ZADATAK

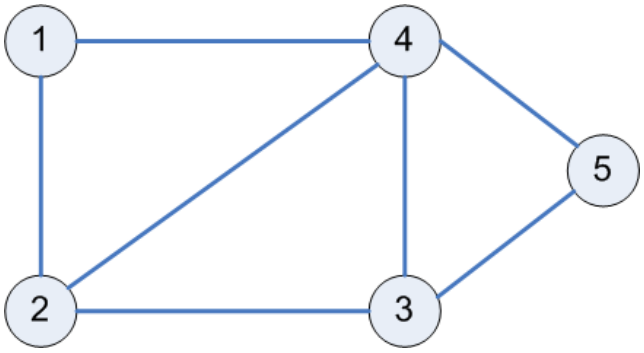
Napisati web aplikaciju koja će Dijkstrin algoritmom izračunati najkraći put, te ispisati taj put i njegovu „cijenu“.

1. TEORIJA

Graf G je uređeni par $G = (V,E)$, gdje je V neprazan skup vrhova, a E je skup bridova. Svaki brid $e \in E$ spaja dva vrha $u, v \in V$ koji se zovu krajevi od e. Brid čiji se krajevi podudaraju zove se petlja, a ako dva ili više bridova povezuju isti par vrhova, zovu se višestruki bridovi. Ako graf sadrži višestruke bridove, zove se ***multigraf***, a ako nema ni petlja ni višestrukih bridova, zove se ***jednostavan graf***. Broj vrhova u grafu označavamo s $v(G)$, a broj bridova s $e(G)$.

1.1 Reprezentacija grafova

Za algoritamsku obradu nekog grafa potreban nam je sustavan i praktičan način da prikažemo njegove vrhove i bridove. Slika grafa koji će se prikazati u primjerima:



1.1.1 Matrica susjedstva (eng. *Adjacency matrix*)

Svakom grafu s *v(G)* vrhova možemo pridružiti kvadratnu matricu M dimenzija *v(G)* × *v(G)* u čijem se *i*-tom retku i *j*-tom stupcu nalazi broj bridova koji spajaju *i*-ti i *j*-ti vrh. Takva matrica se zove *matrica susjedstva*.

	1	**2**	**3**	**4**	**5**
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

1.1.2 Matrica incidencije (eng. *Incidence matrix*)

Graf G je moguće prikazati i matricom incidencije, *v(G)× e(G)* matricom u čijem se *i*-tom retku i *j*-tom stupcu nalazi broj (0, 1 ili 2) koliko su puta vrh v_i i brid e_j incidentni.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	0	1	0	1	1	0
4	0	1	0	1	1	0	1
5	0	0	0	0	0	1	1

1.1.3 Lista susjedstva (eng. *Adjacency list*)

To su vektori od *v(G)* elemenata koji predstavljaju vrhove, gdje je *i*-tom elementu pridružena lista susjednih vrhova od vrha v_i.

```
1: 2, 4
2: 1, 3, 4
3: 2, 4, 5
4: 1, 2, 3, 5
5: 3, 4
```

1.2 Dijkstrin algoritam

Formiraju se dva skupa uređenih parova ***S*** i ***T***. Svaki uređeni par sastoji se od oznake čvora c_i i udaljenosti u_i od tog čvora do ishodišta. U koraku inicijalizacije, u skup ***S*** stavlja se ishodište, dok se svakom čvoru iz ***T*** pridružuje direktna udaljenost od ishodišta, ili ∞ ako ne postoji grana koja spaja taj čvor s ishodištem. U skup ***S*** prebacuje se element iz ***T*** s najmanjom udaljenošću u_i. Ako se uspostavi da je za neki čvor iz ***T*** udaljenost od ishodišta preko prebačenog čvora manja od njegove dotad izračunate udaljenosti, udaljenost se modificira. Postupak se ponavlja dok ima elemenata u ***T***.

Postupak se može opisati algoritamski:

$$S = \{(c_1, 0)\}$$

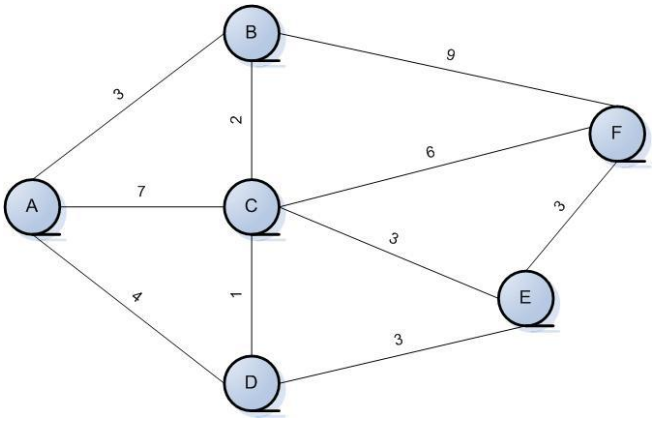
$$T = \{(c_2, u_2), (c_3, u_3), (c_4, u_4), \ldots, (c_n, u_n)\}, \quad u_k = d_{1k}, \forall k$$

dok je $T \neq \emptyset$

- izaberi $(c_k, u_k) \in T | \min_j u_j$
- $T = T \backslash \{(c_k, u_k)\}$
- $S = S \cup \{(c_k, u_k)\}$
- za svaki $(c_j, u_j) \in T$
  - $u_j = min(u_j, u_k + d_{kj})$

Primjer rješavanja:

Potrebno je pronaći najkraći put od čvora A do čvora F grafa sa slike.



U početnom koraku u skup ***S*** stavlja se ishodište, a u ***T*** ostali čvorovi s direktnim udaljenostima od ishodišta:

S = { (A, 0) }
T = { (B, 3), (C, 7), (D, 4), (E, ∞), (F, ∞) }

Čvor sa najmanjom udaljenošću je čvor B, pa ga prebacujemo u ***S***, dok za sve članove skupa ***T*** uspoređujemo udaljenost s udaljenošću preko čvora B, i izabiremo manju:

S = { (A, 0), (B, 3) }
T = { (C, min(7, 3+2)), (D, min(4, 3+∞)),
(E, min(∞, 3+∞)), (F, min(∞, 3+9)) }

```
  = { (C, 5), (D, 4), (E, ∞), (F, 12) }
```

Od članova iz ***T*** sada je najbliži ishodištu čvor D:

```
S = { (A, 0), (B, 3), (D, 4) }
T = { (C, min(5, 4+1) ), (E, min(∞, 4+3) ),
          (F, min(12, 4+∞) ) }
  = { (C, 5), (E, 7), (F, 12) }
```

Sljedeći izbor pada na C:

```
S = { (A, 0), (B, 3), (D, 4), (C, 5) }
T = { (E, min(7, 5+3) ), (F, min(12, 5+6) ) }
  = { (E, 7), (F, 11) }
```

Odabiremo E:

```
S = { (A, 0), (B, 3), (D, 4), (C, 5), (E, 7) }
T = { (F, min(11, 7+3) ) }
  = { (F, 10) }
```

F je jedini element iz ***T***, pa njegovim prebacivanjem u ***S*** postupak završava.
Udaljenost od A do F je 10 jedinica.
Slijed čvorova u najkraćem putu odredit ćemo **unatrag**, vodeći se činjenicom da je čvor *i* prethodnik čvora *j* ako je $u_j - u_i = d_{ij}$. Tako je na primjer $u_F - u_B = 7$ dok je $d_{FB} = 9$, pa zaključujemo da čvor B nije prethodnik čvora F. Slično vrijedi i za čvor C, dok za čvor E imamo $u_F - u_E = 3$, uz $d_{EF} = 3$, pa zaključujemo da je čvor E prethodnik čvora F. Takvo razmatranje treba provesti u čvoru E, i tako do ishodišta, što će kao najkraći put dati A-D-E-F.

2. RAZRADA ZADATKA

Očekujemo implementaciju korištenjem tehnologija:

- Spring boot
- Spring MVC
- Spring Security
- Spring REST
- Spring Data JPA

Za Frontend koristiti AngularJS te neki responsive CSS framework.

Na frontu je potrebno imati dva text boxa u koje se upisuju matrica incidencije (delimiter je zarez i nema razmaka) i težine bridova (brid, dvotočka, razmak težina_brida).
Nadalje potrebno je imati dva textboxa u koje se upisuje početni i završni vrh, te gumb „Izračunaj“.
Pritiskom na gumb „Izračunaj“ potrebno je ispisati „cijenu“ puta, te sam put u obliku:

- „v1 -> v2 -> v3“.

Primjer unosa matrice incidencije:

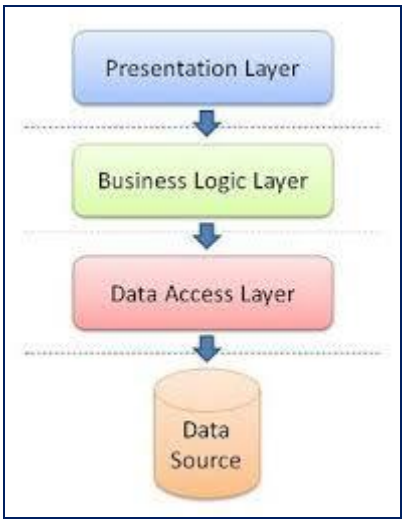
```
1,1,1,0
1,0,0,1
0,1,0,0
```

Primjer unosa težine bridova:

```
1: 3
2: 2
3: 9
4: 2
```

Unosne podatke i rezultat treba spremiti u bazu podataka (H2, „in memory“).

Nadalje, arhitektura servisne komponente mora biti troslojna bazirana na singletonima.



Pri tome valja paziti na to da tipovi objekata definirani na „višim“ slojevima ne smiju ulaziti u „niže slojeve“ već samo suprotno. Na primjer, u web aplikaciji prezentacijski sloj ne smije u poslovni sloj poslati HttpRequest objekt jer se taj objekt odnosi samo na prezentacijski sloj.

Programski kod mora imati 80%-tnu pokrivenost jUnit testovima.
Dizajn frontend komponente mora biti jednostavan i ne treba se koncentrirati na to.

Napraviti svoju implementaciju algoritma u Javi i ne smije se koristiti neki framework/library koji već to ima implementirano.

Ostale „detalje“ prepuštamo vama na volju i na izbor, pokažite sve što znate.