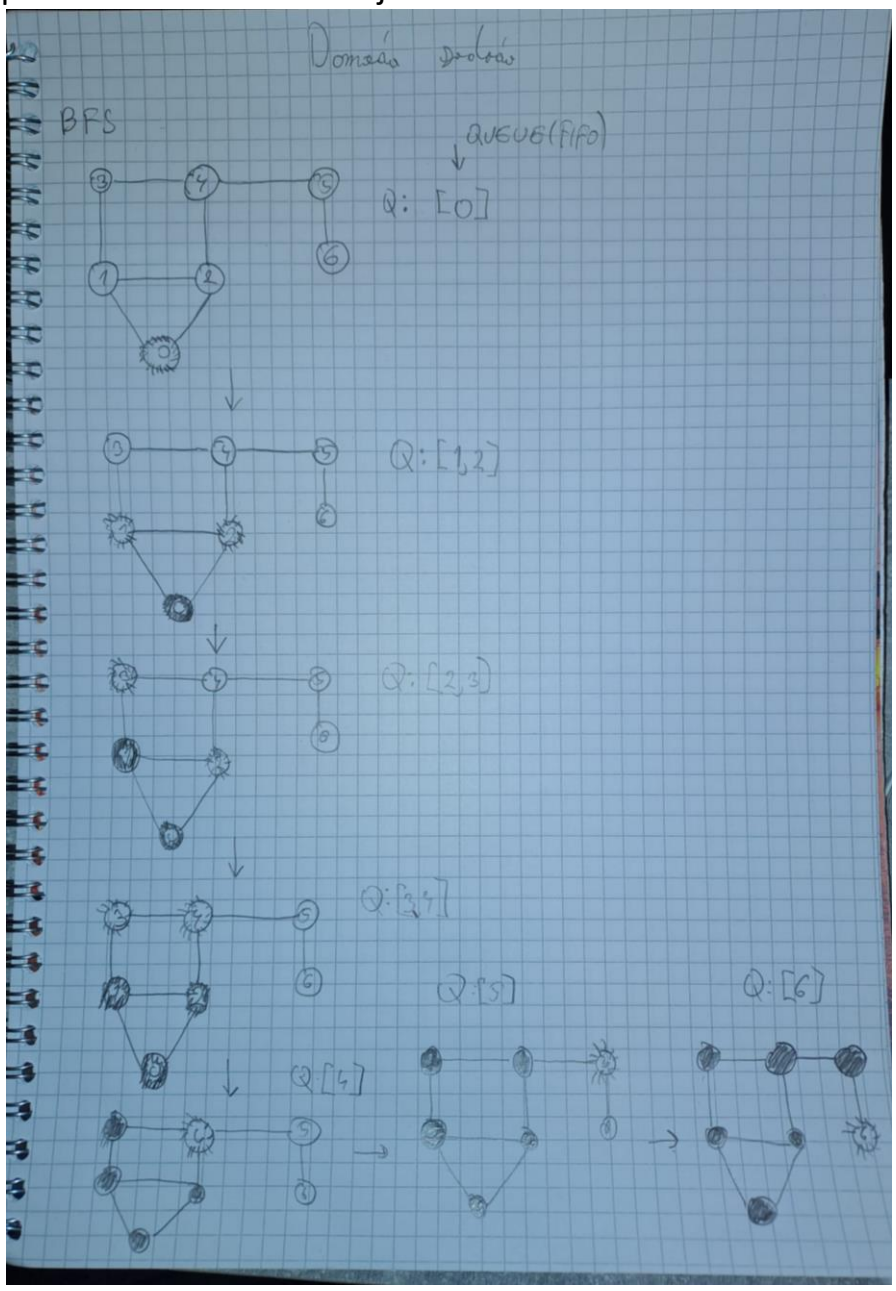
Zadatak 1

BFS

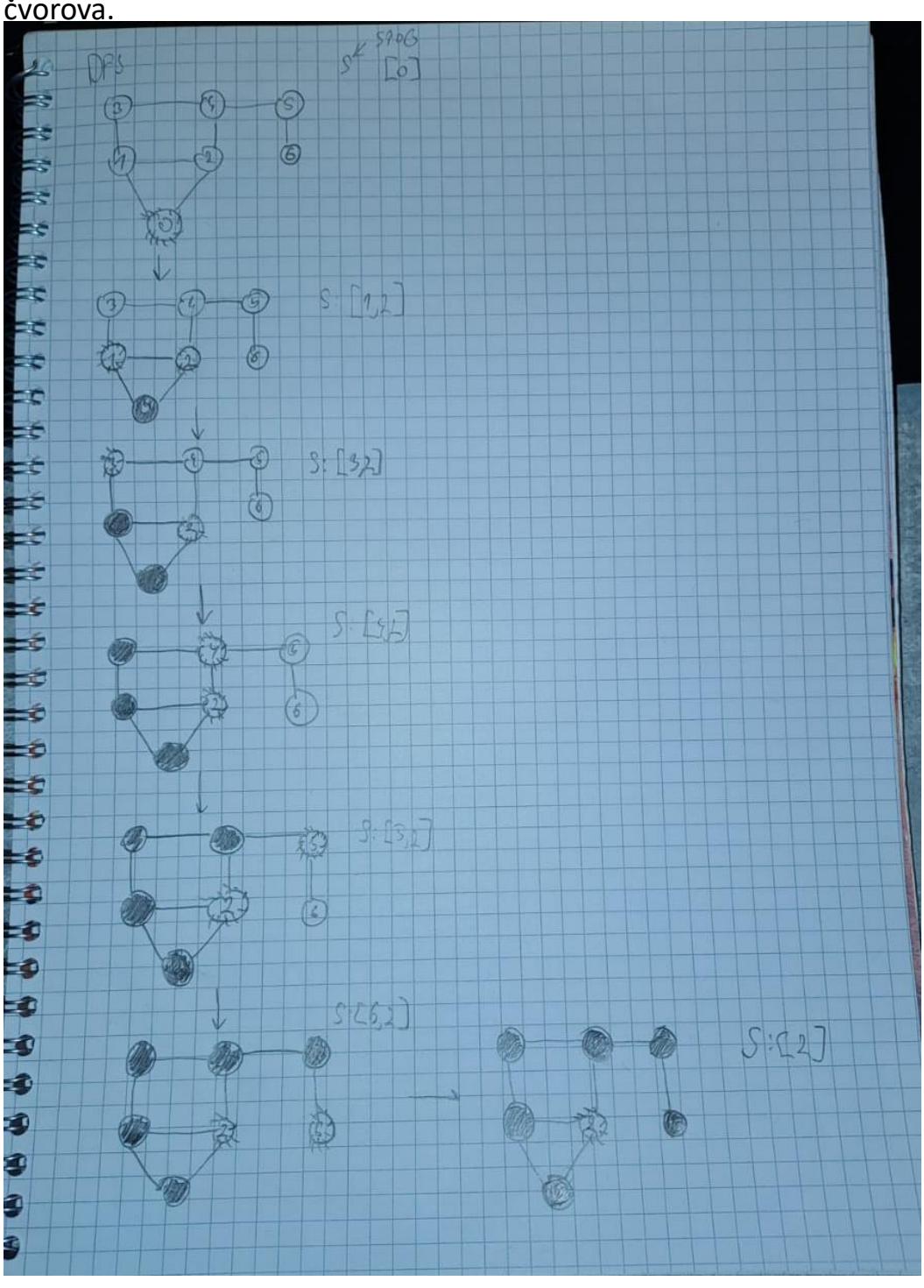
Za BFS algoritam koristimo boje kako bi razaznali koji je čvor u grafu posjećen, a koji nije. Na početku svi čvorovi grafa osim početnog iz kojeg krećemo su označeni bijelom bojom (bijela boja označava da čvor nije otkriven) dok je početni čvor označen sivom (siva boja označava da je čvor otkriven i da se iz njega izvršava pretraga susjednih čvorova). Na kraju za oznaku da je čvor otkriven i da je pretraga završena obojamo ga crnom bojom. Za BFS algoritam potreban nam je strukutra podataka reda. Na početku u red stavimo početni čvor iz koje počinje pretraga. Dok god nije red prazam izvadimo čvor koji je na početku, obojamo ga crno i u red ubacimo njegove susjede (ako su oni bijeli) te njih obojamo u sivo. Na priloženoj slici je skiciran postupak BFS algoritma, bojanje čvorova i stanje reda u tom trenutku. Na slici malo označeni čvorovi označuju sivu boju, dok potpuno crni čvorovi označuju crnu.



Na kraju čvor 6 će se obojati crnom bojom i izbaciti iz reda gdje staje i algoritam.

DFS

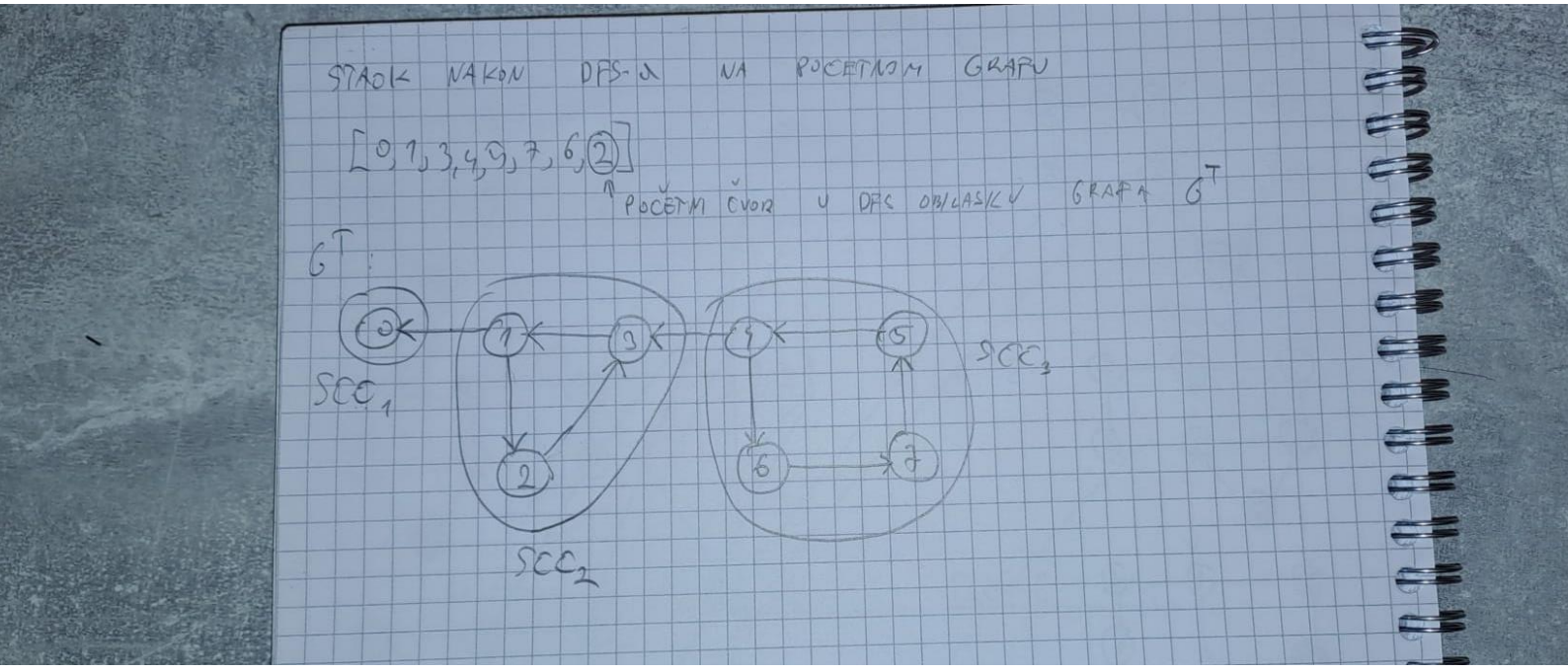
Ovaj algoritam isto tako radi na principu bojanja bijelim, sivim i crnim bojama kao i BFS algoritam ranije objašnjen. DFS radi tako da umjesto reda koristi stog i na stog stavlja susjede od trenutnog čvora. Krećemo od nule bojamo ju sivo. Nakon tog sa stoga popamo nulu i stavljamo njene susjede 1 i 2 i nulu bojamo crno. Sada sa stoga popamo čvor 1, bojamo ga crno i na stog stavljamo njegove susjede u ovom primjeru čvor 3. Postupak ponavljamo dok na stogu imamo čvorova.



