je ako $(i, j) \notin E$
- $a[i, j] = w(i, j)$ za težinske grafove

PRIMER

G1



a)

- Matrica susednosti

$$
\begin{array}{cl}
A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix}
\end{array}
$$

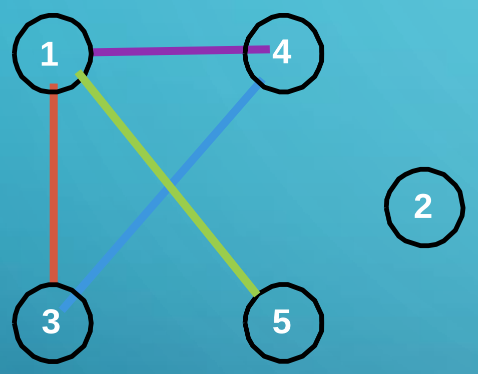
Čvorovi 1,2,3 i 5 imaju po jednu vezu

Čvor 4 ima tri veze prema čvorvima 2,3 i 5

Čvor 6 nema ni jednu vezu

PRIMER

G2



b)

- Matrica susednosti

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}$$

Sve veze su neusmerene prema tome (one su obostrane)

Čvorov 1 ima veze sa čvorovima 3,4 i 5

Čvor 2 nema ni jednu vezu

Čvor 3 ima veze prema čvorvima 1 i 4

Čvor 4 ima veze prema čvorvima 1 i 3

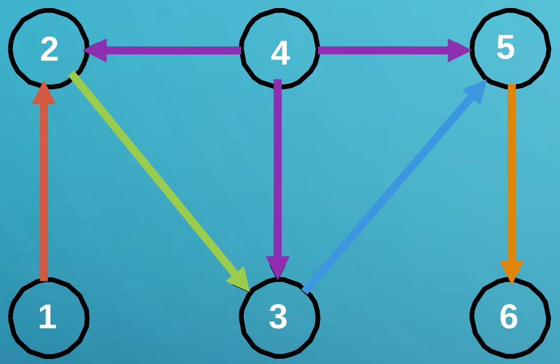
Čvor 5 ima jednu vezu prema čvoru 1

ULANČANA REPREZENTACIJA

- Liste susednosti
- vektor zaglavlja
- ulančane liste suseda
- element liste odgovara grani

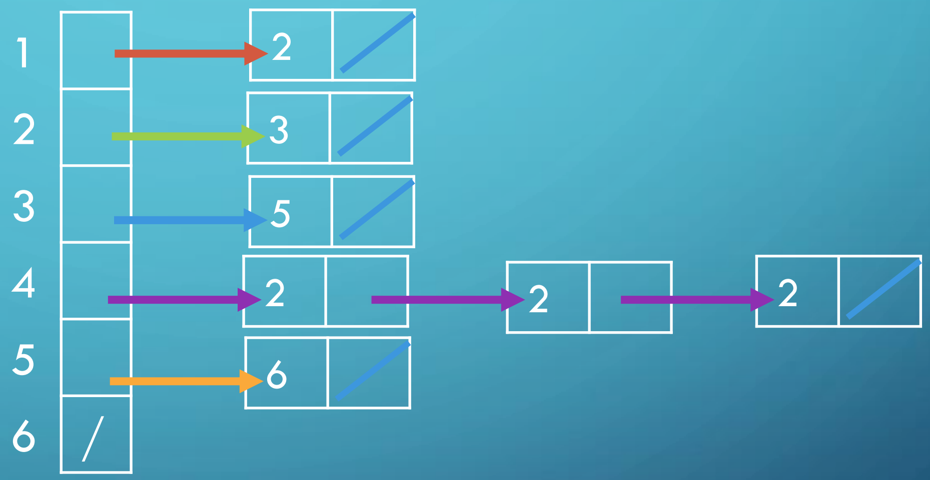
PRIMER

G1



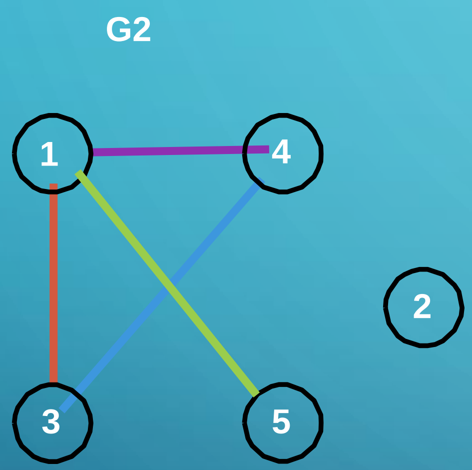
a)

AL

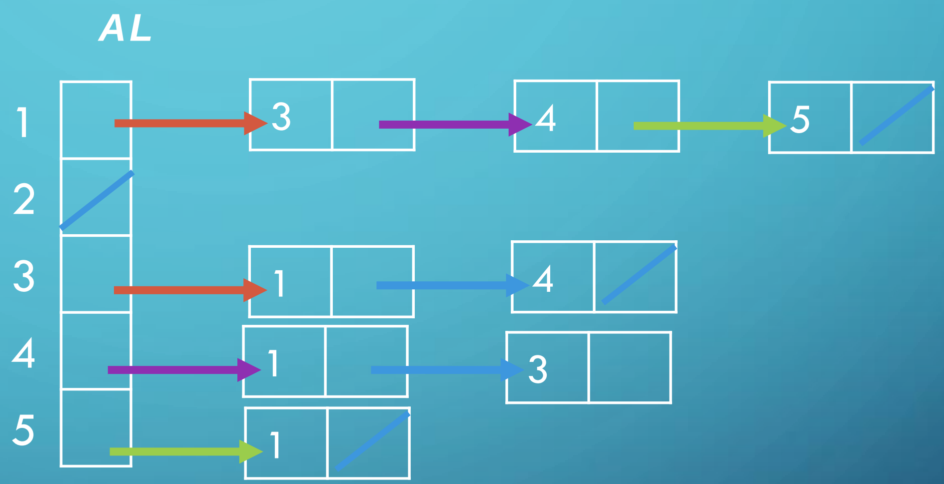


a)

PRIMER



b)



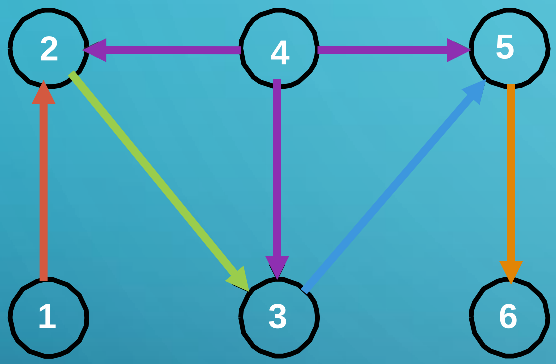
a)

