Codeforces Round #790 (Div. 4)

A. Lucky?

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

A ticket is a string consisting of six digits. A ticket is considered lucky if the sum of the first three digits is equal to the sum of the last three digits. Given a ticket, output if it is lucky or not. Note that a ticket can have leading zeroes.

Input

The first line of the input contains an integer t ($1 \leq t \leq 10^3$) — the number of testcases.

The description of each test consists of one line containing one string consisting of six digits.

Output

Output t lines, each of which contains the answer to the corresponding test case. Output "YES" if the given ticket is lucky, and "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

Example

input
5 213132 973894 045207 000000 055776
output
YES NO YES YES NO

Note

In the first test case, the sum of the first three digits is $2 + 1 + 3 = 6$ and the sum of the last three digits is $1 + 3 + 2 = 6$, they are equal so the answer is "YES".

In the second test case, the sum of the first three digits is $9 + 7 + 3 = 19$ and the sum of the last three digits is $8 + 9 + 4 = 21$, they are not equal so the answer is "NO".

In the third test case, the sum of the first three digits is $0 + 4 + 5 = 9$ and the sum of the last three digits is $2 + 0 + 7 = 9$, they are equal so the answer is "YES".

B. Equal Candies

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

There are n boxes with different quantities of candies in each of them. The i-th box has a_i candies inside.

You also have n friends that you want to give the candies to, so you decided to give each friend a box of candies. But, you don't want any friends to get upset so you decided to eat some (possibly none) candies from each box so that all boxes have the same quantity of candies in them. Note that you may eat a different number of candies from different boxes and you cannot add candies to any of the boxes.

What's the minimum total number of candies you have to eat to satisfy the requirements?

Input

The first line contains an integer t ($1 \leq t \leq 1000$) — the number of test cases.

The first line of each test case contains an integer n ($1 \leq n \leq 50$) — the number of boxes you have.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^7$) — the quantity of candies in each box.

Output

For each test case, print a single integer denoting the minimum number of candies you have to eat to satisfy the requirements.

Example

input
5 5 1 2 3 4 5 6 1000 1000 5 1000 1000 1000 10 1 2 3 5 1 2 7 9 13 5 3 8 8 8

```
1
10000000
```

output

```
10
4975
38
0
0
```

Note

For the first test case, you can eat 1 candy from the second box, 2 candies from the third box, 3 candies from the fourth box and 4 candies from the fifth box. Now the boxes have $[1, 1, 1, 1, 1]$ candies in them and you ate $0 + 1 + 2 + 3 + 4 = 10$ candies in total so the answer is 10.

For the second test case, the best answer is obtained by making all boxes contain 5 candies in them, thus eating $995 + 995 + 0 + 995 + 995 + 995 = 4975$ candies in total.

C. Most Similar Words

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

You are given n words of **equal** length m, consisting of lowercase Latin alphabet letters. The i-th word is denoted s_i.

In one move you can choose **any position in any single word** and change the letter at that position to the previous or next letter in alphabetical order. For example:

- you can change 'e' to 'd' or to 'f';
- 'a' can only be changed to 'b';
- 'z' can only be changed to 'y'.

The *difference* between two words is the **minimum** number of moves required to make them equal. For example, the *difference* between "best" and "cost" is $1 + 10 + 0 + 0 = 11$.

Find the minimum *difference* of s_i and s_j such that $(i < j)$. In other words, find the minimum *difference* over all possible pairs of the n words.

Input

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

The first line of each test case contains 2 integers n and m ($2 \le n \le 50$, $1 \le m \le 8$) — the number of strings and their length respectively.

Then follows n lines, the i-th of which containing a single string s_i of length m, consisting of lowercase Latin letters.

Output

For each test case, print a single integer — the minimum *difference* over all possible pairs of the given strings.

Example

input

```
6
2 4
best
cost
6 3
abb
zba
bef
cdu
ooo
zzz
2 7
aaabbbc
bbaezfe
3 2
ab
ab
ab
2 8
aaaaaaaa
zzzzzzzz
3 1
a
u
y
```

output

```
11
8
35
0
200
4
```

Note

For the second test case, one can show that the best pair is ("abb","bef"), which has *difference* equal to 8, which can be obtained in the following way: change the first character of the first string to 'b' in one move, change the second character of the second string to 'b' in 3 moves and change the third character of the second string to 'b' in 4 moves, thus making in total $1 + 3 + 4 = 8$ moves.

