A. And Matching

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

You are given a set of n (n is always a power of 2) elements containing all integers $0, 1, 2, \ldots, n-1$ exactly once.

Find $\frac{n}{2}$ pairs of elements such that:

- Each element in the set is in exactly one pair.
- The sum over all pairs of the bitwise AND of its elements must be exactly equal to k. Formally, if a_i and b_i are the elements of the i-th pair, then the following must hold:

$$\sum_{i=1}^{n/2} a_i \& b_i = k,$$

  where $\&$ denotes the bitwise AND operation.

If there are many solutions, print any of them, if there is no solution, print -1 instead.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 400$) — the number of test cases. Description of the test cases follows.

Each test case consists of a single line with two integers n and k ($4 \le n \le 2^{16}$, n is a power of 2, $0 \le k \le n-1$).

The sum of n over all test cases does not exceed 2^{16}. All test cases in each individual input will be pairwise **different**.

Output

For each test case, if there is no solution, print a single line with the integer -1.

Otherwise, print $\frac{n}{2}$ lines, the i-th of them must contain a_i and b_i, the elements in the i-th pair.

If there are many solutions, print any of them. Print the pairs and the elements in the pairs in any order.

Example

input

```
4
4 0
4 1
4 2
4 3
```

output

```
0 3
1 2
0 2
1 3
0 1
2 3
-1
```

Note

In the first test, $(0\&3) + (1\&2) = 0$.

In the second test, $(0\&2) + (1\&3) = 1$.

In the third test, $(0\&1) + (2\&3) = 2$.

In the fourth test, there is no solution.

B. Range and Partition

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

Given an array a of n integers, find a range of values $[x, y]$ ($x \le y$), and split a into **exactly** k ($1 \le k \le n$) subarrays in such a way that:

- Each subarray is formed by several continuous elements of a, that is, it is equal to $a_l, a_{l+1}, \ldots, a_r$ for some l and r ($1 \le l \le r \le n$).
- Each element from a belongs to exactly one subarray.
- In each subarray the number of elements inside the range $[x, y]$ (inclusive) is **strictly greater** than the number of elements outside the range. An element with index i is inside the range $[x, y]$ if and only if $x \le a_i \le y$.

Print any solution that minimizes $y - x$.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 3 \cdot 10^4$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$) — the length of the array a and the number of subarrays required in the partition.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) where a_i is the i-th element of the array.

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 $k + 1$ lines.

In the first line, print x and y — the limits of the found range.

Then print k lines, the i-th should contain l_i and r_i ($1 \le l_i \le r_i \le n$) — the limits of the i-th subarray.

You can print the subarrays in any order.

Example

input

```
3
2 1
1 2
4 2
1 2 2 2
11 3
5 5 5 1 5 5 1 5 5 5 1
```

output

```
1 2
1 2
2 2
1 3
4 4
5 5
1 1
2 2
3 11
```

Note

In the first test, there should be only one subarray, which must be equal to the whole array. There are 2 elements inside the range $[1, 2]$ and 0 elements outside, if the chosen range is $[1, 1]$, there will be 1 element inside (a_1) and 1 element outside (a_2), and the answer will be invalid.

In the second test, it is possible to choose the range $[2, 2]$, and split the array in subarrays $(1, 3)$ and $(4, 4)$, in subarray $(1, 3)$ there are 2 elements inside the range (a_2 and a_3) and 1 element outside (a_1), in subarray $(4, 4)$ there is only 1 element (a_4), and it is inside the range.

In the third test, it is possible to choose the range $[5, 5]$, and split the array in subarrays $(1, 4)$, $(5, 7)$ and $(8, 11)$, in the subarray $(1, 4)$ there are 3 elements inside the range and 1 element outside, in the subarray $(5, 7)$ there are 2 elements inside and 1 element outside and in the subarray $(8, 11)$ there are 3 elements inside and 1 element outside.

C. Paint the Middle

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

You are given n elements numbered from 1 to n, the element i has value a_i and color c_i, initially, $c_i = 0$ for all i.

The following operation can be applied:

- Select three elements i, j and k ($1 \le i < j < k \le n$), such that c_i, c_j and c_k are all equal to 0 and $a_i = a_k$, then set $c_j = 1$.

