A: It's All Downhill From Here

John loves winter. Every skiing season he goes heli-skiing with his friends. To do so, they rent a helicopter that flies them directly to any mountain in the Alps. From there they follow the picturesque slopes through the untouched snow.

Of course they want to ski on only the best snow, in the best weather they can get. For this they use a combined condition measure and for any given day, they rate all the available slopes.

Can you help them find the most awesome route?

Input

The input consists of:

- one line with two integers n ($2 \leq n \leq 1000$) and m ($1 \leq m \leq 5000$), where n is the number of (1-indexed) connecting points between slopes and m is the number of slopes.
- m lines, each with three integers s, t, c ($1 \leq s, t \leq n$, $1 \leq c \leq 100$) representing a slope from point s to point t with condition measure c.

Points without incoming slopes are mountain tops with beautiful scenery, points without outgoing slopes are valleys. The helicopter can land on every connecting point, so the friends can start and end their tour at any point they want. All slopes go downhill, so regardless of where they start, they cannot reach the same point again after taking any of the slopes.

Output

Output a single number n that is the maximum sum of condition measures along a path that the friends could take.

Sample visualization

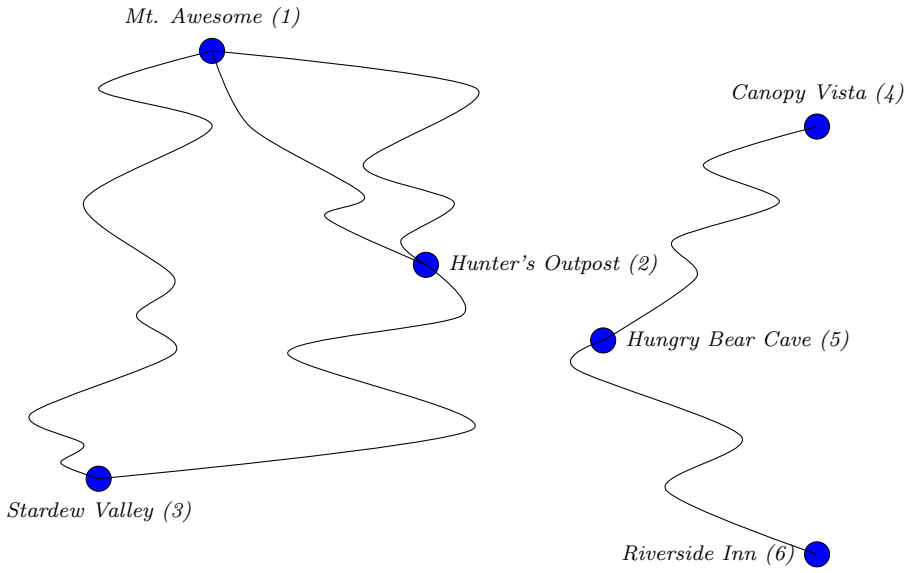


Figure C.1: Map of the second sample case

Sample Input 1

```
5 5
1 2 15
2 3 12
1 4 17
4 2 11
5 4 9
```

Sample Output 1

```
40
```

Sample Input 2

```
6 6
1 2 2
4 5 2
2 3 3
1 3 2
5 6 2
1 2 4
```

Sample Output 2

```
7
```

B: Leyline of Sanctity

Ancient cultures had an enormous and mostly unexpected knowledge of the earth and geometry. When they were building monuments, they did not place them at arbitrary locations. Instead, each building spot was carefully selected and calculated. Nowadays we can observe this in the form of so-called *ley lines*. A ley line is a horizontal or vertical straight line of infinite length that runs through at least two ancient monuments. Ley lines are expected to be a source of some kind of magic energy and every church built along such a line is a *mighty church*.

There are already several monuments and churches. An ancient culture is planning to construct a new monument, but the location is yet to be determined. They are looking for the spot that will turn the most ordinary churches into mighty churches. The monument can be co-located with a church or an existing monument – in that case, the new monument will simply be built around the church or monument.

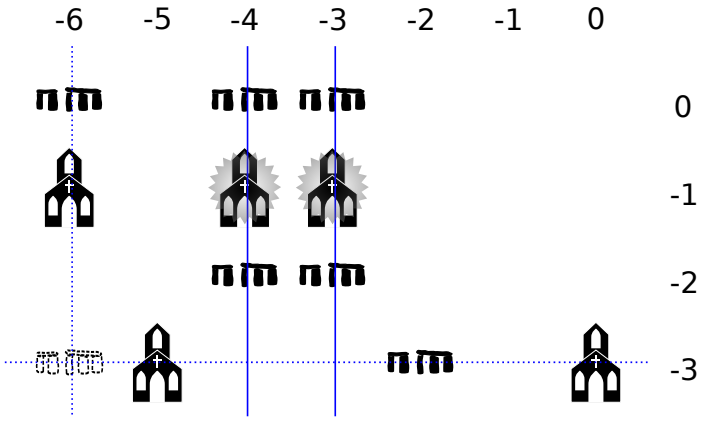


Figure K.1: Illustration of Sample 2 with 2 ley lines (|), 6 monuments ( ), 2 mighty churches ( ) and 3 ordinary churches ( ) that are turned into mighty churches when building the dashed monument.

Input

The input consists of:

- One line with two integers m and c ($0 \leq m, c \leq 1\,000$), the number of already built monuments and churches.
- m lines with the coordinates of the monuments.
- c lines with the coordinates of the churches.

All coordinates are given as two integers x and y ($-10^6 \leq x, y \leq 10^6$). None of the given coordinate pairs coincide with one another, but any may coincide with the location of the new monument.

Output

Output three integers: the coordinates for where to build the new monument and the number of ordinary churches that will be turned into mighty churches with this new monument. The coordinates should be in the range ($-10^6 \leq x, y \leq 10^6$). If there are multiple optimal solutions, you may output any one of them.

Sample Input 1

```
2 3
0 5
5 0
0 1
0 3
3 0
```

Sample Output 1

```
0 0
3
```

Sample Input 2

```
6 5
-6 0
-4 0
-3 0
-4 -2
-3 -2
-2 -3
-6 -1
-4 -1
-3 -1
-5 -3
0 -3
```

Sample Output 2

```
-6 -3
3
```

Sample Input 3

```
1 2
0 0
1 0
0 1
```

Sample Output 3

```
0 0
2
```

C: A Series of Tubes

As part of his new job at the IT security department of his company, Bob got the task to track how fast messages between the companies' different offices are transmitted through the internet. Since some offices are currently rebuilt and not finished yet he is not able to simply measure the transmission speed but needs to compute it somehow. Therefore, Bob created a map of all servers that may be involved in routing the packages through the internet. He also gathered the times each server needs to process a message. The total processing time of a message is the sum of the processing times of the sender, all servers along the path, and the receiver. Furthermore, Bob read that messages are sent through the network of servers along a path such that the total processing time of all servers on the path is minimal.

Bob thought that it might be an easy problem to compute the total transmission time between two offices, but he forgot the intelligence agencies! Each server on the internet is controlled by some agency that can decide which packages are routed and which of them are not. All servers are configured in a way that they read all incoming data, since gathering all kind of information is exactly what the intelligence agencies want to do, but not all data is forwarded to other servers.

Each server has a list of pairs of other servers such that messages from the first of them are not sent to the second one. Can you still help Bob to compute how fast his messages will be transmitted?

Input

The input consists of:

- one line with an integer n ($2 \leq n \leq 100$), where n is the number of servers labeled from 1 to n;
- n blocks describing the servers. Each server is described by:
  - one line with two integers m ($0 \leq m \leq n-1$) and t ($0 \leq t \leq 1000$), where m is the number of outgoing connections from this server and t is the processing time of this server;
  - m lines with two integers s ($0 \leq s \leq n-1$) and x ($1 \leq x \leq n$) and s more distinct integers $a_1, \ldots, a_s$ ($1 \leq a_j \leq n$, $a_j \neq i$ for all $1 \leq j \leq s$) indicating that server i sends messages to server x, but only if it was not directly transmitted from one of the servers $a_1, \ldots, a_s$ to server i.

