WOBURN CHALLENGE

2016-17 Online Round 3

Sunday, February 19th, 2017

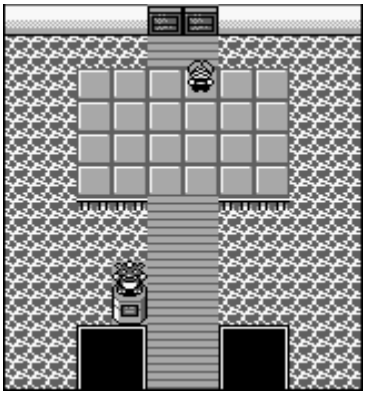
Junior Division Problems

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

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

Problem J1: The Elite *N*

20 Points / Time Limit: 1.00s / Memory Limit: 64M *Submit online: wcipeg.com/problem/wc163j1*



You've been hooked on the latest game in the Pokémon series, Pokémon Navy Green, and have finally reached the final challenge! In the original game, this challenge would've been the Elite 4, a series of 4 dangerous Pokémon trainers to defeat in a row. By now, it's been generalized to be the Elite N ($1 \le N \le 10^9$). Its N members are numbered from 1 to N in the order in which they must be defeated.

You've brought just two Pokémon with you to get the job done, Aeroxis and Brinoble – you've trained both to be incredibly powerful, so they're all you'll need. You initially have Aeroxis as your active Pokémon, with Brinoble stored away in a backup Pokeball. Before each of the N battles (including the first one), you may choose to swap your Pokémon, changing which of the two is active. Doing so takes S ($1 \le S \le 10^9$) seconds. You'll then use your currently active Pokémon to battle against the next of the N trainers. It takes Aeroxis A ($1 \le A \le 10^9$) seconds to defeat a trainer, and it takes Brinoble B ($1 \le B \le 10^9$) seconds to do the same. Note that you may not swap Pokémon during a battle.

Some members of the Elite N will be using different types of Pokémon, however. In particular, M ($1 \le M \le 1000$) of the trainers will be using cloud-type Pokémon, which Brinoble is extremely weak against – as such, you absolutely must use Aeroxis to battle each of them. The i-th of these M cloud-type Pokémon trainers is trainer number T_i of the Elite N. The numbers $T_{1..M}$ are distinct, and are given in increasing order ($1 \le T_1 < T_2 < \ldots < T_M \le N$). Given that you optimally choose when to swap Pokémon, what's the minimum total amount of time required to defeat all N members of the Elite N in order?

In test cases worth 15/20 of the points, $N \le 10^5$.

Input Format

The first line of input consists of five space-separated integers N, M, A, B, and S.
M lines follow, with the i-th of these lines consisting of a single integer, T_i (for $i = 1..M$).

Output Format

Output a single line consisting of a single integer – the minimum total amount of time required to defeat the Elite N, in seconds.

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

```
20 5 5 2 5
5
6
11
17
19
```

Sample Output

```
91
```

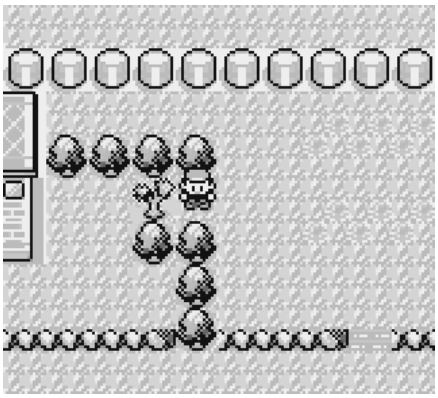
Sample Explanation

One optimal strategy is as follows:

- Swap to Brinboble before the 1st battle (5 seconds).
- Use Brinboble for battles 1..4 (8 seconds).
- Swap to Aeroxis (5 seconds).
- Use Aeroxis for batles 5..6 (10 seconds).
- Swap to Brinboble (5 seconds).
- Use Brinboble for battles 7..10 (8 seconds).
- Swap to Aeroxis (5 seconds)
- Use Aeroxis for battle 11 (5 seconds).
- Swap to Brinboble (5 seconds).
- Use Brinboble for battles 12..16 (10 seconds).
- Swap to Aeroxis (5 seconds).
- Use Aeroxis for battles 17..20 (20 seconds).

Problem J2: Pokéwarehouses

20 Points / Time Limit: 2.00s / Memory Limit: 64M *Submit online: wcipeg.com/problem/wc163j2*



Pokémarts are stores which are always stocked with useful items to help Pokémon trainers on their journeys. However, keeping Pokémarts sufficiently stocked is at least as difficult as catching Pokémon!

You've been placed in charging of planning out an upcoming shipment of N ($1 \le N \le 10^5$) items. The items are numbered from 1 to N, and are currently arranged in a single stack in an initial warehouse `I` such that the i-th item from the top is item number S_i ($1 \le S_i \le N$). Your job is to ensure that they get transported to a certain destination warehouse `D`, and end up arranged in a single stack such that the i-th item from the top is item number i.

Unfortunately, items are only allowed to be moved in a particular fashion. The only type of operation which may be performed consists of taking a single item from the top of one warehouse's stack, and moving it to the top of another warehouse's stack. Items may never be moved away from warehouse `D`, and may never be moved back to warehouse `I`, as this would make it look like progress was not being made. Furthermore, each warehouse is only large enough to contain a single stack, and no stacks may be formed outside of warehouses. As such, in order to make the shipment possible to complete, you may ask for 0 or more intermediate warehouses to first be constructed.

Of course, constructing entire warehouses is expensive, so you'd like to avoid doing so. As such, you'd like to determine the minimum number of intermediate warehouses which must be constructed, such that the shipment may then be completed successfully.

Input Format

The first line of input consists of a single integer N.
N lines follow, with the i-th of these lines consisting of a single integer S_i (for $i = 1..N$).

Output Format

Output one line consisting of a single integer – the minimum number of intermediate warehouses required to complete the shipment.

Sample Input

```
3
3
1
2
```

Sample Output

```
1
```

Sample Explanation

With one intermediate warehouse `W`, the following sequence of actions may be performed:

- Move item 3 from warehouse `I` to `D`.
- Move item 1 from warehouse `I` to `W`.
- Move item 2 from warehouse `I` to `D`.
- Move item 1 from warehouse `W` to `D`.

Problem J3: Cutting Edge

24 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 ($2 \leq N, M \leq 2{,}000{,}000{,}000$). 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 ($0 \leq K \leq 2{,}000{,}000{,}000$) 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 20/24 of the points, $N \leq 10^6$ and $M \leq 10^6$.

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 J4: Training Regimen

36 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 26/36 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.