Find the maximum value of $\sum_{i=1}^{n} c_i$ that can be obtained after applying the given operation any number of times.

Input

The first line contains an integer n $(3 \le n \le 2 \cdot 10^5)$ — the number of elements.

The second line consists of n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$, where a_i is the value of the i-th element.

Output

Print a single integer in a line — the maximum value of $\sum_{i=1}^{n} c_i$ that can be obtained after applying the given operation any number of times.

Examples

input

```
7
1 2 1 2 7 4 7
```

output

```
2
```

input

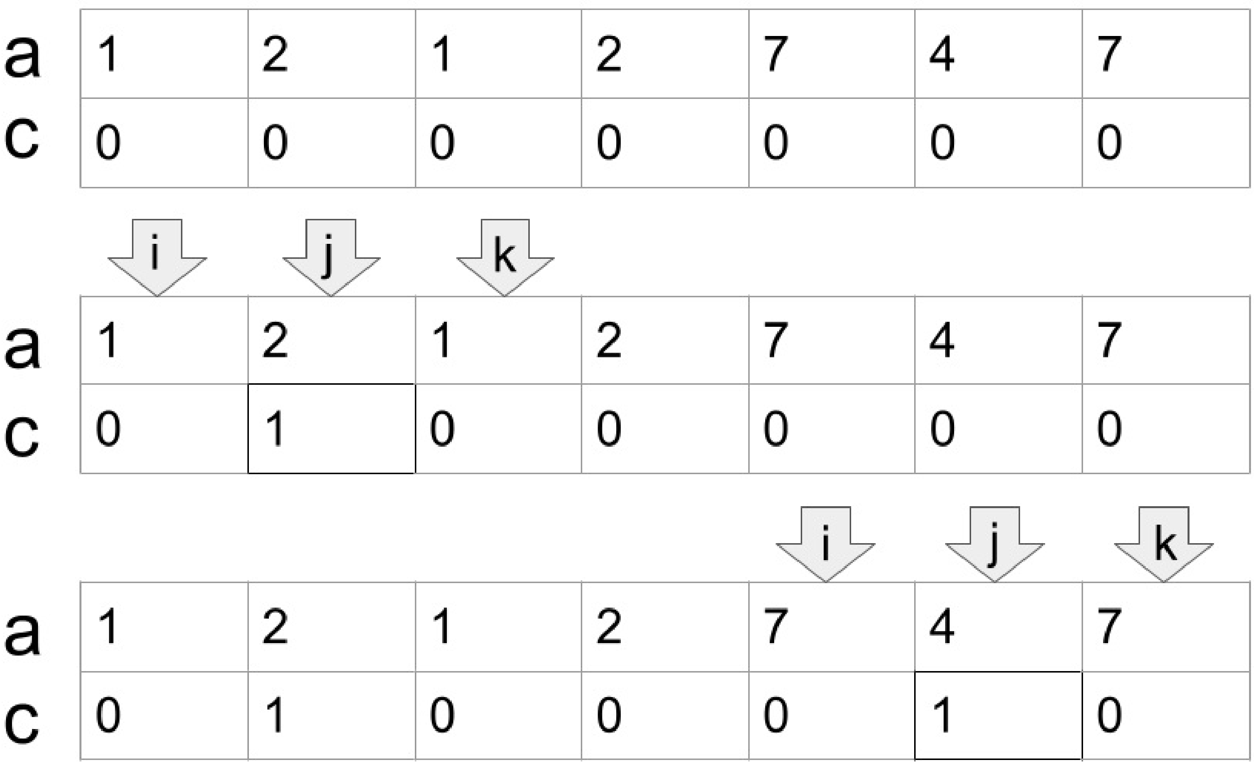
```
13
1 2 3 2 1 3 3 4 5 5 5 4 7
```

output

```
7
```

Note

In the first test, it is possible to apply the following operations in order:



D. Flipping Range

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

You are given an array a of n integers and a set B of m positive integers such that $1 \le b_i \le \lfloor \frac{n}{2} \rfloor$ for $1 \le i \le m$, where b_i is the i-th element of B.

You can make the following operation on a:

1. Select some x such that x appears in B.
2. Select an interval from array a of size x and multiply by -1 every element in the interval. Formally, select l and r such that

$1 \le l \le r \le n$ and $r - l + 1 = x$, then assign $a_i := -a_i$ for every i such that $l \le i \le r$.

