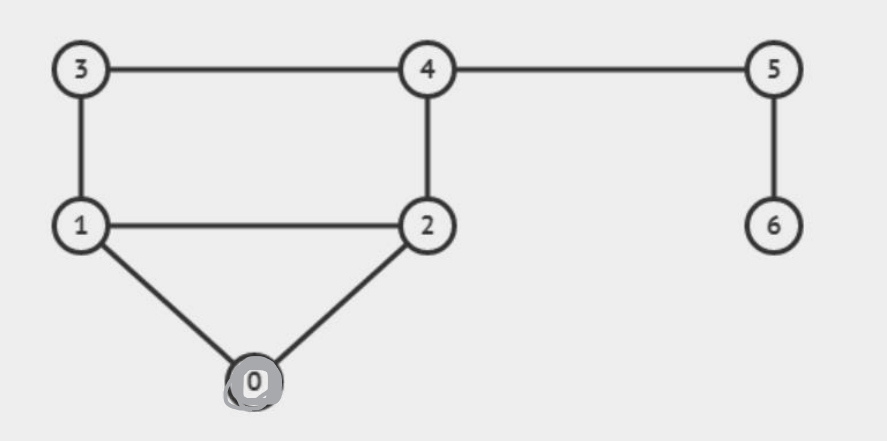
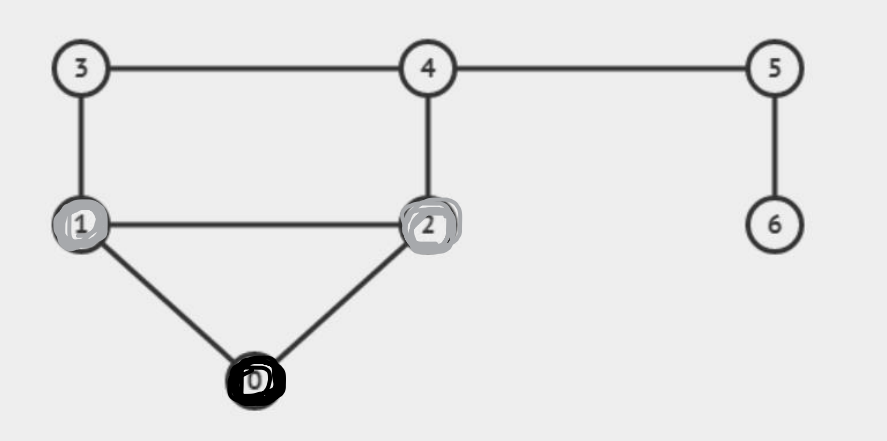
Zadaća 11

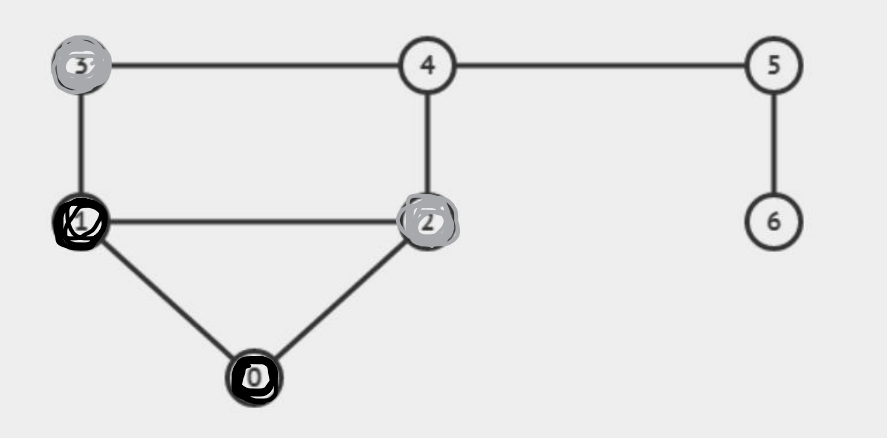
1. BFS algoritam radi tako što na početku svaki vrh osim početnog oboja bijelo. Početni vrh oboja sivo. Bijela boja označava vrh koji nije otkriven, siva označava vrh koji je otkriven ali iz njega nije završena pretraga, a crna vrh iz kojeg je pretraga završena. Inicijaliziramo red kao FIFO strukturu podataka. U njega stavimo početni vrh. Dok red nije prazan prvo dequeueamo početni element iz reda i svaki vrh koji je tom elementu susjedan obojamo sivo (ako je bijel) i stavimo ga u queue. Nakon što smo sve susjedne vrhove obojali sivo obojamo vrh iz kojeg smo obavili pretragu crno što simbolizira kraj pretrage tog vrha. Ovako BFS radi na grafu na slici (počinjemo od vrha 0):