Bob's messages start at server 1 and should go to server n.

Output

Output the sum of the processing times of all servers on the shortest path for Bob's message including the first and last one, or "`impossible`" if there is no such path.

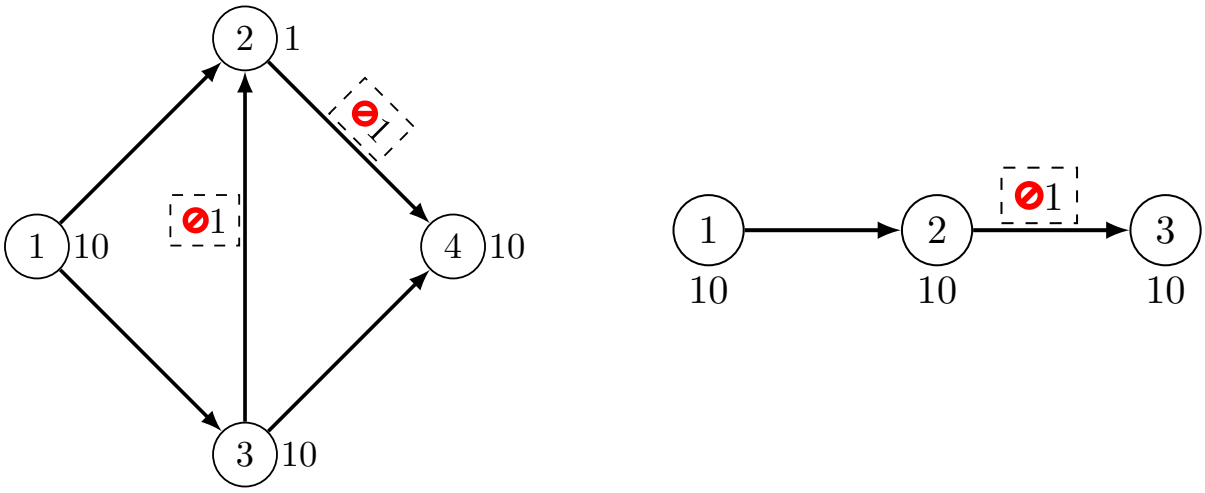


Figure K.1: Illustration of the sample inputs.

Sample Input 1

```
4
2 10
0 2
0 3
1 1
1 4 1
2 10
1 2 1
0 4
0 10
```

Sample Output 1

```
30
```

Sample Input 2

```
3
1 10
0 2
1 10
1 3 1
0 10
```

Sample Output 2

```
impossible
```

D: The Twilight Drone

The *Icelandic Corporation for Parcel Circulation* is the leading carrier for transporting goods between Iceland and the rest of the world. Their newest innovation is a drone link connecting to mainland Europe that has a number of drones travelling back and forth along a single route.

Drone by Hyeri Kim, Pixabay

The drones are equipped with a sophisticated system that allows them to fly evasive manoeuvres whenever two drones come close to each other. Unfortunately, a software glitch has caused this system to break down and now all drones are flying along the route with no way of avoiding collisions between them.

For the purposes of this problem, the drones are considered as points moving along an infinite straight line with constant velocity. Whenever two drones are at the same location, they will collide, causing them to fall off their flight path and plummet into the Atlantic Ocean. The flight schedule of the drones is guaranteed to be such that at no point will there be three or more drones colliding at the same location.

You know the current position of each drone as well as their velocities. Your task is to assess the damage caused by the system failure by finding out which drones will continue flying indefinitely without crashing.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of drones. The drones are numbered from 1 to n.
- n lines, the ith of which contains two integers x_i and v_i ($-10^9 \leq x_i, v_i \leq 10^9$), the current location and the velocity of the ith drone along the infinite straight line.

The drones are given by increasing x coordinate and no two drones are currently in the same position, i.e. $x_i < x_{i+1}$ for each i. You may assume that there will never be a collision involving three or more drones.

