Fakultet elektrotehnike i računarstva
Zavod za primjenjeno računarstvo

Napredni algoritmi i strukture podataka

3. laboratorijska vježba

Filip Kujundžić 0036479155

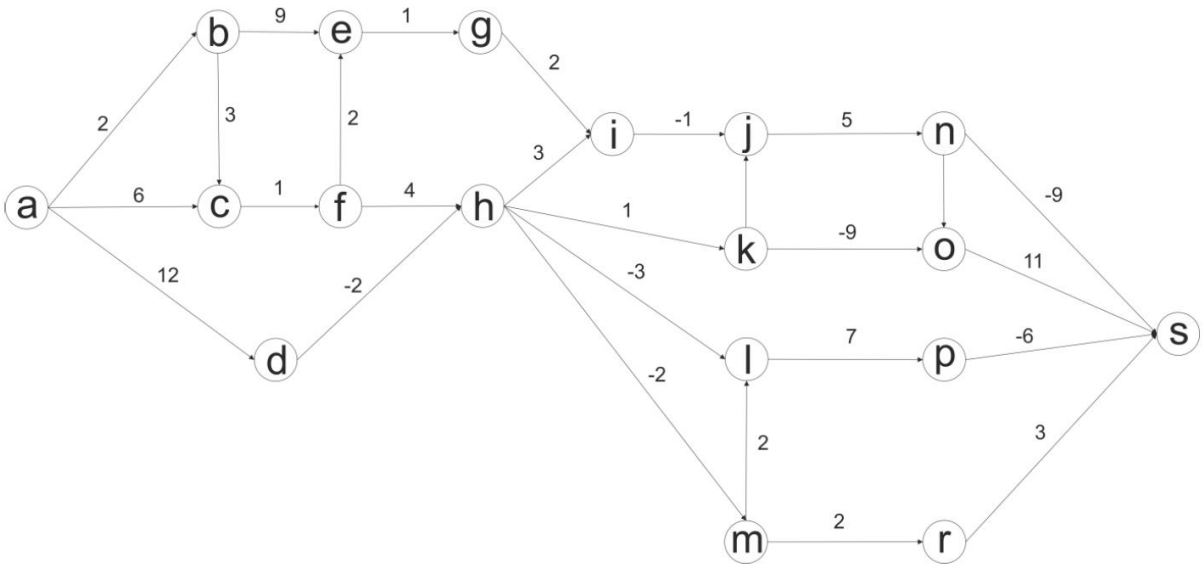
Zagreb, 18.1.2020.

1. Zadatak

Zadatci za 11 bodova

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.
- tko ne zna što modelirati, može raditi s grafom na slici



2. Rješenje zadatka

2.1. Teorijski uvod

Pitanje na kojem se temelji problem riješen u laboratorijskoj vježbi je: „Kako iz jednog vrha grafa doći najkraćim putem do drugog vrha grafa“? Problem rješavamo na jednostavnom usmjerenom grafu, grafu koji između svaka dva vrha ima smjer, najviše jedan brid i u kojem nema pelji. Bellman – Fordov algoritam je primijenjen u laboratorijskoj vježbi za rješavnje opisanog problema. Algoritam je pogodan i kada u grafu postoje bridovi s negativnom težinom, no ne podržava negativne cikluse.

2.2. Implementacija

Laboratorijska vježba je pisana u programskom jeziku Python (verzija 3.7.2.). Korištena je programska knjižnica collections.