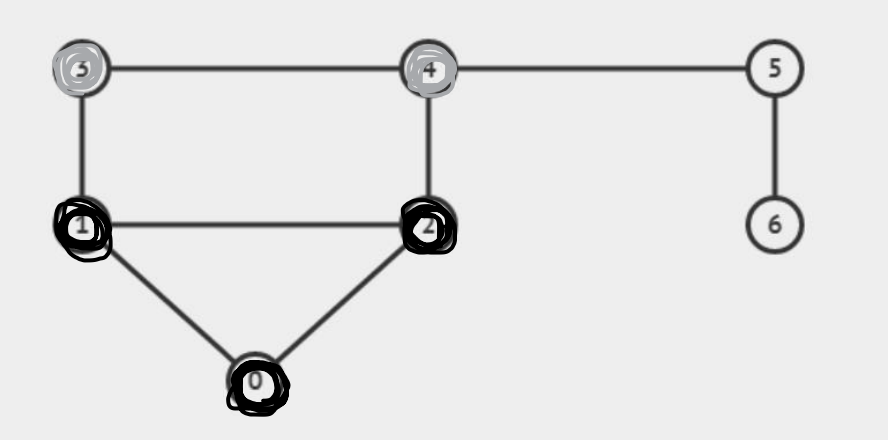
Queue : [0]



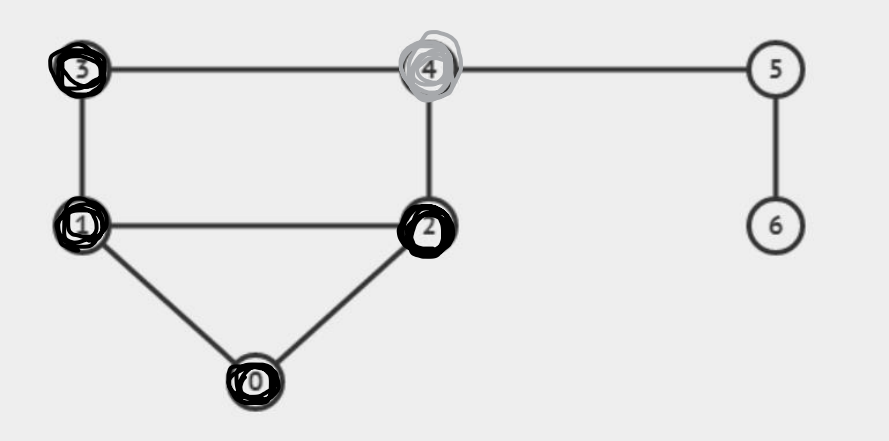
Queue : [1, 2]



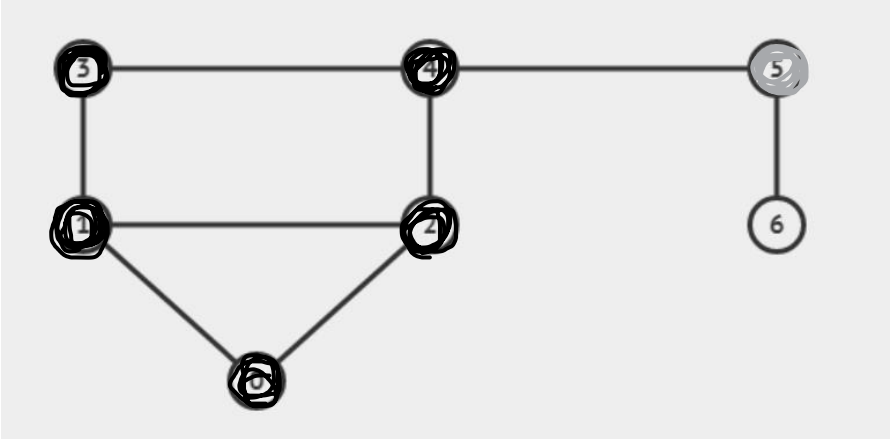
Queue : [2, 3]



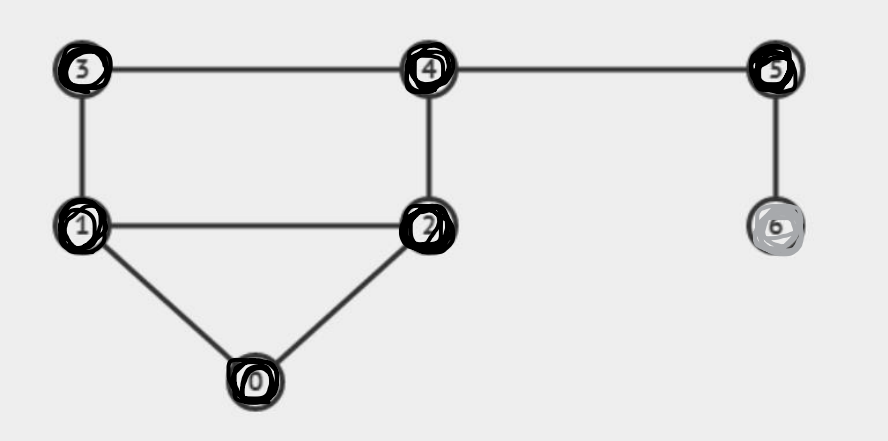
Queue : [3, 4]



