Fakultet organizacionih nauka
Alati i metode softverskog inženjerstva
Januar 2020. godine

Implementacija algoritma za generisanje i rešavanje lavirinta u programskom jeziku Clojure

Cilj ovog rada je primena jednog od algoritama koji su pogodni za generisanje i rešavanje lavirinta korišćenjem programskog jezika Clojure. Izabran je *Depth-first search* (DFS) algoritam koji predstavlja algoritam za pretraživanje grafa po dubini. Naime, algoritam sledi put koji se od početnog čvora udaljava u jednom pravcu što je više moguće. Kada se dalje tim putem ne mogu više posećivati čvorovi, algoritam se vraća na neki čvor na pređenom putu i sledi neki drugi put kroz graf i tako dok se ne posete svi čvorovi. Obilazak počinje iz nekog zadatog čvora *v.* Posle početnog čvora se posećuje i označi jedan njegov sused, a onda jedan neposećeni susedov sused, itd. Ovako se ide u jednom pravcu sve dok se ne dodje do čvora koji nema suseda ili čiji su svi susedi već posećeni. Kada se ovaj put više ne može produžiti, algoritam se vraća na poslednjeg posećenog prethodnika koji ima neposećenih suseda, pa počinje da sledi novi put od tog čvora prema nekom neposećenom susedu. Algoritam se završava kada više nema neposećenih čvorova koji su dostižni iz već posećenih čvorova. Strategija algoritma ga čini veoma pogodnim za rekurzivnu realizaciju, te je u ovom projektu na takav način i implementiran.

Lavirint koji se generiše predstavljen je grafom matrične reprezentacije. U okviru funkcije *initialize* generiše se matrica susednosti ovog grafa, gde jedinica predstavlja to da su čvorovi *i* i *j* povezani, dok nula govori da nisu. U okviru ove funkcije svaki čvor se “povezuje” sa svim svojim susedima kojih će biti dva, tri ili četiri u zavisnosti od toga koji čvor je u pitanju, i ova povezanost zapravo predstavlja zid u lavirintu. Takodje, generiše se i vektor čija je dimenzija jednaka broju čvorova i u njega se upisuje koje čvorove je algoritam do određenog trenutka posetio. Na indeksu posećenog čvora u vektoru se upisuje jedinica.

U nastavku se poziva funkcija *dfs* koja implementira pomenuti algoritam, odnosno “ruši zidove” izmedju čvorova postavljajući nule na odgovarajuća mesta u matrici susednosti. Pri tome, koristi se i vektor *shuffled* koji služi za izdvajanje suseda čvora u kome se algoritam trenutno nalazi i njihovo mešanje. Ovaj korak je ubačen kako bi se izbegla posledica težnje algoritma da pravi dugačke prolaze u lavirintu, s obzirom da je algoritam takav da ide u dubinu grafa koliko god je to moguće.

Za pronalazak puta za izlazak iz lavirinta korišćen je isti algoritam, samo je u ovom slučaju posmatran graf sastavljen od povezanih čvorova u lavirintu. Takođe, pri ovom načinu implementacije moguće je da algoritam neće obići sve čvorove jer mu je zadato da se zaustavi kada stigne do izlaza iz lavirinta, što i jeste cilj u ovom slučaju. Ipak, kada algoritam stigne do poslednjeg čvora u grafu, odnosno izlaza, funkcija će se rekurzivno pozvati za sve neposećene susede posećenih čvorora. U svrhu izbacivanja ovih čvorova iz puta za izlazak iz lavirinta, napisana je funkcija *prune*.