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

dreaming

Georgian — 1.0

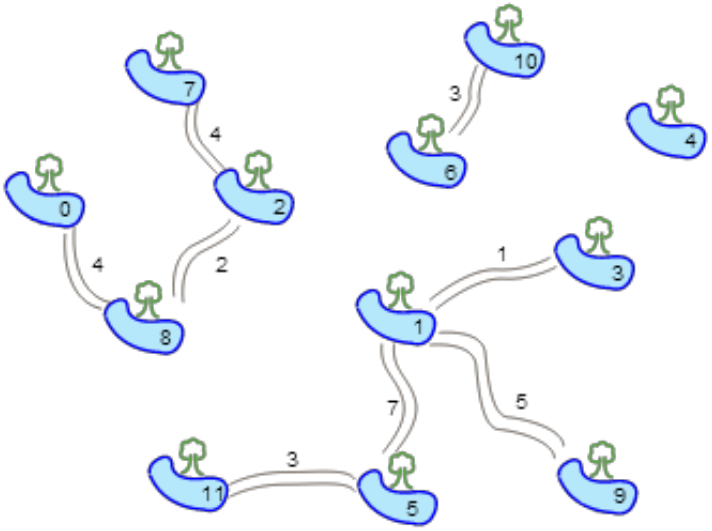
ეს ისტორია დასაბამს იღებს ძველი დროიდან, როცა სამყარო ახალი შექმნილი იყო და IOI-ს ჩატარებაზე ოცნებაც არ შეიძლებოდა.

სერპენტი ცხოვრობს ისეთ ტერიტორიაზე, რომელზეც განლაგებულია N ცალი წყლის გუბურა. გუბურები გადანომრილია რიცხვებით $0, \ldots, N - 1$. მოცემულია M ცალი ორმხრივი ბილიკი, რომლებიც აერთებენ გუბურათა წყვილებს და რომლებზეც სერპენტს შეუძლია მოძრაობა. გუბურების ყოველ წყვილს შორის არსებობს არაუმეტეს ერთი გზისა (გზა წარმოადგენს ბილიკების მიმდევრობას), ხოლო ზოგიერთ წყვილს შორის გზა შეიძლება საერთოდ არ არსებობდეს (ანუ, $M \leq N-1$). ყოველი ბილიკის გასავლელად სერპენტი ხარჯავს დღეების გარკვეულ რაოდენობას და ეს რაოდენობები სხვადასხვა ბილიკისათვის შეიძლება სხვადასხვა იყოს.

სერპენტის მეგობარ კენგურუს სურს შექმნას $N - M - 1$ ახალი ბილიკი ისე, რომ გზით დააკავშიროს გუბურების ნებისმიერი წყვილი. კენგურუს შეუძლია ბილიკის შექმნა გუბურათა ნებისმიერ წყვილს შორის და ყოველი ასეთი ახლადშექმნილი ბილიკის გასავლელად სერპენტს დასჭირდება ერთი და იგივე L რაოდენობის დღე.

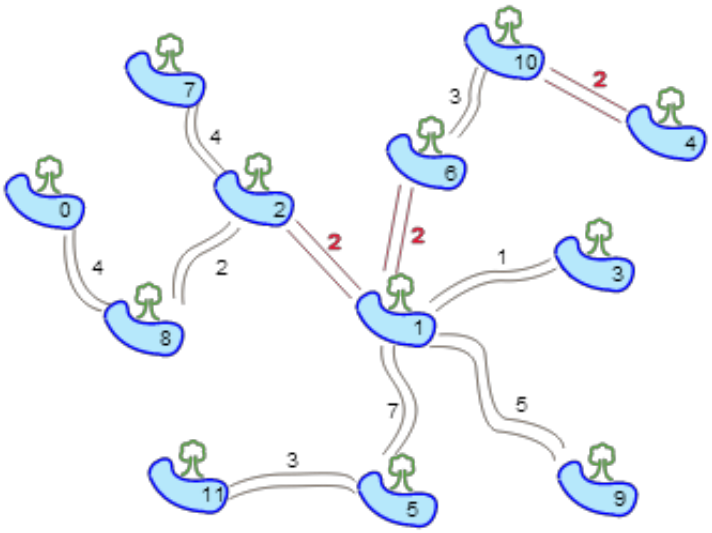
ამასთან, კენგურუს სურს, რომ სერპენტის მოგზაურობის დრო რაც შეიძლება ნაკლები იყოს და ამიტომ ის დაამატებს ახალ ბილიკებს ისე, რომ გუბურათა ნებისმიერ წყვილს შორის არსებული გზის გასავლელად საჭირო დღეების მაქსიმალური რაოდენობა მინიმალური იყოს.

