NOI 2025

22 March 2025

Tasks	Time Limit	Memory Limit
Monsters	1 s	1024 MB
Thumper	1 s	1024 MB
Reachability	1 s	1024 MB
Robots	1 s	1024 MB
Flooding	3 s	1024 MB

Have Fun!

Task 1: `Monsters`

Penguinland is an infinite number line with n monsters. The i-th monster is initially at position $a[i]$ on the number line, with health $h[i]$. It is guaranteed that no two monsters share the same initial position.

Today, Brian the penguin wishes to defeat all the monsters! To defeat them, Brian has planted k mines at certain positions. The j-th mine is located at position $x[j]$. Detonating a mine **instantly destroys all monsters** standing on that position, and each mine can be detonated any number of times. However, each detonation costs 1 dollar. It is guaranteed that no two mines are planted at the same position.

In addition to detonating mines, Brian can also perform two types of operations:

- Move a monster left or right by 1 unit along the number line.
- Increase or decrease a monster's health by 1.

Each operation costs 1 dollar to execute.

A monster is considered defeated if its health reaches 0 or if it is destroyed by a mine. Help Brian find the minimum cost (in dollars) needed to defeat all the monsters.

Input format

Your program must read from standard input.

The first line of input contains two space-separated integers n and k.

The following n lines each contain two space-separated integers. The i-th of these lines contains $a[i]$ and $h[i]$.

The last line of input contains k space-separated integers $x[1], x[2], \ldots, x[k]$.

Output format

Your program must print to standard output.

Output a single integer, the minimum cost (in dollars) needed to defeat all the monsters.

The output should contain only a single integer. Do not print any additional text such as `Enter a number` or `The answer is.`

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \leq n, k \leq 200\,000$
- $1 \leq a[i], h[i] \leq 10^9$ for all $1 \leq i \leq n$
- $1 \leq x[i] \leq 10^9$ for all $1 \leq i \leq k$
- $a[i] \neq a[j]$ for all $i \neq j$
- $x[i] \neq x[j]$ for all $i \neq j$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	**Marks**	**Additional constraints**
0	0	Sample test cases
1	14	$k = 1$
2	6	$k = 2$
3	10	$n, k \leq 18$
4	30	$n, k \leq 3000$
5	29	$h[i] = 10^9$
6	11	No additional constraints

Sample Test Case 1

This test case is valid for subtasks 1, 3, 4, and 6.

Input	Output

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

Sample Test Case 1 Explanation

There are $n = 3$ monsters and $k = 1$ mine. Brian can:

- Decrease the health of monster 1 to 0, costing 2 dollars.
- Move monster 2 to the right by 1 unit (changing its position from 4 to 5), costing 1 dollar.
- Detonate the mine at position 5, defeating both monsters 2 and 3, costing 1 dollar.

The total cost needed is $2 + 1 + 1 = 4$ dollars.

Sample Test Case 2

This test case is valid for subtasks 2, 3, 4, and 6.

Input	Output

`5 2`
`7 7`
`6 3`
`10 4`
`4 4`
`9 1`
`7 10`
	`7`

Sample Test Case 2 Explanation

There are $n = 5$ monsters and $k = 2$ mines. Brian can:

- Decrease the health of monster 5 to 0, costing 1 dollar.
- Detonate mine 2, defeating monster 3, costing 1 dollar.
- Move monster 2 to the right by 1 unit (changing its position from 6 to 7), costing 1 dollar.
- Move monster 4 to the right by 3 units (changing its position from 4 to 7), costing 3 dollars.
- Detonate mine 1, defeating monsters 1, 2, and 4, costing 1 dollar.

The total cost needed is $1 + 1 + 1 + 3 + 1 = 7$ dollars.

Sample Test Case 3

This test case is valid for subtasks 3, 4, and 6.

Input	Output

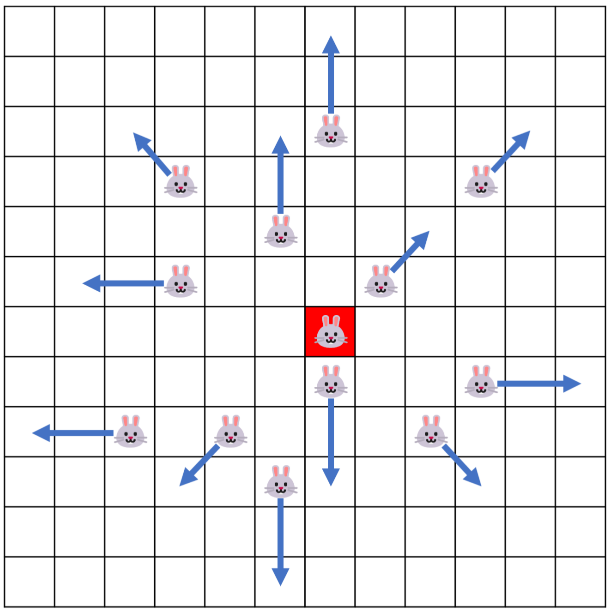
`10 5`
`19 10`
`5 3`
`1 2`
`3 6`
`17 2`
`20 3`
`8 2`
`12 3`
`14 2`
`15 1`
`40 13 37 14 6`
	`23`