2.2.1. Razred Graph

Razred kojim modeliramo graf u laboratorijskoj vježbi je *Graph*. Metoda *addEdge* omogućava nam da u graf dodamo brid koji povezuje dva vrha i ima određenu težinu (može biti i negativna). Konstruktor razreda *Graph* prima samo jedan prirodni broj koji predstavlja broj vrhova u grafu.[2]

2.2.2. Metoda BellmanFord

Metoda *BellmanFord* implementira Bellman – Fordov algoritam za pronalazak najkraćeg puta između dva vrha u grafu.
Pseudokod algoritma:[1]

```
initialisation: for all vertices d(v) = inf // d(v) = (source, v)
d(source) = 0;
while there is an edge(u,v) such that d(u) + edge(u,v) < current
d(v)
     d(v) = d(u) + edge(u,v);
     predecessor(v) = u;
```

Navedeni algoritam uvijek provjerava sve birdove, sve dok se makar i jedna udaljenost među vrhovima mijenja (smanjuje) tijekom obilaska vrhova. Zbog toga se najkraći put do nekog vrha može saznati tek nakon određivanja najkraćih udaljenosti do svih vrhova. Metoda sadrži i provjeru negativnih ciklusa u slučaju kojih se ispisuje poruka: „*Graph contains negative weight cycle*“. Argumenti metode su početni i krajnji vrh između kojih želimo izračunati najkraću udaljenosti, pa će, na primjer, poziv metode *BellmanFord* za najkraću udaljenost između vrhova A i H u grafu izgledati ovako:

```
g.BellmanFord(i('A'),i('H'))
```

2.2.3. Metoda printDst

Ispis najkraće udaljenosti između dva vrha u grafu ostvaruje se metodom *printDst*. Za izvršavanje metode potrebni su argumenti *dist* (udaljenost svih vrhova od početnog vrha), *src* (početni vrh) i *dest* (odredišni vrh). Metoda se poziva nakon izvršenja metode *BellmanFord* kojom smo izračunali udaljenost svih vrhova (*dist*).

2.2.4. Metoda print_path

Algoritam Bellman – Ford daje samo udaljenost između dva vrha u grafu, no i ne informaciju kojim smo vrhovima proši na toj putanji. Zbog toga je potrebno pamtiti prethodne vrhove kojima smo prošli pri odabiru novog vrha. Za tu svrhu koristimo listu previous koja je jedan od argumenata metode *print_path*. Ostali argumenti su početni i krajnji vrh. U metodi se radi nova lista prethodnih vrhova ciljnom vrhu. Postupak završava kad dođemo do početnog vrha. Obrnuto ispisana dobivena lista je tražena putanja.

2.2.5. Pomoćne ispisne metode

Svaki vrh grafa u algoritmu označen je brojem, 0...broj vrhova – 1. Kako bi bilo lakše postavljati bridove u grafu i pozivati samu metodu *BellmanFord*, dodane su dvije dodatne metode koje pretvaraju prirodne brojeve u slova engleske abecede i obrnuto. Metoda *i(character)* vraća ASCII kod slova umanjen za 65, pa tako uneseni vrh 'A' zapravo predstavlja vrh 0 u grafu. Metoda *get_char(number)* ima obrnuti zadatak, vraća slovo povezano s vrhom u grafu.

3. Zaključak

Postupak modeliranja problema grafom danas je prisutan ne samo u računarskoj znanosti, nego i u svakodnevnom životu. Bellman – Fordov algoritam je dobar pristup rješavanju navedenog problema, iako ostavlja mjesta i za poboljšanja. Algoritam bi se mogao unaprijediti tako da se ne moraju svaki puta obilaziti svi bridovi što bi ušedjelo puno vremena prilikom pronalaska najkraćeg puta u grafovima s velikom brojem vrhova. Također, izlazak iz potencijalnih negativnih ciklusa bez prekidanja programa znatno bi poboljšao algoritam.

4. Literatura

Vrsta	**Format**
[1] Prezentacija	Odabrani algoritmi nad grafovima; prezentacija kolegija Napredni algoritmi i strukture podataka; FER; Autori: Nikica Hlupić, Damir Kalpić, 2009
[2] Web stranica	Sandeep Jain,Shikhar Goel, Dharmesh Singh, Shubham Baranwal; https://www.geeksforgeeks.org/ ; posjećeno 10.1.2020.