Codeforces Round #636 (Div. 3)

A. Candies

1 second, 256 megabytes

Recently Vova found n candy wrappers. He remembers that he bought x candies during the first day, $2x$ candies during the second day, $4x$ candies during the third day, $\dots$, $2^{k-1}x$ candies during the k-th day. But there is an issue: Vova remembers neither x nor k but he is sure that x and k are positive integers and $k > 1$.

Vova will be satisfied if you tell him **any positive** integer x so there is an integer $k > 1$ that $x + 2x + 4x + \dots + 2^{k-1}x = n$. It is guaranteed that at least one solution exists. **Note that** $k > 1$.

You have to answer t independent test cases.

Input

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

The only line of the test case contains one integer n ($3 \le n \le 10^9$) — the number of candy wrappers Vova found. It is guaranteed that there is some positive integer x and integer $k > 1$ that $x + 2x + 4x + \dots + 2^{k-1}x = n$.

Output

Print one integer — **any positive** integer value of x so there is an integer $k > 1$ that $x + 2x + 4x + \dots + 2^{k-1}x = n$.

input

```
7
3
6
7
21
28
999999999
999999984
```

output

```
1
2
1
7
4
333333333
333333328
```

In the first test case of the example, one of the possible answers is $x = 1, k = 2$. Then $1 \cdot 1 + 2 \cdot 1$ equals $n = 3$.

In the second test case of the example, one of the possible answers is $x = 2, k = 2$. Then $1 \cdot 2 + 2 \cdot 2$ equals $n = 6$.

In the third test case of the example, one of the possible answers is $x = 1, k = 3$. Then $1 \cdot 1 + 2 \cdot 1 + 4 \cdot 1$ equals $n = 7$.

In the fourth test case of the example, one of the possible answers is $x = 7, k = 2$. Then $1 \cdot 7 + 2 \cdot 7$ equals $n = 21$.

In the fifth test case of the example, one of the possible answers is $x = 4, k = 3$. Then $1 \cdot 4 + 2 \cdot 4 + 4 \cdot 4$ equals $n = 28$.

B. Balanced Array

1 second, 256 megabytes

You are given a positive integer n, it is guaranteed that n is even (i.e. divisible by 2).

You want to construct the array a of length n such that:

- The first $\frac{n}{2}$ elements of a are even (divisible by 2);
- the second $\frac{n}{2}$ elements of a are odd (not divisible by 2);
- **all elements of a are distinct and positive**;
- the sum of the first half equals to the sum of the second half ($\sum\limits_{i=1}^{\frac{n}{2}} a_i = \sum\limits_{i=\frac{n}{2}+1}^{n} a_i$).

If there are multiple answers, you can print any. It is **not guaranteed** that the answer exists.

You have to answer t independent test cases.

Input

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

The only line of the test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the length of the array. It is guaranteed that that n is even (i.e. divisible by 2).

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

Output

For each test case, print the answer — "NO" (without quotes), if there is no suitable answer for the given test case or "YES" in the first line and **any** suitable array $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) satisfying conditions from the problem statement on the second line.

input

```
5
2
4
6
8
10
```

output

```
NO
YES
2 4 1 5
NO
YES
2 4 6 8 1 3 5 11
NO
```

C. Alternating Subsequence

1 second, 256 megabytes

Recall that the sequence b is a a subsequence of the sequence a if b can be derived from a by removing zero or more elements without changing the order of the remaining elements. For example, if $a = [1, 2, 1, 3, 1, 2, 1]$, then possible subsequences are: $[1, 1, 1, 1]$, $[3]$ and $[1, 2, 1, 3, 1, 2, 1]$, but not $[3, 2, 3]$ and $[1, 1, 1, 1, 2]$.

You are given a sequence a consisting of n positive and negative elements (there is no zeros in the sequence).

Your task is to choose **maximum by size** (length) *alternating* subsequence of the given sequence (i.e. the sign of each next element is the opposite from the sign of the current element, like positive-negative-positive and so on or negative-positive-negative and so on). Among all such subsequences, you have to choose one which has the **maximum sum** of elements.

In other words, if the maximum length of *alternating* subsequence is k then your task is to find the **maximum sum** of elements of some *alternating* subsequence of length k.

You have to answer t independent test cases.

Input

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

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in a. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($-10^9 \le a_i \le 10^9, a_i \ne 0$), where a_i is the i-th element of a.

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

Output

For each test case, print the answer — the **maximum sum** of the **maximum by size** (length) *alternating* subsequence of a.

input

```
4
5
1 2 3 -1 -2
4
-1 -2 -1 -3
10
-2 8 3 8 -4 -15 5 -2 -3 1
6
1 -1000000000 1 -1000000000 1 -1000000000
```

output

```
2
-1
6
-2999999997
```

In the first test case of the example, one of the possible answers is $[1, 2, \underline{3}, \underline{-1}, -2]$.

In the second test case of the example, one of the possible answers is $[-1, -2, \underline{-1}, -3]$.

In the third test case of the example, one of the possible answers is $[\underline{-2}, 8, 3, \underline{8}, \underline{-4}, -15, \underline{5}, \underline{-2}, -3, \underline{1}]$.

In the fourth test case of the example, one of the possible answers is $[\underline{1}, \underline{-1000000000}, \underline{1}, \underline{-1000000000}, \underline{1}, \underline{-1000000000}]$.

D. Constant Palindrome Sum

1 second, 256 megabytes

You are given an array a consisting of n integers (it is guaranteed that n is even, i.e. divisible by 2). All a_i does not exceed some integer k.

Your task is to replace the **minimum** number of elements (replacement is the following operation: choose some index i from 1 to n and replace a_i with some integer in range $[1; k]$) to satisfy the following conditions:

- after all replacements, all a_i are positive integers not greater than k;
- for all i from 1 to $\frac{n}{2}$ the following equation is true: $a_i + a_{n-i+1} = x$, where x should be **the same** for all $\frac{n}{2}$ pairs of elements.

You have to answer t independent test cases.

Input

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

The first line of the test case contains two integers n and k ($2 \le n \le 2 \cdot 10^5, 1 \le k \le 2 \cdot 10^5$) — the length of a and the maximum possible value of some a_i correspondingly. It is guratanteed that n is even (i.e. divisible by 2). The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le k$), where a_i is the i-th element of a.

It is guaranteed that the sum of n (as well as the sum of k) over all test cases does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$, $\sum k \le 2 \cdot 10^5$).

Output

For each test case, print the answer — the **minimum** number of elements you have to replace in a to satisfy the conditions from the problem statement.

input

```
4
4 2
1 2 1 2
4 3
1 2 2 1
8 7
6 1 1 7 6 3 4 6
6 6
5 2 6 1 3 4
```

output

```
0
1
4
2
```

E. Weights Distributing

2 seconds, 256 megabytes

You are given an undirected unweighted graph consisting of n vertices and m edges (which represents the map of Bertown) and the array of prices p of length m. It is guaranteed that there is a path between each pair of vertices (districts).

Mike has planned a trip from the vertex (district) a to the vertex (district) b and then from the vertex (district) b to the vertex (district) c. He can visit the same district twice or more. But there is one issue: authorities of the city want to set a price for using the road so if someone goes along the road then he should pay the price corresponding to this road (**he pays each time he goes along the road**). The list of prices that will be used p is ready and they just want to distribute it between all roads in the town in such a way that each price from the array corresponds to exactly one road.

You are a good friend of Mike (and suddenly a mayor of Bertown) and want to help him to make his trip as cheap as possible. So, your task is to distribute prices between roads in such a way that if Mike chooses the optimal path then the price of the trip is the **minimum** possible. **Note that you cannot rearrange prices after the start of the trip**.

You have to answer t independent test cases.

Input

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

The first line of the test case contains five integers n, m, a, b and c ($2 \le n \le 2 \cdot 10^5$, $n-1 \le m \le min(\frac{n(n-1)}{2}, 2 \cdot 10^5)$, $1 \le a, b, c \le n$) — the number of vertices, the number of edges and districts in Mike's trip.

The second line of the test case contains m integers $p_1, p_2, \dots, p_m$ ($1 \le p_i \le 10^9$), where p_i is the i-th price from the array.

