WOBURN CHALLENGE

2016-17 Online Round 3

Sunday, February 19th, 2017

Senior Division Problems

Automated grading is available for these problems at:
wcipeg.com

For more problems from past contests, visit:
woburnchallenge.com

Problem S1: Cutting Edge

12 Points / Time Limit: 1.00s / Memory Limit: 16M *Submit online: wcipeg.com/problem/wc163s1*

You've landed a job as one of the level designers for the latest game in the Pokémon series, Pokémon Navy Green! You've always hated those nerdy Pokémon fans, always having fun with their childish Pokémon nonsense, so this is your chance to get back at them.

One of the screens in the game is a grid of cells with N rows and M columns ($2 \le N, M \le 2{,}000{,}000{,}000$). The player will start in the top-left cell, with the goal of reaching the bottom-right cell by repeatedly moving up, down, left, or right.

As a level designer, you may decide that some subset of the cells in the grid should contain trees. The top-left and bottom-right cells may not contain trees, but any of the others are fair game. The player may not move into a cell which contains a tree. However, they do have the ability to teach a move called Cut to one of their Pokémon. Each time Cut is used, a single tree can be destroyed, allowing the player to move into its cell. However, Cut may only be used at most K ($0 \le K \le 2{,}000{,}000{,}000$) times in total.

You love the idea of frustrating the player with a subtly impossible level, but don't want to make your treachery too obvious, for fear of losing your job before the game gets released. As such, you'd like to determine the minimum number of trees that must be present on the grid such that it's impossible for the player to move from the top-left cell to the bottom-right one, even after optimally using Cut at most K times. Unfortunately, it may also be the case that the level will be possible to complete no matter how many trees you place on the grid.

In test cases worth 10/12 of the points, $N \le 10^6$ and $M \le 10^6$.

Input Format

The first and only line of input consists of three space-separated integers N, M, and K.

Output Format

Output one line consisting of a single integer – the minimum number of trees required to make the level impossible, or −1 if no number of trees will suffice.
Note that the answer may not necessarily fit within a 32-bit signed integer (you may need the `long long` type in C++, or `long` in Java).

Sample Input 1

```
2 5 2
```

Sample Output 1

```
6
```

Sample Explanation 1

One optimal arrangement of trees is shown below (trees are indicated with "`#`", while empty cells are indicated with "`.`"):

```
.###.
.###.
```

If the players cut down any 2 of these 6 trees, they still won't be able to move from the top-left to the bottom-right cell.

Sample Input 2

```
2 5 4
```

Sample Output 2

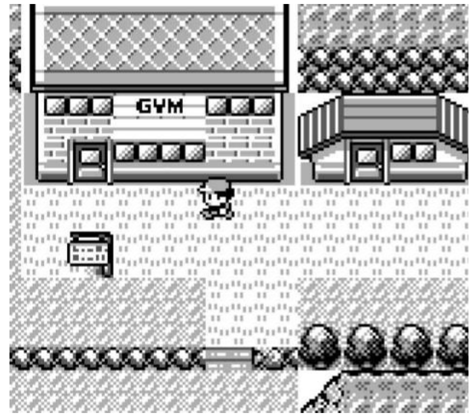
```
-1
```

Sample Explanation 2

Even if trees are placed in all 8 valid cells, the player will still be able to move from the top-left to the bottom-right cell after cutting down 4 of them.

Problem S2: Training Regimen

18 Points / Time Limit: 3.00s / Memory Limit: 64M *Submit online: wcipeg.com/problem/wc163s2*



In the world of Pokémon Navy Green, there are N ($2 \le N \le 200{,}000$) towns, with M ($0 \le M \le 200{,}000$) routes running amongst them. At the beginning of the game, you find yourself in town 1 with a starting Pokémon of your choice – a cute level 1 Pyglion! Your objective is to reach town N by following a sequence of routes.

The i-th route connects distinct towns A_i and B_i ($1 \le A_i, B_i \le N$), and can be walked along in either direction. No pair of towns are directly connected by multiple routes. However, the routes are also crawling with dangerous Pokémon. As such, you can only dare venture along the i-th route if your Pyglion's level is currently no smaller than C_i ($1 \le C_i \le 10^9$).

