Problem 1.

Thanos is hiding from Captain America in a maze full of N rooms connected by M gates. The maze is designed in such a way that each room leads to another room via gates. All connecting gates are unidirectional. Thanos is hiding only in those rooms which are accessible directly/indirectly through every other room in the maze.

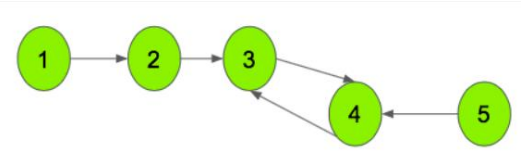
Help Captain America find the number of rooms in which Thanos can hide.

Example 1:

```
Input:
N = 5, M = 5 and V = [[1, 2], [2, 3], [3, 4], [4, 3], [5, 4]]

Output: 2

Explanation:
```




```
Thanos only can hide in a room 3 and 4 because they are the only
room which have gates through them. So, answer is 2.
```

Example 2:

```
Input:
N = 2, M = 1 and V = [[1, 2]]

Output: 1
```

Example 3:

```
Input:
N = 1, M = 1 and V = [[1, 1]]

Output: 1
```

Example 4:

```
Input:
N = 3, M = 0 and V = []

Output: 0
```

Constraints:

$1 \leq n \leq 300$

$1 \leq m \leq 2000$

$1 \leq p,q \leq n$

Problem 2.

You are given a square binary grid. A grid is considered binary if every value in the grid is either 1 or 0. You can change at most one cell in the grid from 0 to 1. You need to find the largest group of connected 1's. Two cells are said to be connected if both are adjacent (top, bottom, left, right) to each other and both have the same value.

Example 1:

```
Input:
grid = [[1, 1]
        [0, 1]]
Output: 4
Explanation: By changing cell (2,1), we can obtain a connected
group of 4 1's
[[1, 1]
[1, 1]]
```

Example 2:

```
Input:
grid = [[1, 0, 1]
        [1, 0, 1]
        [1, 0, 1]]
Output: 7
```

Example 3:

```
Input:
grid = [[1],
         [0],
         [1]]
Output: 3
```

Constraints:

```
1 <= size of the grid<= 500
0 <= grid[i][j] <= 1
```

Problem 3.

In this problem, a rooted tree is a **directed** graph such that, there is exactly one node (the root) for which all other nodes are descendants of this node, plus every node has exactly one parent, except for the root node which has no parents.

The given input is a directed graph that started as a rooted tree with `n` nodes (with distinct values from `1` to `n`), with one additional directed edge added. The added edge has two different vertices chosen from `1` to `n`, and was not an edge that already existed.

The resulting graph is given as a 2D-array of `edges`. Each element of `edges` is a pair $[u_i, v_i]$ that represents a directed edge connecting nodes u_i and v_i, where u_i is a parent of child v_i.

Return an edge that can be removed so that the resulting graph is a rooted tree of `n` nodes. If there are multiple answers, return the answer that occurs last in the given 2D-array.

Examples:

Modify and submit `functions.hpp`. For tests, you can use `main.cpp`.

Compile command: `g++ -o main main.cpp`

Execution command: `./main "given arrays"`

Example 1:

```
>> ./main "[[1,2],[1,3],[2,3]]"
>> [2,3]
>>
```

Example 2:

```
>> ./main "[[1,2],[2,3],[3,4],[4,1],[1,5]]"
>> [4,1]
>>
```

Problem 4.

Given an array `arr[]` with non-negative integers representing the height of blocks. If width of each block is 1, compute how much water can be trapped between the blocks during the rainy season.

Examples:

Modify and submit `functions.hpp`. For tests, you can use `main.cpp`.

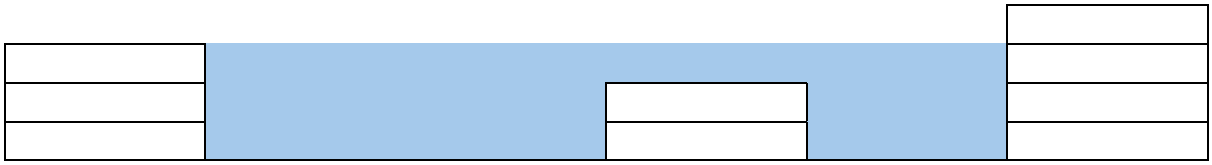
Compile command: `g++ -o main main.cpp`

Execution command: `./main "given array"`

Example 1:

```
>> ./main "3 0 0 2 0 4"
>> 10
>>
```

Explanation:



Example 2:

```
>> ./main "7 4 0 9"
>> 10
>>
```

Explanation: