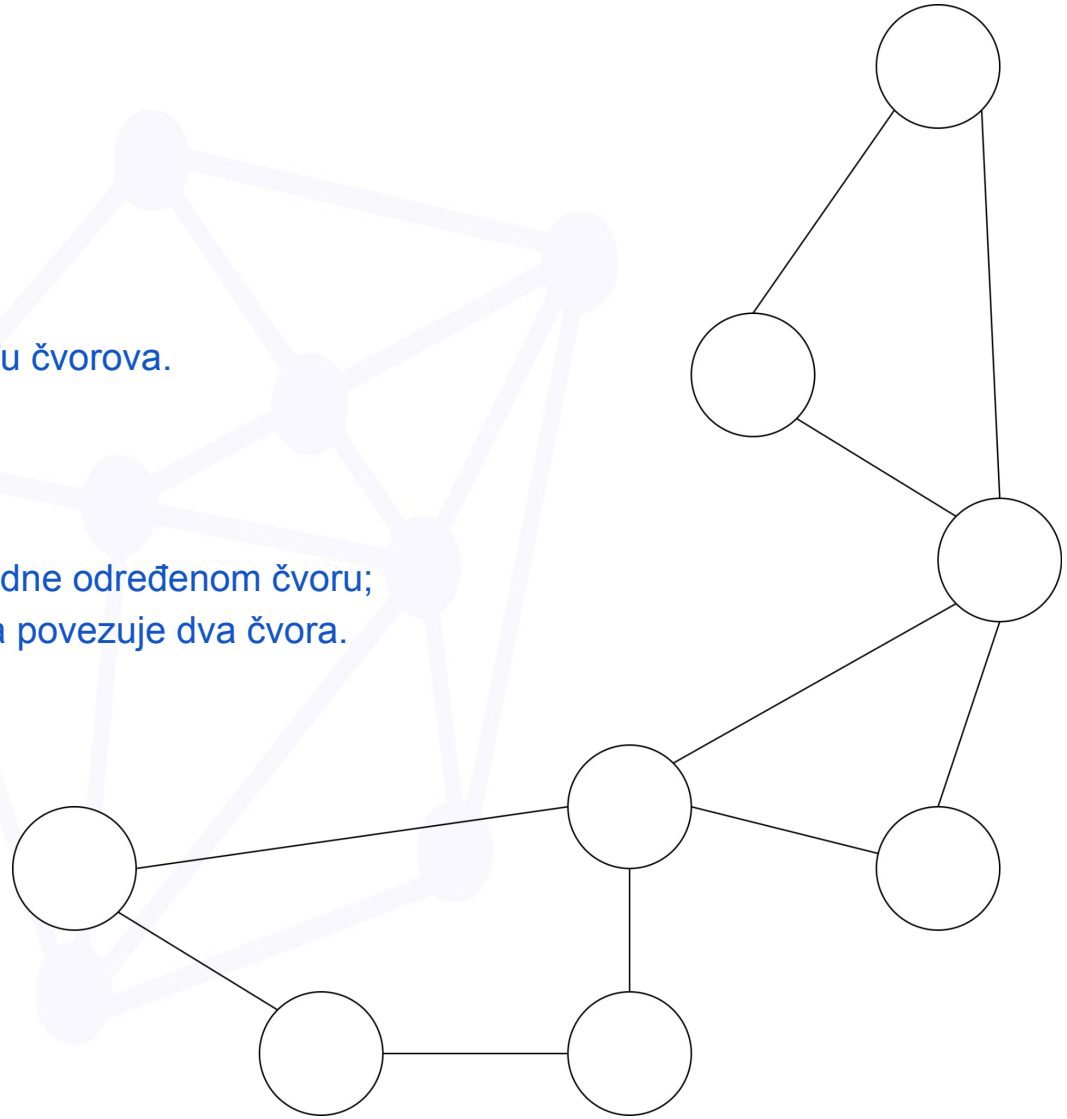
Teorija Algoritama

Grafovi

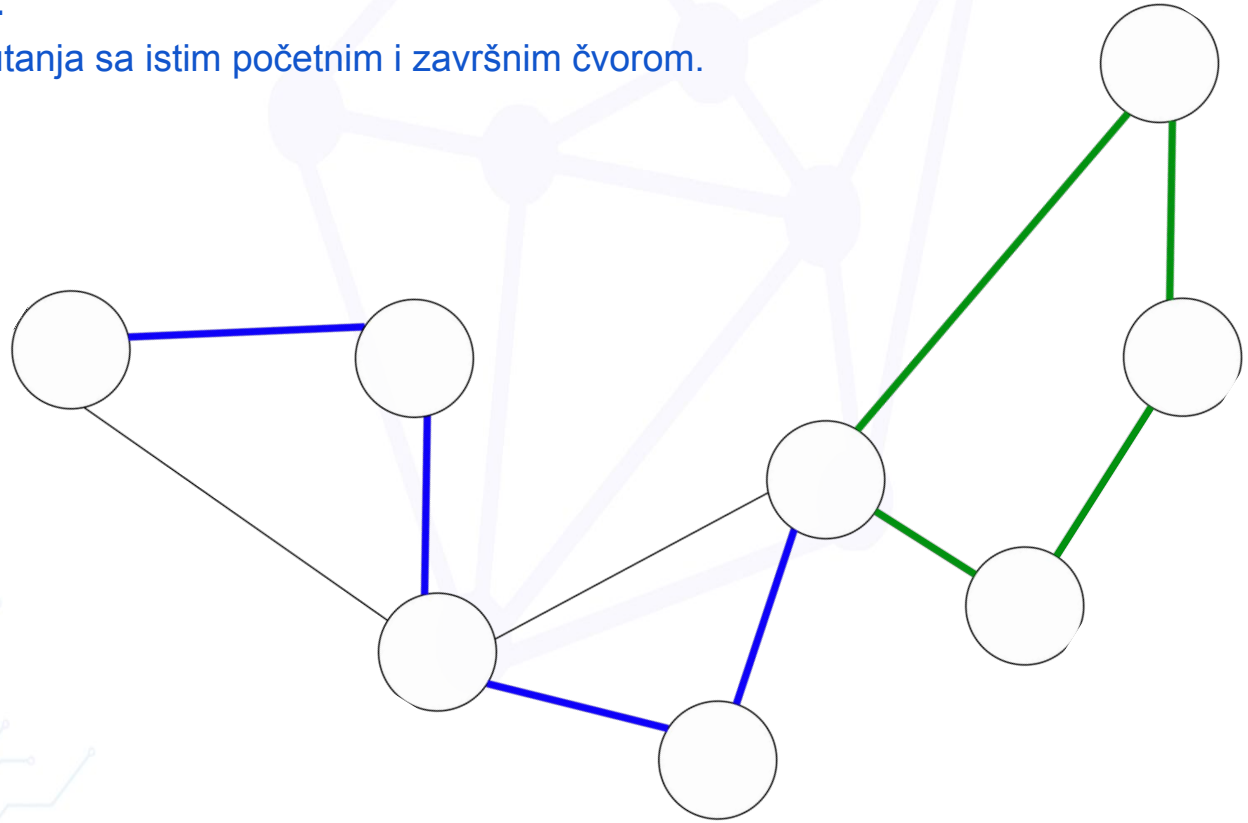
Definicija

- Apstraktni tip podataka.
- Uređeni par $G = (V, E)$:
  - V - skup čvorova;
  - E - skup grana/veza između čvorova.
- Osnovne operacije:
  - Iteracija kroz čvorove;
  - Iteracija kroz grane;
  - Iteracija kroz čvorove susedne određenom čvoru;
  - Upit da li postoji grana koja povezuje dva čvora.



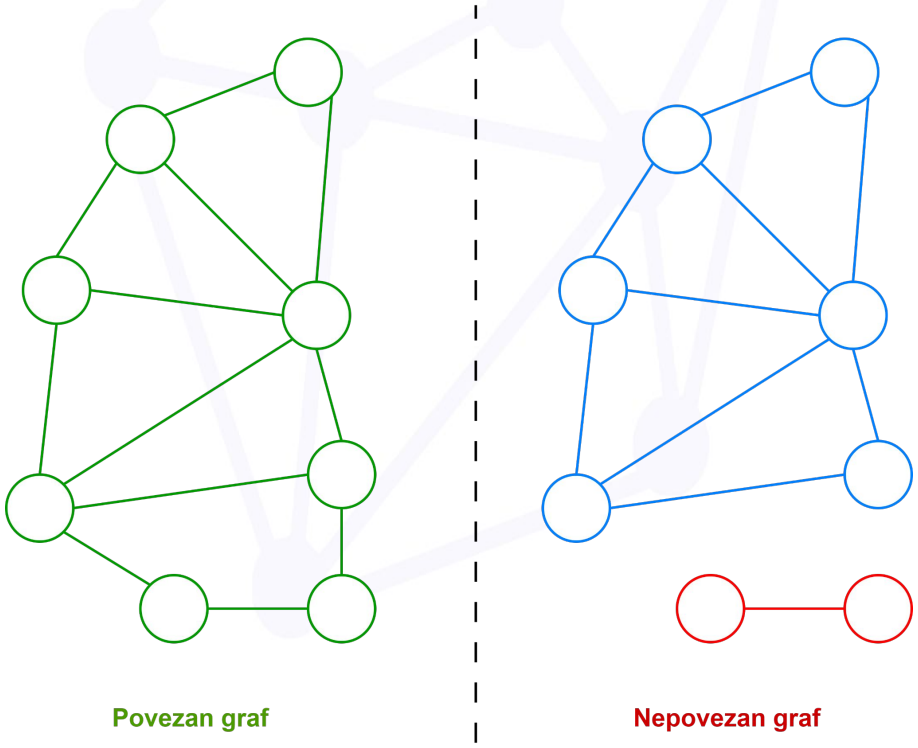
Osnovni pojmovi

- Putanja:
  - Uređena *n*-torka čvorova za koju važi da između svaka dva susedna čvora u *n*-torki postoji grana u grafu.
- Ciklus:
  - Putanja sa istim početnim i završnim čvorom.



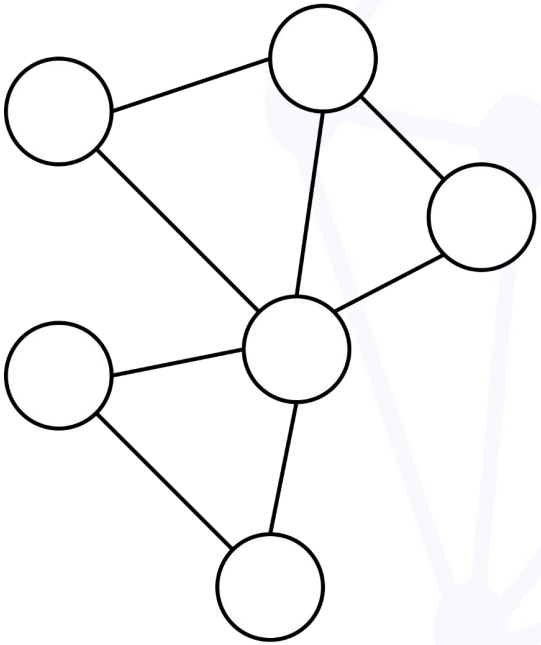
Osnovni pojmovi

- Gustina:
  - Odnos broja čvorova i broja grana.
- Povezanost:
  - Graf je povezan ako između svaka dva čvora postoji putanja.

