Ema Mandal

emamandal@hotmail.com

Prirodno-matematički fakultet

STRUKTURE PODATAKA I ALGORITMI

Projekat br. 3:Algoritam 1

# IDEJA

Traženu funkciju za izračunavanje udaljenosti između dva proslijeđena zida sam implementirala unutar klase Labirint. Klasa sadrži matricu “labirint” u kojoj je zapravo smješten labirint. Nule u matrici predstavljaju prolaze dok ostali brojevi (> 0) predstavljaju zidove. Takođe imamo dvije cjelobrojne varijable “kolone” i “redovi” koje predstavljaju broj kolona i nepraznih redova respektivno. Sve vrijednosti se učitavaju iz fajla. Možemo primjetiti da matrica “labirint” ima zapravo duplo više redova od broja koji je pohranjen u varijablu “redovi”. To je zbog duplih praznih redova koji se nalaze u samoj matrici, a već je rečeno da broj spremljen u varijablu “redovi” predstavlja broj nepraznih redova.

Za implementaciju funkcije koja izračunava udaljenost korištene su dodatne metode : bool Zid(char slovo, int kolona), int Red(char slovo), pair<int, int> pocetak(char slovo, int kolona), vector<int> SviProlazi(int red), bool PrazanRed(int red), koje će biti naknadno objašnjene.

Ideja je zasnovana na tome da računamo samo udaljenosti koje će nas dovesti do cilja, a ne sve moguće udaljenosti. Prvo što trebamo je da nađemo blok ispod prvog proslijeđenog bloka koji se nalazi u redu prviZid u koloni pozicijaPrvog, te njegovu udaljenost postavimo na 1. Dalje se spuštamo do prvog nepraznog reda povećavajući udaljenost, te u njemu tražimo najbliži prolaz. Ukoliko postoji više prolaza koji su jednako udaljeni od trenutne tačke, uzimamo onaj čija je kolona bliža koloni ciljne tačke. Zatim skačemo do mjesta gdje je prolaz povećavajući udaljenost za broj blokova koji smo prešli. Sada nam ostaje da se spustimo kroz prolaz u sljedeća dva reda pa to i uradimo i povećamo udaljenost za 2. Ponavljamo postupak sve dok ne dođemo do ciljne tačke.

# FUNKCIJE

Labirint(string text) – konstruktor sa jednim parametrom koji predstavlja ime fajla iz kojeg se učitavaju podaci potrebni za konstruisanje matrice „labirint“. Prvo se iz fajla prčita broj redova i kolona te smjesti u varijable kolone i redovi redom. Zatim u matricu labirint dodajemo prvi red koji sadrži sve kolone. Nakon njega slijede dva prazna reda. Zatim se čita sljedeći red iz fajla koji sadrži indekse mjesta na kojima se nalaze prolazi. Mjesta sa prolazima se stave na vrijednost 0 a ostali se indeksiraju standardno od 1 do broja kolona. Takva dva reda se dodaju u matricu, a zatim se dodaju dva prazna reda. Postupak se ponavlja dok ne dođemo do kraja fajla. Na kraju dodajemo posljednji red koji je isti kao i prvi.

bool Zid(char slovo, int kolona) – metoda koja prima poziciju i provjerava da li je na toj poziciji zid ili prolaz, tako što provjerava da li je na toj poziciji u matrici labirint smješten broj 0 ili broj različit od 0. U slučaju da je 0, funkcija vraća false, a u suprotnom true.

int Red(char slovo) – metoda koja određuje red proslijeđenog slova. Za parna slova oduzmemo ASCII kodove proslijeđenog slova i slova ‘A’ i taj broj pomnožimo sa 2, a u suprotnom oduzimamo ASCII kodove proslijeđenog slova i slova ‘B’ te taj broj pomnožimo sa 2 i dodamo na njega 3 (zbog praznih redova).

pair<int, int> pocetak(char slovo, int kolona) – pomoću ove metode određujemo slobodnu poziciju od koje možemo početi računati udaljenost. Ta pozicija je ispod proslijeđene pozicije ako slovo ima neparan ASCII kod, a u suprotnom je iznad proslijeđene pozicije.

vector<int> SviProlazi(int red) – metoda koja vraća vektor pozicija svih prolaza u proslijeđenom redu.

bool PrazanRed(int red) – služi za provjeru da li je proslijeđeni red prazan ili ne. Prolazi kroz proslijeđeni red i ukoliko naiđe na nenulti elemenat (ukazuje na zid) vraća false, a u suprotnom vraća true.

int udaljenost (char prviZid, int pozicijaPrvog, char drugiZid, int pozicijaDrugog) – prvo u varijable start i cilj spremimo početnu i krajnu tačku (tj tačke ispod/iznad proslijeđenih). Zatim formiramo matricu udaljenosti u koju ćemo spašavati udaljenosti koje su nam potrebne. Udaljenost tačke start postavimo na 1. Pomoću varijabli trenutniRed i trenutnaKolona ćemo se pomijerati kroz labirint. Koristimo while petlju da bismo ponavljali iteracije dok ne dođemo do ciljne tačke, kada izlazimo iz while petlje. Udaljenost se postavlja u zavisnosti od vrijednosti trenutnog reda i trenutne kolone. Imamo 4 mogućnosti:

1. Ako je trenutni red prazan, ali prethodni red nije prazan, tada postavljamo trenutnu udaljenost na vrijednost udaljenosti tačke iznad (gdje je bio prolaz) uvećanu za 1.
2. Ako su i trenutni i prethodni red prazni, to znači da moramo doći do kolone u kojoj se nalazi prolaz u sljedeća dva reda. To radimo tako što nađemo indekse svih prolaza u sljedećem redu i tražimo onaj najbliži. Ukoliko postoji više prolaza koji su jednako udaljeni od trenutne tačke onda biramo onaj koji je po kolonama najbliži ciljnoj tački. Kada nađemo kolonu u kojoj je najbliži prolaz, udaljenost te kolone u trenutnom redu postavljamo na vrijednost udaljenosti tačke u kojoj smo trenutno + razlika trenutne kolone i kolone prolaza. Trenutnu kolonu postavljamo na vrijednost kolone sa prolazom i trenutni red povećavamo za 1.
3. Ako trenutni red nije prazan, a prethodni jeste, tada postavljamo udaljenost sljedećeg reda i trenutne kolone na udaljenost prethodnog reda i te kolone uvećanu za dva i trenutni red povećavamo za dva.
4. Ako smo došli do ciljnog reda tada tražimo ciljnu kolonu i udaljenost ciljnog mjesta postavljamo na udaljenost trenutnog mjesta uvećanu za broj kolona kojima smo se pomjerili da bismo došli do ciljne.

Na kraju vraćamo udaljenost ciljnog reda i ciljne kolone.

void Ispis() – ispisuje labirint tako što ide kroz matricu i ispisuje joj redove i kolone. Kada naiđe na 0 ispisuje prazno mjesto.

# VRIJEME IZVRŠAVANJA

Označimo sa ***n*** ukupan broj redova, a sa ***m*** broj kolona. Vidimo da se funkcije Zid, Red i pocetak izvršavaju u konstantnom vremenu. Fukncije SviProlazi i PrazanRed imaju vrijeme izvršavanja ***O(m)***, zbog for petlje koja prolazi svim kolonama proslijeđenog reda, dok funkcija za ispis uzima ***Θ(n\*m)*** vremena.

Postamtrajmo sada našu glavnu funkciju, onu koja računa udaljenosti. Na početku imamo jednu for petlju koja se izvršava u vremenu ***Θ(n)***. Zatim slijedi while petlja koja se izvršava otprilike n/2 puta jer se oba neprazna reda posmatraju u jednoj iteraciji. Međutim unutar petlje pozivamo i funkcije SviProlazi i PrazanRed koje se izvršavaju u vremenu ***O(m)***, pa možemo zaključiti da je ukupno vrijeme izvršavanja while petlje ***O(n\*m)*** što takođe predstavlja i vrijeme izvršavanja funkcije udaljenosti.