For the third test case, there is only one possible pair and it can be shown that the minimum amount of moves necessary to make the strings equal is 35.

For the fourth test case, there is a pair of strings which is already equal, so the answer is 0.

D. X-Sum

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

Timur's grandfather gifted him a chessboard to practice his chess skills. This chessboard is a grid a with n rows and m columns with each cell having a **non-negative** integer written on it.

Timur's challenge is to place a bishop on the board such that the sum of all cells attacked by the bishop is **maximal**. The bishop attacks in all directions diagonally, and there is no limit to the distance which the bishop can attack. Note that the cell on which the bishop is placed is also considered attacked. Help him find the maximal sum he can get.

Input

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

The first line of each test case contains the integers n and m ($1 \le n \le 200$, $1 \le m \le 200$).

The following n lines contain m integers each, the j-th element of the i-th line a_{ij} is the number written in the j-th cell of the i-th row ($0 \le a_{ij} \le 10^6$)

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $4 \cdot 10^4$.

Output

For each test case output a single integer, the maximum sum over all possible placements of the bishop.

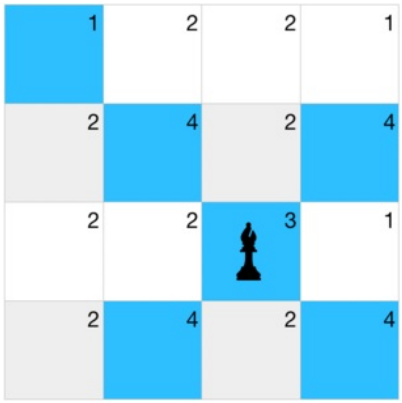
Example

input
4 4 4 1 2 2 1 2 4 2 4 2 2 3 1 2 4 2 4 2 1 1 0 3 3 1 1 1 1 1 1 1 1 1 3 3 0 1 1 1 0 1 1 1 0

output
20 1 5 3

Note

For the first test case here the best sum is achieved by the bishop being in this position:



E. Eating Queries

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Timur has n candies. The i-th candy has a quantity of sugar equal to a_i. So, by eating the i-th candy, Timur consumes a quantity of sugar equal to a_i.

Timur will ask you q queries regarding his candies. For the j-th query you have to answer what is the **minimum** number of candies he needs to eat in order to reach a quantity of sugar **greater than or equal to** x_j or print -1 if it's not possible to obtain such a quantity. In other words, you should print the minimum possible k such that after eating k candies, Timur consumes a quantity of sugar of at least x_j or say that no possible k exists.

Note that he can't eat the same candy twice and queries are independent of each other (Timur can use the same candy in different queries).

Input

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

The first line contains 2 integers n and q ($1 \le n, q \le 1.5 \cdot 10^5$) — the number of candies Timur has and the number of queries you have to print an answer for respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^4$) — the quantity of sugar in each of the candies respectively.

Then q lines follow.

Each of the next q lines contains a single integer x_j ($1 \le x_j \le 2 \cdot 10^9$) – the quantity Timur wants to reach for the given query.

It is guaranteed that the sum of n and the sum of q over all test cases do not exceed $1.5 \cdot 10^5$.

Output

For each test case output q lines. For the j-th line output the number of candies Timur needs to eat in order to reach a quantity of sugar greater than or equal to x_j or print `-1` if it's not possible to obtain such a quantity.

Example

input
3 8 7 4 3 3 1 1 4 5 9 1 10 50 14 15 22 30 4 1 1 2 3 4 3 1 2 5 4 6

output
1 2 -1 2 3 4 8 1 1 -1

Note

For the first test case:

For the first query, Timur can eat any candy, and he will reach the desired quantity.

For the second query, Timur can reach a quantity of at least 10 by eating the 7-th and the 8-th candies, thus consuming a quantity of sugar equal to 14.

For the third query, there is no possible answer.

For the fourth query, Timur can reach a quantity of at least 14 by eating the 7-th and the 8-th candies, thus consuming a quantity of sugar equal to 14.

For the second test case:

For the only query of the second test case, we can choose the third candy from which Timur receives exactly 3 sugar. It's also possible to obtain the same answer by choosing the fourth candy.