Consider the following example, let $a = [0, 6, -2, 1, -4, 5]$ and $B = \{1, 2\}$:

1. $[0, 6, -2, -1, 4, 5]$ is obtained after choosing size 2 and $l = 4$, $r = 5$.
2. $[0, 6, 2, -1, 4, 5]$ is obtained after choosing size 1 and $l = 3$, $r = 3$.

Find the maximum $\sum_{i=1}^{n} a_i$ you can get after applying such operation any number of times (possibly zero).

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains two integers n and m ($2 \le n \le 10^6$, $1 \le m \le \lfloor \frac{n}{2} \rfloor$) — the number of elements of a and B respectively.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

The third line of each test case contains m **distinct** positive integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le \lfloor \frac{n}{2} \rfloor$) — the elements in the set B.

It's guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case print a single integer — the maximum possible sum of all a_i after applying such operation any number of times.

Example

input

```
3
6 2
0 6 -2 1 -4 5
1 2
7 1
1 -1 1 -1 1 -1 1
2
5 1
-1000000000 -1000000000 -1000000000 -1000000000 -1000000000
1
```

output

```
18
5
5000000000
```

Note

In the first test, you can apply the operation $x = 1$, $l = 3$, $r = 3$, and the operation $x = 1$, $l = 5$, $r = 5$, then the array becomes $[0, 6, 2, 1, 4, 5]$.

In the second test, you can apply the operation $x = 2$, $l = 2$, $r = 3$, and the array becomes $[1, 1, -1, -1, 1, -1, 1]$, then apply the operation $x = 2$, $l = 3$, $r = 4$, and the array becomes $[1, 1, 1, 1, 1, -1, 1]$. There is no way to achieve a sum bigger than 5.

E. Expected Components

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

Given a cyclic array a of size n, where a_i is the value of a in the i-th position, **there may be repeated values**. Let us define that a permutation of a is equal to another permutation of a if and only if their values are the same for each position i or we can transform them to each other by performing some cyclic rotation. Let us define for a cyclic array b its number of components as the number of connected components in a graph, where the vertices are the positions of b and we add an edge between each pair of adjacent positions of b with equal values (note that in a cyclic array the first and last position are also adjacents).

Find the expected value of components of a permutation of a if we select it equiprobably over the set of all the different permutations of a.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^6$) — the size of the cyclic array a.

The second line of each test case contains n integers, the i-th of them is the value a_i ($1 \le a_i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

It is guaranteed that the total number of different permutations of a is not divisible by M

Output

For each test case print a single integer — the expected value of components of a permutation of a if we select it equiprobably over the set of all the different permutations of a modulo $998\,244\,353$.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Example

input

```
5
4
1 1 1 1
4
1 1 2 1
4
1 2 1 2
5
4 3 2 5 1
12
1 3 2 3 2 1 3 3 1 3 3 2
```

output

```
1
2
3
5
358642921
```

Note

In the first test case there is only 1 different permutation of a:

- $[1,1,1,1]$ has 1 component.
- Therefore the expected value of components is $\frac{1}{1} = 1$

In the second test case there are 4 ways to permute the cyclic array a, but there is only 1 different permutation of a:

- $[1,1,1,2]$, $[1,1,2,1]$, $[1,2,1,1]$ and $[2,1,1,1]$ are the same permutation and have 2 components.
- Therefore the expected value of components is $\frac{2}{1} = 2$

In the third test case there are 6 ways to permute the cyclic array a, but there are only 2 different permutations of a:

- $[1,1,2,2]$, $[2,1,1,2]$, $[2,2,1,1]$ and $[1,2,2,1]$ are the same permutation and have 2 components.
- $[1,2,1,2]$ and $[2,1,2,1]$ are the same permutation and have 4 components.
- Therefore the expected value of components is $\frac{2+4}{2} = \frac{6}{2} = 3$

In the fourth test case there are 120 ways to permute the cyclic array a, but there are only 24 different permutations of a:

- Any permutation of a has 5 components.
- Therefore the expected value of components is $\frac{24 \cdot 5}{24} = \frac{120}{24} = 5$

F. Making It Bipartite

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an undirected graph of n vertices indexed from 1 to n, where vertex i has a value a_i assigned to it and all values a_i are **different**. There is an edge between two vertices u and v if either a_u divides a_v or a_v divides a_u.

