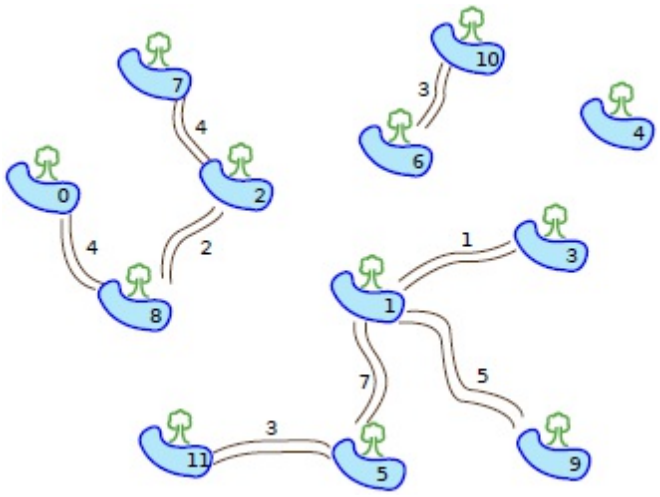
IOI '13 P1 - Dreaming

This story takes place a long time ago, when the world was new and the IOI had not yet been dreamt. Serpent lives in a land which has N billabongs (water holes), numbered $0, \ldots, N-1$. There are M bidirectional trails, joining pairs of billabongs, which Serpent can travel along. Each pair of billabongs is connected (directly or indirectly) by at most one sequence of trails, though some pairs of billabongs may not be connected at all (thus, $M \leq N-1$). Each trail takes a certain number of days for Serpent to travel along it: this number may be different for each trail.

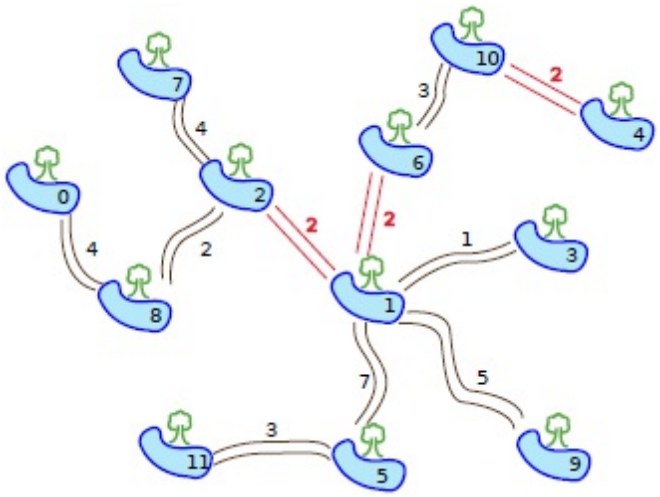
Serpent's friend, Kangaroo, wishes to make $N-M-1$ new trails, so that it is possible for Serpent to travel between any pair of billabongs. Kangaroo can create trails between any pair of billabongs, and every trail that Kangaroo creates will take L days for Serpent to travel along it.

Additionally, Kangaroo wants to make Serpent's travels as fast as possible. Kangaroo will build the new trails so that the longest travel time between any two billabongs is as small as possible. Help Kangaroo and Serpent determine the longest travel time between any two billabongs, after Kangaroo has built the new trails in this way.



In the picture above there are $N=12$ billabongs and $M=8$ trails. Suppose that $L=2$, so that any new trail will require Serpent 2 days to travel. Then Kangaroo could build three new trails:

- between billabongs 1 and 2;
- between billabongs 1 and 6;
- between billabongs 4 and 10.



The picture above shows the final set of trails. The longest travel time is 18 days, between billabongs 0 and 11. This is the smallest result possible—no matter how Kangaroo builds the trails, there will be some pair of billabongs that requires Serpent to travel for 18 days or more.

Implementation

You should submit a file implementing the function `travelTime()`, as follows:

Grader Function: `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;
```

Description

> This function should calculate the greatest travel time (measured in days) between any pair of billabongs, assuming that Kangaroo has added $N - M - 1$ trails in such a way that all billabongs are connected and this greatest travel time is as small as possible.

Parameters

- `N`: The number of billabongs.
- `M`: The number of trails that already exist.
- `L`: The time in days that it takes Serpent to travel along a new trail.
- `A`, `B` and `T`: Arrays of length M that specify the endpoints and travel time of each pre-existing trail, so

that the i^{th} trail joins billabongs $A[i_1]$ and $B[i_1]$, and takes $T[i_1]$ days to travel in either direction.
- *Returns:* The greatest travel time between any pair of billabongs, as described above.

Sample Session

The following session describes the example above:

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

Constraints

- $1 \leq N \leq 100\,000$
- $0 \leq M \leq N - 1$
- $0 \leq A[i], B[i] \leq N - 1$
- $1 \leq T[i] \leq 10\,000$
- $1 \leq L \leq 10\,000$

Subtasks

Subtask	Percent of Points	Additional Input Constraints
1	14	$M = N - 2$, and there are precisely one or two preexisting trails leading from each billabong. In other words, there are two sets of connected billabongs, and in each set the trails form an unbranching path.
2	10	$M = N - 2$ and $N \leq 100$
3	23	$M = N - 2$
4	18	There is at most one pre-existing trail leading from each billabong.
5	12	$N \leq 3\,000$

6	23	(None)