Data Structures and Algorithms (CS F211)
Second Semester 2018-2019
Lab Sheet 7

1. Alice and Bob are playing the tag game. The game should be played on an undirected rooted tree of ***n*** vertices. Vertex 1 is the root of the tree. Alice starts at vertex 1 and Bob starts at vertex ***x*** (***x*** ≠ 1). The moves are made in turns, Bob goes first. In one move, one can either stay at the current vertex or travel to a neighbouring one. The game ends when Alice reaches the same vertex where Bob is standing. Alice wants to minimize the total number of moves and Bob wants to maximize it. Determine how many moves the game will last.

Input:
The first line contains two integer numbers ***n*** and ***x***.
Each of the next ***n - 1*** lines contain two integer numbers ***a*** and ***b*** (***1 ≤ a, b ≤ n***) — edges of the tree. It is guaranteed that the edges will form a valid tree.

Output:
Print the total number of moves Alice and Bob will make.

Example:
Input:
```
4 3
1 2
2 3
2 4
```

Output:
```
4
```

Explanation:
Bob: stay at vertex 3
Alice: go to vertex 2
Bob: stay at vertex 3
Alice: go to vertex 3

2. Your city has ***n*** junctions. There are ***m*** one-way roads between the junctions. As a mayor of the city, you have to ensure the security of all the junctions. To ensure the security, you have to build some police checkposts. Checkposts can only be built in a junction. A checkpost at junction ***i*** can protect junction ***j*** if either ***i = j*** or the police patrol car can go to ***j*** from ***i*** and then come back to ***i***. Building checkposts costs some money. As some areas of the city are more expensive than others, building checkposts at some junctions might require more money than other junctions. You have to determine the minimum possible money needed to ensure the security of all the junctions. Also, you have to find the number of ways in which security can be ensured in minimum price and also using minimum number of checkposts. Two ways are different if any of the junctions contains a checkpost in one of them and do not contain in the other.

Input:
In the first line, you will be given an integer ***n***, number of junctions.
In the next line, ***n*** space-separated integers will be given. The ***i***th integer is the cost of building a checkpost at the ***i***th junction.
The next line will contain an integer ***m***.

Each of the next ***m*** lines contains two integers u_i and v_i. A pair u_i, v_i means, that there is a one-way road which goes from u_i to v_i. There will not be more than one road between two nodes in the same direction.

Output:
Print two integers separated by spaces. The first one is the minimum possible money needed to ensure the security of all the junctions. The second one is the number of ways you can ensure the security with the minimum cost.

Example:
Input:
```
3
1 2 3
3
1 2
2 3
3 2
```

Output:
```
3 1
```

3. Let's consider a rooted undirected tree with ***n*** vertices, numbered ***1*** through ***n***. There are many ways to represent such a tree. One way is to create an array with ***n*** integers $p_1, p_2, \ldots, p_n$ where p_i denotes a parent of vertex ***i*** (for convenience a root is considered its own parent). Given a sequence $p_1, p_2, \ldots, p_n$, one is able to restore a tree subject to the following conditions:

i) There must be exactly one index ***r*** such that $p_r = r$. A vertex ***r*** is a root of the tree.
ii) For all other ***n - 1*** vertices, for every vertex ***i***, there is an edge between vertex ***i*** and vertex p_i.

A sequence $p_1, p_2, \ldots, p_n$ is called valid if the described procedure generates some (any) rooted tree. For example, for ***n*** = 3 sequences (1, 2, 2), (2, 3, 1) and (2, 1, 3) are not valid.
You are given a sequence $a_1, a_2, \ldots, a_n$ not necessarily valid. Your task is to change the minimum number of elements, in order to get a valid sequence using only the vertices from ***1*** to ***n***. Print the minimum number of changes and an example of a valid sequence after that number of changes. If there are more than one valid sequence achievable in the minimum number of changes, print any one of them.