Output

Output the number of drones that never crash, followed by the indices of these drones in numerically increasing order.

Sample Input 1	Sample Output 1
3 10 15 30 5 50 -1	1 3

Sample Input 2	Sample Output 2
6 0 3 2 2 3 1 4 3 5 2 6 3	2 1 6

E: Infinity War

The boardgame Chaos is an exotic variant of Chess, played by two players in alternating turns on an $n \times n$ playing board. All pieces have the same set of n valid moves which are agreed on ahead of the game.

Chess by jplenio, CC0 Public Domain

In a single turn a player can pick exactly one of their pieces and perform one of the following actions:

- Perform up to two valid moves using the chosen piece, capturing any piece that the chosen piece lands on along the way.
- Teleport the chosen piece to any cell on the board that is not already occupied by another piece.
- Leave the chosen piece untouched in its current cell.

Having recently discovered Chaos, Alice and Bob are currently in the endgame of a very exciting match. Each player has a single piece left on the board and there are only two turns left, with Alice going next.

Having analysed the situation, she realises that the only way she can win is to capture Bob's piece in her turn. If that is not possible, Alice may be able to force a tie if she can teleport her piece to a cell that Bob cannot capture in his turn. Otherwise Bob will be able to win by capturing Alice's piece, no matter what she does in her turn. Help Alice determine her optimal outcome.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 10^5$), the size of the playing board and the number of valid moves.
- One line with two integers a_x and a_y ($1 \leq a_x, a_y \leq n$), the column and row in which Alice's piece is currently located.
- One line with two integers b_x and b_y ($1 \leq b_x, b_y \leq n$), the column and row in which Bob's piece is currently located.
- n lines, the ith of which contains two integers x_i and y_i ($-n < x_i, y_i < n$) representing one of the valid moves. This moves the given piece x_i columns to the right and y_i rows up, provided this does not take the piece outside of the board.

Columns are numbered 1 to n from left to right and rows are numbered 1 to n from bottom to top. All valid moves are distinct.

Output

If Alice can capture Bob's piece in her turn, output "`Alice wins`".

If Alice can use her turn to force a tie by teleporting her piece to a cell that Bob cannot capture

in his turn output "`tie`" followed by two integers a'_x and a'_y, the location of any such cell. If there are multiple valid solutions, you may output any one of them.

Otherwise, if Bob is able to capture Alice's piece no matter what she does in her turn, output "`Bob wins`".

Sample Input 1

```
2
2 1
1 2
1 0
0 -1
```

Sample Output 1

```
Bob wins
```

Sample Input 2

```
3
2 3
1 3
-2 1
1 1
1 0
```

Sample Output 2

```
tie 3 1
```

Sample Input 3

```
4
1 1
3 4
0 3
2 0
0 -3
-2 0
```

Sample Output 3

```
Alice wins
```

F: Domiyes

Marie likes Dominoes. She is too young to fully understand the game, so she just creates arrangements based on the following simple rule: Each of the two ends of a domino must be adjacent to an end of another domino with the same number on it.

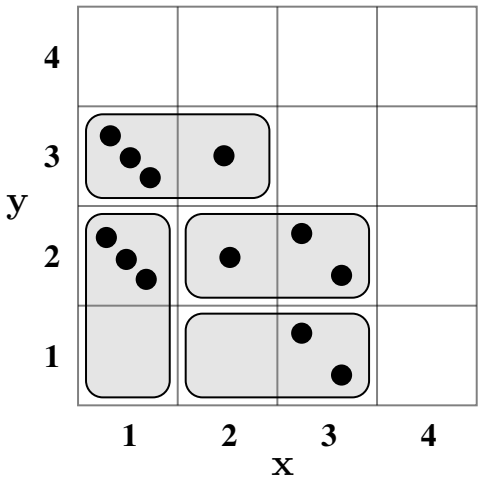


Figure D.1: Visualization of the first sample test case.

Today, Marie found a large box of blank dominoes. This is very exciting for her because now she can show her full creativity by first creating an unrestricted arrangement and then, in a second step, painting numbers on both ends of all dominoes so that her simple rule is fulfilled.

She already decided that putting the same number on each end of every domino is not satisfying enough for her. She only wants to use each number at most twice. However, she does not restrict herself to numbers between 0 and 6, and she also does not care if two dominoes have the same pair of numbers on them.

Marie positions the dominoes along an integer grid, so that each domino occupies exactly two neighbouring grid squares. Note that Marie's arrangement does not necessarily have to be connected.

After Marie decided on an arrangement, she notices that choosing suitable numbers is harder than initially anticipated. Help her to find a valid numbering for her given arrangement or state that this is impossible.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 5\,000$), the number of dominoes in Marie's arrangement.
- n lines, each with four integers x_1, y_1, x_2, y_2 ($1 \leq x_1, y_1, x_2, y_2 \leq 10\,000$), where (x_1, y_1) and (x_2, y_2) are the grid positions of the two ends of one of the dominoes.

It is guaranteed that all dominoes occupy two neighbouring positions in the integer grid and no two dominoes overlap.

Output

If a valid numbering exists, print n lines, the ith of which contains two numbers, the integers that Marie should write on the two ends of the ith domino, respectively. Output the numbers in

the same order as the dominoes (including their two ends) appear in the input. All numbers in the output should be integers between 0 and 10^6 inclusive. In case multiple valid numberings exist, you may output any one of them. If there does not exist a valid numbering, output `impossible` instead.

Sample Input 1

```
4
1 1 1 2
2 2 3 2
2 1 3 1
1 3 2 3
```

Sample Output 1

```
0 3
1 2
0 2
3 1
```

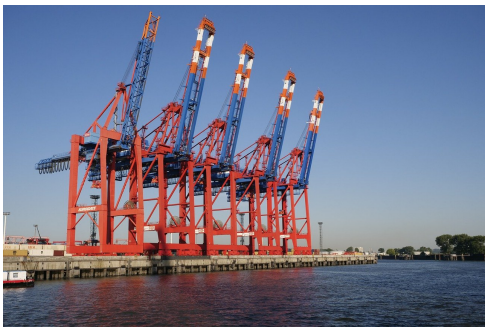
Sample Input 2

```
4
1 1 2 1
1 2 2 2
4 2 4 3
4 4 3 4
```

Sample Output 2

```
impossible
```

G: Safe Harbour



Gantry cranes by wasi1370 on Pixabay

There are two gantry cranes operating on the same gantry of length n. The gantry has some fixed integral positions, labelled from 1 to n, at which the cranes must perform loading/unloading operations. In the beginning the first gantry crane is located on the very left of the gantry at position 1, while the second one is located on the very right of the gantry at position n. In each time step a gantry crane can either move to a neighbouring integral position or stay at its current position (and potentially perform a loading/unloading operation). To prevent the gantry cranes from hitting each other, the first crane needs to stay strictly to the left of the second crane at all times. For both cranes you are given a task list consisting of gantry positions at which the cranes must perform loading/unloading operations. Both cranes must perform their assigned operations in the given order. What is the minimal amount of time necessary for both gantry cranes to finish their tasks? It is guaranteed that the first gantry crane never has to operate at position n of the gantry while the second gantry crane never has to operate at position 1. For both gantry cranes the first and last loading/unloading operation in the task list is their initial position.

Input

The input consists of:

- One line with integers n, a and b where
  - n ($2 \leq n \leq 2\,000$) is the length of the gantry;
  - a ($2 \leq a \leq 50$) is the number of operations in the task list of the first gantry crane;
  - b ($2 \leq b \leq 50$) is the number of operations in the task list of the second gantry crane.
- One line with a integers $k_1, \ldots, k_a$ ($1 \leq k_i \leq n-1$ for all i), the tasks of the first gantry crane.
- One line with b integers $\ell_1, \ldots, \ell_b$ ($2 \leq \ell_i \leq n$ for all i), the tasks of the second gantry crane.

The first and last task of both gantry cranes are at their initial position, i.e., $k_1 = k_a = 1$ and $\ell_1 = \ell_b = n$.

Output

Output the minimum number of time steps necessary for both gantry cranes to finish their assigned tasks.

Sample Explanation

In the first sample test case the gantry is of length 3, the first gantry crane has 2 operations in its task list while the second gantry crane has 4 operations in its task list. At least 6 time steps are necessary for both gantry cranes to finish their assigned tasks.

Time	Gantry Crane 1	Gantry Crane 2
1	Operate at 1	Operate at 3
2	Operate at 1	Operate at 3
3	Idle at 1	Move from 3 to 2
4	Idle at 1	Operate at 2
5	Idle at 1	Move from 2 to 3
6	Idle at 1	Operate at 3

In the second sample test case the gantry is of length 4 and both gantry cranes have to perform 4 operations. At least 9 time steps are necessary for both gantry cranes to finish their assigned tasks.

Time	Gantry Crane 1	Gantry Crane 2
1	Operate at 1	Operate at 4
2	Move from 1 to 2	Move from 4 to 3
3	Operate at 2	Operate at 3
4	Move from 2 to 3	Move from 3 to 4
5	Operate at 3	Idle at 4
6	Move from 3 to 2	Move from 4 to 3
7	Move from 2 to 1	Operate at 3
8	Operate at 1	Move from 3 to 4
9	Idle at 1	Operate at 4

Sample Input 1

```
3 2 4
1 1
3 3 2 3
```

Sample Output 1

```
6
```

Sample Input 2

```
4 4 4
1 2 3 1
4 3 3 4
```

Sample Output 2

```
9
```

H: Black Mole Son

The mole family recently decided to dig a new tunnel network. The layout, which has already been decided, consists of chambers and bidirectional tunnels connecting them, forming a connected graph. Mother mole wants to use the opportunity to teach her two mole kids how to dig a tunnel network.

Mole by Ahmad Kanbar, Unsplash

As an initial quick demonstration, mother mole is going to start by digging out a few of the chambers and tunnels, in the form of a non-self-intersecting path in the planned tunnel network. She will then divide the remaining chambers between the two mole kids, making sure that each mole kid has to dig out the same number of chambers, or else one of the mole kids will become sad. (The tunnels are much easier to dig out, and thus of no concern.) The kids may work on their assigned chambers in any order they like.

Since the mole kids do not have much experience with digging tunnel networks, mother mole realises one issue with her plan: if there is a tunnel between a pair of chambers that are assigned to different mole kids, there is a risk of an accident during the excavation of that tunnel if the other mole kid happens to be digging in the connecting chamber at the same time.

Help mother mole decide which path to use for her initial demonstration, and how to divide the remaining chambers evenly, so that no tunnel connects a pair of chambers assigned to different mole kids. The initial path must consist of at least one chamber and must not visit a chamber more than once.

Input

The input consists of:

- One line with two integers c and t ($1 \leq c \leq 2 \cdot 10^5$, $0 \leq t \leq 2 \cdot 10^5$), the number of chambers and tunnels in the planned tunnel network.
- t lines, each containing two integers a and b ($1 \leq a, b \leq c$, $a \neq b$), describing a bidirectional tunnel between chambers a and b.

The chambers are numbered from 1 to c. There is at most one tunnel between any pair of chambers, and there exists a path in the network between any pair of chambers.

Output

First output two integers p and s, the number of chambers on the path in mother mole's initial demonstration and the number of chambers each mole kid has to dig out. Then output a line containing the p chambers in mother mole's initial path, in the order that she digs them out. Then output two more lines, each containing the s chambers that the respective mole kid has to dig out, in any order.

The input is chosen such that there exists at least one valid solution. If there are multiple valid solutions, you may output any one of them.

Sample Input 1

```
3 2
3 1
2 1
```

Sample Output 1

```
3 0
3 1 2
```

Sample Input 2

```
4 3
1 3
2 3
3 4
```

Sample Output 2

```
2 1
3 4
2
1
```

Sample Input 3

```
7 7
1 2
2 3
4 2
2 5
4 5
6 7
7 2
```

Sample Output 3

```
3 2
5 2 7
6 1
3 4
```

K: Teardown

You are tasked with bulldozing some buildings that stand along a long, straight road. The buildings are modelled as evenly spaced stacks of identical square blocks along an infinite line. Your powerful bulldozer is capable of moving any one of these blocks one unit of distance to the left or to the right. This may push other blocks out of the way, and blocks which sit atop moving blocks will move along. Blocks which are pushed over a gap fall down until they reach either the ground or another block.

For instance, consider the stacks of blocks shown on the left in Figure B.1 below. If you push the block labelled C to the right, the blocks D and E would be pushed along to the right, since they are in the way. Blocks A, B and F would also move along because they are sitting on top of moving blocks. After pushing C to the right, E would be sitting over a gap, so E and F drop down to fill that gap. The resulting stacks are shown in the middle of Figure B.1. Pushing block C one further step to the right would result in the configuration shown on the right.

Your goal is to *level* all the buildings: bulldoze until all stacks are of height at most 1, i.e., all blocks are on the ground. Note that the road stretches out infinitely far on either side, so this is always possible.

Given the initial heights of the stacks, determine the smallest number of moves you need to make to level all the buildings, where a move consists of using the bulldozer to push one block one step to the left or right.

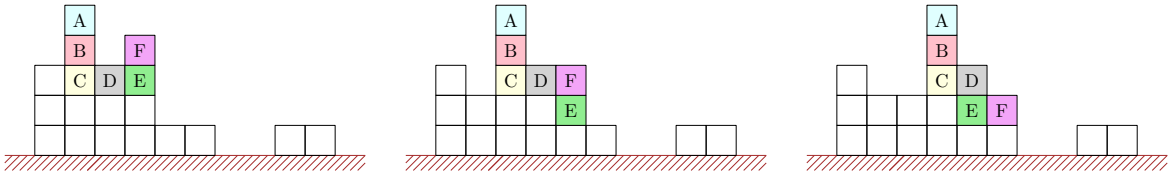


Figure B.1: Illustration of a configuration of stacks of blocks, and the results of pushing the block labelled C towards the right twice (the blocks labelled A–F are coloured and labelled only for illustrative purposes).

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of stacks of blocks.
- One line with n integers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^9$ for each i), the initial heights of the stacks from left to right.

The example shown on the left in Figure B.1 could be given by $3, 5, 3, 4, 1, 1, 0, 0, 1, 1$, but it could also be left- or right-padded with additional zeros.

Output

Output the minimum number of moves required to level every building.

Sample Input 1

```
5
1 1 2 1 1
```

Sample Output 1

```
2
```

Sample Input 2

```
5
1 4 3 1 1
```

Sample Output 2

```
7
```

Sample Input 3

```
9
1 0 0 0 6 0 0 0 1
```

Sample Output 3

```
5
```

Sample Input 4

```
10
1 3 0 0 1 9 1 1 1 1
```

Sample Output 4

```
13
```

L: Well Spoken

Richard and Janet are going on their first date. Richard has offered to meet her at home with his bicycle, and Janet tells him she will call when she is ready in 10 to 20 minutes. But Richard is an impatient person; while he could wait at home for Janet's signal, he might also leave early and travel around the neighbourhood for a bit, in order to minimise the time it takes him to reach her once she calls. Due to his impatience, once Richard is on his bicycle, he does not want to ride any slower than the legal speed limit, stop at intersections, or wait outside Janet's house (but he does not mind passing by Janet's house and returning to it later).

Given the directed graph representing the neighbourhood around Richard's and Janet's houses, Richard wants to devise a route around the neighbourhood (after an optional waiting period at his own house) which minimises the time that Janet has to wait in the worst case. He can travel for as long as he likes and visit each intersection as many times as he likes.

Janet will call Richard as soon as she is ready, and at that point Richard will take the shortest path to her that he can. Richard does not know exactly when Janet will be ready, but he knows it will be in somewhere between a and b minutes (not necessarily at a whole minute).

If Richard is passing through an intersection at the exact same instant Janet calls, the call is considered to happen before he chooses what to do at the intersection. For example, if he is passing by Janet's house at the moment she calls, he can immediately stop there and she does not have to wait for him at all.

It could happen that Janet never has to wait for w minutes, but that she might have to wait for $w - \epsilon$ minutes for arbitrarily small $\epsilon > 0$, if she calls Richard at some inopportune moment (say, nanoseconds after he has left an intersection). In this case, we still define the worst case waiting time to be w.

Input

The input consists of:

- One line with two integers a, b ($0 \leq a \leq b \leq 10^{12}$), indicating that Janet will be ready in at least a minutes and at most b minutes.
- One line with two integers n, m ($2 \leq n \leq m \leq 10^5$), the number of intersections and the number of roads in the neighbourhood. The intersections are numbered from 1 to n.
- m lines, each with three integers u, v and t ($1 \leq u, v \leq n$, $1 \leq t \leq 10^6$), indicating that there is a one-way road from intersection u to intersection v, and that it takes Richard exactly t minutes to travel along this road.

Richard's house is at intersection 1 and Janet's house is at intersection n. It is guaranteed that it is possible to travel from Richard's house to Janet's house, and that it is possible to exit each intersection through at least one road, even if that road just loops back to the same intersection.

Output

Output the time Janet has to wait in the worst case assuming she will be ready in at least a minutes and at most b minutes and Richard plans his route optimally.

It can be shown that the worst case waiting time is always an integer.

Sample Input 1

```
10 20
3 5
1 3 7
2 1 1
2 3 2
2 3 5
3 2 4
```

Sample Output 1

```
6
```

Sample Input 2

```
4 10
5 7
1 4 6
4 5 5
4 5 3
5 5 30
1 2 1
2 3 1
3 2 1
```

Sample Output 2

```
5
```

M: Awesome Games Done Quick

The classic video game "Prince of Python" comprises n levels, numbered from 1 to n. You are going to speedrun this game by finishing all of the levels as fast as possible, and you can beat them in any order that you want.

You enter each level equipped with one of $n + 1$ magical items. In the beginning you only have item 0 in your inventory. Once you beat a level, you get to keep the item numbered the same as that level. For example, on finishing level 5, you obtain a mighty *Gauntlet of 5 Fingers* you may equip thereafter instead of the less-acclaimed *Sword of 0 Damage* you always start out with.

Beating a level can take different amounts of time depending on which item you take into the level with you. Higher-numbered items are more powerful, so if playing by the rules it is always at least as fast to finish the level with a higher-numbered item as with a lower-numbered item.

However, each level also has a shortcut left in by the developers. The shortcut for a level can be accessed by applying a specific item in an unconventional way. By doing so you can finish the level as fast as, or even faster than, if you had used any of the other items.

How long will it take you to beat all of the levels of the game?

Input

The input consists of:

- One line containing an integer n ($1 \leq n \leq 2\,500$), the number of levels.
- n lines, describing the levels.

  The ith such line starts with two integers x_i and s_i ($0 \leq x_i \leq n$, $1 \leq s_i \leq 10^9$), the shortcut item for level i and the completion time for level i when using the shortcut.

  The remainder of the line has $n + 1$ integers $a_{i,0}, \ldots, a_{i,n}$ ($10^9 \geq a_{i,0} \geq a_{i,1} \geq \ldots \geq a_{i,n} \geq s_i$), where $a_{i,j}$ is the completion time for level i when playing by the rules using item j.

Output

Output the minimum time it takes to beat, in any order, all of the levels in the game.

Sample Input 1

```
3
1 1 40 30 20 10
3 1 95 95 95 10
2 1 95 50 30 20
```

Sample Output 1

```
91
```

Sample Input 2

```
4
4 4 5 5 5 5 5
4 4 5 5 5 5 5
4 4 5 5 5 5 5
4 4 5 5 5 5 5
```

Sample Output 2

```
17
```