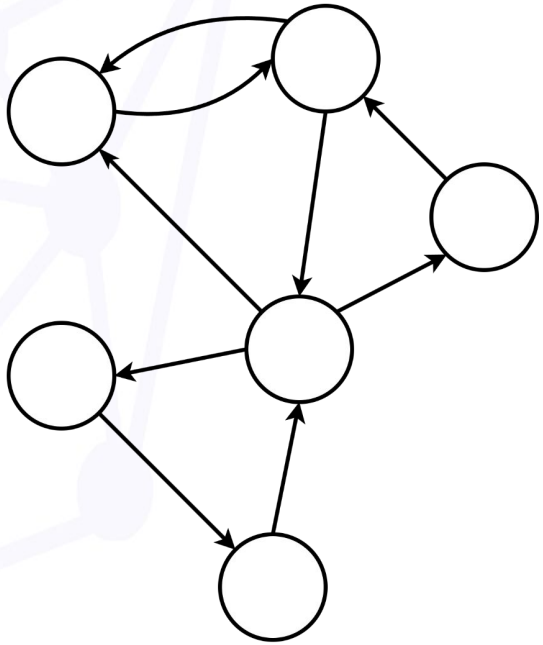


Vrste grafova

- Neusmeren - grane ne poseduju smer.
- Usmeren - grane su usmerene od jednog čvora ka drugom.



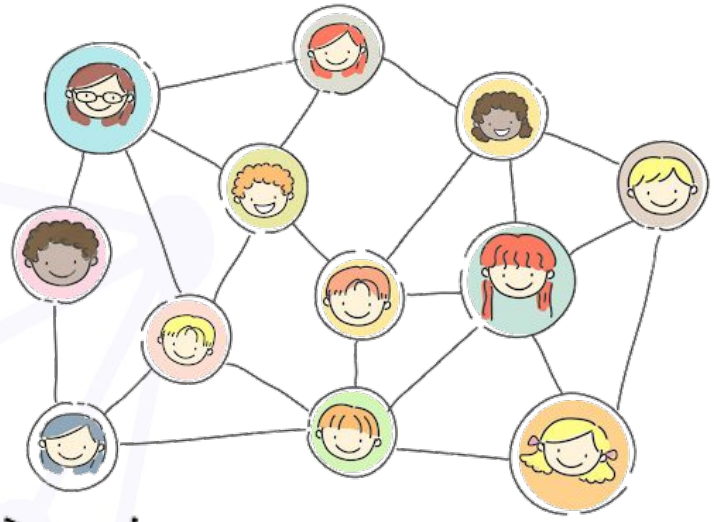
Neusmeren graf

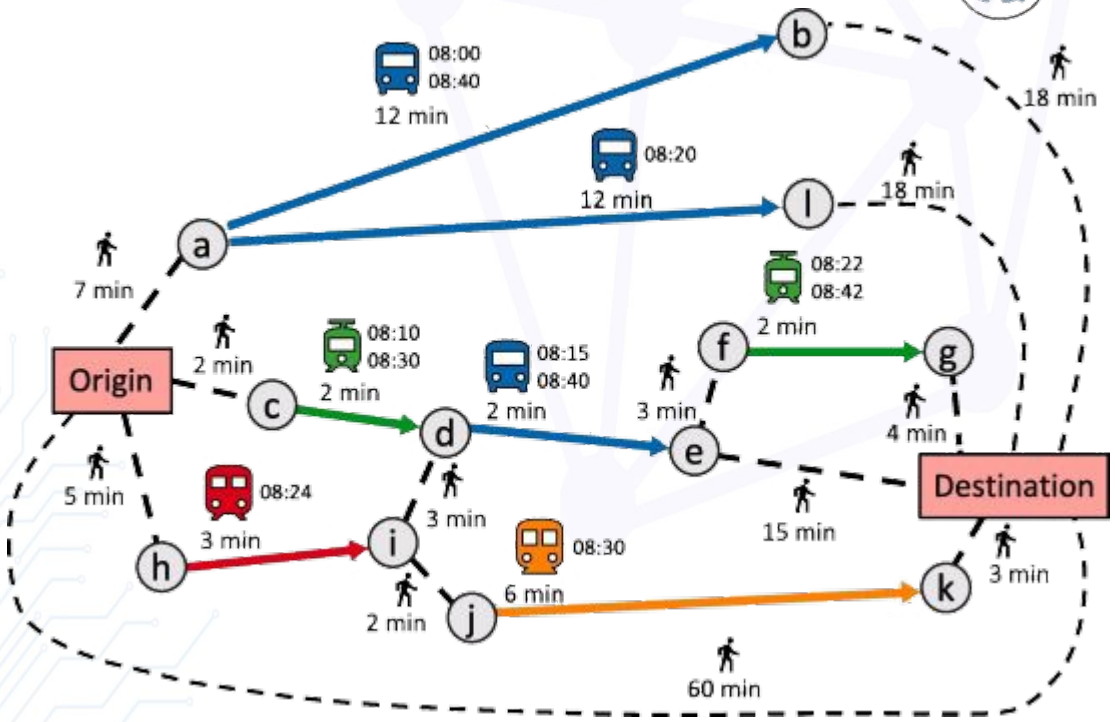


Usmeren graf

Primeri grafova

- Društvene mreže;
- Računarske mreže (internet);
- GPS;
- Saobraćajna infrastruktura;
- Organizaciona struktura preduzeća.