ULANČANA REPREZENTACIJA

- Inverzne liste susednosti
  - ✓ lista za čvor sadrži sve čvorove kojima je sused
  - ✓ pogodno za izračunavanje ulaznog stepena
- Multiliste
  - ✓ element koji predstavlja granu ulančan u dve liste
  - ✓ identifikacija oba čvora i dva pokazivača

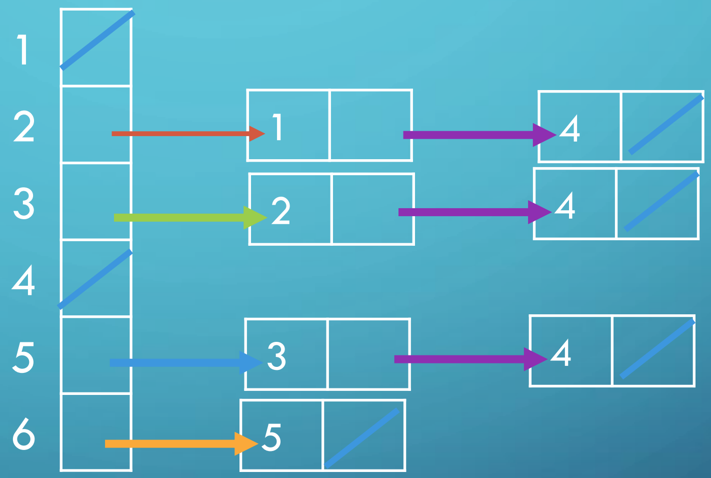
PRIMER

G1



a)

AL



a)

PREDSTAVLJANJE GRAFOVA

- Prostorna složenost

	matrica	liste (netežinski)	liste (težinski)
usmereni	n^2	$n + 2e$	$n + 3e$
neusmereni	$n(n + 1)/2$	$n + 4e$	$n + 6e$

- Matrična reprezentacija pogodnija
  - za dinamičku promenu broja grana
  - za direktan pristup grani
- Ulančana reprezentacija pogodnija za
  - dinamičku promenu broja čvorova
  - za odredjivanje suseda

OBILAZAK GRAFA

- Svi čvorovi se posete samo jednom u nekom linearnom poretku
- Poredak zavisi od izbora početnog čvora
- Ako se ne posete svi zbog nedostižnosti, nastavlja se sa nekim od neposećenih
- Na čvor se može naići više puta, ali samo prvi put se poseti
- Osnovni algoritmi zasnovani na susednosti:
  - obilazak po širini (BFS)
  - obilazak po dubini (DFS)

OBILAZAK GRAFA PO ŠIRINI

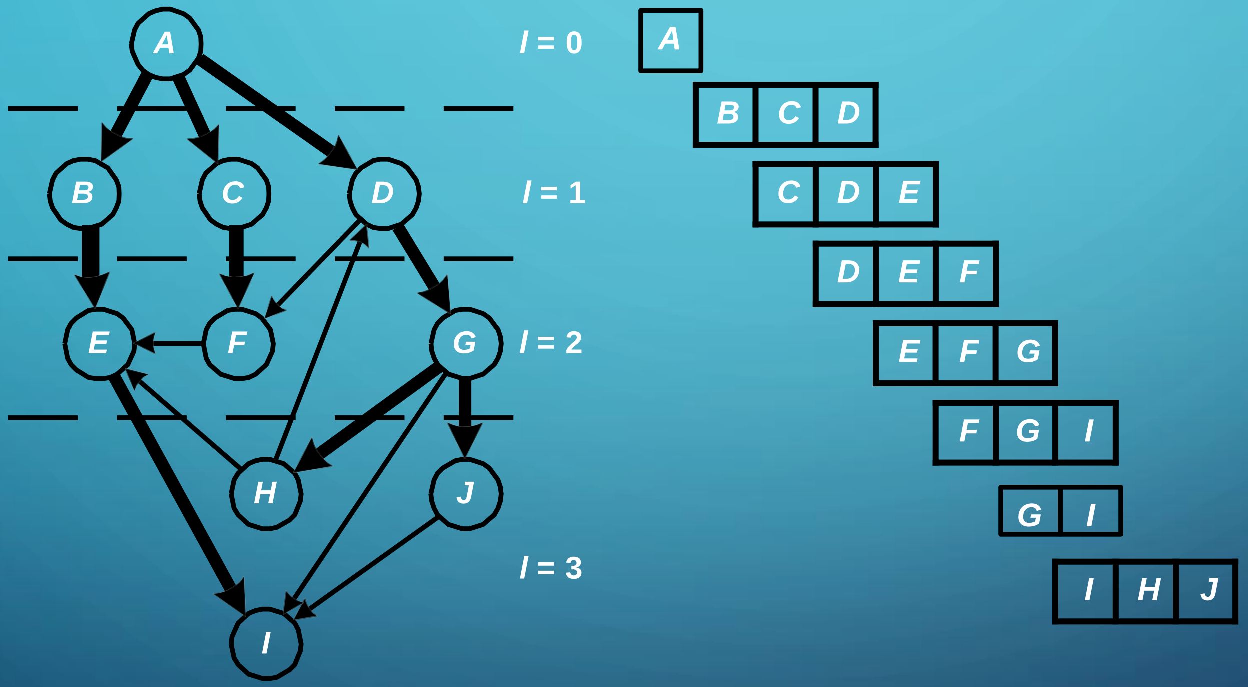
- Strategija algoritma
  - ✓ poseti početni čvor
  - ✓ poseti njegove susede
  - ✓ poseti njihove neposećene susede istim redom, ...
- Posete u “talasima”, po nivoima iste udaljenosti
- Koristi se neprioritetni red za čekanje i vektor posećenosti
- Vremenska složenost
  - ✓ $O(n^2)$ za matričnu reprezentaciju
  - ✓ $O(\max(n, e))$ za ulančanu reprezentaciju

OBILAZAK GRAFA PO ŠIRINI

```
BFS(G, v)
for i = 1 to n do
    visit [i] = false
end_for
visit [v] = true
INSERT(Q, v)
while (not QUEUE-EMPTY(Q)) do
    v = DELETE(Q)
    for { u : (v, u) ∈ E} do
        if (not visit [u]) then
            visit [u] = true
            INSERT(Q, u)
        end_if
    end_for
end_while
```