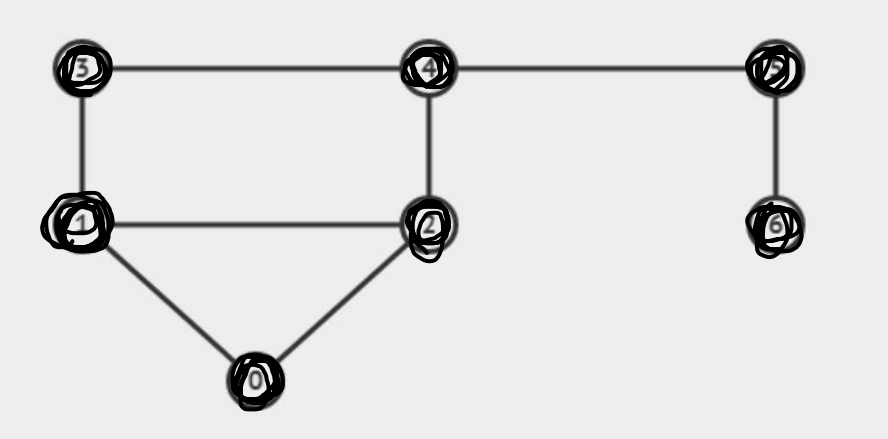
Queue : [4]



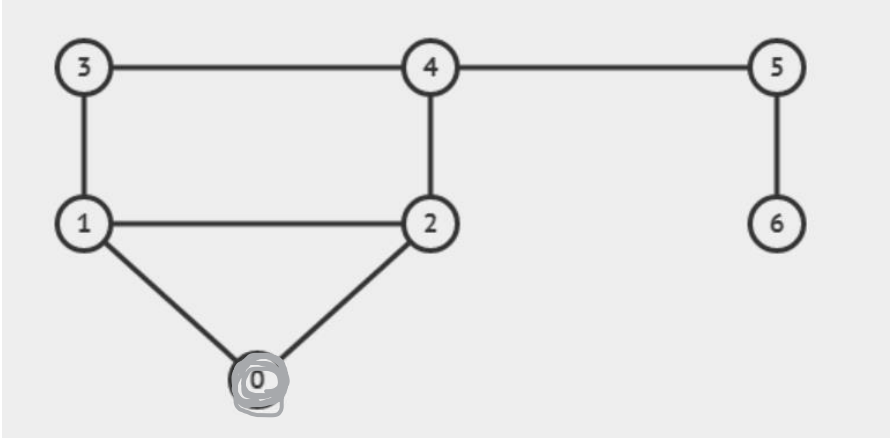
Queue : [5]

