Podatkovne strukture in algoritmi 1
Fakulteta za Matematiko in Fiziko
5.1.2024

Domača naloga 3 - Segmenti

1 Opis

Naj bo $G = (V(G), E(G))$ neusmerjen graf z n vozlišči in m povezavami, ter utežmi na povezavah in vozliščih. Naj bo

- $f(u)$ utež za vozlišče $u \in V(G)$,
- $g(e)$ utež za povezavo $e \in E(G)$.

Za podano število k, vsako vozlišče dodelimo v natanko enega izmed k segmentov. Pri tem mora vsak segment vsebovati vsaj eno vozlišče po drugi strani pa mora biti segment povezan (torej obstaja pot znotraj segmenta med vsakim parom vozlišč v segmentu). To si lahko predstavljamo, tako da vsako vozlišče pobarvamo z eno izmed k barv, kjer pa mora biti vsaka barva uporabljena vsaj enkrat. Vozlišča, ki so pobarvana z isto barvo so torej v svojem segmentu (segmentu po angleško rečemo cluster). Označimo sedaj segmente kot $C_1, \ldots, C_k$. Sedaj definiramo razdaljo med dvema segmentoma $d(C_i, C_j)$ kot najmanjša utež na povezavi, ki povezuje vozlišče iz C_i z vozliščem iz C_j. Bolj natančno

$$d(C_i, C_j) = \min_{e=(u,v)\in E(G)} \{g(e) \mid u \in C_i,\ v \in C_j\}.$$

Če taka povezava ne obstaja si mislimo, da je $d(C_i, C_j) = \infty$. Poleg tega vsakemu segmentu dodelimo skupno utež

$$f(C_i) = \sum_{\{u,v\}\in\binom{C_i}{2}} f(u) \cdot f(v).$$

Skupna utež segmenta je torej vsota, ki za vse pare vozlišč znotraj segmenta sešteje produkte uteži.
Vaša naloga bo, da za dan k določite segmente $C_1, \ldots, C_k$, tako da bo najmanjša razdalja med segmenti čim večja. Lahko se zgodi, da graf ni povezan. Če ima torej graf strogo več kot k povezanih komponent, zgornja razdelitev v segmente ni možna.

2 Vhodni podatki

V prvi vrstici se nahajajo tri števila n, m, k. Nato sledi vrstica, ki vsebuje n števil, ki predstavljajo uteži na vozliščih. Nato sledi m vrstic oblike u, v, w, ki predstavlja povezavo od vozlišča u do v z utežjo w.
Omejitve vhodnih podatkov:

- $n \leq 10^5$

- $m \leq 10n$
- $2 \leq k \leq \frac{n}{2}$
- $f(u) \leq 10$ za vsak $u \in V(G)$.
- $g(e) \leq 1000$ za vsak $e \in E(G)$.
- Vse uteži so decimalna števila

3 Izhodni podatki

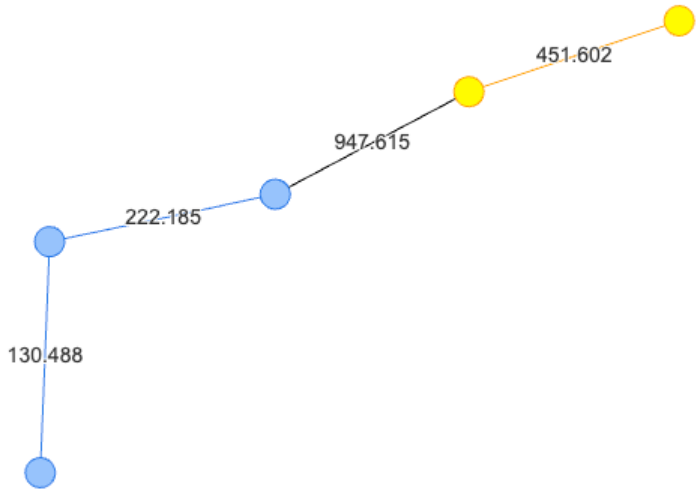
Izpišite k vrstic v vsaki pa naj bosta dve števili c, f, kjer je c velikost segmenta, f pa njegova skupna utež. Vrstice naj bodo urejene padajoče po leksikografski ureditvi, število f pa naj bo zaokroženo na 4 decimalna mesta. V kolikor razdelitev v k segmentov ni mogoča, izpišite eno vrstico z vrednostjo -1. Testni primeri naj bi bili genererani na tak način, da je razdelitev v segmente enolična.

Časovna zahtevnost: Vaš algoritem mora delovati v $O((m+n)\log m)$.

4 Primer 1

Vhod	Izhod
`5,4,2`	`3,84.6822`
`3.8,3.6,4.9,3.0,7.5`	`2,10.935`
`2,0,130.488`	`1,0`
`1,3,451.602`	
`4,1,947.615`	
`0,4,222.185`	

V tem primeru je graf kar drevo, iščemo pa dva segmenta. Če želimo, da je utež na povezavi med dvema segmentoma čim večja, moramo drevo razdeliti preko največje povezave (saj ko odstranimo eno povezavo iz drevesa, drevo razpade na natanko dve komponenti). To je primer `test00.in`.

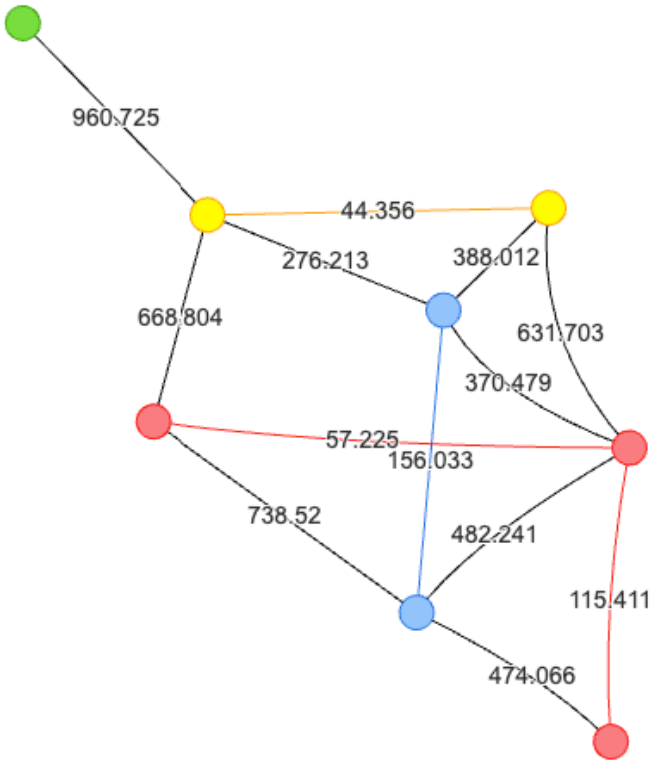


Shematični prikaz prvega primera.

5 Primer 2

Vhod	Izhod
8,13,4	3,12.0124
5.1,2.4,5.8,4.5,0.8,1.7,4.1,1.6	2,29.6665
0,1,388.012	2,11.1316
3,0,276.213	1,0
3,7,668.804	
0,2,156.033	
3,1,44.356	
6,7,57.225	
5,3,960.725	
4,2,474.066	
2,6,482.241	
2,7,738.52	
6,0,370.479	
4,6,115.411	
6,1,631.703	

V tem primeru je graf povezan, poiskati pa moramo 4 segmente. Preverimo lahko, da je najmanjša povezava med različnimi segmenti enaka 276.213 in tega ne moremo izboljšati z kakšno drugo ureditvijo.



Shematični prikaz drugega primera.

6 Dodatni napotki

Morda se bo nekaterim ta problem zdel težak in neboste imeli dobre ideje kako se lotit. Kot namig vam povem, da smo algoritem, ki to reši obravnavali tako na vajah kot predavanjih. Potreba je le manjša modifikacija. Kakšen doda

7 Testni primeri

Na voljo imate več testnih primerov ter pripadajočih izhodov. Vsak testni primer ima vhodne podatke v datoteki `testXY.in` ter izhodne podatke v `testXY.out`. Testni primeri se nahajajo v mapi `Tests`.
S pomočjo priložene skripte `test_runner.py` lahko zaženete enega, ali pa vse testne primere. Datoteko `resitev.py`, kjer ste rešili nalogo prestavite v isto mapo, kjer se nahaja `test_runner.py` ter mapa `Tests`. Skripta `test_runner.py` ima 3 možne argumente:

- `-script` (obvezen): ime oziroma pot do vaše datoteke z rešitvijo
- `-test_nb` (neobvezen): številka testa, ki naj se izvede. Če ni podano se izvedejo vsi testi.
- `-pypath` (neobvezen): Pot do Python interpreterja oz. okrajšava. Če ni podano, se vzame `python3`.

Opozorilo: Na različnih sistemih ima Python lahko različne okrajšave. Na linux in MacOS operacijskih sistemih Python ponavadi zaganjamo z `python3`, medtem ko na Windows operacijskih sistemih ponavadi z `python` (v kolikor je python dodan v okoljsko spremenljivko). V kolikor boste imeli kakšne težave me kontaktirajte.

Odprite terminal v mapi, kjer imate zgornje datoteke. Nato izvedite ukaz:

```
> python3 test_runner.py -script resitev.py -test_nb 0
```

Ob uspešno izvedenem testu se izpiše:

```
> Test test00.in uspešen.  Čas:  0.024 sekunde.
```