- Algoritam polazi od zadatog čvora v
- Postavlja vektor posecenig cvorova za svih n na netacno
- Svaki poseceni cvor se postavlja u neprioritetni red Q (Prvo se spusta cvor v)
- Sve dok ima elemenata u redu Q skida se element iz reda i posecuju se svi njegovi neposeceni susedi . Svaki neposeceni sused se spusta u red Q i menja se vrednost vektora v[u] =true;
- Kad ispraznimo red to znaci da su svi cvorovi poseceni odnosno da poslednji cvorovi koji su bilu u redu nisu imali neposecene susede

OBILAZAK GRAFA PO ŠIRINI



A

ABCD

ABCDE

ABCDEF

ABCDEFG

ABCDEFGI

ABCDEFGI

ABCDEFGIHJ

OBILAZAK GRAFA PO DUBINI

- Strategija algoritma
  - ✓ poseti početni čvor
  - ✓ poseti jednog njegovog suseda
  - ✓ poseti jednog neposećenog suseda prethodnog
  - ✓ ako ga nema, vraća se do poslednjeg prethodnika koji ima neposećenog suseda

- Slično preorderu kod stabla

- Vremenska složenost
  - ✓ $O(n^2)$ za matričnu reprezentaciju
  - ✓ $O(\max(n, e))$ za ulančanu reprezentaciju

OBILAZAK GRAFA PO DUBINI

- Rekurzivna realizacija

```
DFS(G, v)
for i = 1 to n do
      visit [i] = false
end_for
DFS-VISIT(v)
```

```
DFS-VISIT(v)
visit [v] = true
for { u, (v, u) ∈ E } do
      if (not visit [u]) then
            DFS-VISIT(u)
      end_if
end_for
```

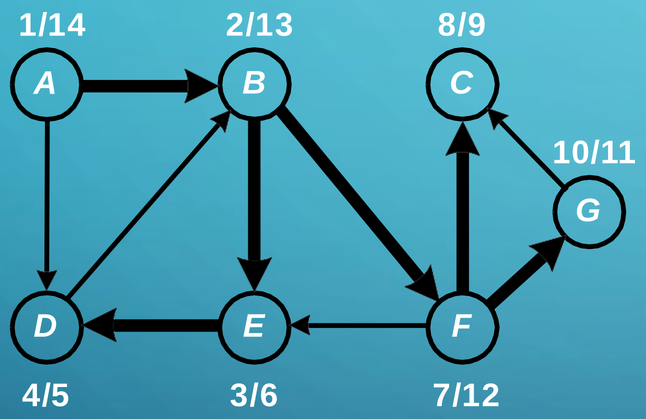
Postavi se vektor posecenosti v na netacno za sve čvorove

Polazimo od jednog čvora posetimo ga zatim za sve njegove sinove pozivamo istu metodu

Može i iterativna realizacija korišćenjem steka

OBILAZAK GRAFA PO DUBINI

- Primer



ABEDFCG

- Uz svaki čvor stoje dve vremenske oznake trenutak kada je prvi put cvor obidjen i vreme zavrsetka obilaska svih suseda cvora

PRIMENE OBILASKA GRAFA

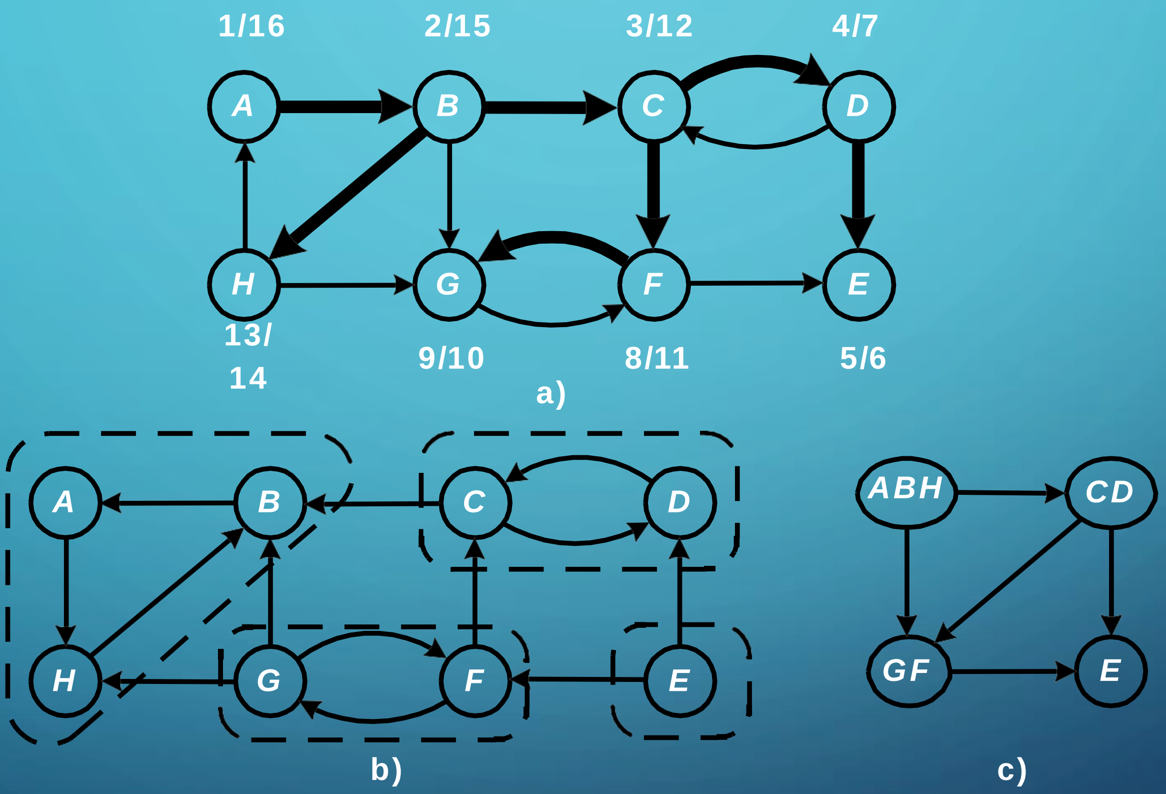
- Određivanje najkraćeg rastojanja
  - između dva čvora (*i* i *j*) u netežinskom grafu
    - ✓ rastojanje do *j* jednako broju nivoa *l*[*j*]
    - ✓ BFS se startuje od polaznog čvora *i* (*l*[*i*] = 0)
    - ✓ pri poseti nekog čvora *u* preko drugog čvora *v*
      - postavi se njegov nivo na *l*[*u*] = *l*[*v*] + 1
    - ✓ kad se poseti čvor *j* rastojanje je nađeno kao minimalni broj grana od čvora *i*
- Provera cikličnosti
    - ✓ postojanje poprečnih ili povratnih grana