მაგალითი



ზემოთ მოყვანილ ნახაზზე მოცემულია `N = 12` გუბურა და `M = 8` ბილიკი. დავუშვათ, `L = 2`, ანუ ყოველი ახალი ბილიკის გასავლელად სერპენტს 2 დღე დასჭირდება. კენგურუს შეუძლია ააგოს სამი ახალი ბილიკი:

- 1 და 2 გუბურებს შორის;
- 1 და 6 გუბურებს შორის;
- 4 და 10 გუბურებს შორის.



ზემოთ მოყვანილ ნახაზზე ნაჩვენებია ბილიკების საბოლოო სიმრავლე. უგრძელესი მოგზაურობის დრო არის 18 დღე 0 და 11 ნომრების მქონე გუბურებს შორის. ეს უმცირესი შესაძლო შედეგია. ყველა სხვა შემთხვევაში მოიძებნება გუბურათა წყვილი, რომელთა შორის მოგზაურობის ხანგრძლივობა 18 ან მეტი დღე იქნება.

იმპლემენტაცია

თქვენ უნდა გააგზავნოთ ფაილი, რომელშიც უნდა მოახდინოთ ფუნქცია `travelTime()`–ის იმპლემენტაცია შემდეგნაირად:

თქვენი ფუნქცია: **`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;
```

აღწერა

ამ ფუნქციამ უნდა გამოთვალოს უდიდესი სამოგზაურო დრო (დღეებში) გუბურათა ყველა შესაძლო წყვილს შორის. ამასთან, იგულისხმება, რომ კენგურუმ დაამატა `N - M - 1` ბილიკი და ნებისმიერ ორ გუბურას შორის არსებობს ერთადერთი გზა, ხოლო უდიდესი სამოგზაურო დრო რაც შეიძლება მინიმალური უნდა იყოს.

პარამეტრები

- `N`: გუბურათა რაოდენობა.
- `M`: უკვე არსებული ბილიკების რაოდენობა.
- `L`: დრო (დღეების რაოდენობა), რომელიც სჭირდება სერპენტს ახალი ბილიკების გასავლელად.
- `A`, `B` და `T`: მასივები ელემენტების `M` რაოდენობით, რომლებიც განსაზღვრავენ არსებულ ბილიკებს შემდეგნაირად: `i`–ური ბილიკი აერთებს `A[i-1]` და `B[i-1]` გუბურებს და ნებისმიერი მიმართულებით მის გასავლელად საჭიროა `T[i-1]` დღე.
- *Returns*: გუბურათა ნებისმიერ წყვილს შორის არსებული გზის გასავლელად საჭირო დღეების მაქსიმალურ რაოდენობებს შორის მინიმალური.

ტექსტში მოტანილი მაგალითის შეტანის ნიმუში

შემდეგი ნიმუში აღწერს ზემოთ მოყვანილ მაგალითს:

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

შეზღუდვები

- დროის ლიმიტი: 1 წამი
- მეხსიერების ლიმიტი: 64 MiB
- 1 ≤ N ≤ 100,000
- 0 ≤ M ≤ N - 1
- 0 ≤ A[i], B[i] ≤ N - 1
- 1 ≤ T[i] ≤ 10,000
- 1 ≤ L ≤ 10,000

ქვეამოცანები

ქვეამოცანა	ქულა	შესატანი მონაცემების დამატებითი შეზღუდვა
1	14	M = N - 2, ყოველი გუბურისთვის არსებობს მასთან მისასვლელი ერთი ან ორი ბილიკი. სხვა სიტყვებით გუბურების სიმრავლე გაყოფილია ორ ბმულ ქვესიმრავლედ და თითოეულ ქვესიმრავლეში ბილიკები ჰქმნიან არაგანშტოებად გზას.
2	10	M = N - 2 და N ≤ 100
3	23	M = N - 2
4	18	ყველა გუბურიდან გამოდის არაუმეტეს ერთი თავდაპირველად არსებული ბილიკი
5	12	N ≤ 3,000
6	23	*(None)*

ექსპერიმენტირება

სანიმუშო გრადერმა თქვენი კომპიუტერიდან უნდა წაიკითხოს შესატანი მონაცემები ფაილიდან `dreaming.in`, შემდეგი ფორმატით:

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

მაგალითად, ზემოთ მოყვანილი მაგალითი წარმოდგენილი უნდა იყოს შემდეგი ფორმატით:

```
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
```

შენიშვნები პროგრამული ენებისათვის

C/C++	თქვენ უნდა გამოიყენოთ `#include "dreaming.h"`.
Pascal	You must define the `unit Dreaming`. All arrays are numbered beginning at `0` (not `1`).

მაგალითებისათვის ამოხსნათა შაბლონები იხილეთ თქვენს კომპიუტერზე.