Input:
The first line of the input contains an integer ***n*** — the number of vertices in the tree.
The second line contains ***n*** integers $a_1, a_2, \ldots, a_n$.

Output:
In the first line, print the minimum number of elements to be changed, in order to get a valid sequence.
In the second line, print any valid sequence possible to get from $a_1, a_2, \ldots, a_n$ in the minimum number of changes. If there are many such sequences, any of them is acceptab;e.

Example:
Input:
```
4
2 3 3 4
```
Output:
```
1
2 3 4 4
```

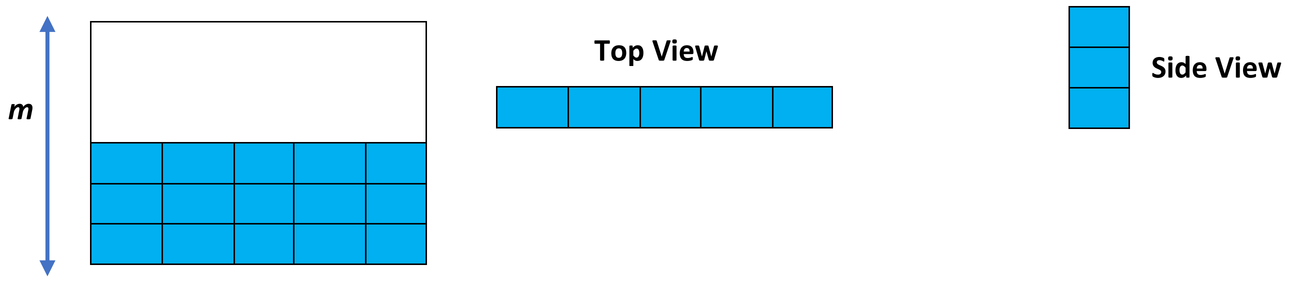
Comment: Note that 4 3 3 2 is also a valid tree, however, you end up changing two elements instead of one.

4. You came to an exhibition and one exhibit has drawn your attention. It consists of ***n*** stacks of blocks, where the ***i***th stack consists of $\boldsymbol{a_i}$ blocks resting on the surface. The height of the exhibit is equal to ***m***. Consequently, the number of blocks in each stack is less than or equal to ***m***. There is a camera on the ceiling that sees the top view of the blocks and a camera on the right wall that sees the side view of the blocks. Find the maximum number of blocks you can remove such that the views for both the cameras do not change. Note, that while originally all blocks are stacked on the floor, it is not required for them to stay connected to the floor after some blocks are removed. There is no gravity in the whole exhibition, so no block will fall down, even if the block underneath is removed. **It not allowed to move blocks from one stack to another, i.e., you are not allowed to move blocks across the stacks in order to preserve both the views.**

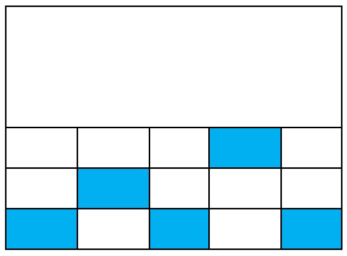
Input:
The first line contains two integers ***n*** and ***m***— the number of stacks and the height of the exhibit.
The second line contains ***n*** integers $\mathbf{a_1, a_2, \dots, a_n}$— the number of blocks in each stack from left to right.

Output:
Print exactly one integer — the maximum number of blocks that can be removed without changing the views.

Example1:
Input:
5 6
3 3 3 3 3



Output:
10



Explanation:
Here, total 10 blocks have been removed. As shown in the figure above, even after removal, from the side, 3 blocks can be seen and from the top, 5 blocks can be seen. The top camera does not care whether the 5 blocks are at the same height or not. Similarly, the side camera does not care whether the 3 blocks belong to the same stack or not.