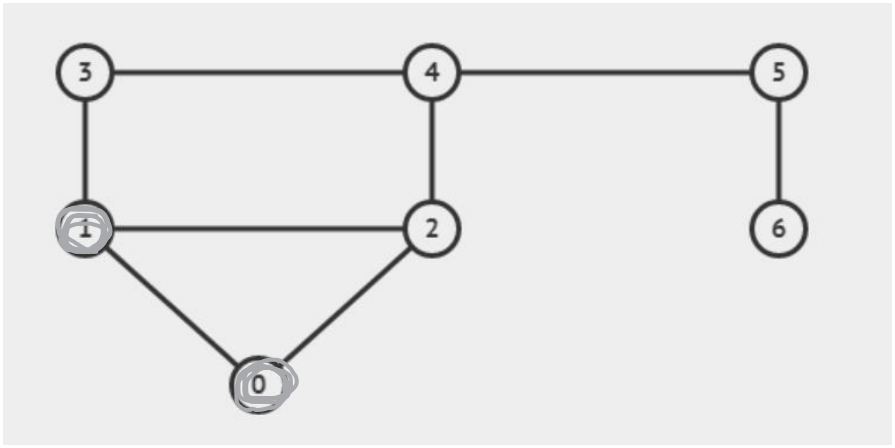


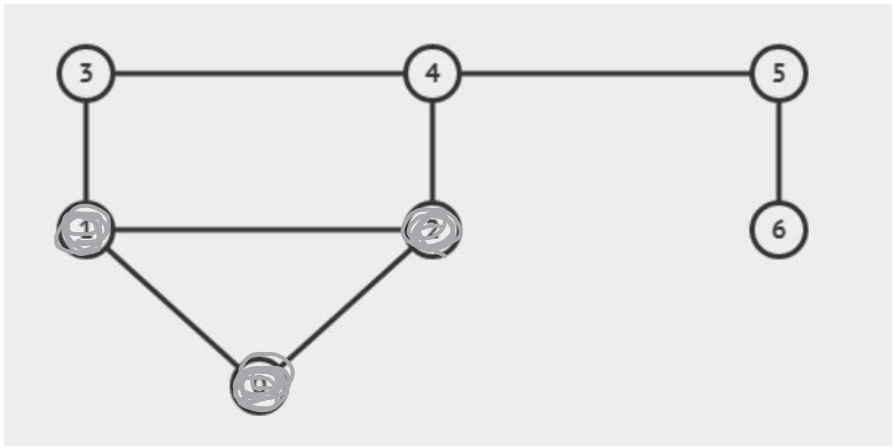
Queue : [6]

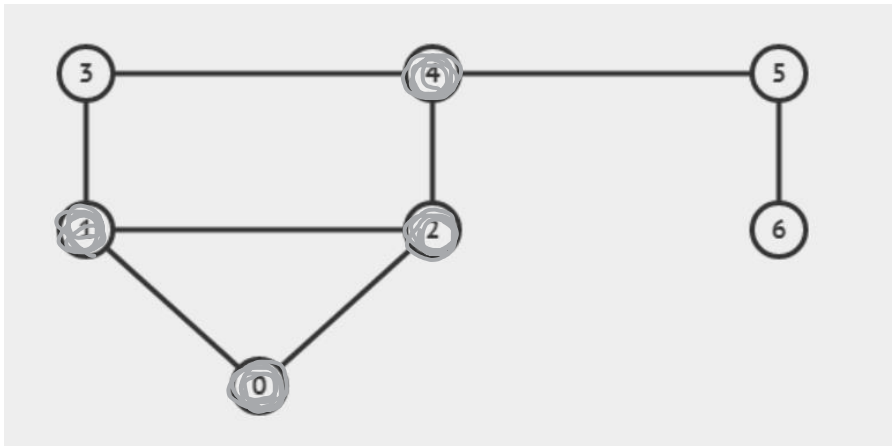


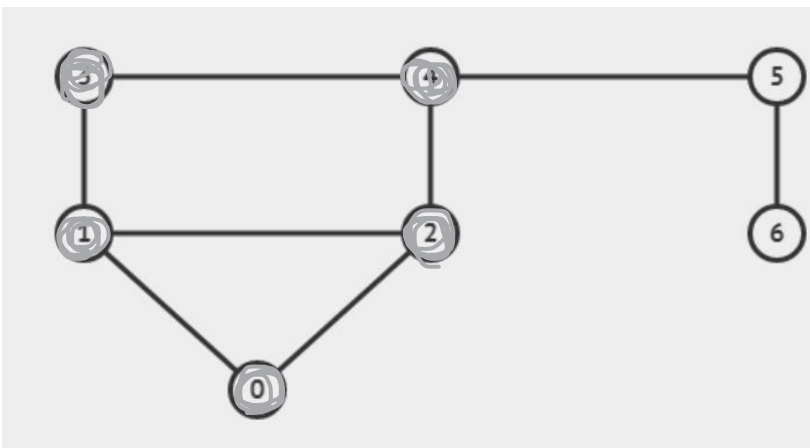
DFS algoritam radi tako što na početku stavi sve vrhove na bijelu boju i za svaki vrh u skupu vrhova napravi proceduru DFS-VISIT. DFS-VISIT prima čvor i boja ga sivo. Nakon toga prolazi kroz čvorove susjedne tom čvoru i provjerava jesu li bijele boje. Ako je vrh bijele boje, rekurzivno poziva DFS-VISIT na tom vrhu. Nakon što je petlja koja provjerava susjedne čvorove završena, oboja čvor crno što simbolizira završetak pretrage iz tog čvora. Ovako DFS radi na grafu na slici (krećemo od čvora 0):