As mentioned, your Pyglion's level is initially 1, but it can be increased through intensive training. Each town has its own training gym for that purpose. However, some gyms are more efficient than others – in particular, in the i-th town, it takes T_i ($1 \le T_i \le 10^9$) minutes of training to increase your Pyglion's level by 1. This T_i–minute process can be repeated as many times as you'd like.

You'd hate to tucker out your poor Pyglion more than necessary, so you'd like to reach town N after spending as little total time training in gyms as possible. Note that time spent walking along routes isn't relevant, and that you may revisit towns on your adventure as often as you'd like.

Please determine the minimum number of minutes of training required to reach town N, or determine that you can't complete your trip no matter how much training you put your Pyglion through.

In test cases worth 13/18 of the points, $N \le 500$, $M \le 500$, and $C_i \le 500$.

Input Format

The first line of input consists of two space-separated integers N and M.
N lines follow, with the i-th of these lines consisting of a single integer, T_i (for $i = 1..N$).
M lines follow, with the i-th of these lines consisting of three space-separated integers A_i, B_i, and C_i, (for $i = 1..M$).

Output Format

Output one line consisting of a single integer – the minimum number of minutes of training required to reach town N, or –1 if it can't be done.
Note that the answer may not necessarily fit within a 32-bit signed integer (you may need the `long long` type in C++, or `long` in Java).

Sample Input

```
6 8
14
5
8
10
2
4
1 4 5
1 2 8
4 5 12
3 1 2
6 3 11
2 3 14
5 6 4
2 4 6
```

Sample Output

```
71
```

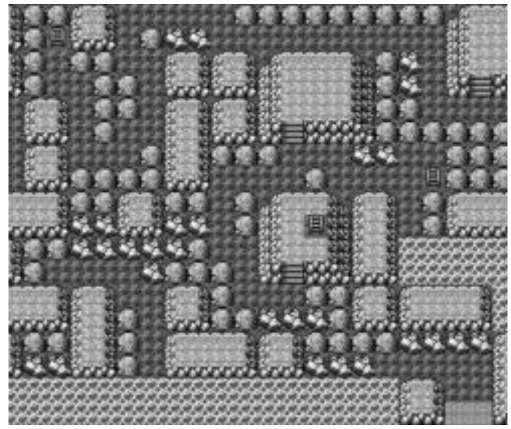
Sample Explanation

One optimal sequence of actions is as follows:

- Train in town 1 for 14 minutes (up to level 2).
- Walk to town 3.
- Train in town 3 for 32 minutes (up to level 6).
- Walk to town 1.
- Walk to town 2.
- Train in town 2 for 25 minutes (up to level 11).
- Walk to town 1.
- Walk to town 3.
- Walk to town 6.

Problem S3: Puzzle Rooms

35 Points / Time Limit: 5.00s / Memory Limit: 16M *Submit online: wcipeg.com/problem/wc163s3*



Among other things, the Pokémon series of video games is known for having interesting 2D puzzles for the player to solve. The latest installment, Pokémon Navy Green, is no exception!

As one of the game's level designers, you've been given the responsibility of turning N ($1 \le N \le 50$) different rooms into a series of puzzling challenges. However, rather than designing thought-provoking puzzles which would require careful solutions from the player, you'd much rather annoy them by crafting as tedious a gaming experience as possible.

The i-th room is a grid of cells with 4 rows and C_i ($1 \le C_i \le 100$) columns. You'll designate one of the cells to be the player's starting cell, and another one to be their destination. You may then choose some subset of the remaining cells to contain walls. The player's objective will then be to navigate from the starting cell to the destination cell by moving up, down, left, and right, without entering any cells that contain walls. Mazes are passable as puzzles, right?

Since your own objective is to make the players' lives miserable, you want to design each room in such a way that the shortest possible path from the starting cell to the destination cell is as long as possible. The distance covered by a path between two cells is the number of up/down/left/right moves that it involves. For each room, you'll need to determine the maximum possible length of such a shortest path, and come up with a room design (an arrangement of walls and starting/destination cells) which yields that optimal distance between its starting and destination cells. A room can be described as 4 strings of N characters each, with the j-th character of the i-th string representing the cell in the i-th row and the j-th column. The single starting cell should be indicated with an "`S`", the single destination cell with an "`E`", each wall with a "`#`", and each remaining empty cell with a "`.`". If there exist multiple optimal room designs, any of them will do.

In test cases worth 8/35 of the points, $C_i \le 6$.
In test cases worth another 8/35 of the points, $C_i \le 12$.

Input Format

The first line of input consists of a single integer N.
N lines follow, the i-th of which consists of a single integer C_i (for i = 1..N)

Output Format

For each room, output five lines. The first of these lines should consist of a single integer – the largest possible shortest distance between the starting and destination cells.
The last four of these lines shall describe any valid room design which yields that optimal distance.

Sample Input

```
2
2
3
```

Sample Output

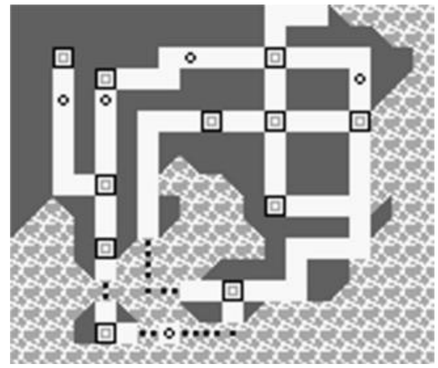
```
5
S.
#.
..
E#
8
...
.#E
..#
..S
```

Sample Explanation

Note that, for both rooms, there exist other valid room designs (besides the ones shown here) which would also yield the same optimal distances of 5 and 8, and which would also be accepted.

Problem S4: Replay Value

35 Points / Time Limit: 4.00s / Memory Limit: 64M *Submit online: wcipeg.com/problem/wc163s4*



In the world of Pokémon Navy Green, there are N ($2 \le N \le 500{,}000$) towns, with $N - 1$ routes running amongst them. The i-th route runs between distinct towns A_i and B_i ($1 \le A_i, B_i \le N$), and can be travelled along in either direction. No pair of towns have multiple routes running directly between them, and it's possible to reach every town from every other town by following a sequence of one or more routes. The player starts the game in some initial town, with the objective of reaching some other given final town by travelling along a sequence of one or more routes, without ever retracing their path.

Every time you play through the game, you get to choose which ordered pair of towns will be the initial and final towns for your adventure. That means that you might get to experience a whopping $N \times (N - 1)$ distinct playthroughs of Pokémon Navy Green – what a bargain! Unfortunately, some of those playthroughs may not turn out well, though.

Each town i has a difficulty rating D_i ($1 \le D_i \le 10^9$), indicating the strength of the Pokémon Gym Leader residing in it. It would sure be a poor gameplay experience if you were to visit a town with a smaller difficulty rating than that of the previous town you visited. As such, it's vital that the sequence of difficulty ratings of the towns you visit on the path from the initial to the final town (including both the initial and final town) is non-decreasing.

Despite this limitation, it seems that this game may still have plenty of replay value. Just how many times can you play through the game such that each playthrough features a distinct ordered pair of initial and final towns, while also resulting in you visiting towns with a non-decreasing sequence of difficulty ratings?

In test cases worth 8/35 of the points, $N \le 2000$ and $D_i \le 2$.
In test cases worth another 14/35 of the points, $D_i \le 2$.

Input Format

The first line of input consists of a single integer N.
N lines follow, the i-th of which consists of a single integer D_i (for $i = 1..N$).
$N - 1$ lines follow, the i-th of which consists of two space-separated integers A_i and B_i (for $i = 1..N-1$).

Output Format

Output one line consisting of a single integer – the number of distinct valid playthroughs.
Note that the answer may not necessarily fit within a 32-bit signed integer (you may need the `long long` type in C++, or `long` in Java).

Sample Input

```
6
10
3
11
10
8
42
1 2
1 3
1 4
5 3
6 3
```

Sample Output

```
13
```

Sample Explanation

One of the 13 valid playthroughs has initial town 4 and final town 6. The sequence of towns involved in this playthrough is [4, 1, 3, 6], which have the non-decreasing sequence of difficulty ratings [10, 10, 11, 42].