Task 2: `Thumper`

Bunnyland has large fields where the Bunnyland Dwarf (a native species of rabbit) roams freely. One such field can be modelled as a $10^9 \times 10^9$ grid of cells. The rows of the grid are numbered 1 to 10^9 from north to south, and the columns of the grid are numbered 1 to 10^9 from west to east. We refer to the cell located at row r and column c of the grid as cell (r, c).

In this field, there are n rabbits numbered from 1 to n. The i-th rabbit is initially located at cell $(r[i], c[i])$. **No two rabbits are initially located at the same cell**.

Rabbits lift their hind legs and kick the ground when annoyed, an action known as thumping. These n rabbits will execute a sequence of m thumps. At the start of the j-th second, rabbit $t[j]$ will thump. When a rabbit thumps, all other rabbits will move away from the thumping rabbit.



To be precise, when rabbit A thumps, rabbit B will move as follows:

- If the number of rows between A and B is less than the number of columns between A and B, B will move two columns away from A.
- If the number of rows between A and B is equal to the number of columns between A and B, B will move one column and one row away from A.
- If the number of rows between A and B is more than the number of columns between A and B, B will move two rows away from A.

It can be shown that the location of the rabbits will remain distinct after a thump has occurred.

Benson the Rabbit has come to find his brethren after his retirement from investigating toxic bacteria, but all the thumping has caused the rabbits to scatter. Help Benson determine the final locations of the n rabbits after all thumps have occurred!

It is guaranteed that a rabbit will never leave the grid during the sequence of thumps. You may also assume that rabbits do not move in any other circumstances except thumping.

Input format

Your program must read from standard input.

The first line of input contains two space-separated integers n and m.

The following n lines of input each contain two space-separated integers. The i-th of these lines contains $r[i]$ and $c[i]$.

The last line of input contains m space-separated integers $t[1], t[2], \ldots, t[m]$.

Output format

Your program must print to standard output.

The output should contain n lines. The i-th of these lines should contain two space-separated integers, indicating the row and the column of rabbit i after all thumps have occurred.

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \leq n, m \leq 500\,000$
- $1 \leq r[i], c[i] \leq 10^9$ for all $1 \leq i \leq n$
- $1 \leq t[j] \leq n$ for all $1 \leq j \leq m$
- $(r_i, c_i) \neq (r_j, c_j)$ for all $i \neq j$
- It is guaranteed that a rabbit will never exit the grid during the sequence of thumps.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional constraints
0	0	Sample test cases
1	18	$n, m \leq 2000$
2	21	$r[i] = 1$
3	32	$n \leq 2000$
4	13	$n \leq 100\,000$
5	16	No additional constraints

Sample Test Case 1

This test case is valid for subtasks 1, 3, 4, and 5.

Input	Output

2 1
1 1
2 2
1
	1 1
	3 3

Sample Test Case 1 Explanation

Rabbit 1 is at cell $(1, 1)$ and rabbit 2 is at cell $(2, 2)$.

Since the number of rows between rabbit 1 and rabbit 2 is equal to the number of columns between rabbit 1 and rabbit 2, when rabbit 1 thumps, rabbit 2 will move one column and one row away from rabbit 1 in the southeast direction, landing at cell $(3, 3)$. The position of the thumping rabbit 1 does not change.

Sample Test Case 2

This test case is valid for subtasks 1, 3, 4, and 5.

Input	Output
`13 1`	`7 7`
`7 7`	`1 7`
`3 7`	`3 3`
`4 4`	`3 11`
`4 10`	`3 6`
`5 6`	`6 2`
`6 4`	`5 9`
`6 8`	`10 7`
`8 7`	`8 12`
`8 10`	`9 1`
`9 3`	`10 4`
`9 5`	`10 10`
`9 9`	`12 6`
`10 6`	
`1`	

Sample Test Case 2 Explanation

The diagram in the statement corresponds to this test case. The blue arrows show how the other rabbits move when rabbit 1 at cell $(7, 7)$ thumps.