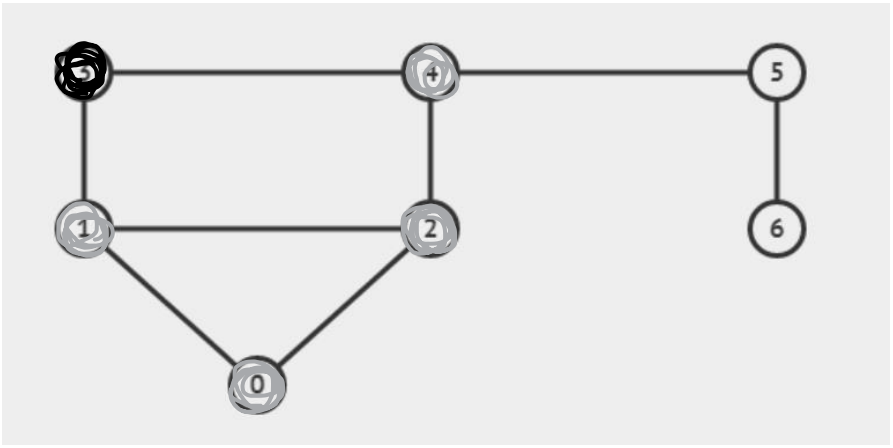


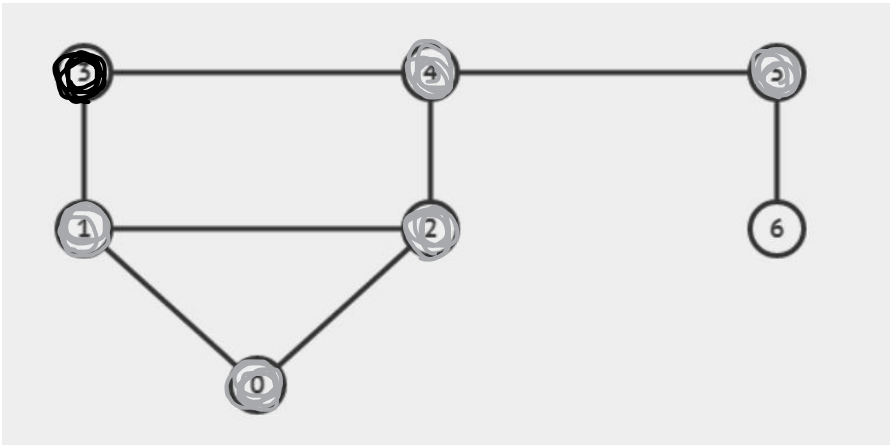

3
4
5
1
2
6
0

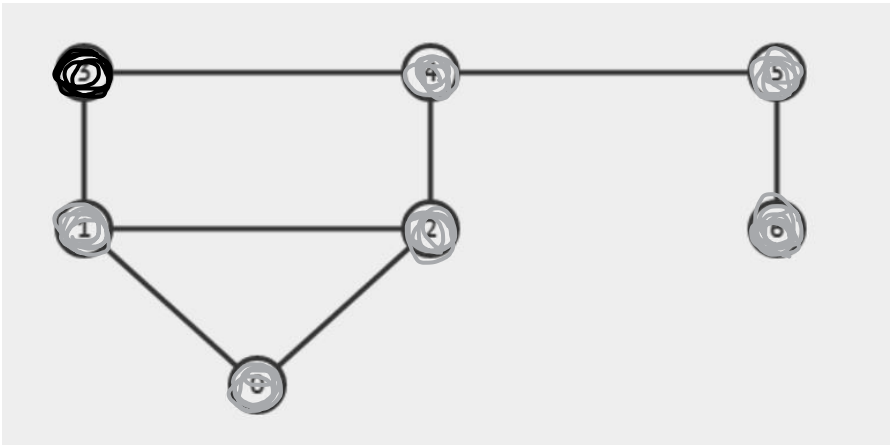

3
4
5
1
2
6
0

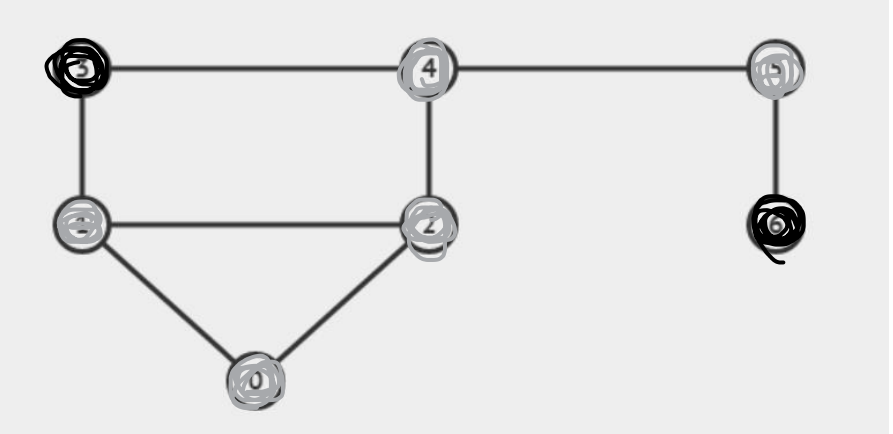

3
4
5
1
2
6
0

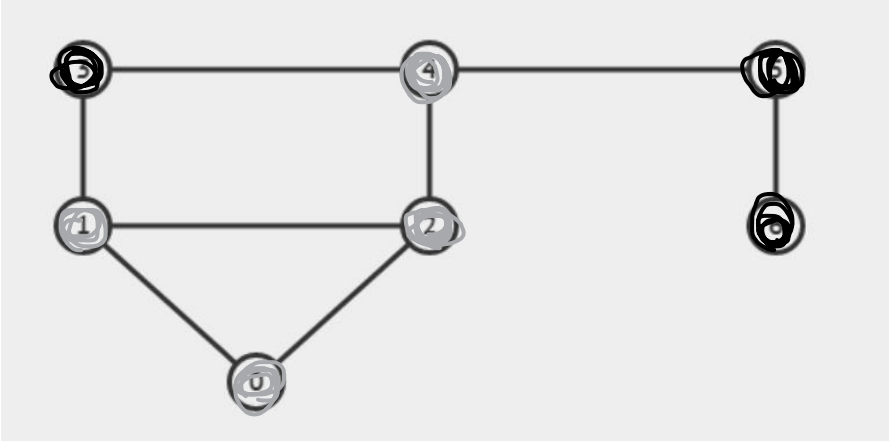

3
4
5
1
2
6
0

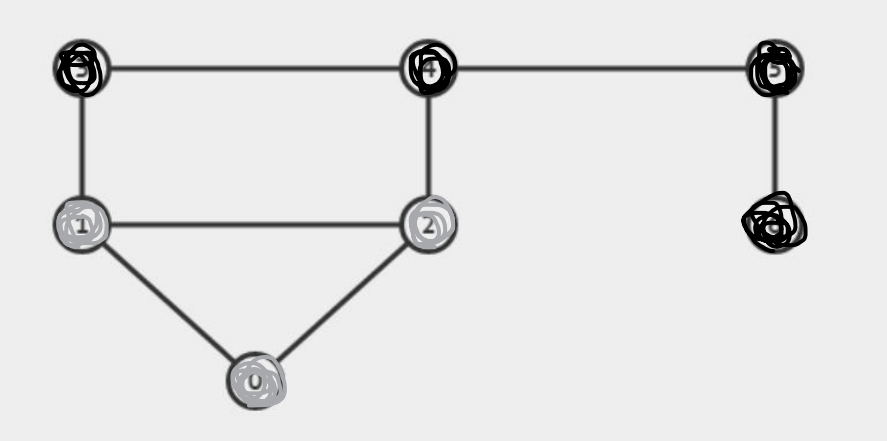

3
4
5
1
2
6
0

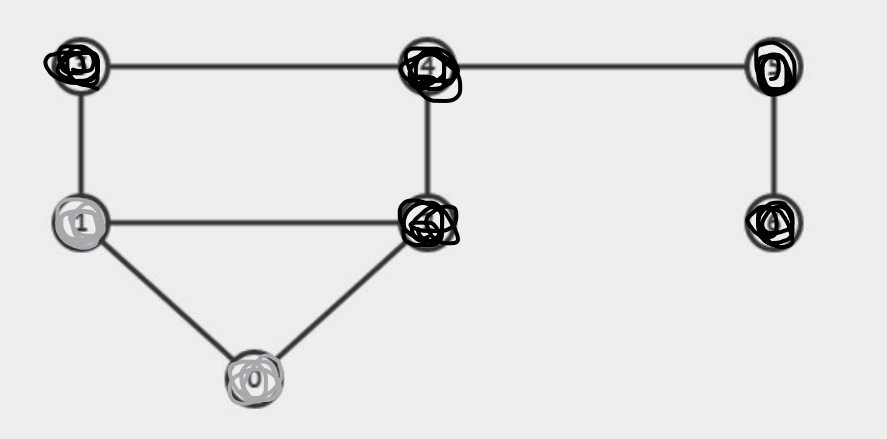

3
4
5
1
2
6
0

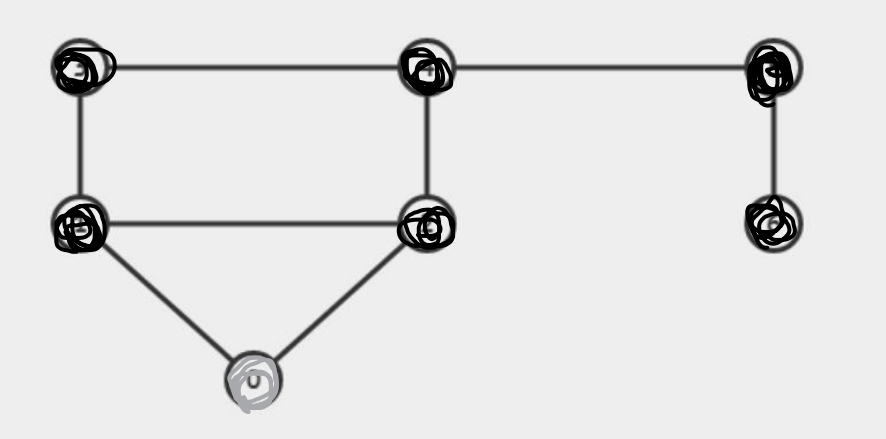

3
4
5
1
2
6
0

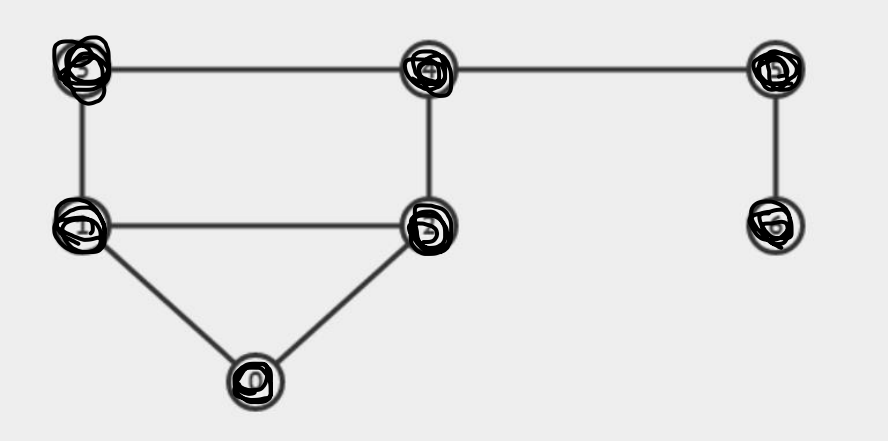

3
4
5
1
2
6
0


3
4
5
1
2
6
0


3
4
5
1
2
6
0

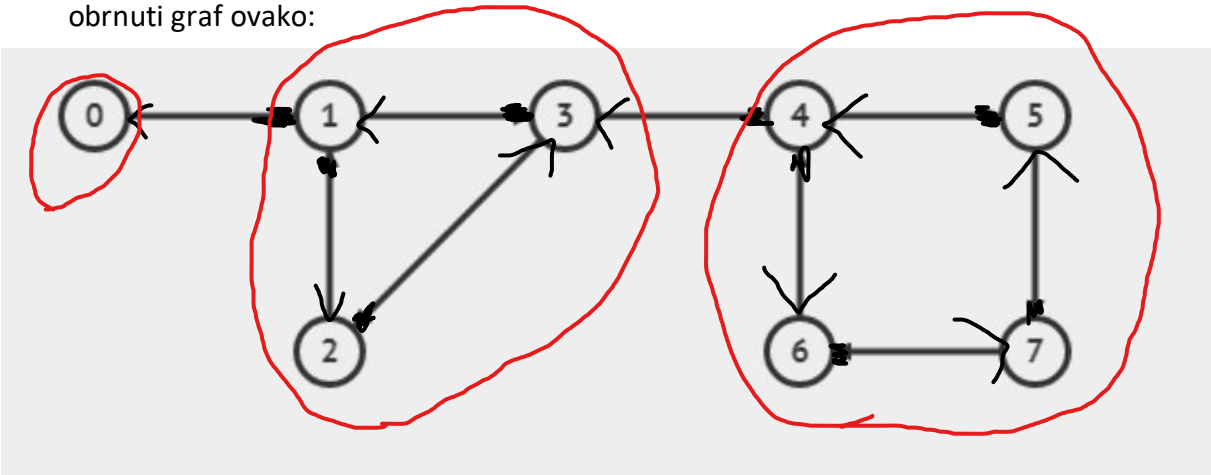

3
4
5
1
2
6
0


3
4
5
1
2
6
0



2. Komponente jake povezanosti izračunat ćemo sljedećim algoritmom:
   Prvo ćemo izračunati završna vremena pretrage za svaki vrh u grafu u DFS pretrazi, tj. računamo f[u] za svaki u koji je vrh u V uz pomoć DFS pretrage. Izračunat ćemo transponirani graf od početnog. Nakon toga ponovo ćemo pozvati DFS prema padajućem poretku od f[u]. Na kraju ćemo vratiti vrhove svakog stabla u DFS šumi kao SCC. Taj algoritam zove se Kosaraju algoritam. Možemo koristiti stack u koji ćemo pushati svaki vrh iz kojeg smo završili pretragu, transponirati graf (obrnuti bridove koji spajaju dva vrha) i nakon toga pozivati DFS na transponiranom grafu iz vrhova na vrhu stacka i popati ih sa stacka. U svakom DFS prolazu u tom redoslijedu posjećeni vrhovi tvorit će SCC.

