A. Another String Minimization Problem

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a sequence $a_1, a_2, \ldots, a_n$ of length n, consisting of integers between 1 and m. You also have a string s, consisting of m characters B.

You are going to perform the following n operations.

- At the i-th ($1 \le i \le n$) operation, you replace either the a_i-th **or** the $(m + 1 - a_i)$-th character of s with A. You can replace the character at any position multiple times through the operations.

Find the lexicographically smallest string you can get after these operations.

A string x is lexicographically smaller than a string y of the same length if and only if in the first position where x and y differ, the string x has a letter that appears earlier in the alphabet than the corresponding letter in y.

Input

The first line contains the number of test cases t ($1 \le t \le 2000$).

The first line of each test case contains two integers n and m ($1 \le n, m \le 50$) — the length of the sequence a and the length of the string s respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$) — the sequence a.

Output

For each test case, print a string of length m — the lexicographically smallest string you can get. Each character of the string should be either capital English letter A or capital English letter B.

Example

input
6 4 5 1 1 3 1 1 5 2 4 1 1 1 1 1 2 4 1 3 2 7 7 5 4 5 5 5 3 5
output
ABABA BABBB A AABB ABABBBB ABABA

Note

In the first test case, the sequence $a = [1, 1, 3, 1]$. One of the possible solutions is the following.

- At the 1-st operation, you can replace the 1-st character of s with A. After it, s becomes ABBBB.
- At the 2-nd operation, you can replace the 5-th character of s with A (since $m + 1 - a_2 = 5$). After it, s becomes ABBBA.
- At the 3-rd operation, you can replace the 3-rd character of s with A. After it, s becomes ABABA.
- At the 4-th operation, you can replace the 1-st character of s with A. After it, s remains equal to ABABA.

The resulting string is ABABA. It is impossible to produce a lexicographically smaller string.
In the second test case, you are going to perform only one operation. You can replace either the 2-nd character or 4-th character of s with A. You can get strings BABBB and BBBAB after the operation. The string BABBB is the lexicographically smallest among these strings.

In the third test case, the only string you can get is A.

In the fourth test case, you can replace the 1-st and 2-nd characters of s with A to get AABB.

In the fifth test case, you can replace the 1-st and 3-rd characters of s with A to get ABABBBB.

B. Making Towers

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a sequence of n colored blocks. The color of the i-th block is c_i, an integer between 1 and n.

You will place the blocks down in sequence on an infinite coordinate grid in the following way.

1. Initially, you place block 1 at $(0, 0)$.
2. For $2 \leq i \leq n$, if the $(i-1)$-th block is placed at position (x, y), then the i-th block can be placed at one of positions $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ (**but not at position** $(x, y-1)$), as long no previous block was placed at that position.

A *tower* is formed by s blocks such that they are placed at positions $(x, y), (x, y+1), \ldots, (x, y+s-1)$ for some position (x, y) and integer s. The *size* of the tower is s, the number of blocks in it. A *tower of color* r is a tower such that all blocks in it have the color r.

For each color r from 1 to n, solve the following problem **independently**:

- Find the maximum size of a tower of color r that you can form by placing down the blocks according to the rules.

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 10^5$).

The second line of each test case contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq n$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers. The r-th of them should be the maximum size of an tower of color r you can form by following the given rules. If you cannot form any tower of color r, the r-th integer should be 0.

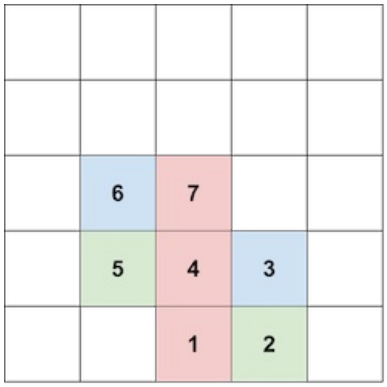
Example

input
6 7 1 2 3 1 2 3 1 6 4 2 2 2 4 4 1 1 5 5 4 5 3 5 6 3 3 3 1 3 3 8 1 2 3 4 4 3 2 1
output
3 2 2 0 0 0 0 0 3 0 2 0 0 1 0 0 1 1 1 1 0 4 0 0 0 2 2 2 2 0 0 0 0

Note

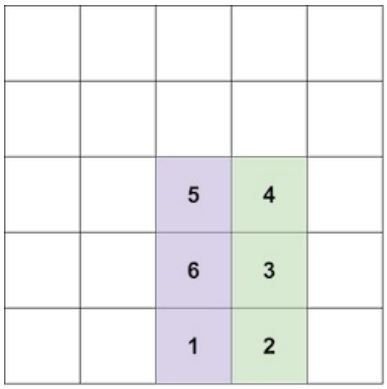
In the first test case, one of the possible ways to form a tower of color 1 and size 3 is:

- place block 1 at position $(0, 0)$;
- place block 2 to the right of block 1, at position $(1, 0)$;
- place block 3 above block 2, at position $(1, 1)$;
- place block 4 to the left of block 3, at position $(0, 1)$;
- place block 5 to the left of block 4, at position $(-1, 1)$;
- place block 6 above block 5, at position $(-1, 2)$;
- place block 7 to the right of block 6, at position $(0, 2)$.



The blocks at positions $(0, 0)$, $(0, 1)$, and $(0, 2)$ all have color 1, forming an tower of size 3.

In the second test case, note that the following placement is **not valid**, since you are not allowed to place block 6 under block 5:



It can be shown that it is impossible to form a tower of color 4 and size 3.

C. Qpwoeirut And The City

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Qpwoeirut has taken up architecture and ambitiously decided to remodel his city.

Qpwoeirut's city can be described as a row of n buildings, the i-th ($1 \le i \le n$) of which is h_i floors high. You can assume that the height of every floor in this problem is equal. Therefore, building i is taller than the building j if and only if the number of floors h_i in building i is larger than the number of floors h_j in building j.

Building i is *cool* if it is taller than both building $i - 1$ and building $i + 1$ (and both of them exist). Note that neither the 1-st nor the n-th building can be cool.

To remodel the city, Qpwoeirut needs to maximize the number of cool buildings. To do this, Qpwoeirut can build additional floors on top of any of the buildings to make them taller. Note that he cannot remove already existing floors.

Since building new floors is expensive, Qpwoeirut wants to minimize the number of floors he builds. Find the minimum number of floors Qpwoeirut needs to build in order to maximize the number of cool buildings.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains the single integer n ($3 \le n \le 10^5$) — the number of buildings in Qpwoeirut's city.

The second line of each test case contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$) — the number of floors in each of the buildings of the city.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the minimum number of additional floors Qpwoeirut needs to build in order to maximize the number of cool buildings.

Example

input
6 3 2 1 2 5 1 2 1 4 3 6 3 1 4 5 5 2

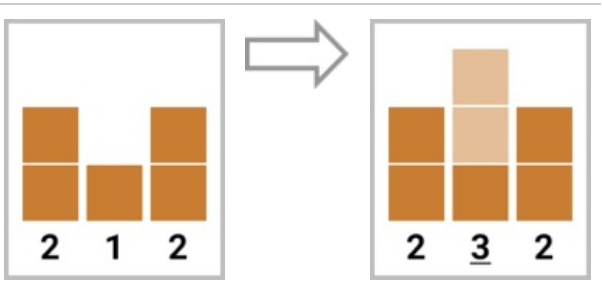
```
8
4 2 1 3 5 3 6 1
6
1 10 1 1 10 1
8
1 10 11 1 10 11 10 1
```