F. Longest Strike

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

Given an array a of length n and an integer k, you are tasked to find any two numbers l and r ($l \le r$) such that:

- For each x ($l \le x \le r$), x appears in a at least k times (i.e. k or more array elements are equal to x).
- The value $r - l$ is maximized.

If no numbers satisfy the conditions, output `-1`.

For example, if $a = [11, 11, 12, 13, 13, 14, 14]$ and $k = 2$, then:

- for $l = 12$, $r = 14$ the first condition fails because 12 does not appear at least $k = 2$ times.
- for $l = 13$, $r = 14$ the first condition holds, because 13 occurs at least $k = 2$ times in a and 14 occurs at least $k = 2$ times in a.
- for $l = 11$, $r = 11$ the first condition holds, because 11 occurs at least $k = 2$ times in a.

A pair of l and r for which the first condition holds and $r - l$ is maximal is $l = 13$, $r = 14$.

Input

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

The first line of each test case contains the integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le n$) — the length of the array a and the minimum amount of times each number in the range $[l, r]$ should appear respectively.

Then a single line follows, containing n integers describing the array a ($1 \le a_i \le 10^9$).

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 2 numbers, l and r that satisfy the conditions, or "-1" if no numbers satisfy the conditions.

If multiple answers exist, you can output any.

Example

input
4 7 2 11 11 12 13 13 14 14 5 1 6 3 5 2 1 6 4 4 3 4 3 3 4 14 2 1 1 2 2 2 3 3 3 3 4 4 4 4 4

output
13 14 1 3 -1 1 4

G. White-Black Balanced Subtrees

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

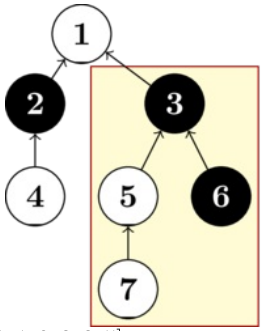
You are given a rooted tree consisting of n vertices numbered from 1 to n. The root is vertex 1. There is also a string s denoting the color of each vertex: if $s_i = \texttt{B}$, then vertex i is black, and if $s_i = \texttt{W}$, then vertex i is white.

A subtree of the tree is called balanced if the number of white vertices equals the number of black vertices. Count the number of balanced subtrees.

A *tree* is a connected undirected graph without cycles. A *rooted tree* is a tree with a selected vertex, which is called the *root*. In this problem, all trees have root 1.

The tree is specified by an array of parents $a_2, \ldots, a_n$ containing $n - 1$ numbers: a_i is the parent of the vertex with the number i for all $i = 2, \ldots, n$. The parent of a vertex u is a vertex that is the next vertex on a simple path from u to the root.

The *subtree* of a vertex u is the set of all vertices that pass through u on a simple path to the root. For example, in the picture below, 7 is in the subtree of 3 because the simple path $7 \to 5 \to 3 \to 1$ passes through 3. Note that a vertex is included in its subtree, and the subtree of the root is the entire tree.



The picture shows the tree for $n = 7$, $a = [1, 1, 2, 3, 3, 5]$, and $s = \texttt{WBBWWBW}$. The subtree at the vertex 3 is balanced.

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

The first line of each test case contains an integer n ($2 \le n \le 4000$) — the number of vertices in the tree.

The second line of each test case contains $n - 1$ integers $a_2, \ldots, a_n$ ($1 \le a_i < i$) — the parents of the vertices $2, \ldots, n$.

The third line of each test case contains a string s of length n consisting of the characters $\texttt{B}$ and $\texttt{W}$ — the coloring of the tree.

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

Output
For each test case, output a single integer — the number of balanced subtrees.

Example

input
3 7 1 1 2 3 3 5 WBBWWBW 2 1 BW 8 1 2 3 4 5 6 7 BWBWBWBW

output
2 1

```
4
```

Note
The first test case is pictured in the statement. Only the subtrees at vertices 2 and 3 are balanced.

In the second test case, only the subtree at vertex 1 is balanced.

In the third test case, only the subtrees at vertices 1, 3, 5, and 7 are balanced.

