Fakultet elektrotehnike i računarstva

Zavod za primjenjeno računarstvo

**Napredni algoritmi i strukture podataka**

1. laboratorijska vježba

Tyrannizer, 10563

Zagreb, 11. 1. 2017.

# Zadatak

Zadatak za 11 bodova

2) Modelirati grafom dio nekog naselja i programski odrediti najkraći put između dva mjesta (dva vrha). Tko želi, može modelirati i nešto drugo, gdje bi bilo čak i bridova negativnih težina.

- za kolokviranje vježbe važno je pregledno i jasno opisati model te organizaciju podataka u programu

- nije potrebno graditi komplicirane modele, dovoljni su grafovi s 10...20 vrhova. Naravno, kompliciraniji modeli će vjerojatno biti i izazovniji te kao takvi zanimljivija i „plodonosnija“ laboratorijska vježba.

- program mora riješiti pohranu grafa u kompjutoru, pronalaženje najkraćeg puta i ispis (iscrtavanje) rješenja

- iscrtavanje grafa i najkraćeg puta nije obavezno, nego samo poželjno, ali prikladan ispis najkraćeg puta je obavezan

- iscrtavanje se brzo i relativno lako može postići prepuštanjem tog posla slobodnom (open source) programu Graphviz koji možete preuzeti sa stranice http://www.graphviz.org/, gdje su i podrobne upute za njegovo korištenje. Dovoljno je iskoristiti samo njegovu osnovnu funkcionalnost, bez

posebnog dotjerivanja rješenja, a ni njegovo pozivanje ne mora biti automatsko, nego je dovoljno programski pripremiti podatke za Graphviz, a pozivati ga možete i „ručno“ iz komandne linije.

# Rješenje zadatka

## Teorijski uvod

Problem težine najkraćeg puta je pronalazak najkraćeg puta u usmjerenom grafu s težinskom funkcijom od početnog čvora do nekog drugog čvora.

Graf je definiran skupom čvorova i skupom bridova .

Težinska funkcija preslikava čvor u realan broj, to jest njegovu težinu .

Težina puta je suma svih težina u putu, definirana sljedećom funkcijom

Težina najkraćeg puta definirana je kao

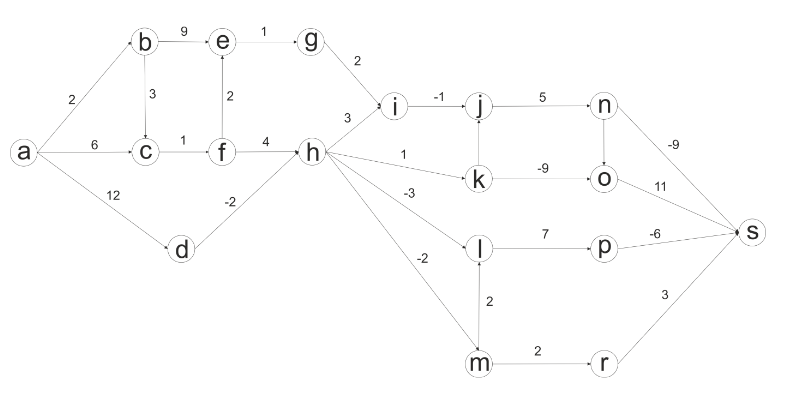
Najčešće korišteni algoritmi za problem težine najkraćeg puta su Dijkstrin algoritam i Bellman-Fordov algoritam. Ograničenje Dijkstrinog algoritma je da graf ne smije imati negativne težine bridova. Ograničenje Bellman-Fordovog algoritma je da graf ne smije imati negativan ciklus, međutim u tom slučaju najkraći put ni ne postoji, jer uvijek možemo dobiti manju težinu puta još jednim prolaskom kroz ciklus.