output

```
2
0
3
3
0
4
```

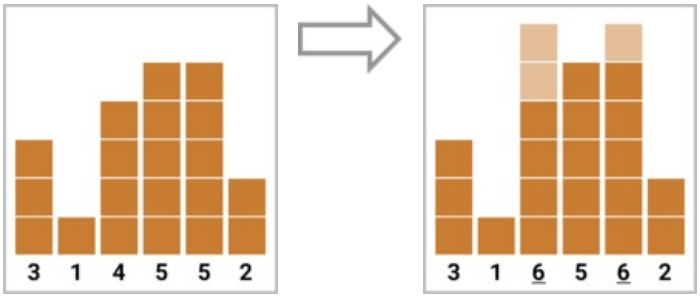
Note

In the first test case, it is optimal for Qpwoeirut to make the second building cool by building 2 additional floors on top of it, making it taller than both of its adjacent buildings. The final heights of buildings will be $[2, \underline{3}, 2]$.



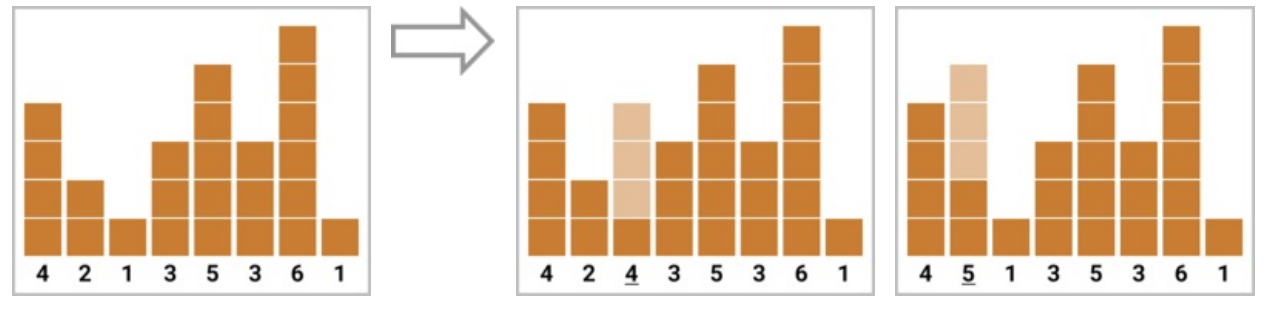
In the second test case, the number of cool buildings is already maximized, so Qpwoeirut does not need to do anything.

In the third test case, it is optimal for Qpwoeirut to make the third and fifth buildings cool by building 2 additional floors onto the third building and 1 additional floor onto the fifth building. The final heights of buildings will be $[3, 1, \underline{6}, 5, \underline{6}, 2]$.



It can be shown that it is impossible to make more than 2 of the buildings cool, or to make 2 buildings cool using fewer than 3 additional floors.

In the fourth test case, Qpwoeirut can either make the second building cool, or he can make the third building cool. Either way, he will be building 3 additional floors and maximizing the number of cool buildings. The final heights of buildings will be $[4, 2, \underline{4}, 3, 5, 3, 6, 1]$ or $[4, \underline{5}, 1, 3, 5, 3, 6, 1]$.



D1. Chopping Carrots (Easy Version)

time limit per test: 4 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The only difference between the versions is the constraints on n, k, a_i, and the sum of n over all test cases. You can make hacks only if both versions of the problem are solved.

Note the unusual memory limit.

You are given an array of integers $a_1, a_2, \ldots, a_n$ of length n, and an integer k.

The *cost* of an array of integers $p_1, p_2, \ldots, p_n$ of length n is

$$\max_{1 \le i \le n} \left(\left\lfloor \frac{a_i}{p_i} \right\rfloor \right) - \min_{1 \le i \le n} \left(\left\lfloor \frac{a_i}{p_i} \right\rfloor \right).$$

Here, $\lfloor \frac{x}{y} \rfloor$ denotes the integer part of the division of x by y. Find the minimum cost of an array p such that $1 \le p_i \le k$ for all $1 \le i \le n$.