The following m lines of the test case denote edges: edge i is represented by a pair of integers v_i, u_i ($1 \le v_i, u_i \le n$, $u_i \ne v_i$), which are the indices of vertices connected by the edge. There are no loops or multiple edges in the given graph, i. e. for each pair (v_i, u_i) there are no other pairs (v_i, u_i) or (u_i, v_i) in the array of edges, and for each pair (v_i, u_i) the condition $v_i \ne u_i$ is satisfied. It is guaranteed that the given graph is connected.

It is guaranteed that the sum of n (as well as the sum of m) does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$, $\sum m \le 2 \cdot 10^5$).

Output

For each test case, print the answer — the **minimum** possible price of Mike's trip if you distribute prices between edges optimally.

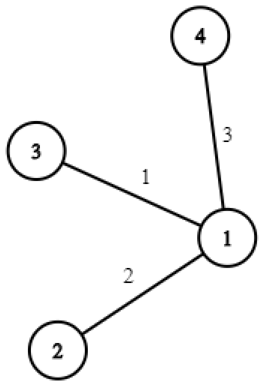
input

```
2
4 3 2 3 4
1 2 3
1 2
1 3
1 4
7 9 1 5 7
2 10 4 8 5 6 7 3 3
1 2
1 3
1 4
3 2
3 5
4 2
5 6
1 7
6 7
```

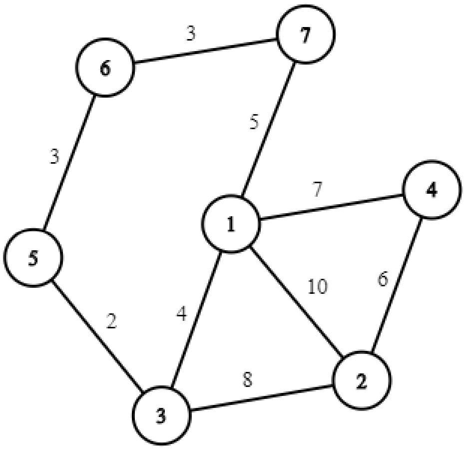
output

```
7
12
```

One of the possible solution to the first test case of the example:



One of the possible solution to the second test case of the example:



F. Restore the Permutation by Sorted Segments

2 seconds, 256 megabytes

We guessed a permutation p consisting of n integers. The permutation of length n is the array of length n where each element from 1 to n appears exactly once. This permutation is a secret for you.

For each position r from 2 to n we chose some other index l ($l < r$) and gave you the segment $p_l, p_{l+1}, \dots, p_r$ in **sorted** order (i.e. we rearranged the elements of this segment in a way that the elements of this segment are sorted). Thus, you are given exactly $n-1$ segments of the initial permutation but elements inside each segment are sorted. The segments are given to you in random order.

For example, if the secret permutation is $p = [3, 1, 4, 6, 2, 5]$ then the possible given set of segments can be:

- $[2, 5, 6]$
- $[4, 6]$
- $[1, 3, 4]$
- $[1, 3]$
- $[1, 2, 4, 6]$

Your task is to find **any** suitable permutation (i.e. any permutation corresponding to the given input data). It is guaranteed that the input data corresponds to some permutation (i.e. such permutation exists).

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($2 \le n \le 200$) — the length of the permutation.

The next $n-1$ lines describe given segments.

The i-th line contains the description of the i-th segment. The line starts with the integer k_i ($2 \le k_i \le n$) — the length of the i-th segment. Then k_i integers follow. All integers in a line are distinct, sorted in ascending order, between 1 and n, inclusive.

It is guaranteed that the required p exists for each test case.

It is also guaranteed that the sum of n over all test cases does not exceed 200 ($\sum n \le 200$).

Output

For each test case, print the answer: n integers $p_1, p_2, \dots, p_n$ ($1 \le p_i \le n$, all p_i should be distinct) — **any** suitable permutation (i.e. any permutation corresponding to the test case input).

input

```
5
6
3 2 5 6
2 4 6
3 1 3 4
2 1 3
4 1 2 4 6
5
2 2 3
2 1 2
2 1 4
2 4 5
7
3 1 2 6
4 1 3 5 6
2 1 2
3 4 5 7
6 1 2 3 4 5 6
3 1 3 6
2
2 1 2
5
2 2 5
3 2 3 5
4 2 3 4 5
5 1 2 3 4 5
```

output

```
3 1 4 6 2 5
3 2 1 4 5
2 1 6 3 5 4 7
1 2
2 5 3 4 1
```