Na kraju i čvor 2 obojamo crno i maknemo ga sa stacka.

Zadatak 2

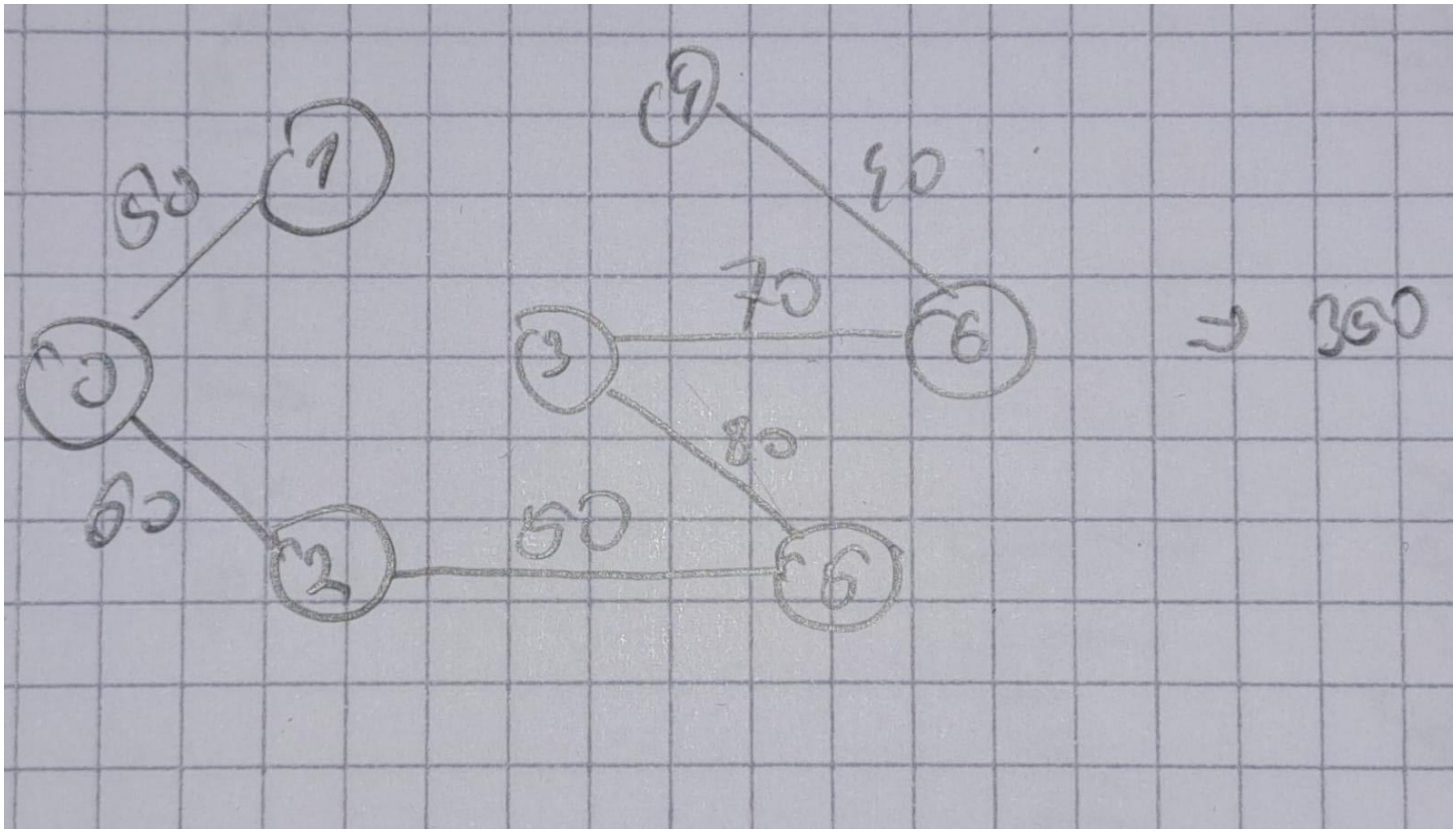
Komponente jake povezanosti možemo izračunati Kosaraju algoritmom. Prvo moramo pozvati DFS na grafu i izračunati završno vrijeme pretrage f[u] svakog čvora u grafu. Nakon toga moramo izračunati transponirani graf početnog grafa (zamjenjujemo usmjerenje strelica u grafu). Nakon toga pozivamo DFS na transponirani graf prema padajućem poretku od f[u]. Nakon toga trebamo vratiti vrhove svakog stabla u DFS šumi kao SCC komponente. Koristimo stack u koji pushamo sve čvorove u kojima je završena obrada vrhova, te nakon konstrukcije transponiranog grafa pozivamo DFS na transponiram grafu u poretku čvorova na vrhu stacka i popamo ih sa stacka. Ako čvor nema susjede, DFS za taj čvor staje i on čini jedan SCC. Isto tako vrijedi i za više čvorova, odnosno svi posjećeni čvorovi u jednom prolasku DFS-a čine jedan SCC.



Zadatak 3

Primov algoritam

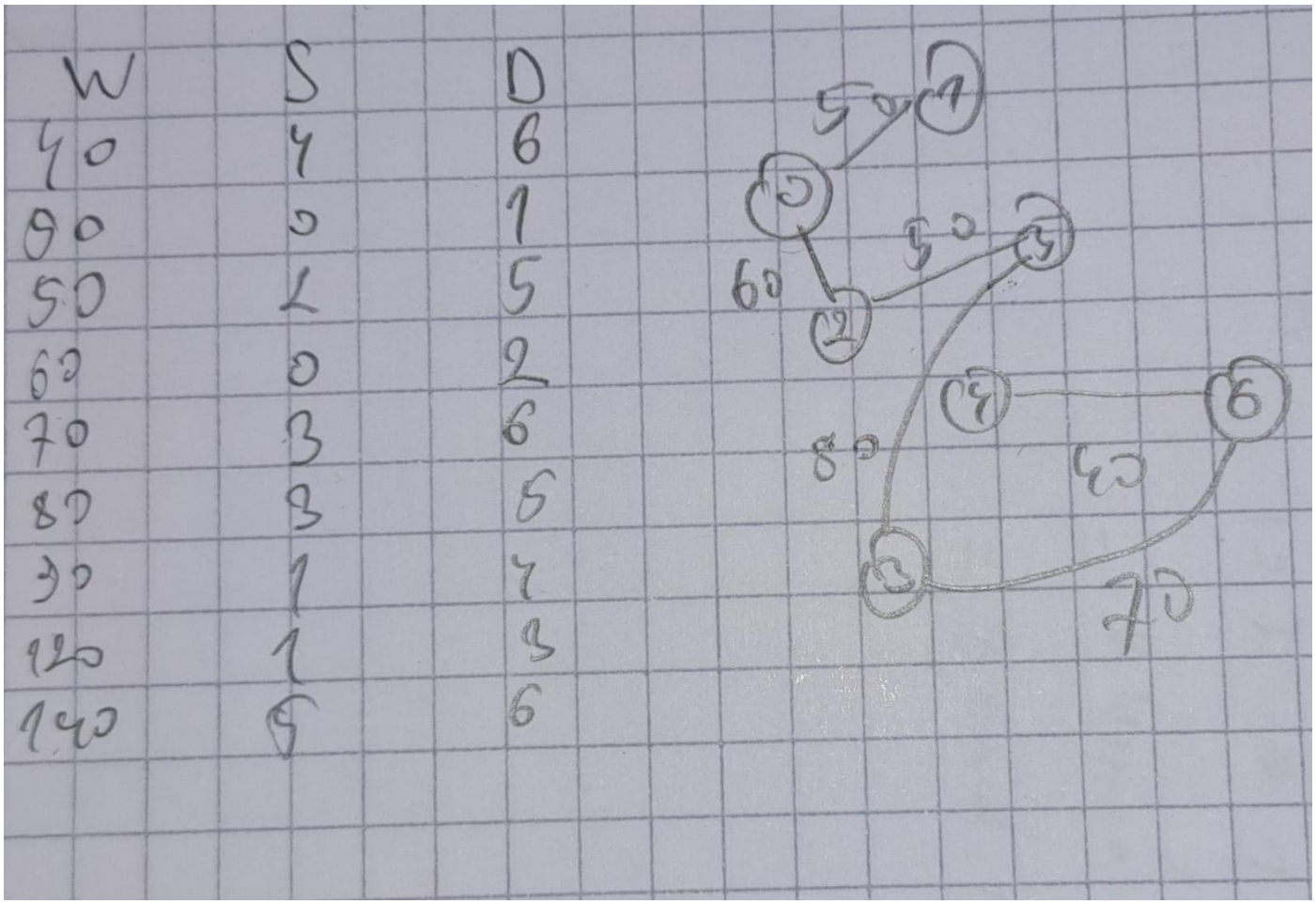
Algoritam počinje s praznim razapinjućim stablom. Ideja je održati dva skupa vrhova. Prvi skup sadrži vrhove koji su već uključeni u MST, a drugi skup sadrži vrhove koji nisu uključeni. U svakom koraku razmatra sve rubove koji povezuju dva skupa i odabire rub minimalne težine s tih rubova. Nakon odabira ruba, pomiče drugu krajnju točku ruba u skup koji sadrži MST. Ovo je primjer greedy algoritma jer uvijek uzima čvor s najmanjom težinom.