PRIMENE OBILASKA GRAFA

- Određivanje povezanih komponenenata u neusmerenom grafu

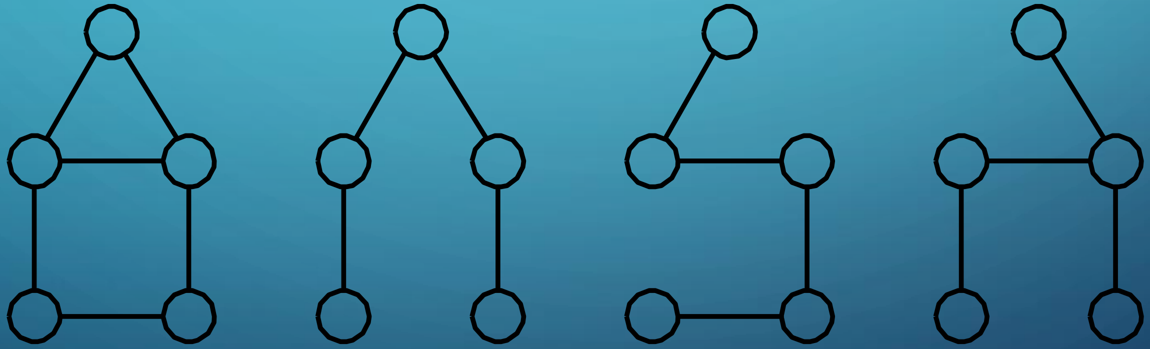
```
CONN-COMP(G)
n_cc = 0
for i = 1 to n do
      visit[i] = false
end_for
for i = 1 to n do
      if (not (visit[i]) then
            n_cc = n_cc + 1
            DFS-VISIT(i) → CC(n_cc)
      end_if
end_for
```

PRIMENE OBILAZAKA GRAFA



OBUHVATNA STABLA

- Obuhvatno stablo $ST = (U, E')$
  - neusmerenog, povezanog grafa $G = (V, E)$
    - ✓ sadrži sve čvorove grafa, $U = V$
    - ✓ sadrži određen broj grana, $E' \subseteq E$, tako da su svi čvorovi povezani, ali da nema ciklusa

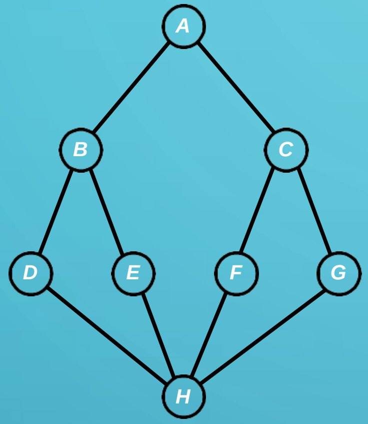


a) b) c) d)

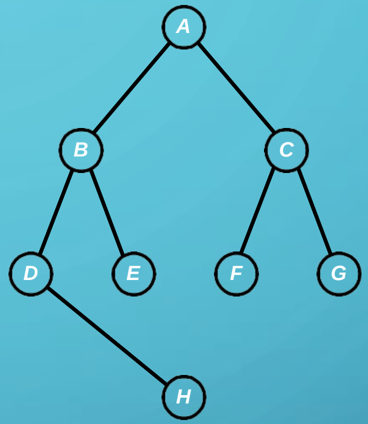
OBUHVATNA STABLA

- Obuhvatno stablo – slobodno stablo
- Za nepovezan graf – obuhvatna šuma $ST_i = (V_i, E_i)$
- Generiše se pomoću algoritama obilaska BFS ili DFS
  - ✓ na početku E' prazan skup
  - ✓ kada se doĎe do neposećenog čvora u, dolazna grana (v, u) se uključi u stablo ($E' = E' + \{(v, u)\}$ u **then** delu algoritma
- Broj grana u obuhvatnom stablu - n - 1
- Primene

OBUHVATNA STABLA

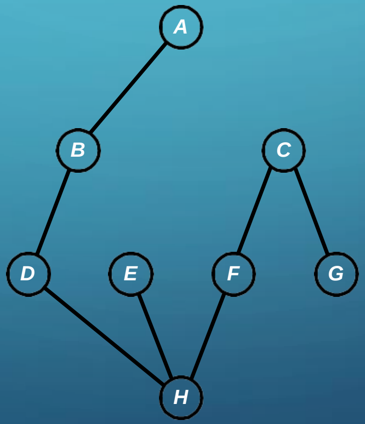


a)



b)

BFS stablo

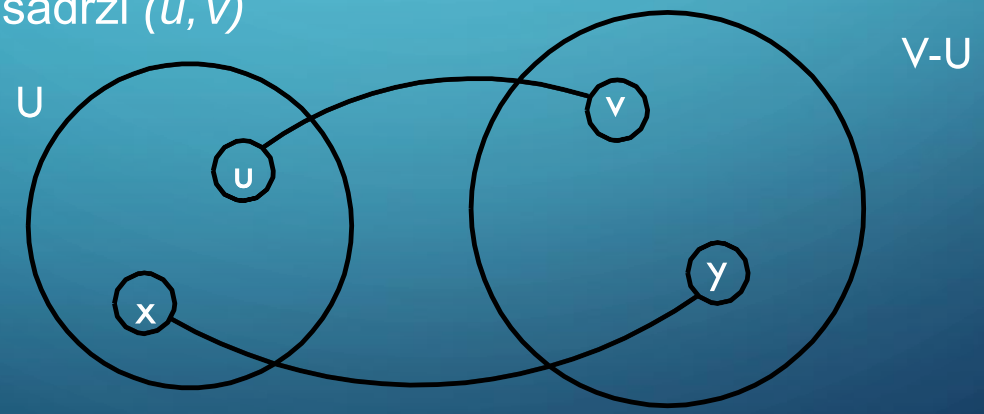


c)

DFS stablo

MINIMALNA OBUHVATNA STABLA

- Cena obuhvatnog stabla – $\sum w(u, v)$ $\quad (u, v) \in E'$
- MST – obuhvatno stablo čija je cena **minimalna**
- Ako je $U \subseteq V$ i ako je (u, v) grana najmanje težine takva da je $u \in U$ i $v \in (V - U)$,
  - onda postoji MST koje sadrži (u, v)