Sample Test Case 3

This test case is valid for all subtasks.

Input	Output
`3 2`	`1 8`
`1 10`	`1 20`
`1 20`	`1 32`
`1 30`	
`1 3`	

Task 3: `Reachability`

Sheepland is a country with n cities. There are $n-1$ roads connecting pairs of cities with each other. Road j directly connects cities $u[j]$ and $v[j]$. Initially, it is possible to travel from any city to any other city using only these roads.

All $n-1$ roads in Sheepland are planned to be renovated. Under the renovation plan, each road j will be in one of four states:

1. Two-way: citizens from both cities $u[j]$ and $v[j]$ may cross this road into the other city.
2. One-way from city $u[j]$ to city $v[j]$: only citizens from city $u[j]$ may cross this road into city $v[j]$.
3. One-way from city $v[j]$ to city $u[j]$: only citizens from city $v[j]$ may cross this road into city $u[j]$.
4. Closed: no citizens from cities $u[j]$ or $v[j]$ may cross this road into the other city.

Unfortunately, the renovation plan has gone missing!

To try to recover it, you ask the mayor of each city how many cities are **reachable** from their city under the renovation plan. The mayor of the i-th city replies with $l[i]$. However, some mayors may have provided incorrect values.

A city v is considered **reachable** from a city u if there exists a sequence $c_1, c_2, c_3, \ldots, c_k$ where $c_1 = u$, $c_k = v$ and a crossable road exists from c_x to c_{x+1} for all $1 \leq x \leq k-1$. In particular, a city is reachable from itself.

Help Sheepland determine whether there exists a renovation plan that is consistent with the number of cities reachable from each city, as reported by all mayors!

Input format

Your program must read from standard input.

The first line of input contains one integer n.

The second line of input contains n space-separated integers $l[1], l[2], \ldots, l[n]$.

The following $n-1$ lines of input each contain two space-separated integers. The j-th of these lines contains $u[j]$ and $v[j]$.

Output format

Your program must print to standard output.

Output `YES` if a renovation plan that is consistent with the number of cities reachable from each city, as reported by all mayors, exists and `NO` otherwise.

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \leq n \leq 5000$
- $1 \leq l[i] \leq n$ for all $1 \leq i \leq n$
- $1 \leq u[j], v[j] \leq n$ for all $1 \leq j \leq n-1$
- $u[j] \neq v[j]$ for all $1 \leq j \leq n-1$
- Initially, it is possible to travel from any city to any other city using roads only.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample test cases
1	4	$n \leq 7$
2	5	$n \leq 15$
3	11	$l[1] = l[2] = \cdots = l[n]$
4	10	If a plan exists, at least one such plan has no two-way roads
5	45	$n \leq 400$
6	25	No additional constraints

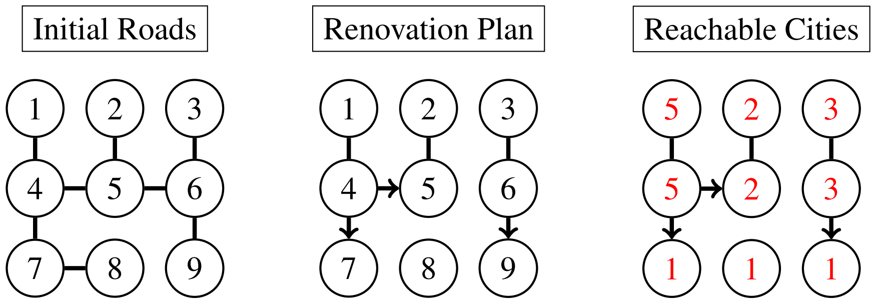
Sample Test Case 1

This test case is valid for subtasks 2, 5, and 6.

Input	Output
`9`	`YES`
`5 2 3 5 2 3 1 1 1`	
`1 4`	
`4 5`	
`2 5`	
`3 6`	
`5 6`	
`6 9`	
`7 8`	
`4 7`	

Sample Test Case 1 Explanation

Refer to the diagram below. The renovation plan is consistent with the number of cities reachable from each city, as reported by all mayors.



Sample Test Case 2

This test case is valid for subtasks 2, 4, 5, and 6.

Input	Output

9
5 2 3 5 2 3 1 1 2
1 4
4 5
2 5
3 6
5 6
6 9
7 8
4 7
	NO

Sample Test Case 2 Explanation

There does not exist a renovation plan consistent with the number of cities reachable from each city, as reported by all mayors.

Sample Test Case 3

This test case is valid for subtasks 1, 2, 5, and 6.

Input	Output

7
3 3 1 3 2 1 2
3 4
1 2
6 2
7 3
5 6
4 2
	YES

Task 4: `Robots`

Funnyland is modelled as a grid of size $(h + 2) \times (w + 2)$. The rows of the grid are numbered 0 to $h + 1$ from top to bottom, and the columns of the grid are numbered 0 to $w + 1$ from left to right. We refer to the cell located at row r and column c of the grid as cell (r, c).

Initially, all cells in this grid are coloured *white*, except for the rightmost column of cells, which are all coloured *black*.

The columns 1 to w each contain exactly one special cell. Each of these special cells are to be coloured *red* or *blue*. The edges of the grid (i.e., the topmost row, the bottommost row, the leftmost column, and the rightmost column) will never contain special cells.

Several robots will be placed on cells in the leftmost column and will move according to the colour of their cell:

- If the cell is white, the robot moves right.
- If the cell is red, the robot moves upward.
- If the cell is blue, the robot moves downward.
- If the cell is black, the robot does not move.

Robots do not collide with each other; each robot moves independently of other robots. Multiple robots are allowed to occupy the same cell.

A query consists of two integers a and b ($1 \leq a < b \leq h$). For each $a \leq c \leq b$, there will be a robot starting at $(c, 0)$. Across all possible configurations of colouring the special cells red or blue, determine the minimum number of distinct final cells the robots can occupy.

Note that different queries may result in different configurations of colouring the cells red and blue.

Input Format

Your program must read from standard input.

The first line of input contains two space-separated integers h and w.

The second line of input contains w space-separated integers $x[1], x[2], \ldots, x[w]$, indicating that the special cells are $(x[1], 1), (x[2], 2), \ldots, (x[w], w)$.

The third line of input contains one integer q.

The following q lines of input each contain two space-separated integers. The i-th of these lines contains $a[i]$ and $b[i]$.

Output format

Your program must print to standard output.

The output should contain q lines. The i-th of these lines should contain one integer, the answer to the i-th query.

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \leq w, q \leq 200\,000$
- $2 \leq h \leq 200\,000$
- $1 \leq x[j] \leq h$ for all $1 \leq j \leq w$
- $1 \leq a[i] < b[i] \leq h$ for all $1 \leq i \leq q$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample test cases
1	10	$h, w \leq 16$, $q \leq 20$
2	4	$a[i] + 1 = b[i]$
3	12	$x[1] < x[2] < \cdots < x[w]$
4	43	$h, w, q \leq 5000$
5	31	No additional constraints

Sample Test Case 1

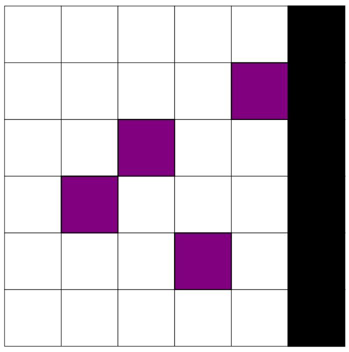
This test case is valid for subtasks 1, 4, and 5.

Input	Output

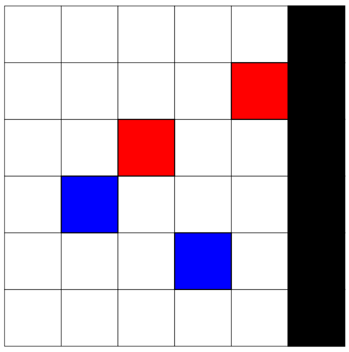
```
4 4
3 2 4 1
3
1 4
1 3
2 4
```
	```
	2
	1
	1
	```

Sample Test Case 1 Explanation

The grid is coloured as follows, with the special cells in violet.



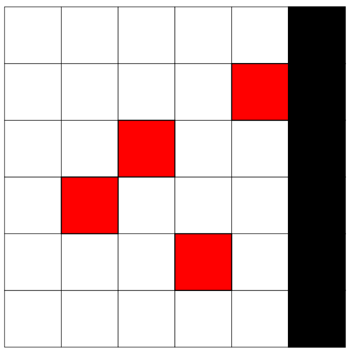
For the first query, we can colour the special cells at $(3, 1)$ and $(4, 3)$ blue, $(2, 2)$ and $(1, 4)$ red to achieve the following:



- The robot starting at $(1, 0)$ moves to $(1, 1), (1, 2), (1, 3), (1, 4), (0, 4), (0, 5)$, and ends at $(0, 5)$.
- The robot starting at $(2, 0)$ moves to $(2, 1), (2, 2), (1, 2), (1, 3), (1, 4), (0, 4), (0, 5)$, and ends at $(0, 5)$.
- The robot starting at $(3, 0)$ moves to $(3, 1), (4, 1), (4, 2), (4, 3), (5, 3), (5, 4)$, and ends at $(5, 5)$.
- The robot starting at $(4, 0)$ moves to $(4, 1), (4, 2), (4, 3), (5, 3), (5, 4)$, and ends at $(5, 5)$.

