Codeforces Round #809 (Div. 2)

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 a1,a2,…,ana1,a2,…,an of length nn, consisting of integers between 11 and mm. You also have a string ss, consisting of mm characters B.

You are going to perform the following nn operations.

- At the ii-th (1≤i≤n1≤i≤n) operation, you replace either the aiai-th **or** the (m+1−ai)(m+1−ai)-th character of ss 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 xx is lexicographically smaller than a string yy of the same length if and only if in the first position where xx and yy differ, the string xx has a letter that appears earlier in the alphabet than the corresponding letter in yy.

Input

The first line contains the number of test cases tt (1≤t≤20001≤t≤2000).
The first line of each test case contains two integers nn and mm (1≤n,m≤501≤n,m≤50) — the length of the sequence aa and the length of the string ss respectively.
The second line contains nn integers a1,a2,…,ana1,a2,…,an (1≤ai≤m1≤ai≤m) — the sequence aa.

Output

For each test case, print a string of length mm — 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

Copy

```
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

Copy

```
ABABA
BABBB
A
AABB
```

```
ABABBBB
ABABA
```

Note

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

- At the 11-st operation, you can replace the 11-st character of ss with A. After it, ss becomes ABBBB.
- At the 22-nd operation, you can replace the 55-th character of ss with A (since m+1−a2=5m+1−a2=5). After it, ss becomes ABBBA.
- At the 33-rd operation, you can replace the 33-rd character of ss with A. After it, ss becomes ABABA.
- At the 44-th operation, you can replace the 11-st character of ss with A. After it, ss 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 22-nd character or 44-th character of ss 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 11-st and 22-nd characters of ss with A to get AABB.

In the fifth test case, you can replace the 11-st and 33-rd characters of ss 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 nn colored blocks. The color of the ii-th block is cici, an integer between 11 and nn.

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

1. Initially, you place block 11 at (0,0)(0,0).
2. For 2≤i≤n2≤i≤n, if the (i−1)(i−1)-th block is placed at position (x,y)(x,y), then the ii-th block can be placed at one of positions (x+1,y)(x+1,y), (x−1,y)(x−1,y), (x,y+1)(x,y+1) (**but not at position** (x,y−1)(x,y−1)), as long no previous block was placed at that position.

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

For each color rr from 11 to nn, solve the following problem **independently**:

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

Input

The first line contains a single integer tt (1≤t≤1041≤t≤104) — the number of test cases.

The first line of each test case contains a single integer nn (1≤n≤1051≤n≤105).

The second line of each test case contains nn integers c1,c2,…,cnc1,c2,…,cn (1≤ci≤n1≤ci≤n).

It is guaranteed that the sum of nn over all test cases does not exceed 2·1052·105.

Output

For each test case, output nn integers. The rr-th of them should be the maximum size of an tower of color rr you can form by following the given rules. If you cannot form any tower of color rr, the rr-th integer should be 00.

Example

input
Copy

```
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
```

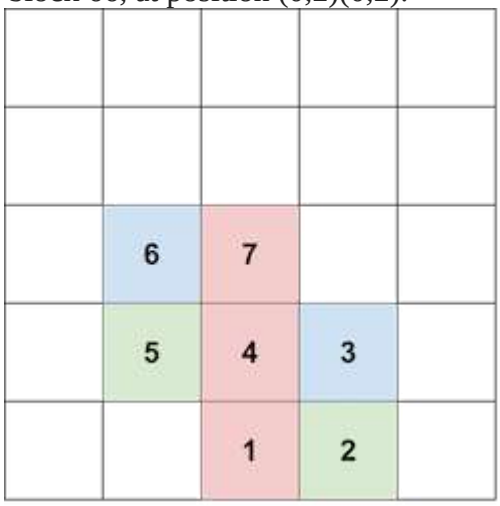
output
Copy

```
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
```

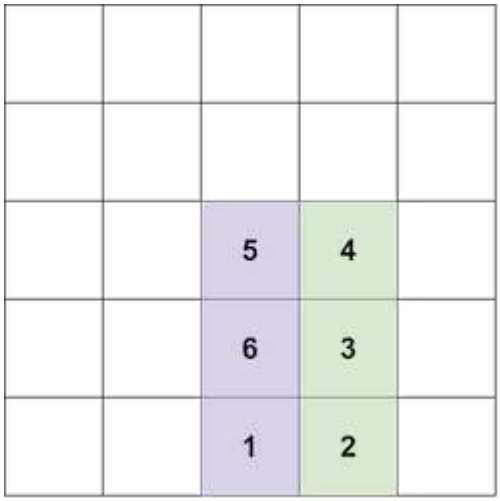
Note

In the first test case, one of the possible ways to form a tower of color 11 and size 33 is:

- place block 11 at position (0,0)(0,0);
- place block 22 to the right of block 11, at position (1,0)(1,0);
- place block 33 above block 22, at position (1,1)(1,1);
- place block 44 to the left of block 33, at position (0,1)(0,1);
- place block 55 to the left of block 44, at position (−1,1)(−1,1);
- place block 66 above block 55, at position (−1,2)(−1,2);
- place block 77 to the right of block 66, at position (0,2)(0,2).



The blocks at positions (0,0)(0,0), (0,1)(0,1), and (0,2)(0,2) all have color 11, forming an tower of size 33.
In the second test case, note that the following placement is **not valid**, since you are not allowed to place block 66 under block 55:



It can be shown that it is impossible to form a tower of color 44 and size 33.

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 nn buildings, the ii-th ($1 \le i \le n1 \le i \le n$) of which is hihi floors high. You can assume that the height of every floor in this problem is equal. Therefore, building ii is taller than the building jj if and only if the number of floors hihi in building ii is larger than the number of floors hjhj in building jj.
Building ii is *cool* if it is taller than both building $i-1i-1$ and building $i+1i+1$ (and both of them exist). Note that neither the 11-st nor the nn-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 tt ($1 \le t \le 1041 \le t \le 104$) — the number of test cases.
The first line of each test case contains the single integer nn ($3 \le n \le 1053 \le n \le 105$) — the number of buildings in Qpwoeirut's city.
The second line of each test case contains nn integers h1,h2,…,hnh1,h2,…,hn ($1 \le hi \le 1091 \le hi \le 109$) — the number of floors in each of the buildings of the city.
It is guaranteed that the sum of nn over all test cases does not exceed $2 \cdot 1052 \cdot 105$.

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

Copy