PRIM-OV ALGORITAM

- Inkrementalno gradi MST počevši od polaznog čvora dodajući po jednu granu i jedan čvor
- Bira granu najmanje težine od onih koje povezuju čvorove koji su već uključeni u MST i čvorove koji još nisu uključeni

<u>PRIM(*G*, *s*)</u>
$U = \{s\}$
$E' = \varnothing$
while $(U \neq V)$ **do**
    find $(u, v) \Rightarrow \min \{w(u, v) : (u \in U) \text{ and } (v \in (V - U))\}$
    $U = U + \{v\}$
    $E' = E' + \{(u, v)\}$
end_while
$MST = (U, E')$

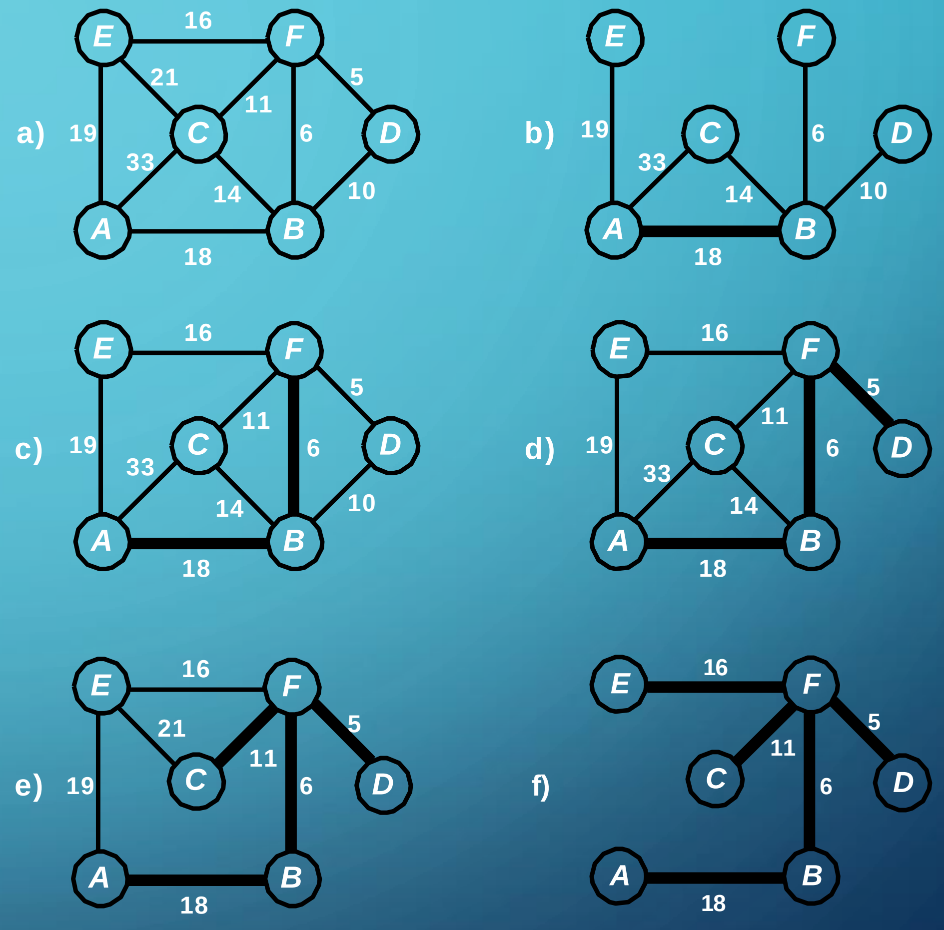
PRIM-OV ALGORITAM

a) Polazimo od cvora A I dodajemo ga skupu U

b) Najkraci put od A do bilo kog drugo cvora je preko B I iznosi 14 zato se B dodaje minimalnom obilaznom stablu (U) kao I odgovarajuca veza izmedju A I B.

c) Najkraca putanja od cvorova iz skupu U (A,B) ka drugim cvorovima (izvan U) je preko veze (B,F)=6 =>F dodajemo U, (B,F)=6 dodajemo E'

d) Najkraca putanja od od skupa U ka ostalim cvorovima je preko veze (F,D)=5, pa se skupu U dodaje F,….



KRUSKAL-OV ALGORITAM

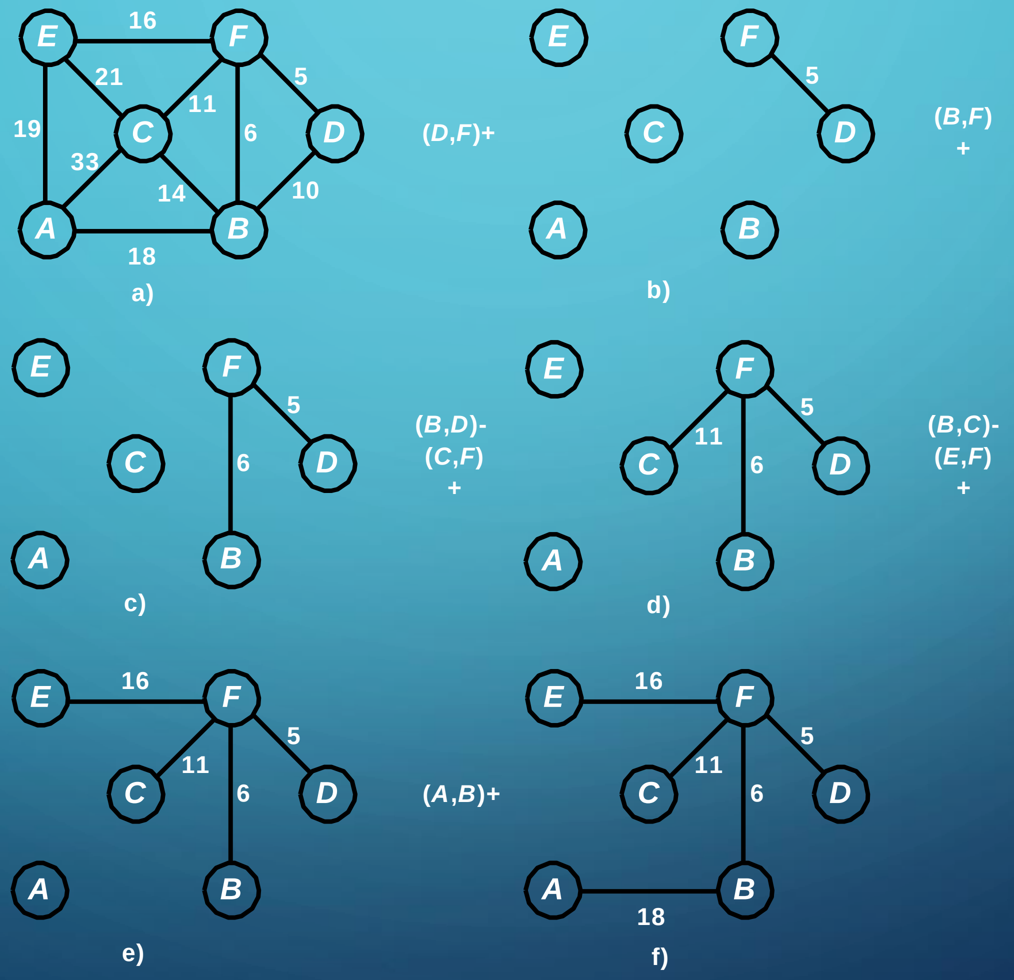
- Još jedan algoritam za konstrukciju obuhvatnog stablo čija je cena minimalna
- Polazi od čvorova koji svaki za sebe predstavlja celinu o dodaje redom grane sa najmanjom težinom ukoliko one povezuju do sada nepovezane celine

```
KRUSKAL(G)
E' = ∅
for each (u, v) ∈ E do
    PQ-INSERT(PQ, w(u, v))
end_for
num = 0
while (num < n - 1) do
    w(u,v) = PQ-MIN-DELETE(PQ)
    if ((u ∈ T_i) and (v ∈ T_j) and (i ≠ j)) then
        E' = E' + {(u, v)}
        T_k = T_i + T_j
        num = num + 1
    end_if
end_while
MST = (V, E')
```

KRUSKAL-OV ALGORITAM

- Polazi od šume nepovezanih čvorova - podstabala
- Inkrementalno dodaje po jednu granu
- Bira granu koja:
  - ✓ ima najmanju težinu od preostalih, neuključenih
  - ✓ ne zatvara ciklus
- Završava kada se uključi n - 1 grana
- Vremenska složenost - $O(e \log e)$

KRUSKAL-OV ALGORITAM



ODREĐIVANJE DOSTIŽNOSTI

- Veoma često postoji potreba da se utvrdi da li postoji putanja kojom su povezana dva čvora to jeste da li je jedan čvor dotižan iz drugog čvora
- Matrica puta daje informaciju da li postoji putanja između dva čvora

ODREĐIVANJE DOSTIŽNOSTI

- Matrica puta (dostižnosti) P
  - ✓ $p[i, j] = 1$ ako postoji put od i do j
  - ✓ $p[i, j] = 0$ ako ne postoji put od i do j
- $a[i, k]a[k, j] = 1$ ako postoji put od i do j preko k
- $a_{ij}^{(2)} = \sum a[i, k]a[k, j]$ - broj puteva dužine 2 od i do j
- $A(2)=A^2$
- $a_{ij}^{(3)} = \sum a[i, k]^{(2)}a[k, j]$ - broj puteva dužine 3 od i do j
- $A(3)=A^3$, ... , $A^{(m)}$, ...

ODREĐIVANJE DOSTIŽNOSTI

Algoritam za određivanje matrice puta

- Naći $B^{(n)} = A + A^{(2)} + \ldots + A^{(n)} = A + A^2 + \ldots + A^n$
- Članovi matrice puta P se određuju kao:
  - ✓ $p[i, j] = 1$ ako je $b[i, j]^{(n)} > 0$
  - ✓ $p[i, j] = 0$ ako je $b[i, j]^{(n)} = 0$
- Optimizacije:
  - ✓ u prostoru - matrica bitova
  - ✓ u vremenu – **or** i **and** umesto ***** i **+**
- Vremenska složenost - $O(n^4)$

WARSHALL-OV ALGORITAM

```
WARSHALL(A)
P = A
for k = 1 to n do
      for  i = 1 to n do
            for  j = 1 to n do
                  p[i, j] = p[i, j] or (p[i, k] and p[k, j])
            end_for
      end_for
end_for
```

```
if (p[i, k] = 1) then
      for j = 1 to n do
            p[i, j] = p[i, j] or p[k, j]
      end_for
end_if
```

- Ovaj algoritam iterativno pronalazi da li se dva cvora mogu povezati preko nekog drugog cvora.
- Prolazi od matrice susednosti A
- Promazi kroz sve cvorove grafa k I dodaje jedinicu u matricu puta P za svaka dva cvora i i j koja su do sada bila ne povezana a za koje je postojala veza sa cvorom k

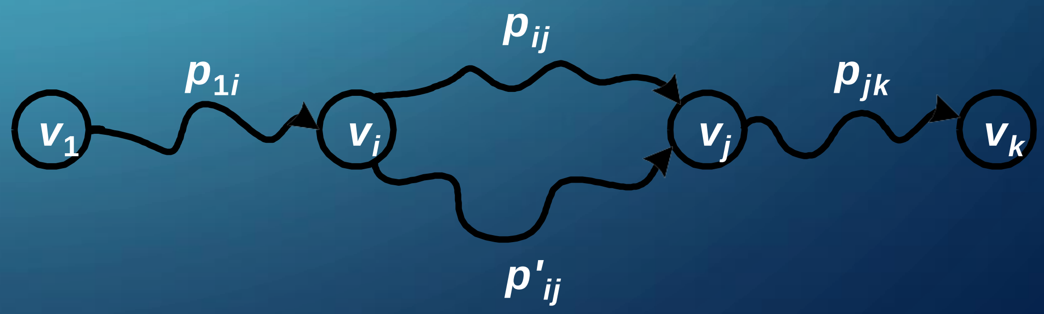
p[i,k]=1 i p[j,k]=1 => p[i,j]=1

WARSHALL-OV ALGORITAM

- Složenost:
  - vremenska - $O(n^3)$
  - prostorna - $O(n^2)$
- U neusmerenom grafu može lakše:
  - naći povezane komponente
  - prostorna - $O(n^2)$
  - $p[i, j] = p[j, i] = 1$ za sve parove*i* i *j* u istoj povezanoj komponenti
  - $p[i, j] = p[j, i] = 0$ za sve parove *i* i *j* u različitim povezanim komponentama
  - vremenska složenost - $O(n^2)$

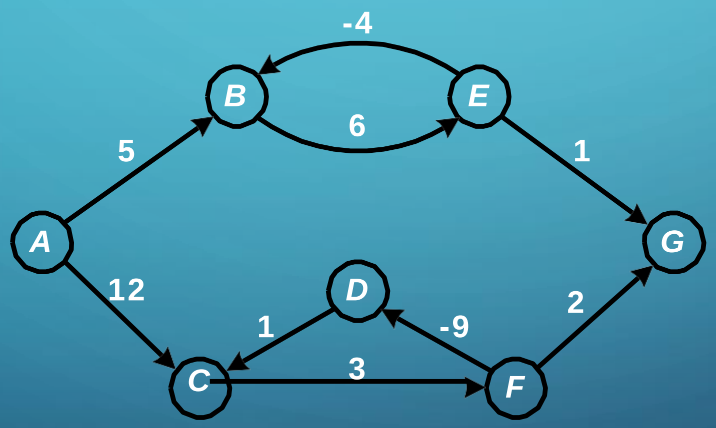
NAJKRAĆA RASTOJANJA

- $G = (V, E)$ – usmereni težinski graf
  - ✓ $w(i, j)$ - težina grane od i do j
  - ✓ $w(p_{1k}) = \sum w(i, j)$ – dužina puta od 1 do k
- Najkraće rastojanje između čvorova i i j je:
  - ✓ $d(i, j) = \min\{w(p)\}$ po svim putevima od i do j
  - ✓ ∞ ako j nije dostižan iz i



NAJKRAĆA RASTOJANJA

- Grane mogu imati i negativne težine
- Nisu dozvoljeni ciklusi sa negativnom težinom



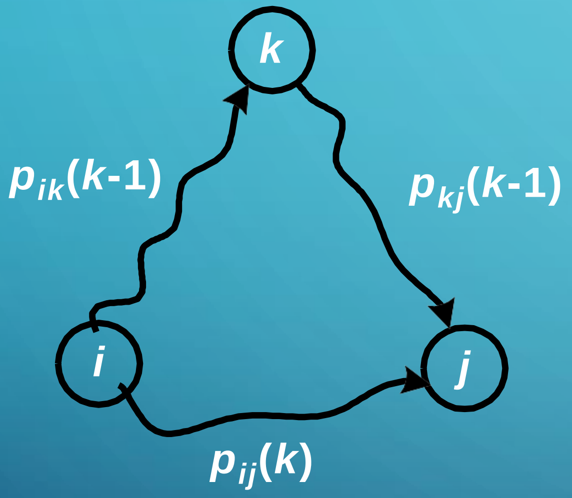
FLOYD-OV ALGORITAM

- Najkraća rastojanja i putevi između svih parova čvorova

- Ulaz – matrica težina W
  - $w[i, j] = 0$ ako je $i = j$
  - $w[i, j] = w(i,j)$ ako je $i \neq j$ i $(i, j) \in E$
  - $w[i, j] = \infty$ ako je $i \neq j$ i $(i, j) \notin E$
- Izlaz – matrica najkraćih rastojanja D i matrica prethodnika T

- Inicijalizacija matrice prethodnika T
  - ✓ $t[i, j] = 0$ ako je $i = j$ ili $w[i, j] = \infty$
  - ✓ $t[i, j] = i$ ako je $i \neq j$ ili $w[i, j] < \infty$

FLOYD-OV ALGORITAM



Princip relaksacije

```
FLOYD(W)
D = W
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if (d[i, j] > d[i, k] + d[k, j]) then
                t[i, j] = t[k, j]
                d[i, j] = d[i, k] + d[k, j]
            end_if
        end_for
    end_for
end_for
```

$\sim O(n^3)$

FLOYD-OV ALGORITAM

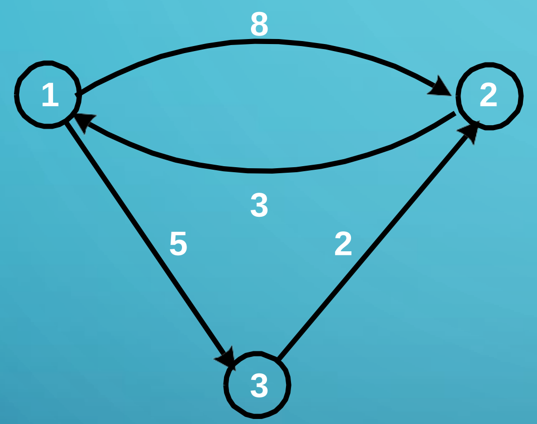
- Matrica težine grana prikazuje težine grana kojima su čvorovi direktno povezani dok za čvorove koji nemaju direktnu vezu težina je beskonačna.
- Pomoću Floydovog algorima za svaki čvor k ispituje se da li je put od čvora i do čvora j (za sve i i j) kraći preko čvora k nego od njegove trenutne vrednosti. Pri tome se u posebnoj matrici matrici prethodnika ukoliko je w[i,k]+w[j,k]<w[i,j] postavlja indeks cvora k što znači da se od i dolazi do j tako što se krene prvo preko čvora k. Dalju putanju će pokazati vrednost iz tabele prethodnika na poziciji (i,k),...

FLOYD-OV ALGORITAM

- Rekonstrukcija puta

```
PATH(i, j)
if (i = j) then
    PRINT(i)
    return
else
    if (t[i, j] = 0) then
        PRINT(Nema puta izmeĎu i i j)
    else
        PATH(i, t[i, j])
        PRINT(j)
    end_if
end_if
```

FLOYD-OV ALGORITAM



$$D^{(1)} = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \quad T^{(1)} = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix} \quad T^{(2)} = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix}$$

$$D^{(0)} = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{bmatrix} \quad T^{(0)} = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} \quad D^{(3)} = \begin{bmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix} \quad T^{(3)} = \begin{bmatrix} 0 & 3 & 1 \\ 2 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix}$$

FLOYD-OV ALGORITAM

- Primena – određivanje središta grafa
- $ecc(v) = \max \{d[i, v] : i \in V\}$ - ekscentričnost čvora
- $\min \{ecc(v), v \in V\}$ - središte
- OdreĐivanje središta:
  - naći matricu najkraćih rastojanja D
  - naći maksimume kolona
  - naći minimume od ovih maksimuma
  - čvor koji odgovara minimumu - središte

DIJKSTRA-IN ALGORITAM

- Najkraća rastojanja i putevi između jednog čvora (1) i svih ostalih
- Ne dozvoljava negativne težine grana
- Ulaz – matrica težina grana *W*
- Izlaz – vektor najkraćih rastojanja *D* i vektor prethodnika *T*
- Inicijalizacija:
  - ✓ vektora *D* sa prvom vrstom matrice W
  - ✓ vektora *T* ($t[i, j] = 1$ ako je $w[1, i] < \infty$)