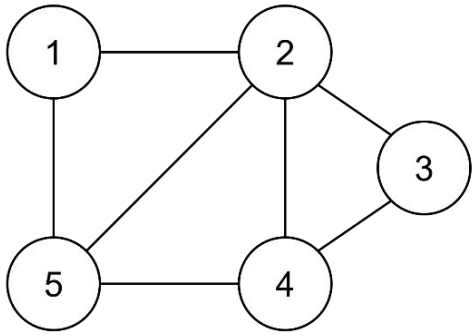


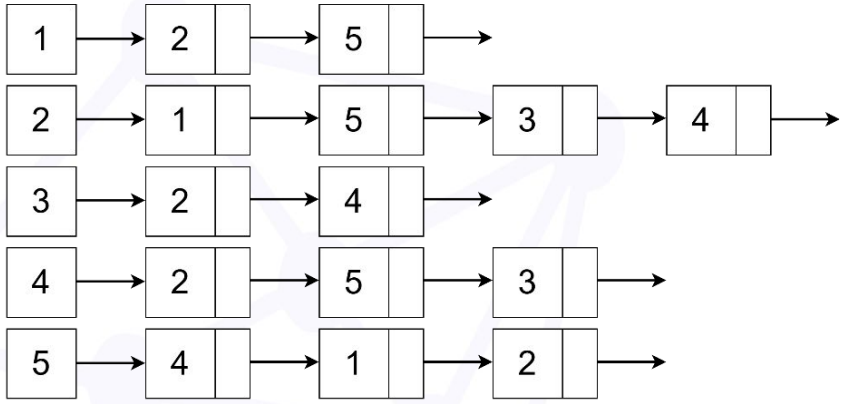


Reprezentacija grafa

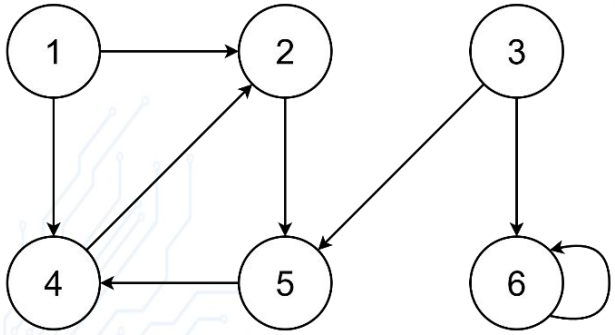
- Elementi grafa:
  - Lista čvorova;
  - Lista grana.
- Sekvencijalna reprezentacija - matrica susedstva:
  - Redovi i kolone odgovaraju čvorovima, elementi matrice odgovaraju granama između čvorova.
- Spregnuta reprezentacija - lista susedstva:
  - Za svaki čvor se beleži lista njemu susednih čvorova.

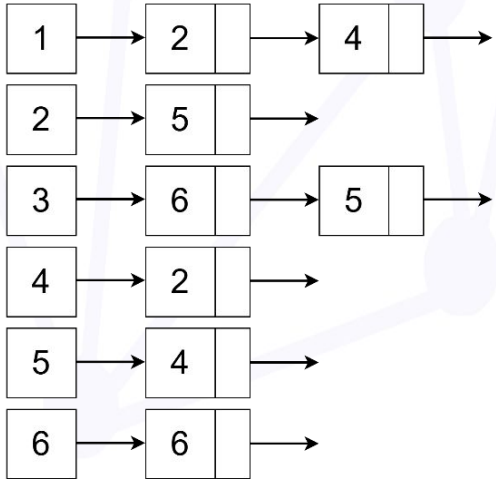
Reprezentacija grafa





	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Pretraga u širinu - BFS (*Breadth-First Search*)

- Za zadati graf ***G*** i izvorišni čvor ***s***, pretraga (obilazak) u širinu daje sve čvorove do kojih postoji putanja iz čvora ***s***.
- Obilazak u širinu koristi bojenje čvorova u belu, sivu i crnu boju:
  - Na početku su svi čvorovi beli;
  - Čvor postaje crne boje kada su svi njegovi susedni čvorovi posećeni;
  - Čvor je sive boje ako je posećen, ali postoje njegovi susedi koji nisu posećeni.
- Koristi se lista susedstva.
- Za svaki čvor se beleži njegov prethodnik u obilasku, kao i njegova udaljenost od izvorišnog čvora ***s***.
- Kao pomoćna struktura za neobrađene čvorove koristi se red (*queue*).

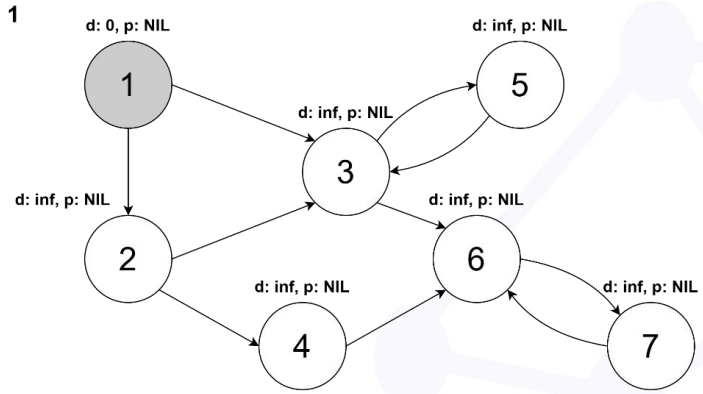
BFS - *pseudo-kod*

```
function BFS(G, s)

    for u in G.V
        u.color = WHITE
        u.d = inf
        u.p = NIL

    s.color = GRAY
    s.d = 0
    s.p = NIL

    Q = initQueue()
    Q.push(s)

    ...
```

```
...

while Q.notEmpty()
    u = Q.pop()
    for v in G.adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.p = u
            Q.push(v)
    u.color = BLACK
```

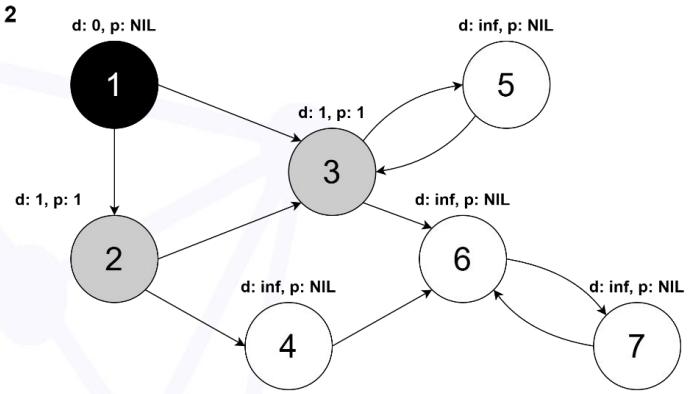
BFS - primer

1



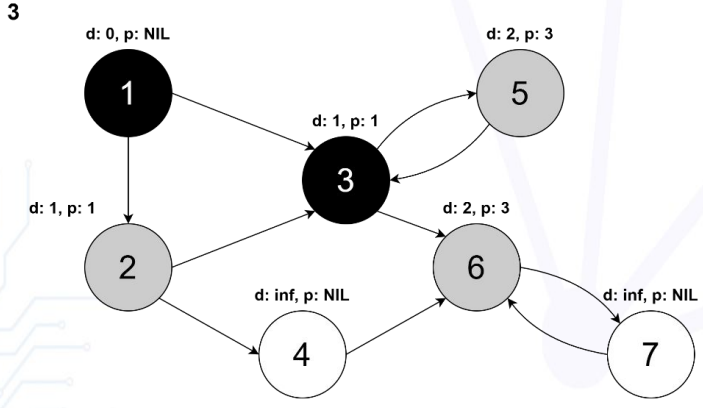
Q : 1

2



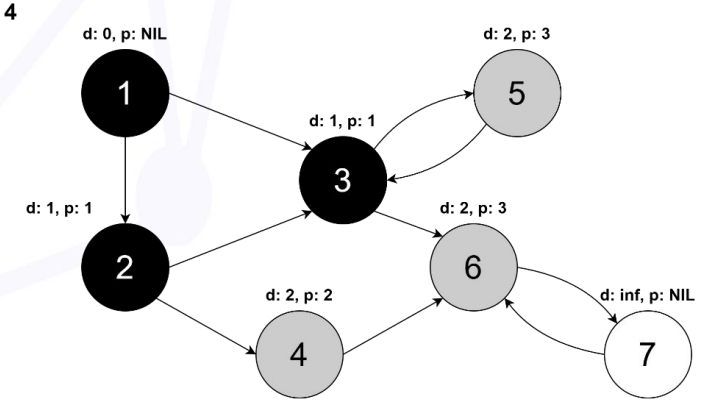
Q : 3 2

3



Q : 2 5 6

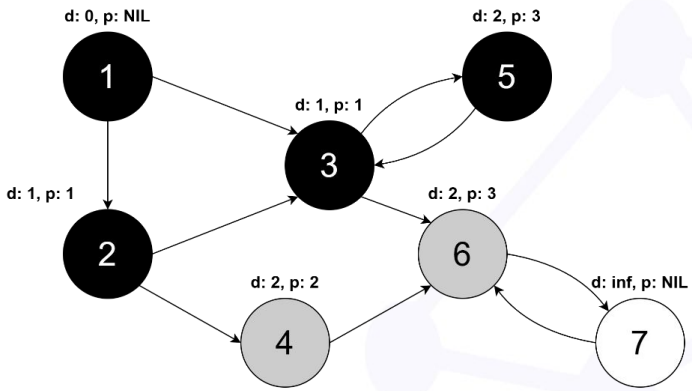
4



Q : 5 6 4

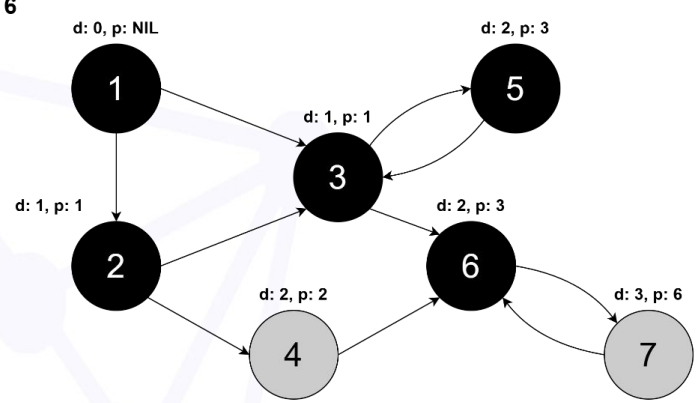
BFS - primer

5



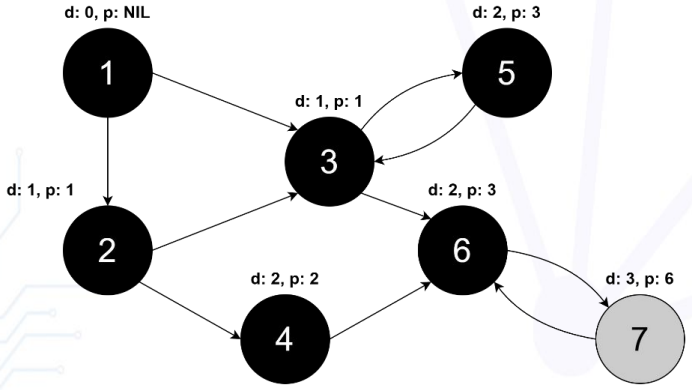
Q : 6 4

6



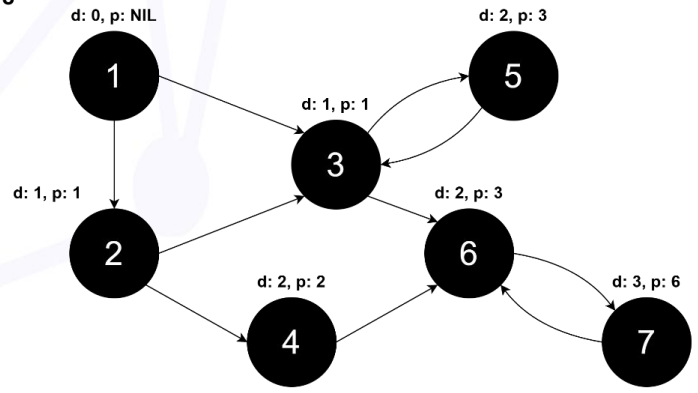
Q : 4 7

7



Q : 7

8



Q :

Pretraga u dubinu - DFS (*Depth-First Search*)

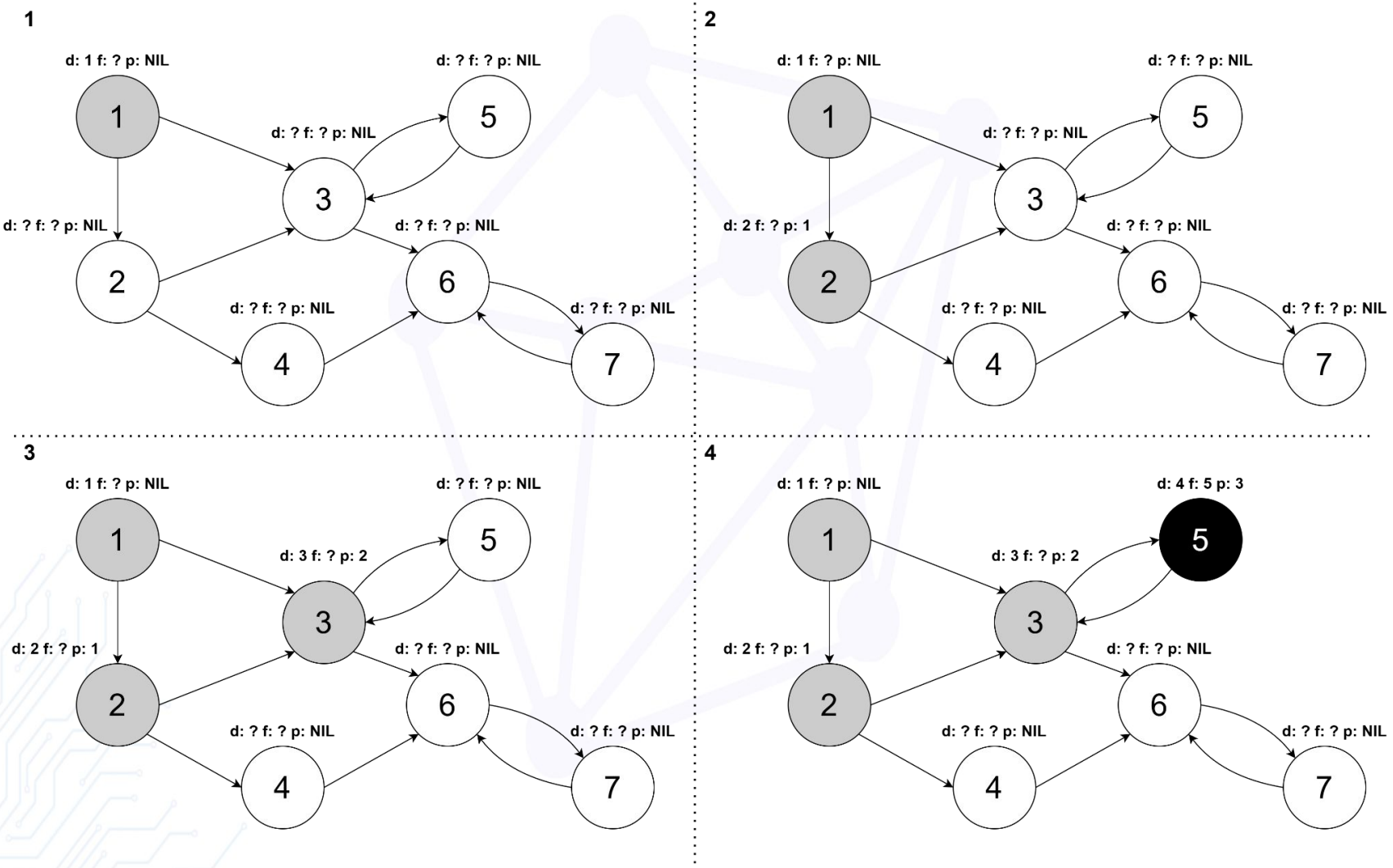
- Za zadati graf ***G***, obilazak u dubinu posećuje sve čvorove za koje postoji putanja iz nekog izvorišnog čvora, tako što se posećuju čvorovi u dubinu koliko god je to moguće, pa se potom obilazak vraća na prethodno izostavljene čvorove.
- Slično kao i kod *BFS*, čvorovi se boje u belu, sivu i crnu boju.
- DFS obilazi sve čvorove koristeći *DFS-VISIT* za obilazak iz određenog izvorišnog čvora ***s***.
- Kao pomoćna struktura za posećene neobrađene čvorove koristi se stek ili se implementira rekurzivno.

DFS - *pseudo-kod*

```
function DFS(G, s)
    for u in G.V
        u.color = WHITE
        u.p = NIL
        u.d = -1
        u.f = -1
    time = 0
    DFS-VISIT(G, s, time)
```

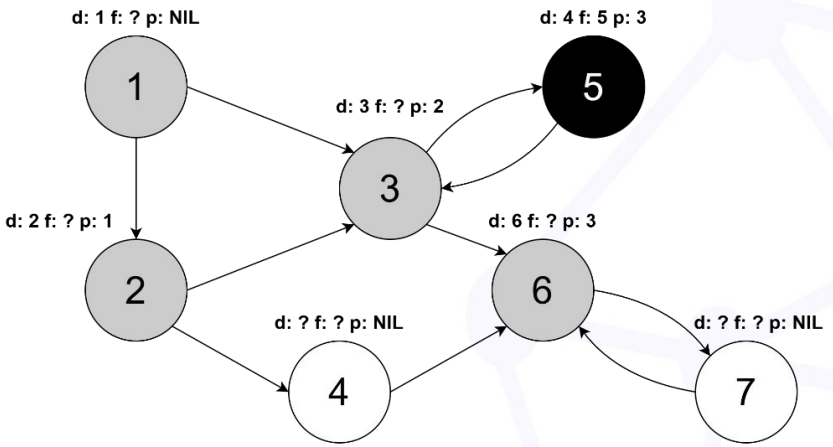
```
function DFS-VISIT(G, u, time)
    time = time + 1
    u.d = time
    u.color = GRAY
    for v in G.adj[u]
        if v.color == WHITE
            v.p = u
            DFS-VISIT(G, v, time)
    u.color = BLACK
    time = time + 1
    u.f = time
```

DFS - Primer

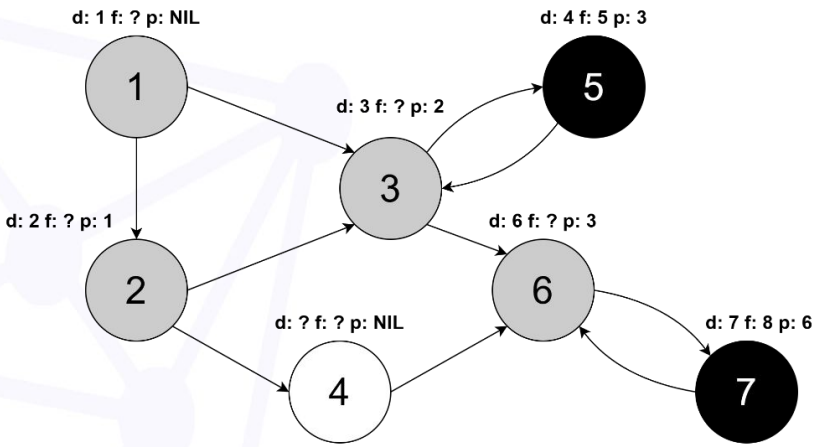

1
d: 1 f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
1
2
3
4
5
6
7
2
d: 1 f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: 2 f: ? p: 1
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
1
2
3
4
5
6
7
3
d: 1 f: ? p: NIL
d: ? f: ? p: NIL
d: 3 f: ? p: 2
d: 2 f: ? p: 1
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
1
2
3
4
5
6
7
4
d: 1 f: ? p: NIL
d: 4 f: 5 p: 3
d: 3 f: ? p: 2
d: 2 f: ? p: 1
d: ? f: ? p: NIL
d: ? f: ? p: NIL
d: ? f: ? p: NIL
1
2
3
4
5
6
7

DFS - Primer

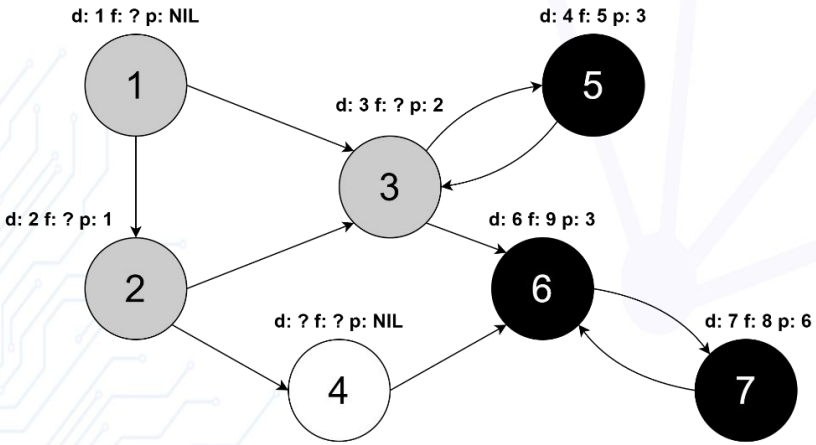
5



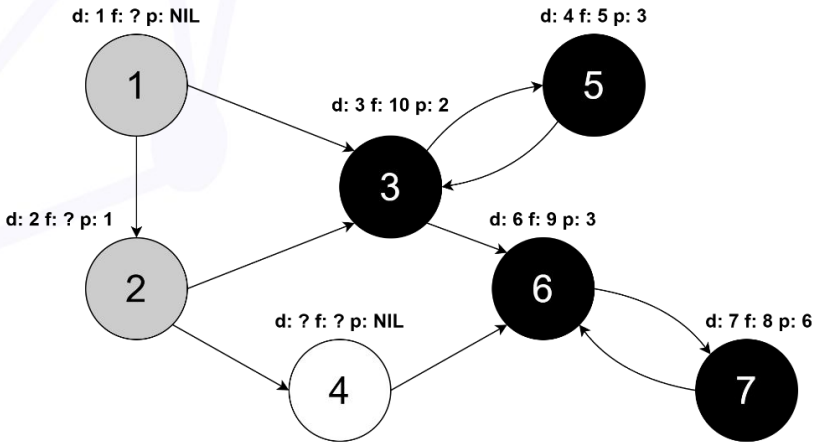
6



7

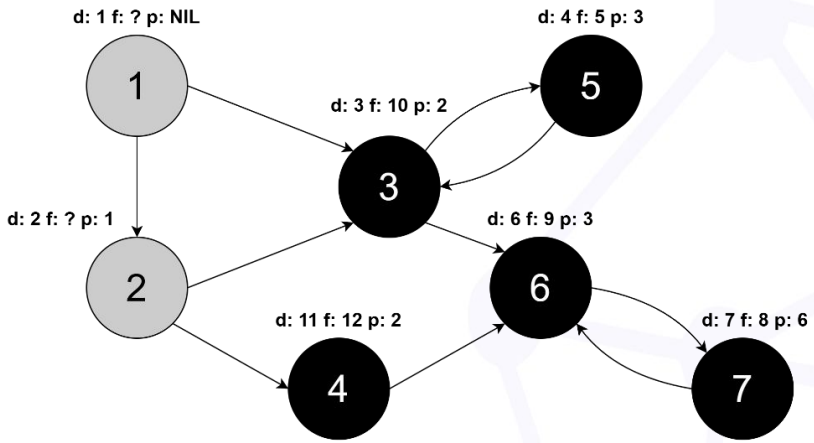


8

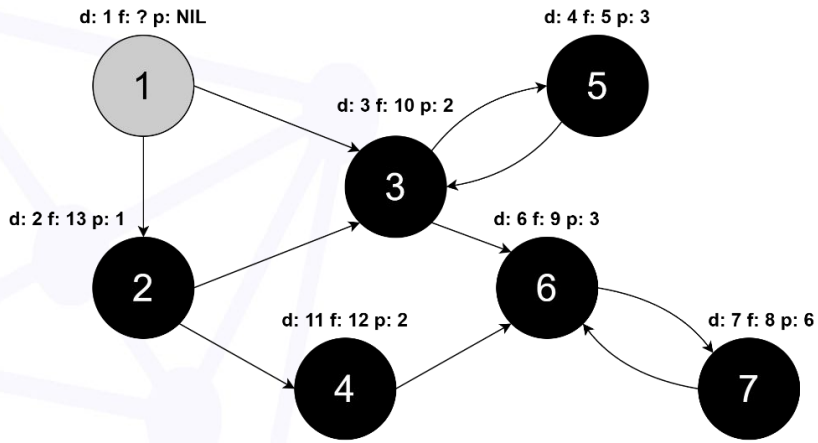


DFS - Primer

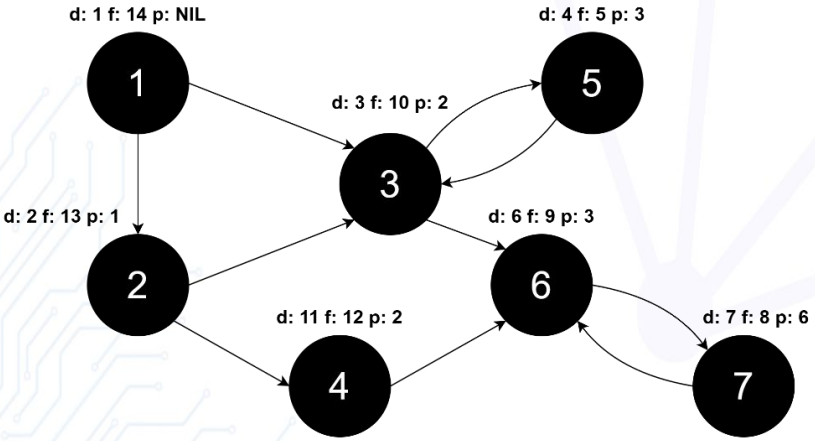
9



10



11

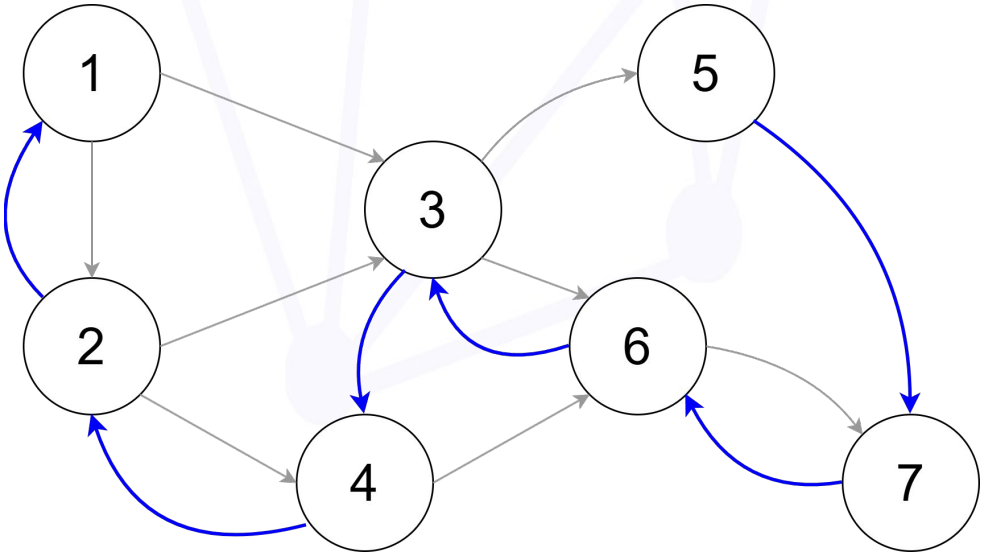


Ispis obilaska:

```
1: First visit, d:1 p: NIL 3: Last visit, f: 10
2: First visit, d:2 p: 1   2: Revisit
3: First visit, d:3 p: 2   4: First visit, d:11 p: 2
5: First visit, d:4 p: 3   4: Last visit, f: 12
5: Last visit, f: 5        2: Revisit
3: Revisit                 2: Last visit, f: 13
6: First visit, d:6 p: 3   1: Revisit
7: First visit, d:7 p: 6   1: Last visit, f: 14
7: Last visit, f: 8
6: Revisit
6: Last visit, f: 9
3: Revisit
```

Topološki sort

- DAG - *Directed Acyclic Graph*.
- Topološki sort predstavlja uređivanje grafa.
- Skup grana predstavlja relaciju poretka.
- Postupak:
  - Izvršiti DFS nad grafom G;
  - Svaki čvor, nakon što se završi njegova obrada (*last visit*), dodaje se u spregnutu listu;
  - Rezultujuća spregnuta lista predstavlja topološko uređenje grafa G.
- Primer:



Zadatak 1

- Za zadate čvorove ***s*** i ***t*** u usmerenom grafu ***G*** odrediti najkraću putanju u pogledu broja grana koje pripadaju putanju.
- Ako putanja ne postoji vratiti vrednost -1.

Zadatak 2

- Implementirati DFS algoritam tako da umesto rekurzije koristi stek kao pomoćnu strukturu podataka.

Zadatak 3

- Koristeći DFS, implementirati algoritam koji će u neusmerenom povezanom grafu naći putanju u kojoj se svaka grana pojavljuje po jednom u svakom smeru.
  - Ukoliko pravite zatvorenu putanju svaki čvor mora imati paran broj suseda;
  - Ukoliko pravite otvorenu putanju, tačno dva čvora moraju imati neparan broj suseda;
  - Imate slobodu da birate koju putanju pravite/tražite.
- Kao pomoć možete na internetu tražiti kako radi algoritam:
  - "Hierholzer’s Algorithm for Eulerian Circuit"