# Collisions

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

You are given $N$ columns numbered $1, 2, \ldots N$ from left to right. For each $i$, the $i^{\text{th}}$ column contains $c_i$ objects. For each $1 \leq j \leq c_i$, the $j^{\text{th}}$ object in the $i^{\text{th}}$ column is positioned at a height of $pos_{i,j}$. The positions (heights) of the objects in any **particular** column are all **distinct**. Your task is to assign a direction to each column: either *left* or *right*.

- The *leftmost column* can only move *right*, and the *rightmost column* can only move *left*.
- If a column $i$ is assigned the *left* direction, all objects in that column move to the **immediate** *left neighbor column* $(i-1)$, while retaining their height.
- If a column $i$ is assigned the *right* direction, all objects in that column move to the **immediate** *right neighbor column* $(i+1)$, while retaining their height.

In this process, two objects are said to "collide" if they:

1. cross each other, **or**
2. end up at the same position in the same column.

Please note that, an object may collide with multiple others.

Your goal is to assign directions to the columns, such that the total number of collisions is minimized.

## Input

- The first line contains a single integer $N$ $(2 \leq N \leq 10^4)$, the number of columns.
- The second line contains $N$ integers $c_1, c_2, \ldots c_N$, $(1 \leq \sum c_i \leq 10^6)$, where $c_i$ represents the number of objects in the $i^{\text{th}}$ column.
- The next $N$ lines describe the heights of the objects in each column:
  - The $i^{\text{th}}$ line contains $c_i$ **distinct** integers $pos_{i,1}, pos_{i,2}, ..., pos_{i,c_i}$ $(1 \leq pos_{i,j} \leq 10^9)$, the heights of the objects in the $i^{\text{th}}$ column.

## Output

Output a single integer: the minimum possible number of collisions.

## Examples

| standard input | standard output |
|---|---|
| 3<br>2 3 2<br>1 5<br>2 4 6<br>1 5 | 2 |
| 2<br>2 2<br>1 3<br>2 4 | 0 |
| 4<br>2 2 2 2<br>1 5<br>2 6<br>3 7<br>4 8 | 0 |
| 4<br>1 1 1 1<br>1<br>1<br>1<br>1 | 2 |

## Note

In the first example, an optimal solution is:

1. Column 1 objects move *right*, column 2 objects move *left*, column 3 objects move *left*.
2. Objects from column 1 and column 3 end up at the same position in column 2. Hence, there are a total of two collisions.

In the second example, an optimal solution is:

1. Objects in column 1 move to the right, and objects in column 2 move to the left.
2. Objects in the two columns are at different heights, and hence there are no collisions.

For the third example, an optimal solution is Column 1 right, Column 2 right, Column 3 left, Column 4 left.

In the fourth example, an optimal solution is:

1. Column 1 moves right, Column 2 moves right, Column 3 moves right, Column 4 moves left.
2. The object in column 4 collides with the object that moves from column 3 and the object that moves from column 2.