DIJKSTRA-IN ALGORITAM

```
DIJKSTRA(W)
S = {1}
for i = 2 to n do
    d [i] = w [1, i]
    if (w [1, i] ≠ ∞) then
        t [i] = 1
    else
        t [i] = 0
    end_if
end_for
```

```
for k = 1 to n - 1 do
    find min {d [i] : i ∈ (V – S) }
    S = S + {i}
    for each j ∈ (V – S) do
        if (d [i] + w [i, j] < d [j]) then
            d [j] = d [i] + w [i,j]
            t [j] = i
        end_if
    end_for
end_for
```

DIJKSTRA-IN ALGORITAM

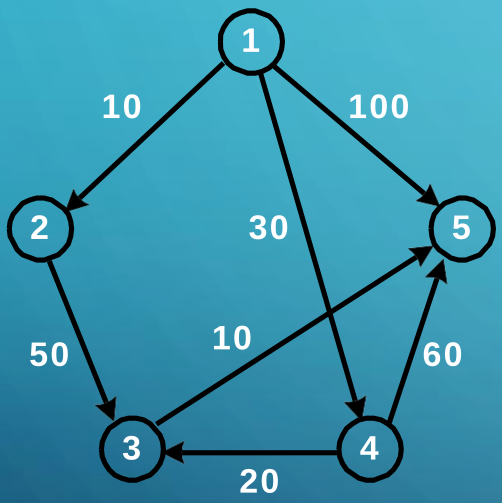
- Polazimo od čvora (1) i matrice težine grana
- U svakom koraku biramo čvor i iz grafa koji ima put najmanje težine do čvora (1) dodajemo ga u skup U i osvežavamo matricu težina i prethodnika sa novim vrednostima ukoliko se preko ovog čvora u smanjuje težina puta od (1) do nekog drugog čvora j
- Dobijeni graf je graf sa putebima minimalne dužine od čvora (1) do svih njemu dostižinih čvorova

DIJKSTRA-IN ALGORITAM

- Rekonstrukcija puta od čvora 1 do *i*

```
PATH(i)
if (i = 1) then
    PRINT(1)
    return
else
    if (t [i ] = 0) then
        PRINT(Nema puta do i )
    else
        PATH(t [i ])
        PRINT(i)
    end_if
end_if
```

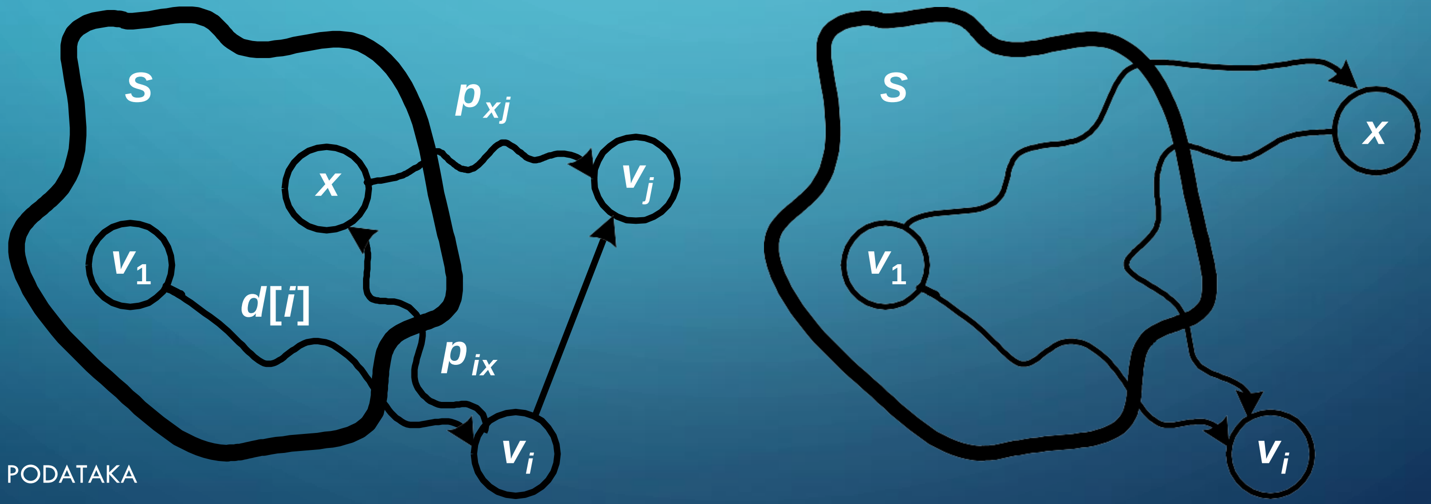
DIJKSTRA-IN ALGORITAM



		D[i]				T[i]			
		2	3	4	5	2	3	4	5
1	-	10	∞	30	100	1	0	1	1
1,2	2	10	60	30	100	1	2	1	1
1,2,4	4	10	50	30	90	1	4	1	4
1,2,4,3	3	10	50	30	60	1	4	1	3
1,2,4,3,5	5	10	50	30	60	1	4	1	3

DIJKSTRA-IN ALGORITAM

- Dokaz korektnosti:
  - korektnost postupka relaksacije
  - korektnost određivanja najkraćeg rastojanja



DIJKSTRA-IN ALGORITAM

- Vremenska složenost:
  - za matricu susednosti - $O(n^2)$
  - za liste susednosti - $O(e + n \log n)$
- Algoritam se može iskoristiti i za određivanje:
  - najkraćih rastojanja između svih parova čvorova
  - najkraćeg rastojanja između dva čvora
- Bellman-Ford algoritam – ako su dozvoljene i grane sa negativnom težinom

TEST PITANJA

1. Dajte primer jednog usmerenog, neusmerenog, težinskog, cikličanog I acikicnog grafa.
2. Napraviti matricu susednosti za jedan od grafova koji ste naveli u prethodnom primeru.
3. Napraviti listu susednosti I inverznu liste susednosti za prethodno prikazani primer.
4. Obiđite predstavljeni graf po širini I po dubini
5. Napravit obuhvatno stablo (ne obavezno minimalno) od povezanog neusmerenog cikličnog grafa
6. Napraviti obuhvatno stablo čija je cena **minimalna pomoću Primovog algoritma**
7. Na kom principu radi Kruskal-ov algoritam. Sta je njegovo početno stanje?
8. Napraviti matrice puta za čvorove iz prethodnog primera pomoću Warshalovog algoritma
9. Odrediti matricu težina i matricu prethodnika koristeci Floydov algoritam za prethodni primer.
10. Primeniti Dijkstrin algoritam za pronalaženje grafika sa putevima minimalne tezine od izbranog čvora do ostalih dostižnih čvorova u grafu.