```
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
```

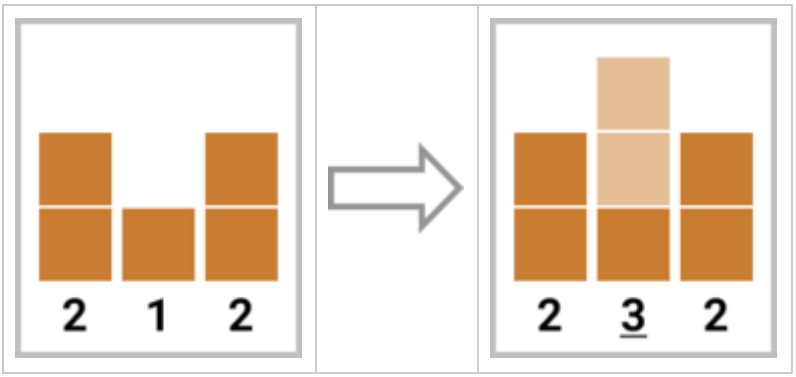
output

Copy

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

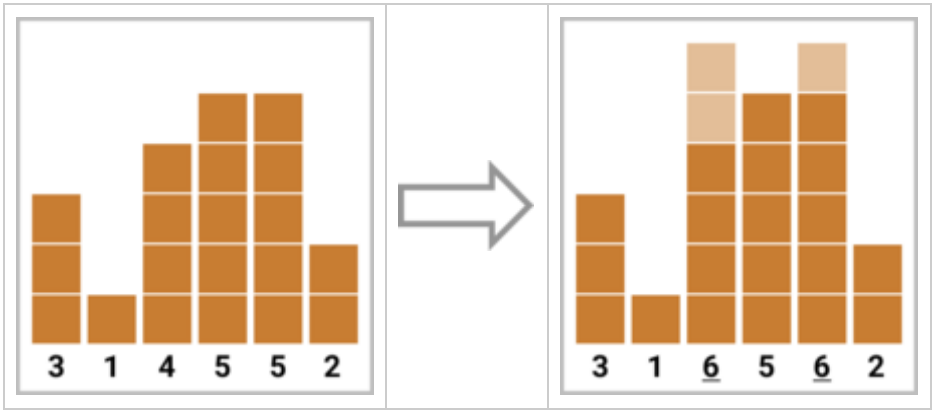
Note

In the first test case, it is optimal for Qpwoeirut to make the second building cool by building 22 additional floors on top of it, making it taller than both of its adjacent buildings. The final heights of buildings will be [2,3–,2][2,3_,2].

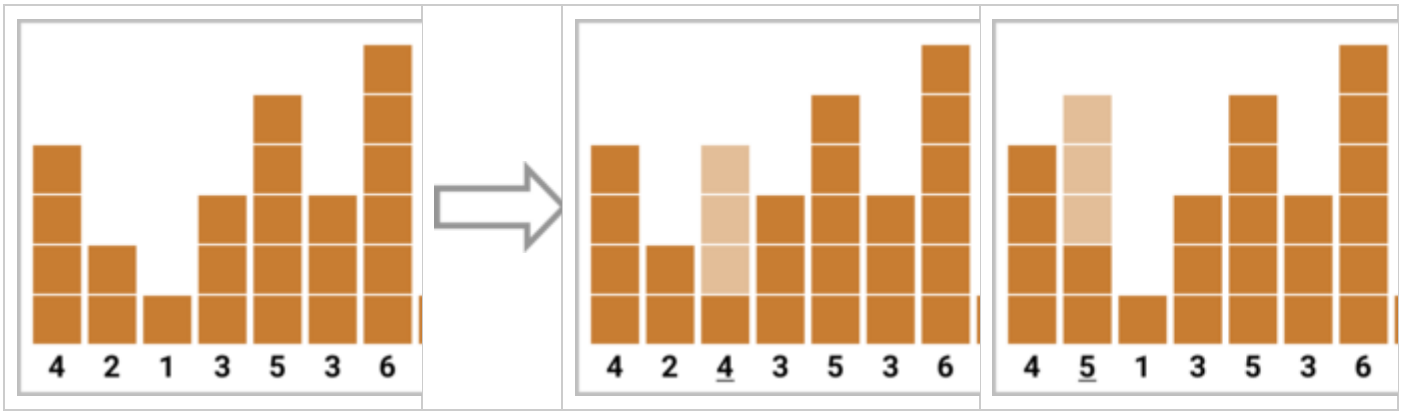


In the second test case, the number of cool buildings is already maximized, so Qpwoeirut does not need to do anything.

In the third test case, it is optimal for Qpwoeirut to make the third and fifth buildings cool by building 22 additional floors onto the third building and 11 additional floor onto the fifth building. The final heights of buildings will be [3,1,6–,5,6–,2][3,1,6_,5,6_,2].



It can be shown that it is impossible to make more than 22 of the buildings cool, or to make 22 buildings cool using fewer than 33 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 33 additional floors and maximizing the number of cool buildings. The final heights of buildings will be [4,2,4–,3,5,3,6,1][4,2,4_,3,5,3,6,1] or [4,5–,1,3,5,3,6,1][4,5_,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 nn, kk, aiai, *and the sum of* nn *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 a1,a2,…,ana1,a2,…,an of length nn, and an integer kk.
The *cost* of an array of integers p1,p2,…,pnp1,p2,…,pn of length nn is

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

Here, ⌊xy⌋⌊xy⌋ denotes the integer part of the division of xx by yy. Find the minimum cost of an array pp such that 1≤pi≤k1≤pi≤k for all 1≤i≤n1≤i≤n.
Input
The first line contains a single integer tt (1≤t≤1001≤t≤100) — the number of test cases.
The first line of each test case contains two integers nn and kk (1≤n,k≤30001≤n,k≤3000).
The second line contains nn integers a1,a2,…,ana1,a2,…,an (1≤a1≤a2≤…≤an≤30001≤a1≤a2≤…≤an≤3000).
It is guaranteed that the sum of nn over all test cases does not exceed 30003000.
Output
For each test case, print a single integer — the minimum possible cost of an array pp satisfying the condition above.

Example
input
Copy

```
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

Copy

```
2
0
13
1
4
7
0
```

Note

In the first test case, the optimal array is p=[1,1,1,2,2]p=[1,1,1,2,2]. The resulting array of values of ⌊aipi⌋⌊aipi⌋ is [4,5,6,4,5][4,5,6,4,5]. The cost of pp is max1≤i≤n(⌊aipi⌋)−min1≤i≤n(⌊aipi⌋)=6−4=2max1≤i≤n(⌊aipi⌋)−min1≤i≤n(⌊aipi⌋)=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]p=[12,12,12,12,12], which results in all ⌊aipi⌋⌊aipi⌋ being 00.

In the third test case, the only possible array is p=[1,1,1]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 nn, kk, aiai, *and the sum of* nn *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 a1,a2,…,ana1,a2,…,an of length nn, and an integer kk.

The *cost* of an array of integers p1,p2,…,pnp1,p2,…,pn of length nn is

max1≤i≤n(⌊aipi⌋)−min1≤i≤n(⌊aipi⌋).max1≤i≤n(⌊aipi⌋)−min1≤i≤n(⌊aipi⌋).

Here, ⌊xy⌋⌊xy⌋ denotes the integer part of the division of xx by yy. Find the minimum cost of an array pp such that 1≤pi≤k1≤pi≤k for all 1≤i≤n1≤i≤n.

Input

The first line contains a single integer tt (1≤t≤1001≤t≤100) — the number of test cases.

The first line of each test case contains two integers nn and kk (1≤n,k≤1051≤n,k≤105).

The second line contains nn integers a1,a2,…,ana1,a2,…,an (1≤a1≤a2≤…≤an≤1051≤a1≤a2≤…≤an≤105).

It is guaranteed that the sum of nn over all test cases does not exceed 105105.

Output

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

Example
input
Copy

```
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
Copy

