International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

dreaming

Slovenščina — 1.0

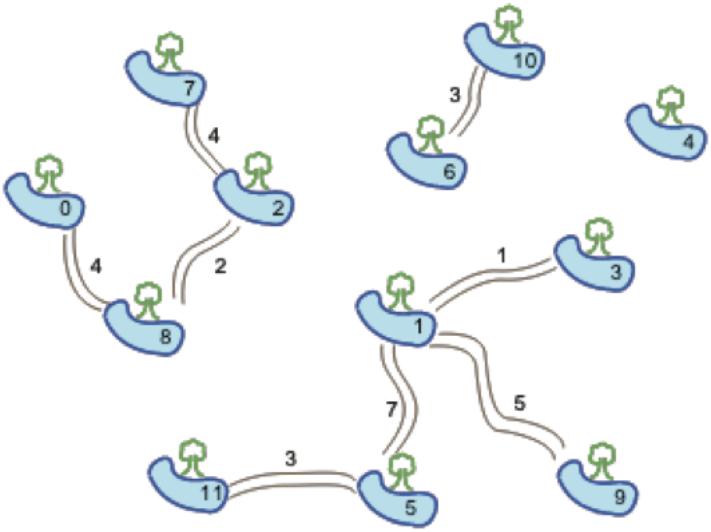
Zgodba se je odvijala pred davnimi časi, ko je bil svet še nov in o IOI še niso niti sanjali.

Kača živi v deželi, ki ima N bilabongov (mrtvih rokavov), oštevilčenih z $0, \ldots, N-1$. Obstaja M dvosmernih *povezav*, ki povezujejo pare bilabongov, po katerih lahko Kača potuje. Vsak par bilabongov je povezan (neposredno ali posredno) preko največ enega niza povezav, vendar nekateri pari morda sploh niso povezani (zato $M \leq N-1$). Vsaka povezava zahteva od Kače določeno število dni potovanja; to število je lahko za vsako povezavo različno.

Kačin prijatelj Kenguru želi narediti $N-M-1$ novih povezav, tako da lahko Kača potuje med poljubnima dvema bilabongoma. Kenguru lahko ustvari povezavo med katerimkoli parom bilabongov. Vsaka povezava, ki jo Kenguru ustvari, zahteva od Kače L dni potovanja.

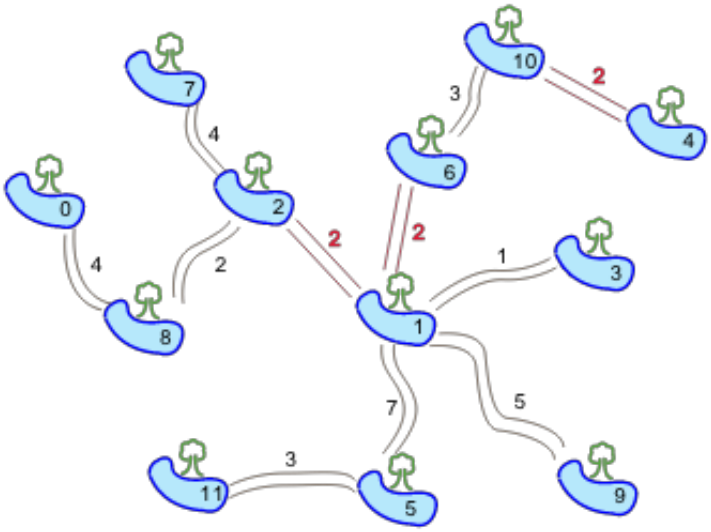
Poleg tega želi Kenguru omogočiti Kači čim hitrejša potovanja. Kenguru bo ustvaril nove povezave tako, da bo najdaljši potovalni čas med poljubnima dvema bilabongoma čim krajši. Pomagaj Kenguruju in Kači določiti najdaljši čas potovanja med poljubnima dvema bilabongoma, potem ko je Kenguru ustvaril manjkajoče povezave.