## Implementacija

Implementacija je napisana u jeziku C++ koristeći razvojno okruženje Visual Studio 15, za implementaciju nije korištena nijedna vanjska biblioteka već isključivo standardna biblioteka jezika C++. Implementiran je konzolni program koji se pokreće iz naredbene linije. Program očekuje jedan ulazni parametar koji označava ime ulazne datoteke iz koje čita graf.

### Graf

Ulazna datoteka za graf sadrži samo popis bridova u obliku “izlazni\_cvor ulazni\_cvor težina”, npr. “G I 12” za definiranje brida koji izlazi Iz čvora G u čvor I s težinom 12. Uz implementaciju je prilozena datoteka “graph.txt” koja sadrži definiciju grafa danog u uputama labaratorijske vježbe.



U implementaciji graf je reprezentiran skupom čvorova i listom bridova.

### Traženje najkraćeg puta

Budući da graf sadrži negativne težine ne može se koristiti Djikstrin algoritam, već je implementiran Bellman-Fordov algoritam, pseudokod ovog algoritma dan je u nastavku.

Bellman-Ford(graph, source)

distance = map

predecessor = map

for v in graph.vertices:

distances[v] = inf

predecessor[v] = null

distances[source] = 0

for i = 0 to |graph.vertices| - 1:

for e(u, v, w) in graph.edges:

if distance[u] + w < distance[v]:

distance[v] := distance[u] + w

predecessor[v] := u

for e(from, to, weight) in graph.edges:

if distance[u] + w < distance[v]:

error "Graph contains a negative-weight cycle"

return distance, predecessor

Na početku algoritma se provodi inicijalizacija struktura podataka koje se koriste, a to su mapa udaljenosti po čvorovima i mapa prethodnog čvora za taj čvor, te se udaljenost za početni čvor postavlja na 0. Nakon toga se puta provjeravaju svi bridovi I ažuriraju udaljenosti, te se na kraju provjerava da li graf ima negativnih ciklusa i u tom slučaju javlja greška, ovu provjeru nije obavezno provoditi.

Za inicijalizaciju struktura je potrebno vremena, a nakon toga ima provjera bridova za što je potrebno vremena, pa je ukupna složenost algoritma .

Ograničenje Bellman-Fordovog algoritma je da graf ne smije imati negativne cikluse, ali u tom slučaju najkraći put ni ne postoji, jer se uvijek može dobiti manja udaljenost prolaskom još jednom kroz taj ciklus.

Nakon što je pronađen najkraći put na zaslon se ispisuju rezultati za svaki čvor grafa koji je moguće dohvatiti iz zadanog početnog čvora. Ispis sadrži ukupnu težinu puta i čvorove kroz koje taj put prolazi. U nastavku je dan ispis za priloženi graf i početnu točku A.

A-B (2): A B

A-C (5): A B C

A-D (12): A D

A-E (8): A B C F E

A-F (6): A B C F

A-G (9): A B C F E G

A-H (10): A D H

A-I (11): A B C F E G I

A-J (10): A B C F E G I J

A-K (11): A D H K

A-L (7): A D H L

A-M (8): A D H M

A-N (15): A B C F E G I J N

A-O (2): A D H K O

A-P (14): A D H L P

A-R (10): A D H M R

A-S (6): A B C F E G I J N S

# Zaključak

Bellman-Fordov algoritam dobro rješava problem najkraćeg puta, moguće poboljšanje originalnog algoritma je da se provjeravaju samo vrhovi kojima se u prethodnoj iteraciji promijenila udaljenost budući da se osim njima jedino udaljenost mogla promijeniti njihovim susjedima.

# Literatura

Tablica . Literatura

|  |  |
| --- | --- |
| **Vrsta** | **Format** |
| Poglavlje knjige | Cormen, et. al.; Single-Source Shortest Paths, Introduction to Algorithms, MIT Press, 2009, 643 - 683 |
| Web stranica | Bellman-Ford algorithm <https://en.wikipedia.org/wiki/Bellman–Ford_algorithm>, 11.1.2017. |
|  |  |
|  |  |