```
2
0
13
1
4
7
0
```

Note
In the first test case, the optimal array is p=[1,1,1,2,2]p=[1,1,1,2,2]. The resulting array of values of ⌊aipi⌋⌊aipi⌋ is [4,5,6,4,5][4,5,6,4,5]. The cost of pp is max1≤i≤n(⌊aipi⌋)−min1≤i≤n(⌊aipi⌋)=6−4=2max1≤i≤n(⌊aipi⌋)−min1≤i≤n(⌊aipi⌋)=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]p=[12,12,12,12,12], which results in all ⌊aipi⌋⌊aipi⌋ being 00.
In the third test case, the only possible array is p=[1,1,1]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 nn vertices and mm edges. Vertices of the graph are numbered by integers from 11 to nn and edges of the graph are numbered by integers from 11 to mm.

Your task is to answer qq queries, each consisting of two integers ll and rr. The answer to each query is the smallest non-negative integer kk such that the following condition holds:

- For all pairs of integers (a,b)(a,b) such that l≤a≤b≤rl≤a≤b≤r, vertices aa and bb are reachable from one another using only the first kk edges (that is, edges 1,2,…,k1,2,…,k).

Input

The first line contains a single integer tt (1≤t≤10001≤t≤1000) — the number of test cases.

The first line of each test case contains three integers nn, mm,
and qq (2≤n≤1052≤n≤105, 1≤m,q≤2·1051≤m,q≤2·105) — the number of vertices, edges, and queries respectively.

Each of the next mm lines contains two integers uiui and vivi (1≤ui,vi≤n1≤ui,vi≤n) — ends of the ii-th edge.

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

Each of the next qq lines contains two integers ll and rr (1≤l≤r≤n1≤l≤r≤n) — descriptions of the queries.

It is guaranteed that that the sum of nn over all test cases does not exceed 105105, the sum of mm over all test cases does not exceed 2·1052·105, and the sum of qq over all test cases does not exceed 2·1052·105.

Output

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

Example

input

Copy

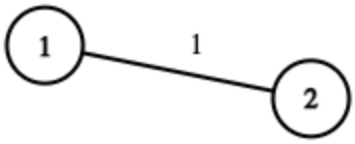
```
3
2 1 2
1 2
1 1
1 2
5 5 5
1 2
1 3
2 4
3 4
3 5
1 4
3 4
2 2
2 5
3 5
3 2 1
1 3
2 3
1 3
```

output

Copy

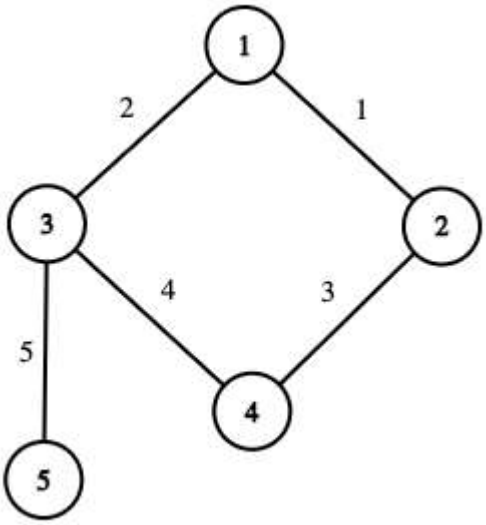
```
0 1
3 3 0 5 5
2
```

Note



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

In the first test case, the graph contains 22 vertices and a single edge connecting vertices 11 and 22.
In the first query, l=1l=1 and r=1r=1. It is possible to reach any vertex from itself, so the answer to this query is 00.
In the second query, l=1l=1 and r=2r=2. Vertices 11 and 22 are reachable from one another using only the first edge, through the path 1↔21↔2. It is impossible to reach vertex 22 from vertex 11 using only the first 00 edges. So, the answer to this query is 11.



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

In the second test case, the graph contains 55 vertices and 55 edges.
In the first query, l=1l=1 and r=4r=4. It is enough to use the first 33 edges to satisfy the condition from the statement:

- Vertices 11 and 22 are reachable from one another through the path 1↔21↔2 (edge 11).
- Vertices 11 and 33 are reachable from one another through the path 1↔31↔3 (edge 22).
- Vertices 11 and 44 are reachable from one another through the path 1↔2↔41↔2↔4 (edges 11 and 33).
- Vertices 22 and 33 are reachable from one another through the path 2↔1↔32↔1↔3 (edges 11 and 22).
- Vertices 22 and 44 are reachable from one another through the path 2↔42↔4 (edge 33).
- Vertices 33 and 44 are reachable from one another through the path 3↔1↔2↔43↔1↔2↔4 (edges 22, 11, and 33).

If we use less than 33 of the first edges, then the condition won't be satisfied. For example, it is impossible to reach vertex 44 from vertex 11 using only the first 22 edges. So, the answer to this query is 33.

In the second query, l=3l=3 and r=4r=4. Vertices 33 and 44 are reachable from one another through the path 3↔1↔2↔43↔1↔2↔4 (edges 22, 11, and 33). If we use any fewer of the first edges, nodes 33 and 44 will not be reachable from one another.