Example 2:
Input:
3 5
1 2 4

Output:
3

5. You live in a town with ***n*** junctions and ***m*** bi-directional roads. Each road connects two distinct junctions and no two roads connect the same pair of junctions. It is possible to go from any junction to any other junction by these roads. The distance between two junctions is equal to the minimum possible number of roads on a path between them. Your city mayor wants to build a new road in such a way that the distance between the junctions ***s*** and ***t*** will not decrease. You are assigned a task to compute the number of pairs of junctions that are not connected by the road, such that if the new road between these two junctions is built the distance between ***s*** and ***t*** won't decrease.

Input:
The first line of the input contains integers ***n***, ***m***, ***s*** and ***t*** --- the number of junctions and the number of roads, as well as the indices of junctions ***s*** and ***t***.
The ***i***th of the following ***m*** lines contains two integers, u_i and v_i, meaning that this road connects junctions u_i and v_i directly.

Output:
Print one integer — the number of pairs of junctions not connected by a direct road, such that building a road between these two junctions won't decrease the distance between junctions ***s*** and ***t***.

Example1:
Input:
5 4 1 5
1 2
2 3
3 4
4 5

Output:
0

Explanation:
The pairs of junctions which are not connected by a direct road are: (1, 3), (1, 4), (1, 5), (2, 4), (2, 5) and (3, 5). If you build any road between any one of these city pairs, then the distance between 3 and 5 will reduce. That is why, no road can be built.

Example 2:
Input:
5 4 3 5
1 2
2 3
3 4
4 5

Output:
5

Explanation:
The pairs of junctions which are not connected by a direct road are: (1, 3), (1, 4), (1, 5), (2, 4), (2, 5) and (3, 5). Out of these, if you build roads between (1, 3), (1, 4), (1, 5), (2, 4) and (2, 5), then the distance between 3 and 5 will not reduce. That is why, the answer is 5.

6. Luka Doncic is a big fan of substring. His mentor Dirk knows that and decided to challenge him for that. He gave him a number ***n*** and a set of characters ***P***. Size of ***P*** is ***k***. Find a string ***S*** such that every possible string that can be constructed from the characters of ***P***, each having length ***n*** appears exactly once as a substring in ***S***. You have to do this in O(k^n) time.

Input:
n k
Array of size ***k***

Example:
Input:
3 2
0 1

Output:
0011101000

Example:
Input:
2 2
0 1

Output:
01100

7. You have to the count number of crossovers while sorting a set of unsorted elements.

Input:
The elements to be sorted

Output:
The number of crossovers required to sort the array.

Example:
Input:
3 2 1 4 5
Output: 3

Explanation:

```
Before Sorting:       3 2 1 4 5
                      \ | /  | |
                       \|/   | |
                      / | \  | |
After sorting:        1 2 3 4 5
```

line (1 to 1) crosses line (2 to 2)
line (1 to 1) crosses line (3 to 3)
line (2 to 2) crosses line (3 to 3)

8. Naruto is now participating in Chunin Exams. He needs to solve a riddle to get a secret weapon. He is given a number ***n*** and he needs to find a set of numbers that are smaller than ***n*** but follow an *adjacent_one* condition*.* *adjacent_one* condition is fulfilled if all the digits in the number differ from their respective adjacent digits by 1. For example, 456 follows *adjacent_one* condition but 782 doesn't.

Note: All single digit numbers fulfill *adjacent_one* condition.

Note: Difference between 9 and 0 is not 1.

Input:
n

Output:
The set of numbers satisfying the condition

Example 1:
Input:
20

Output:
0 1 2 3 4 5 6 7 8 9 10 12

Example 2:
Input:
105

Output:
0 1 2 3 4 5 6 7 8 9 10 12 21 23 32 34 43 45 54 56 65 67 76 78 87 89 98 101

9. Mohit is a traveller. He travels all around the world collecting non-negative numbers. After completing all of his journeys, he decides to form the largest number from the list of numbers. Help Mohit to fulfil his quest.

Input:
The first line contains the total number of non-negative numbers collected.
The second line contains the non-negative numbers.

Output:
The largest number.

Example:
Input:
5
3 30 34 5 9

Output:
9534330

****