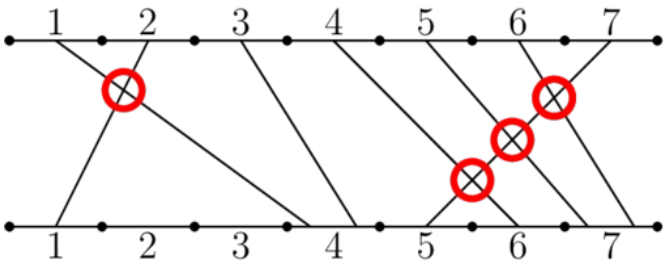
H1. Maximum Crossings (Easy Version)

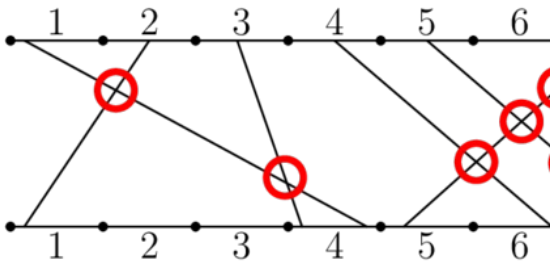
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between the two versions is that in this version $n \le 1000$ and the sum of n over all test cases does not exceed 1000.

A *terminal* is a row of n equal segments numbered 1 to n in order. There are two terminals, one above the other.

You are given an array a of length n. For all $i = 1, 2, \ldots, n$, there should be a straight wire from some point on segment i of the top terminal to some point on segment a_i of the bottom terminal. You can't select the endpoints of a segment. For example, the following pictures show two possible wirings if $n = 7$ and $a = [4, 1, 4, 6, 7, 7, 5]$.





A *crossing* occurs when two wires share a point in common. In the picture above, crossings are circled in red.

What is the **maximum** number of crossings there can be if you place the wires optimally?

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

The first line of each test case contains an integer n ($1 \le n \le 1000$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

The sum of n across all test cases does not exceed 1000.

Output
For each test case, output a single integer — the **maximum** number of crossings there can be if you place the wires optimally.

Example

input
4 7 4 1 4 6 7 7 5 2 2 1 1 1 3 2 2 2

output
6 1 0 3

Note
The first test case is shown in the second picture in the statement.

In the second test case, the only wiring possible has the two wires cross, so the answer is 1.

In the third test case, the only wiring possible has one wire, so the answer is 0.

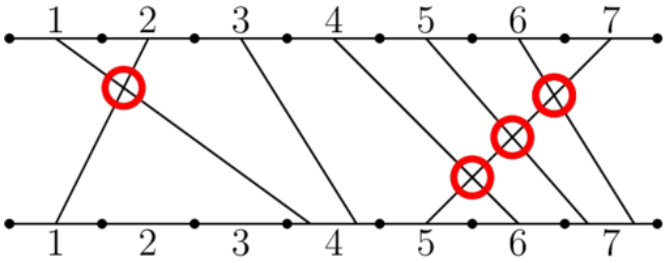
H2. Maximum Crossings (Hard Version)

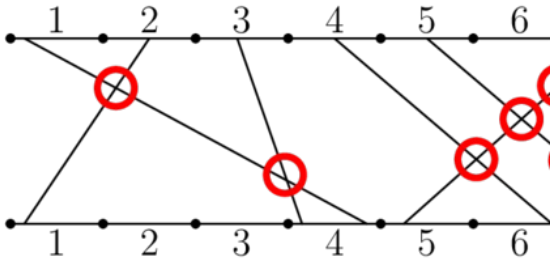
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between the two versions is that in this version $n \le 2 \cdot 10^5$ and the sum of n over all test cases does not exceed $2 \cdot 10^5$.

A *terminal* is a row of n equal segments numbered 1 to n in order. There are two terminals, one above the other.

You are given an array a of length n. For all $i = 1, 2, \ldots, n$, there should be a straight wire from some point on segment i of the top terminal to some point on segment a_i of the bottom terminal. You can't select the endpoints of a segment. For example, the following pictures show two possible wirings if $n = 7$ and $a = [4, 1, 4, 6, 7, 7, 5]$.





A *crossing* occurs when two wires share a point in common. In the picture above, crossings are circled in red.

What is the **maximum** number of crossings there can be if you place the wires optimally?

Input

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

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

The sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the **maximum** number of crossings there can be if you place the wires optimally.

Example

input
4 7 4 1 4 6 7 7 5 2 2 1 1 1 3 2 2 2

output
6 1 0 3

Note

The first test case is shown in the second picture in the statement.

In the second test case, the only wiring possible has the two wires cross, so the answer is 1.

In the third test case, the only wiring possible has one wire, so the answer is 0.

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