Za dani graf krećemo od čvora 0. Imamo dva ruba težine 50 prema čvoru 1 i 60 prema čvoru 2. Uzimamo manji rub koji spaja 0 i 1 i stavljamo ga u MST. Sada biramo između rubova težina 90, 120 i 60. Uzimamo rub 60 koji spaja čvor 0 sa čvor 2. Sada biramo između 90, 120 i 50. Biramo 50 i stavljamo krajnji čvor 5 u MST. Sada uzimamo opet minimum 80 i stavljamo čvor 3 u MST. Nakon tog biramo između bridova 90, 120 i 70 te uzimamo 70 i stavljamo čvor 6 i za kraj uzimamo težinu brida 40 i stavljamo čvor 4 u MST. Sada imamo sve čvorove u MST i algoritam je gotov. Ukupna težina MST-a je 350.

Kruskal algoritam

Kruskalov algoritam radi tako da prvo sortiramo bridove po težini u rastućem poretku. Nakon toga dodajemo bridove u MST samo ako ti bridovi ne formiraju ciklus unutar MST-a. Ovo je kao i Primov algoritam primjer greedy algoritma. Za provjerju nastanka ciklusa dodavanjem bridova koristi se funkcija FIND_SET, a spajanje se vrši operacijom UNION.



Tablica sa slike prikazuje sortirane bridove u rastućem poretku. S označava source, a D destination. Za prvi brid odabiremo najmanji i spajamo čvor 4 sa čvorom 6. Nakon toga soajamo čvor 0 sa čvorom 1, pa nakont toga 2 sa 5, nakon toga 3 sa 6, no sada smo spojili i 3 sa 4 pa imamo vezu 3-6-4. Nakon toga spajamo 3 sa 5, to isto možemo jer se ne pojavljuje nigdje ciklus, i na kraju dodajemo brid težine 90 i završavamo MST. MST nam je kompletan svi su čvorovi spojeni i vrijedi relacija [broj bridova = broj čvorova -1], ako bismo dodavali iduće rubove stvarali bi nam se ciklusi što isto tako nebi odgovaralo MST strukturi.

Vidimo i iz skice i iz programerskog dijela da su nam oba algortima dala isto MST stablo ukupne težine 350.