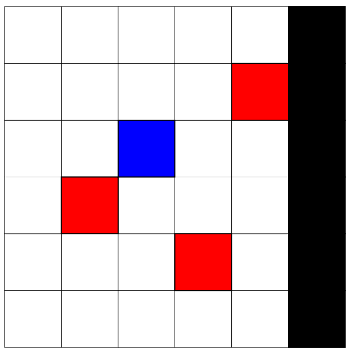
The robots end at 2 distinct cells, $(0, 5)$ and $(5, 5)$, hence the correct output is 2.

For the second query, we can colour the special cells as follows:



The robots starting at $(1, 0)$, $(2, 0)$, and $(3, 0)$ all end at $(0, 5)$.

For the third query, we can colour the special cells as follows:



The robots starting at $(2, 0)$, $(3, 0)$, and $(4, 0)$ all end at $(3, 5)$.

Sample Test Case 2

This test case is valid for subtasks 1, 4, and 5.

Input	Output
`15 16`	`2`
`5 15 3 4 13 8 3 7 14 6 2 10 11 12 9 1`	`2`
`20`	`3`
`4 10`	`2`
`7 15`	`2`
`5 15`	`3`
`2 6`	`1`
`7 15`	`1`
`5 15`	`3`
`12 15`	`1`
`13 14`	`3`
`5 14`	`2`
`13 14`	`1`
`2 11`	`1`
`3 11`	`3`
`2 5`	`3`
`9 11`	`4`
`3 15`	`1`
`5 14`	`2`
`1 13`	`1`
`3 7`	
`7 12`	
`4 8`	

Task 5: `Flooding`

Pavementland is a rectangle-shaped city, which can be modelled as a $h \times w$ grid of cells. The rows of the grid are numbered 1 to h from north to south, and the columns of the grid are numbered 1 to w from west to east. We refer to the cell located at row r and column c of the grid as cell (r, c).

In the grid, each cell is either empty or contains a building. At least one cell is empty.

Due to a monsoon surge, flash floods are occurring throughout Pavementland. Initially, one empty cell becomes flooded with water by the rain. Then, the water flows according to the following rules:

- If an empty cell is adjacent to at least one flooded cell, it becomes flooded.
- If a cell containing a building is adjacent to at least two flooded cells, the building collapses and the cell becomes flooded.

Note that a cell is adjacent to another cell if they share an edge. A cell is adjacent to at most four other cells. Further note that water may not flow outside the grid. Let $f((r, c))$ be the number of cells that would be flooded after the process if the cell (r, c) were initially flooded.

City officials are seeking to forecast the extent of flash floods in all possible scenarios. Help them determine the sum of $f((r, c))$ over all empty cells (r, c).

Input Format

Your program must read from standard input.

The first line of input contains two space-separated integers h and w.

The next h lines of input each contain a binary string of length w. If the c-th character of the r-th line is `0`, then the cell (r, c) is empty. If the c-th character of the r-th line is `1`, then the cell (r, c) contains a building.

Output Format

Your program must print to standard output.

Output a single integer, the sum of $f((r, c))$ over all empty cells (r, c).

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \leq h, w \leq 5000$
- There is at least one empty cell in the grid.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample test cases
1	5	$h = 1$
2	7	$h, w \leq 80$
3	16	$h, w \leq 500$
4	32	$h, w \leq 2000$
5	40	No additional constraints

Sample Test Case 1

This test case is valid for subtasks 2 to 5.

Input

```
3 3
000
011
010
```

Output

```
46
```

Sample Test Case 1 Explanation

If cells $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 1)$, or $(3, 1)$ were initially flooded, the entire grid would become flooded after the process. If cell $(3, 3)$ were initially flooded, only 1 cell would become flooded after the process. Hence, the output is $9 + 9 + 9 + 9 + 9 + 1 = 46$.

Sample Test Case 2

This test case is valid for subtasks 2 to 5.

Input	Output
`5 5`	`182`
`00101`	
`01011`	
`11010`	
`01101`	
`11000`	

Sample Test Case 3

This test case is valid for all subtasks.

Input	Output
`1 10`	`6`
`1101011100`	