Find the minimum number of vertices to remove such that the remaining graph is bipartite, when you remove a vertex you remove all the edges incident to it.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 5 \cdot 10^4$) — the number of vertices in the graph.

The second line of each test case contains n integers, the i-th of them is the value a_i ($1 \le a_i \le 5 \cdot 10^4$) assigned to the i-th vertex, all values a_i are **different**.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^4$.

Output

For each test case print a single integer — the minimum number of vertices to remove such that the remaining graph is bipartite.

Example

input

```
4
4
8 4 2 1
4
30 2 3 5
5
12 4 6 2 3
10
85 195 5 39 3 13 266 154 14 2
```

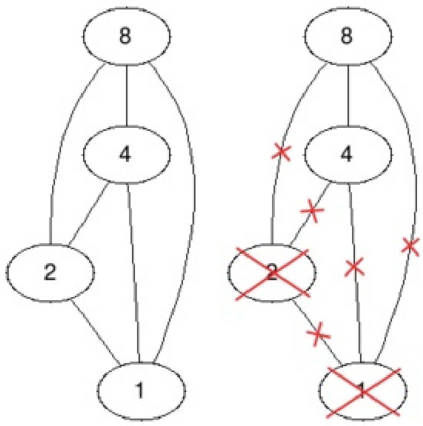
output

```
2
0
1
2
```

Note

In the first test case if we remove the vertices with values 1 and 2 we will obtain a bipartite graph, so the answer is 2, it is impossible to remove less than 2 vertices and still obtain a bipartite graph.

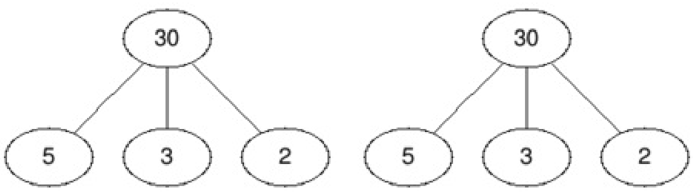
Before	After



test case #1

In the second test case we do not have to remove any vertex because the graph is already bipartite, so the answer is 0.

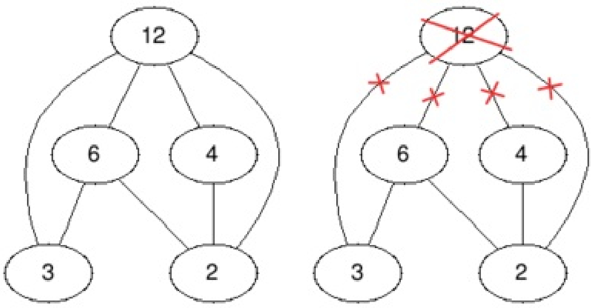
Before	After



test case #2

In the third test case we only have to remove the vertex with value 12, so the answer is 1.

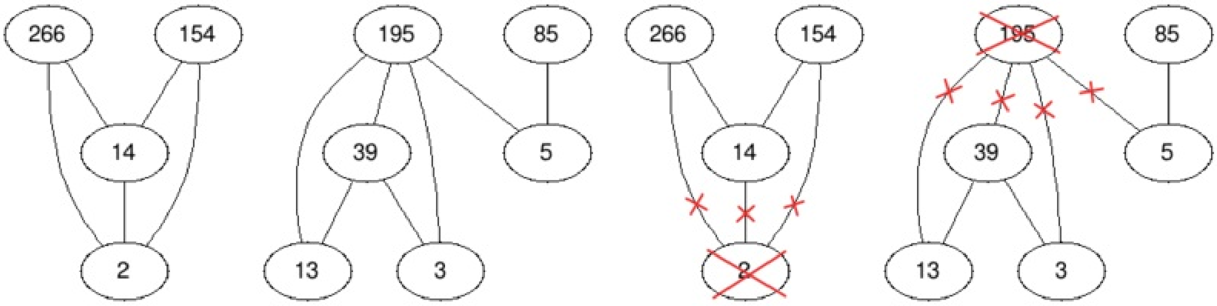
Before	After



test case #3

In the fourth test case we remove the vertices with values 2 and 195, so the answer is 2.

Before	After



test case #4