Input
The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n, k \le 3000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 \le a_2 \le \ldots \le a_n \le 3000$).

It is guaranteed that the sum of n over all test cases does not exceed 3000.

Output
For each test case, print a single integer — the minimum possible cost of an array p satisfying the condition above.

Example

input
7 5 2 4 5 6 8 11 5 12 4 5 6 8 11 3 1 2 9 15 7 3 2 3 5 5 6 9 10 6 56 54 286 527 1436 2450 2681 3 95 16 340 2241 2 2 1 3
output
2 0 13 1 4 7 0

Note
In the first test case, the optimal array is $p = [1, 1, 1, 2, 2]$. The resulting array of values of $\lfloor \frac{a_i}{p_i} \rfloor$ is $[4, 5, 6, 4, 5]$. The cost of p is $\max\limits_{1 \le i \le n} (\lfloor \frac{a_i}{p_i} \rfloor) - \min\limits_{1 \le i \le n} (\lfloor \frac{a_i}{p_i} \rfloor) = 6 - 4 = 2$. We can show that there is no array (satisfying the condition from the statement) with a smaller cost.

In the second test case, one of the optimal arrays is $p = [12, 12, 12, 12, 12]$, which results in all $\lfloor \frac{a_i}{p_i} \rfloor$ being 0.

In the third test case, the only possible array is $p = [1, 1, 1]$.

D2. Chopping Carrots (Hard Version)

time limit per test: 4 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The only difference between the versions is the constraints on n, k, a_i, and the sum of n over all test cases. You can make hacks only if both versions of the problem are solved.

Note the unusual memory limit.

You are given an array of integers $a_1, a_2, \ldots, a_n$ of length n, and an integer k.

The *cost* of an array of integers $p_1, p_2, \ldots, p_n$ of length n is

$$\max_{1 \le i \le n} \left(\left\lfloor \frac{a_i}{p_i} \right\rfloor \right) - \min_{1 \le i \le n} \left(\left\lfloor \frac{a_i}{p_i} \right\rfloor \right).$$

Here, $\lfloor \frac{x}{y} \rfloor$ denotes the integer part of the division of x by y. Find the minimum cost of an array p such that $1 \le p_i \le k$ for all $1 \le i \le n$.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n, k \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 \le a_2 \le \ldots \le a_n \le 10^5$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output
For each test case, print a single integer — the minimum possible cost of an array p satisfying the condition above.

Example

input
7 5 2 4 5 6 8 11 5 12 4 5 6 8 11 3 1 2 9 15 7 3 2 3 5 5 6 9 10 6 56 54 286 527 1436 2450 2681 3 95 16 340 2241 2 2 1 3

output
2 0 13 1 4 7 0

Note
In the first test case, the optimal array is $p = [1, 1, 1, 2, 2]$. The resulting array of values of $\lfloor \frac{a_i}{p_i} \rfloor$ is $[4, 5, 6, 4, 5]$. The cost of p is $\max\limits_{1 \le i \le n} (\lfloor \frac{a_i}{p_i} \rfloor) - \min\limits_{1 \le i \le n} (\lfloor \frac{a_i}{p_i} \rfloor) = 6 - 4 = 2$. We can show that there is no array (satisfying the condition from the statement) with a smaller cost.

In the second test case, one of the optimal arrays is $p = [12, 12, 12, 12, 12]$, which results in all $\lfloor \frac{a_i}{p_i} \rfloor$ being 0.

In the third test case, the only possible array is $p = [1, 1, 1]$.

E. Qpwoeirut and Vertices

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a connected undirected graph with n vertices and m edges. Vertices of the graph are numbered by integers from 1 to n and edges of the graph are numbered by integers from 1 to m.

Your task is to answer q queries, each consisting of two integers l and r. The answer to each query is the smallest non-negative integer k such that the following condition holds:

- For all pairs of integers (a, b) such that $l \le a \le b \le r$, vertices a and b are reachable from one another using only the first k edges (that is, edges $1, 2, \ldots, k$).

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains three integers n, m, and q ($2 \le n \le 10^5$, $1 \le m, q \le 2 \cdot 10^5$) — the number of vertices, edges, and queries respectively.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) — ends of the i-th edge.

It is guaranteed that the graph is connected and there are no multiple edges or self-loops.

Each of the next q lines contains two integers l and r ($1 \le l \le r \le n$) — descriptions of the queries.

It is guaranteed that that the sum of n over all test cases does not exceed 10^5, the sum of m over all test cases does not exceed $2 \cdot 10^5$, and the sum of q over all test cases does not exceed $2 \cdot 10^5$.

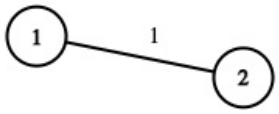
Output

For each test case, print q integers — the answers to the queries.

Example

input
3 2 1 2 1 2 1 1 1 2 5 5 5 1 2 1 3 2 4 3 4 3 5 1 4 3 4 2 2 2 5 3 5 3 2 1 1 3 2 3 1 3
output
0 1 3 3 0 5 5 2

Note

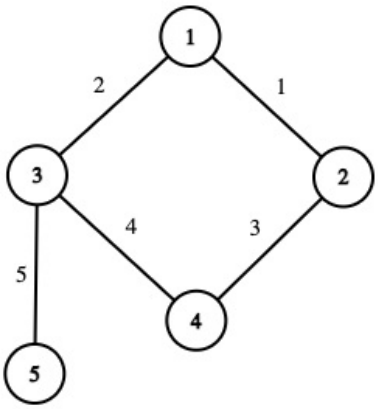


Graph from the first test case. The integer near the edge is its number.

In the first test case, the graph contains 2 vertices and a single edge connecting vertices 1 and 2.

In the first query, $l = 1$ and $r = 1$. It is possible to reach any vertex from itself, so the answer to this query is 0.

In the second query, $l = 1$ and $r = 2$. Vertices 1 and 2 are reachable from one another using only the first edge, through the path $1 \longleftrightarrow 2$. It is impossible to reach vertex 2 from vertex 1 using only the first 0 edges. So, the answer to this query is 1.



Graph from the second test case. The integer near the edge is its number.

In the second test case, the graph contains 5 vertices and 5 edges.

In the first query, $l = 1$ and $r = 4$. It is enough to use the first 3 edges to satisfy the condition from the statement:

- Vertices 1 and 2 are reachable from one another through the path $1 \longleftrightarrow 2$ (edge 1).
- Vertices 1 and 3 are reachable from one another through the path $1 \longleftrightarrow 3$ (edge 2).
- Vertices 1 and 4 are reachable from one another through the path $1 \longleftrightarrow 2 \longleftrightarrow 4$ (edges 1 and 3).
- Vertices 2 and 3 are reachable from one another through the path $2 \longleftrightarrow 1 \longleftrightarrow 3$ (edges 1 and 2).

- Vertices 2 and 4 are reachable from one another through the path $2 \longleftrightarrow 4$ (edge 3).
- Vertices 3 and 4 are reachable from one another through the path $3 \longleftrightarrow 1 \longleftrightarrow 2 \longleftrightarrow 4$ (edges 2, 1, and 3).

If we use less than 3 of the first edges, then the condition won't be satisfied. For example, it is impossible to reach vertex 4 from vertex 1 using only the first 2 edges. So, the answer to this query is 3.

In the second query, $l = 3$ and $r = 4$. Vertices 3 and 4 are reachable from one another through the path $3 \longleftrightarrow 1 \longleftrightarrow 2 \longleftrightarrow 4$ (edges 2, 1, and 3). If we use any fewer of the first edges, nodes 3 and 4 will not be reachable from one another.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform