



International Olympiad in Informatics 2013

6-13 July 2013

Brisbane, Australia

dreaming

Spanish — 1.0

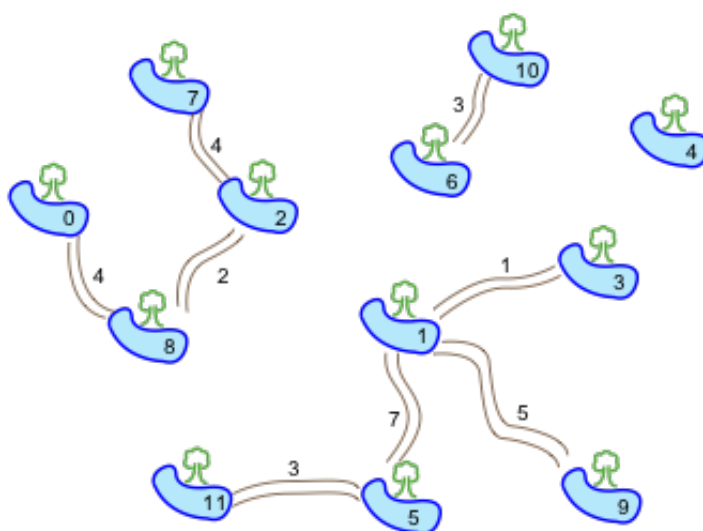
Esta historia sucedió hace mucho tiempo, cuando se creó el mundo y la IOI aún no había sido soñada.

Serpiente vive en un lugar que tiene N estanques (agujeros llenos de agua), numerados $0, \dots, N-1$. Hay M senderos bidireccionales uniendo pares de estanques, por los cuales Serpiente puede viajar. Cada par de estanques están conectados (directamente o indirectamente) por a lo más una secuencia de senderos, aunque algunos pares de estanques podrían no estar conectados en absoluto (o sea $M \leq N-1$). Serpiente le toma una cierta cantidad de días viajar por cada uno de los senderos: este número puede ser distinto para cada sendero.

Canguro, el amigo de Serpiente, desea construir $N - M - 1$ senderos nuevos, para que así Serpiente pueda viajar entre cualquier par de estanques. Canguro puede crear senderos entre cualquier par de estanques, y Serpiente necesitará L días para viajar a través de cada sendero creado.

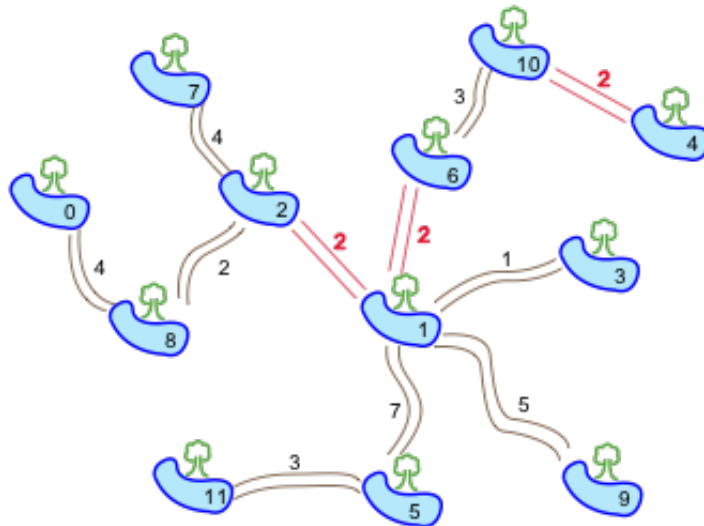
Además, Canguro quiere que Serpiente viaje tan rápido como sea posible. Canguro construirá nuevos senderos de modo que la distancia entre cualquier par de estanques sea tan pequeña como sea posible. Ayuda a Canguro y Serpiente a determinar el tiempo de viaje más largo entre dos estanques después de que Canguro haya constuido los senderos necesarios.

Ejemplos



En la figura anterior hay $N = 12$ estanques y $M = 8$ senderos. Supongamos $L = 2$, de modo que cualquier sendero nuevo requerirá a Serpiente 2 días para cruzarlo. Entonces Canguro podría construir 3 nuevos senderos:

- entre los estanques 1 y 2;
- entre los estanques 1 y 6;
- entre los estanques 4 y 10.



La figura anterior muestra el conjunto final de senderos. El viaje más largo es de 18 días, entre los estanques 0 y 11. Este es el resultado más pequeño posible - no importa como Canguro construya los senderos, siempre existirá un par de estanques que harán que Serpiente necesite viajar 18 días o más.

Implementación

Debes enviar un archivo que implemente la función `travelTime()` como se describe:

Tu función: `travelTime()`

C/C++	<pre>int travelTime(int N, int M, int L, int A[], int B[], int T[]);</pre>
Pascal	<pre>function travelTime(N, M, L : LongInt; var A, B, T : array of LongInt) : LongInt;</pre>

Descripción

Esta función debe calcular el tiempo máximo de viaje (medido en días) para viajar entre cualquier par de estanques, asumiendo que canguro ha agregado $N - M - 1$ senderos de modo que todos los estanques estén conectados y este tiempo máximo sea tan pequeño como sea posible.

Parámetros

- N : Número de estanques.
- M : Número de senderos que ya existen.
- L : Tiempo en días que Serpiente tardará en cruzar si viaja a través de los senderos nuevos.
- A , B y T : Arreglos de largo M que especifican los extremos y el tiempo de viaje de cada sendero pre-existente, de modo que el sendero i -ésimo une los estanques $A[i-1]$ y $B[i-1]$, y toma $T[i-1]$ días cruzarlo en cualquier dirección.
- *Returns*: El tiempo de viaje máximo entre cualquier par de estanques, como se describió anteriormente.

Sesión de Muestra

La siguiente sesión describe el ejemplo anterior:

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

Restricciones

- Time limit: 1 second
- Memory limit: 64 MiB
- $1 \leq N \leq 100,000$
- $0 \leq M \leq N - 1$
- $0 \leq A[i], B[i] \leq N - 1$
- $1 \leq T[i] \leq 10,000$
- $1 \leq L \leq 10,000$

Sub-tareas

Subtarea	Puntos	Restricciones adicionales de la entrada
1	14	$M = N - 2$, y hay exactamente uno o dos senderos preexistentes saliendo de cada estanque. En otras palabras, hay dos conjuntos de estanques conectados, y en cada conjunto los senderos forman un camino sin ramificaciones.
2	10	$M = N - 2$ y $N \leq 100$
3	23	$M = N - 2$
4	18	Como mucho hay un sendero pre-existente saliendo de cada estanque.
5	12	$N \leq 3,000$
6	23	(Ninguna)

Experimentación

El evaluador en tu ordenador leerá la entrada del archivo `dreaming.in`, que tendrá el siguiente formato:

- línea 1: `N M L`
- líneas 2, ..., $M + 1$: `A[i] B[i] T[i]`

Por ejemplo el caso mostrado anteriormente debe ser codificado como de la siguiente forma:

```
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 del Lenguaje

- C/C++ Tienes que incluir la línea `#include "dreaming.h"`.
- Pascal Tienes que definir la `unit Dreaming`. Todos los arreglos están numerados desde `0` (no `1`).

Mira las plantillas de solución en tu máquina como ejemplo.