Izvještaj

Zadatak ovog projekta je bio implementirati klasu „Graf“ koja pruža podršku za rad sa strukturom podataka nazvanom graf. Preciznije rečeno, potrebno je bilo uzeti implementaciju postavljenu na courseware-u i nju dodati jednu od metoda po našem izboru: metodu koja pronalazi Eulerov ciklus (ako postoji) ili metodu za topološko soritranje grafa. Mi smo se odlučili za metodu za topološko sortiranje grafa. Pretražujući raznu literaturu, pronašli smo algoritam koji je dovoljno jednostavan ali i efikasan, koji vrši topološko sortiranje grafa. Pseudokod algoritma glasi:

```
L ← Empty list that will contain the sorted elements
S ← Set of all nodes with no incoming edges
while S is non-empty do
    remove a node n from S
    insert n into L
    for each node m with an edge e from n to m do
        remove edge e from the graph
        if m has no other incoming edges then
            insert m into S
if graph has edges then
    return error (graph has at least one cycle)
else
    return L (a topologically sorted order)
```

U implementacije klase „Graf“ preuzete sa corseware-a, graf je reprezentovan pomoću matrice povezanosti, te je ta matrica iskorištena i za ovaj algoritam. Na početku algoritma dinamički alociramo novi vektor koji će nam ujedno biti i rezultat metode. Na početku rada, vektor je prazan. Još, pravimo praznu listu, tj. S iz prethodnog algoritma. Nakon toga vršimo početnu obradu. Pod početnom obradom se podrazumijeva da za svaki čvor izračunamo njegov ulazni stepen. Ulazni stepen predstavlja broj grana koje ulaze u dati čvor. Nakon što to uradimo, u listu S stavimo sve čvorove čiji je ulazni stepen jednak nuli. Algoritam se nastavlja sve dok lista S nije prazna. U jednoj iteraciji algoritma uzmemo proizvoljan čvor iz liste. Pošto možemo uzeti proizvoljan čvor, možemo uzeti i prvi čvor, što se za liste radi u kontantnom vremenu. Nakon toga, oznaku tog čvora stavljamo u vektor koji predstavljar

rezultat metode. Algoritam nastavlja sa brisanjem svake ivice koja vodi iz izabranog čvora u bilo koji drugi čvor. Nakon što se ivica obriše, potrebno je promijeniti i izlazni stepen čvora. Nakon toga, mora se još provjeriti da li je izlazni stepen jednak nuli, tj. da li ima grana koje vode u taj susjedni čvor. Ukoliko nema, taj čvor se ubacuje u listu S, te se time završava jedna iteracija algoritma. Nakon što smo došli u situaciju da je lista S prazna, tj. da više nemamo čvorova za ispitivanje, algoritam je skoro gotov sa radom. Potrebno je još samo testirati da li smo obrisali sve ivice. Ukoliko jesmo, ti znači da je graf bilo moguće sortirati, tj. da nismo imali ciklusa. U tom slučaju kao rezultat iz metode treba vratiti pokazivač na vektor koji smo dinamički alocirali, a koji sadrži topološki sortiran poredak čvorova. Ukoliko je ostalo još ivica, to znači da je u grafu bilo ciklusa, što uzrokuje da je graf bilo nemoguće topološki sortirati. U tom slučaju se kao rezultat metode vraća nul-pokazivač. To je zgodno, jer kada koristimo implementiranu metodu „toploskoSortiranje“ i njen rezultat dodijelimo nekom pokazivaču, postaje prilično trivijalno testirati da li je bilo moguće topološki sortirati graf. Bilo je moguće ako i samo ako je vrijednost pokazivača različita od nule.