**GPS**

*Projekt iz kolegija Računarski praktikum 3*

Sandro Lovnički, Ivan Čeh, Sebastijan Horvat

0. Kratak opis aplikacije

Aplikacija ima plan nekog grada. U plan se dodaju čvorovi koji predstavljaju križanja ili trgove i linije koje ih povezuju i koje predstavljaju ulice. Novi čvorovi se dodaju preko koordinata. Čvorovi i linije imaju neke karakteristike (trgovina, pošta, benzinska, garaža i sl.). Detalji o ulici ili nekom objektu se pamte u bazi i mogu na zahtjev korisnika biti prikazani u jednom dijelu prozora. Korisnik može unositi odredište i dobiti plan puta na temelju različitih kriterija poput najkraćeg puta ili puta u kojem može obaviti i neki posao (otići u neki tip trgovine, poštu, itd.). Program iscrtava što idealniju putanju.

1. Opis funkcionalnosti

(…znači popis: npr. korisnik može klikom na gumb *Sviraj nešto* pokrenuti glazbu u pozadini aplikacije koja svira sve dok korisnik ponovo ne pritisne na isti gumb, koji je sada promijenio naziv u *Zaustavi glazbu*, a može se iskoristiti i tipkovni prečac, pritiskom na kombinaciju tipaka …+…; gumb se nalazi u padajućem izborniku koji se dobiva pritiskom na *Personaliziraj*, a glazbeni broj koji se izvodi se naziva …)

predviđeno – od 3 do 4 stranice ukupno (ali sve funkcionalnosti moraju biti popisane (na objašnjenu se, **ako je nužno**, može štediti)

2. Implementacija

(znači – plan grada je klasa …, čvorov i bridovi su klase…., kako ih dodajemo…, algoritmi traženja puteva, komunikacija s bazom,…)

predviđeno – 4 do 5 stranica, ne mora biti cijeli kod, samo najbitniji dijelovi, **mora biti popraćeno dijelovima koda** (ali ne nužno čitavim kodom – samo najbitnijim dijelovima – treba li detaljno objašnjavati konstruktor za čvor?)

2.x. Traženje puteva

**Traženje najkraćeg puta**

Najkraći put od čvora A do čvora B tražimo Dijkstrinim algoritmom koji se temelji na re-evaluaciji najkraćeg puta od vrha A do trenutnog vrha B' i to prolaskom po svih izlaznim bridovima iz trenutnog vrha, krećući iz vrha A.

Algoritam kreće s pretpostavkom da su svi čvorovi beskonačno udaljeni (što spremamo kao Double.MaxValue) od čvora A, osim A koji je udaljen 0 od samog sebe. Također, za nijedan čvor ne postoji brid prethodnik (brid kojim smo došli do čvora) što označujemo s null.

Algoritam prati koje čvorove još nije prošao, i to po prioritetu, tako da je sljedeći najbliži čvor prvi. To spremamo u SortedSet<Čvor>, gdje smo u klasi Čvor prilagodili funkciju CompareTo.

Općenito, Dijkstrin algoritam staje ili kad je skup neprođenih čvorova prazan (što bi značilo da čvor B ne postoji, a ta mogućnost u našoj aplikaciji nije moguća) ili kad naiđe na završni vrh B te u tom trenutku imamo označenu vezanu listu bridova koji vode unatrag iz B u A, tj. svaki čvor "zna" kojim bridom smo došli do njega.

Funkcija koju pozivamo je

**public** void nadiPutIzmedu**(**Čvor c1**,** Čvor c2**)**

**1. korak:** ispraznimo skup neprođenih čvorova te ga napunimo svih čvorovima iz Plana po pravilu opisanom gore.

neprođeniČvorovi**.**Clear**();**

**foreach** **(**Čvor cv **in** Plan**.**skupČvorova**.**Values**)**

**{**

cv**.**udaljenostOdPocetka **=** Double**.**MaxValue**;**

cv**.**bridPrethodnik **=** **null;**

**if** **(**cv**.**naziv **==** c1**.**naziv**)**

**{**

cv**.**udaljenostOdPocetka **=** 0**;**

**}**

neprođeniČvorovi**.**Add**(**cv**);**

**}**

**2. korak:** zatim krećemo s glavnim dijelom algoritma. Dok god ima neprođenih čvorova, uzmemo prvi po prioritetu i nazovemo ga trenutni. Ako je taj čvor upravo traženi čvor c2, gotovi smo i našli smo najkraći put **od** c1 do c2. Ako pak ne, krećemo se po svim susjednim bridovima trenutno čvora, računamo zbroj udaljenosti od c1 do trenutnog čvora i duljine trenutnog brida (udaljenost drugog kraja tog brida od početka puta - čvora c1), te ako je ta udaljenosti manja od dosadašnje udaljenosti tog čvora do početka, zamijenimo ju te upišemo novi, bolji, brid kojim smo došli do tog čvora. Na kraju, brišemo trenutni čvor iz neprođenih čvorova.

**while(**neprođeniČvorovi**.**Count **!=** 0**)**

**{**

Console**.**WriteLine**(**"NEPROĐENI ČVOROVI:"**);**

**foreach** **(**Čvor cv **in** neprođeniČvorovi**)**

**{**

Console**.**WriteLine**(**cv**);**

**}**

Čvor trenutni **=** neprođeniČvorovi**.**First**();**

Console**.**WriteLine**(**"trenutni cvor: " **+** trenutni**);**

**if** **(**trenutni**.**naziv **==** c2**.**naziv**)**

**{**

Console**.**WriteLine**(**"KRAJ"**);**

**break;**

**}**

**foreach** **(**Brid susjedniBrid **in** trenutni**.**skupSusjednihBridova**)**

**{**

Console**.**WriteLine**(**"susjed od " **+** trenutni **+** " -> " **+** susjedniBrid**);**

double alt **=** trenutni**.**udaljenostOdPocetka **+** susjedniBrid**.**VrijemeProlaska**;**

Čvor susjedniČvor **=** susjedniBrid**.**PočetniČvor **==** trenutni **?** susjedniBrid**.**ZavršniČvor **:** susjedniBrid**.**PočetniČvor**;**

**if** **(**alt **<** susjedniČvor**.**udaljenostOdPocetka**)**

**{**

neprođeniČvorovi**.**Remove**(**susjedniČvor**);**

susjedniČvor**.**udaljenostOdPocetka **=** alt**;**

susjedniČvor**.**bridPrethodnik **=** susjedniBrid**;**

neprođeniČvorovi**.**Add**(**susjedniČvor**);**

**}**

**}**

neprođeniČvorovi**.**Remove**(**trenutni**);**

**}**

**Traženje najkraćeg puta uz posao usput**

Najkraći put od čvora A do čvora C uz posao B usput, tražimo primjenom gornjeg algoritma prvo na (A,B) pa zatim na (B,C).

3. Prikaz sučelja

stranica dvije, dvije do tri slike po stranici, slike bi trebale biti različite i na njima bi trebali biti različiti prikazi što se gdje, kako otvorilo (pritiskom na koji gumb). Vjerojatno ne treba biti cijelo sučelje na slici ako se promijenio jedan gumb, nego samo izrezana slika promjene tog gumba.