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

# dreaming

Spanish — 1.0

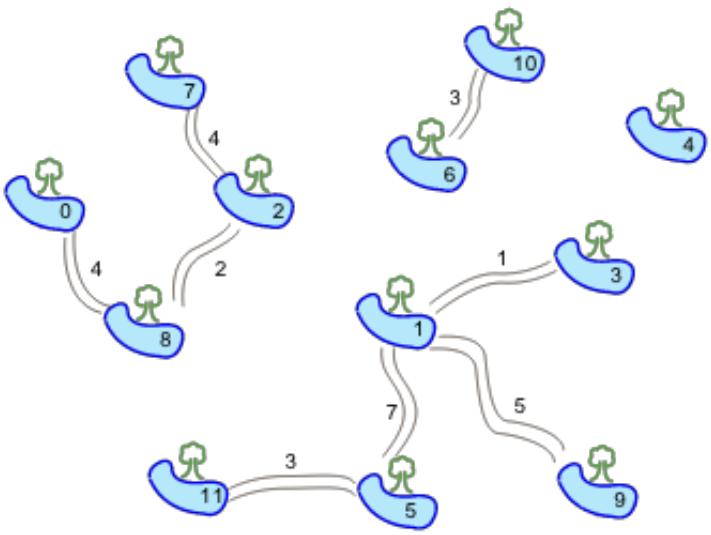
Esta historia sucedió hace mucho tiempo, poco después de que se creara el mundo, y mucho antes de que existiera la IOI.

Serpiente vive en un lugar en el que hay $N$ estanques (llamados billabong en australiano), numerados como $0, \ldots, N-1$. Hay $M$ senderos bidireccionales que unen pares de estanques, y por los que Serpiente puede viajar para ir de uno a otro. Cada par de estanques estará conectado como mucho por una secuencia de senderos (directa o indirectamente), aunque es posible que haya pares de estanques que no estén conectados en absoluto (es decir, $M \leq N-1$). Serpiente tarda un número determinado de días en atravesar un sendero; este número puede ser diferente para senderos diferentes.

Canguro, el amigo de Serpiente, quiere construir $N-M-1$ senderos nuevos, de forma que Serpiente pueda viajar entre cualquier par de estanques. Canguro puede crear senderos entre cualquier par de estanques, y Serpiente tardará $L$ días en atravesar los senderos que crea Canguro.

Además, Canguro quiere que Serpiente viaje lo más rápido posible. Canguro construirá nuevos senderos para que la máxima distancia entre cualquier pareja de estanques sea lo menor posible. Ayuda a Canguro y Serpiente y determina el viaje más largo entre dos estanques después de que Canguro haya construido los senderos de esta manera.

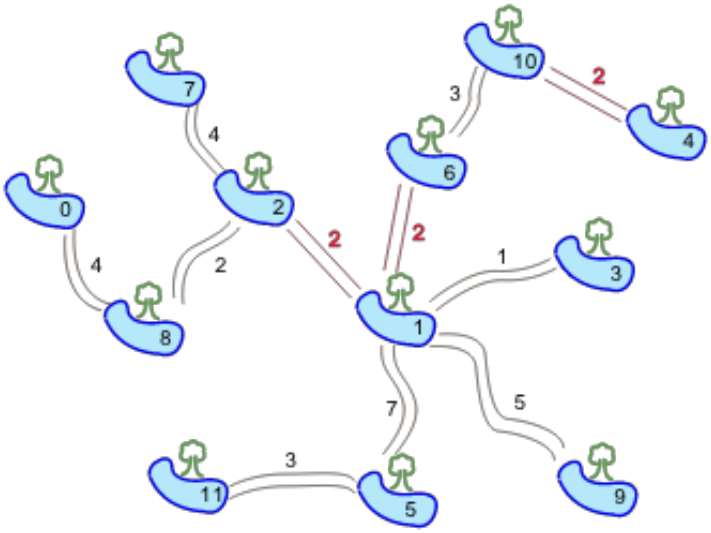
## Ejemplos



En la figura adjunta hay `N = 12` estanques y `M = 8` senderos. Supongamos `L = 2`, eso es, 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 superior muestra el conjunto de los senderos finales. El viaje más largo es de 18 días, entre los estanques 0 y 11. Este es el menor resultado posible - no importa como Canguro construya los senderos: siempre existirá un par de estanques que harán que Serpiente necesite 18 días o más para viajar de uno a otro.

---

## Implementación

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

### Tu función: `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;
```

Descripción

Esta función debería calcular el mayor de los tiempos (medido en días) para viajar entre cualquier par de estanques, asumiendo que Canguro ha añadido `N - M - 1` senderos de modo que todos los estanques estén conectados y este tiempo máximo sea lo menor posible.

Parámetros

- `N`: Número de estanques.
- `M`: Número de senderos que ya existen.
- `L`: Tiempo en días que tardará Serpiente en cruzar los senderos nuevos.
- `A`, `B` y `T`: Arrays de longitud `M` que especifican, por cada sendero ya existente, el estanque inicial y final, y el tiempo que se tarda en atravesarlo. Es decir, el sendero `i`-ésimo une los estanques `A[i-1]` y `B[i-1]`, y se tardan `T[i-1]` días en cruzarlo en cualquier dirección.
- *Returns*: El máximo tiempo de viaje entre cualquier par de estanques, según la descripción previa.

---

## Ejemplo de entrada

La siguiente entrada 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 ≤ N ≤ 100,000`
- `0 ≤ M ≤ N - 1`
- `0 ≤ A[i], B[i] ≤ N - 1`
- `1 ≤ T[i] ≤ 10,000`
- `1 ≤ L ≤ 10,000`

## Subtareas

| Subtarea | Puntos | Restricciones adicionales de la entrada |
|---|---|---|
| 1 | 14 | $M = N - 2$, y hay exactamente uno o dos senderos ya existentes 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 ya existente saliendo de cada estanque. |
| 5 | 12 | $N \leq 3{,}000$ |
| 6 | 23 | *(Ninguna)* |

## Experimentación

El grader 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]`

El ejemplo anterior se daría en el siguiente 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
```

## Apuntes del Lenguaje

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

Mira las plantillas de solución en tu ordenador como ejemplo.