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

dreaming

Português — 1.0

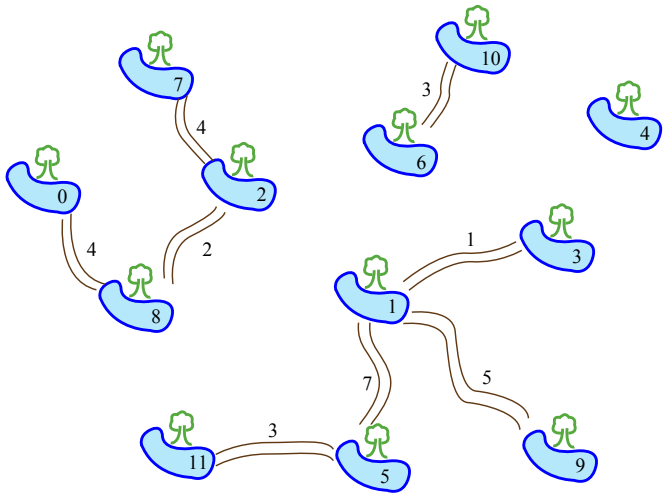
Esta história aconteceu há muito tempo atrás, quando o mundo era novo e a IOI não havia sido sonhada.

Uma Serpente vive em um local que tem N lagos (burracos com água), numerados de $0, \ldots, N-1$. Existem M "trilhas" bidimensionais ligando pares de lagos, nos quais a Serpente pode viajar. Cada par de lago é conectado (diretamente ou indiretamente) por no máximo uma sequência de trilhas, apesar de alguns pares de lagos não serem ligados (isto é, $M \le N-1$). Cada trilha consome um certo número de dias para Serpente viajar por ela: este número pode ser diferente para cada trilha.

Um Canguru, amigo de Serpente, quer fazer $N - M - 1$ novos caminhos, de modo que seja possível para Serpente viajar entre qualquer par de lagos. O Canguru pode criar trilhas entre qualquer par de lagos. A Serpente vai demorar L dias para viajar ao longo de qualquer trilha criada pelo Canguru.

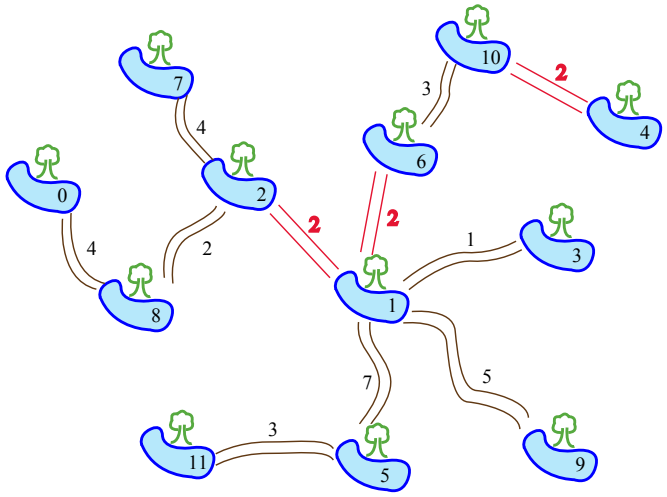
Além disso, o Canguru quer fazer com que as viagens da Serpente sejam as mais rápidas possíveis. O Canguru vai construir os novos caminhos para que o maior tempo de viagem entre quaisquer dois lagos seja o menor possível. Ajude o Canguru e a Serpente a determinar qual é o maior tempo de viagem entre dois quaisquer lagos, depois de o Canguru fazer as novas trilhas.

Exemplos



Na figura acima existem `N = 12` lagos e `M = 8` trilhas. Suponha que `L = 2`, assim qualquer trilha nova consumirá dois dias de viagem da Serpente. Então o Canguru pode construir três novas trilhas:

- entre o lago 1 e 2;
- entre o lago 1 e 6;
- entre o lago 4 e 10.



A figura acima mostra a configuração final das trilhas. A viagem mais longa é de 18 dias, entre os lagos 0 e 11. Este é o menor resultado possível—não importa como o Canguru construa as trilhas, sempre existirá um par de lagos que consumirá 18 dias ou mais de viagem da Serpente.

Implementação

Você deve submeter um arquivo que implemente a função `travelTime()`, da seguinte forma:

Sua função: `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;
```

Descrição

Está função deve calcular a maior distância viajada (mensurada em dias) entre qualquer par de lagos, assumindo que o Canguru adicionou `N-M-1` trilhas de tal forma que os lagos estão conectados e a maior viagem feita é a menor possível.

Parâmetros

- `N`: O número de lagos.
- `M`: O número de trilhas existentes.
- `L`: o tempo em dias que a Serpente leva para viajar nas trilhas novas.
- `A`, `B` e `T`: Vetores de tamanho `M` que especificam os terminos e tempo de viagem de cada trilha pre-existente, assim `i`-ésimo trilha ligando os lagos `A[i-1]` e `B[i-1]`, e consome `T[i-1]` dias para viajar em qualquer direção.
- *Devolve*: O maior tempo de viagem entre qualquer par de ilhas, como descrito acima.

Exemplo de sessão

A seguinte sessão descreve o exemplo acima:

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`

Restrições

- Tempo limite: 1 segundo
- Limite de memóriat: 64 MiB
- $1 \le N \le 100,000$
- $0 \le M \le N - 1$
- $0 \le A[i], B[i] \le N - 1$
- $1 \le T[i] \le 10,000$
- $1 \le L \le 10,000$

Sub-tarefas

Sub-tarefa	Pontos	Restrição adiciona de entreda
1	14	$M = N - 2$, e existe precisamente uma ou duas trilhas pré-existe para cada lago. Em outras palavras, existem dois conjuntos de lagos conectados, e cada um deles forma um caminho sem ramos.
2	10	$M = N - 2$ e $N \le 100$
3	23	$M = N - 2$
4	18	Existe no máximo um caminho pré-existente partindo de cada lago.
5	12	$N \le 3,000$
6	23	*(Nenhuma)*

Implementação

O validador de exemplo no seu computador lê a entrada do arquivo `dreaming.in`, no seguinte formato:

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

O exemplo anterior deve ser dado no seguinte formato:

```
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
```

Notas sobre as Linguagens

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

See the solution templates on your machine for examples.