   U primjeru sa slike stack nakon prvog DFS-a će izgledati ovako:
   [0, 1, 3, 4, 5, 7, 6, 2],
   obrnuti graf ovako:



   , a SCC su označene na grafu crvenom bojom.

3. Primov algoritam je greedy algoritam koji starta na nekom vrhu grafa, promotri svaki brid iz tog vrha i izabere onaj s najmanjom težinom koji doda u MST. Nastavi tako što promatra svaki brid koji spaja čvorove koji su dodani u MST s onima koji nisu dodani. Izabere najmanji i ako on ne formira ciklus, odabere ga. Tako nastavi sve dok ne dobije gotov MST.
   Za graf na slici prvo izaberemo vrh 0 i dodamo ga u MST. Pogledamo koji brid ima najmanju težinu iz vrha 0; to je brid 0 – 1 s težinom 50. Sada u MST-u imamo brid 0 – 1 s težinom 50. Pogledamo koji su vrhovi dohvatljivi iz vrhova 0 i 1. To su vrhovi 3, 4 i 2. Brid 0 – 2 ima najmanju težinu (60) pa njega dodamo. Sada opet gledamo koji su vrhovi dohvatljivi iz 0, 1 i 2. Brid s najmanjom težinom je brid 2 – 5 (50) pa njega dodamo u stablo. Tako nastavljamo pa dodajemo brid 5 – 3 (80), pa 3 – 6 (70) i na kraju 6 – 4 (40). Tako smo povezali sve vrhove s bridovima najmanje težine i imamo MST. Ovako izgleda MST:
   0 – 1 (50)
   0 – 2 (60)
   2 – 5 (50)
   3 – 5 (80)
   3 – 6 (70)
   4 – 6 (40) i totalna težina MST-a je 350.

   Kruskalov algoritam prvo sortira sve bridove u grafu u rastućem poretku (po težinama). Tim redoslijedom dodaje bridove u MST ako bridovi ne formiraju cikluse. To je također greedy algoritam jer u svakom koraku uvijek dodaje najmanji (lokalno optimalni) brid. Da bi odabrali brid koji ne stvara ciklus koristimo operaciju disjunktnih skupova FIND-SET (za svaki brid provjeravamo je li FIND-SET(prvog vrha) != FIND-SET(drugog vrha), ako je onda brid ne formira ciklus). Vrhove spajamo operacijom UNION.
   Za graf na slici prvo ćemo odabrati brid 4 – 6 jer on ima najmanju težinu (40). Nakon toga odabiremo npr. brid 0 – 1 (50). Ta dva brida ne formiraju ciklus pa ih možemo odabrati. Nadalje, odabiremo brid 2 – 5 (50). On ne formira ciklus s odabranim bridovima pa ga možemo odabrati. Nakon njega odabiremo brid 0 – 2 (60). Sada smo spojili vrhove 0 – 2 – 5 no to i dalje nije ciklus pa možemo odabrati taj brid. Nakon njega odabiremo brid 3 – 6 (70). Tako smo spojili vrhove 3 – 6 – 4 no to nije ciklus pa možemo odabrati taj brid. Nakon njega odabiremo brid 3 – 5 (80). To i dalje nije ciklus jer smo spojili vrhove 0 – 2 – 5 – 3 – 6 – 4, no 4 nije spojena s 1 pa ne formiramo ciklus. Nakon toga, broj bridova u MST-u jednak je (V - 1) pa algoritam tu staje. Kad ne bi tu stajao, i dalje nakon tog brida odabiremo brid 1 – 4 (90) no on će spojiti 4 s 1 pa će 0 – 2 – 5 – 3 – 6 – 4 – 1 – 0 biti ciklus (taj brid ne dodajemo u MST). Nakon toga odabiremo 1 – 3 (120). To će također tvoriti ciklus npr. 0 – 2 – 5 – 3 – 1 – 0, pa taj brid ne dodamo u MST. Na kraju, odabiremo brid 5 – 6 (140) no on će opet formirati ciklus, npr. 3 – 6 – 5 pa ni njega ne bi mogli dodati u MST. Dakle, algoritam će formirati ovakav MST:
   0 – 1 (50)
   0 – 2 (60)
   2 – 5 (50)
   3 – 5 (80)

3 – 6 (70)

4 – 6 (40) (sortirano po rastućem poretku od prvog vrha)

i totalna težina MST-a je ponovo 350. Vidimo da su MST-ovi formirani Primovim i Kruskalovim algoritmom jednaki.