Primer



Na zgornji sliki je `N = 12` bilabongov in `M = 8` povezav. Predpostavi, da je `L = 2`, tako da vse nove povezave od Kače zahtevajo 2 dni potovanja. Denimo, da Kenguru ustvari naslednje tri nove povezave:

- med bilabongoma 1 in 2;
- med bilabongoma 1 in 6;
- med bilabongoma 4 in 10.



Zgornja slika prikazuje končno množico povezav. Najdaljši čas potovanja je 18 dni, in sicer med bilabongoma 0 in 11. Prikazan je najmanjši možen rezultat — ne glede na to, kako Kenguru ustvari povezave, bo vedno obstajal nek par bilabongov, ki bo od Kače zahteval 18 ali več dni potovanja.

Implementacija

Oddaj datoteko, v kateri je implementirana funkcija `travelTime()` po sledečih navodilih:

Tvoja funkcija: `travelTime()`

C/C++

```
int travelTime(int N, int M, int L,
        int A[], int B[], int T[]);
```

Pascal

```
function travelTime(N, M, L : LongInt;
        var A, B, T : array of LongInt) : LongInt;
```

Opis

Funkcija naj izračuja najdaljši čas potovanja (v dneh) med poljubnim parom bilabongov ob predpostavki, da je Kenguru dodal `N - M - 1` povezav na tak način, da so vsi bilabongi povezani in da je najdaljši čas potovanja najkrajši možen.

Parametri

- `N`: Število bilabongov.
- `M`: Število že obstoječih poti.
- `L`: Število dni, ki jih od Kače terjajo nove povezave.
- `A`, `B` in `T`: Polja dolžine `M`, ki opisujejo krajišča obstoječih poti in čas potovanja med njimi, tako da `i`-ta povezava povezuje bilabonga `A[i-1]` in `B[i-1]` ter zahteva `T[i-1]` dni za potovanje v katerokoli smer.
- *Vrača*: Najdaljši čas potovanja med poljubnim parom bilabongov, kot je opisano zgoraj.

Vzorčni klic

Gornji primer je opisan z naslednjimi parametri:

Parameter	Value
N	`12`
M	`8`
L	`2`
A	`[0, 8, 2, 5, 5, 1, 1, 10]`
B	`[8, 2, 7, 11, 1, 3, 9, 6]`
T	`[4, 2, 4, 3, 7, 1, 5, 3]`
Returns	`18`

Omejitve

- Časovna omejitev: 1 sekunda
- Prostorska omejitev: 64 MiB
- $1 \le N \le 100\,000$
- $0 \le M \le N - 1$
- $0 \le A[i], B[i] \le N - 1$
- $1 \le T[i] \le 10\,000$
- $1 \le L \le 10\,000$

Podnaloge

Podnaloga	Točke	Dodatne omejitve vhoda
1	14	$M = N - 2$, za vsak bilabong pa obstajata natanko ena ali dve obstoječi povezavi. Z drugimi besedami: obstajata dve povezani množici bilabongov, v vsaki komponenti pa obstaja preprosta pot med bilabongi (ki se ne cepi).
2	10	$M = N - 2$ in $N \leq 100$
3	23	$M = N - 2$
4	18	Za vsak bilabong obstaja največ ena že obstoječa povezava.
5	12	$N \leq 3000$
6	23	*(Brez)*

Preizkušanje

Vzorčni ocenjevalnik na tvojem računalniku bere vhod iz datoteke `dreaming.in`, ki mora ustrezati naslednji obliki:

- vrstica 1: `N M L`
- vrstica 2, …, $M + 1$: `A[i] B[i] T[i]`

Gornji primer, denimo, je podan v sledečem formatu:

```
12 8 2
0 8 4
8 2 2
2 7 4
5 11 3
5 1 7
1 3 1
1 9 5
10 6 3
```

Jezikovne opombe

C/C++ Potrebuješ `#include "dreaming.h"`.

Pascal Definiraj `unit Dreaming`. Oštevilčenje vseh polj se prične z `0` (in ne z `1`).

Za primer glej predlogo rešitve na svojem računalniku.