A. Circular Local MiniMax

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

You are given n integers $a_1, a_2, \ldots, a_n$. Is it possible to arrange them on a circle so that each number is strictly greater than both its neighbors or strictly smaller than both its neighbors?

In other words, check if there exists a rearrangement $b_1, b_2, \ldots, b_n$ of the integers $a_1, a_2, \ldots, a_n$ such that for each i from 1 to n at least one of the following conditions holds:

- $b_{i-1} < b_i > b_{i+1}$
- $b_{i-1} > b_i < b_{i+1}$

To make sense of the previous formulas for $i = 1$ and $i = n$, one shall define $b_0 = b_n$ and $b_{n+1} = b_1$.

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 10^5$) — the number of integers.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

The sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, if it is not possible to arrange the numbers on the circle satisfying the conditions from the statement, output `NO`. You can output each letter in any case.

Otherwise, output `YES`. In the second line, output n integers $b_1, b_2, \ldots, b_n$, which are a rearrangement of $a_1, a_2, \ldots, a_n$ and satisfy the conditions from the statement. If there are multiple valid ways to arrange the numbers, you can output any of them.

Example

input

```
4
3
1 1 2
4
1 9 8 4
4
2 0 2 2
6
1 1 1 11 111 1111
```

output

```
NO
YES
1 8 4 9
NO
YES
1 11 1 111 1 1111
```

Note

It can be shown that there are no valid arrangements for the first and the third test cases.

In the second test case, the arrangement $[1, 8, 4, 9]$ works. In this arrangement, 1 and 4 are both smaller than their neighbors, and $8, 9$ are larger.

In the fourth test case, the arrangement $[1, 11, 1, 111, 1, 1111]$ works. In this arrangement, the three elements equal to 1 are smaller than their neighbors, while all other elements are larger than their neighbors.

B. Linguistics

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

Alina has discovered a weird language, which contains only 4 words: $\mathtt{A}$, $\mathtt{B}$, $\mathtt{AB}$, $\mathtt{BA}$. It also turned out that there are no spaces in this language: a sentence is written by just concatenating its words into a single string.

Alina has found one such sentence s and she is curious: is it possible that it consists of precisely a words $\mathtt{A}$, b words $\mathtt{B}$, c words $\mathtt{AB}$, and d words $\mathtt{BA}$?

In other words, determine, if it's possible to concatenate these $a+b+c+d$ words in some order so that the resulting string is s. Each of the $a+b+c+d$ words must be used exactly once in the concatenation, but you can choose the order in which they are concatenated.

Input

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

The first line of each test case contains four integers a, b, c, d ($0 \le a, b, c, d \le 2 \cdot 10^5$) — the number of times that words $\mathtt{A}$, $\mathtt{B}$, $\mathtt{AB}$, $\mathtt{BA}$ respectively must be used in the sentence.

The second line contains the string s (s consists only of the characters $\mathtt{A}$ and $\mathtt{B}$, $1 \le |s| \le 2 \cdot 10^5$, $|s| = a+b+2c+2d$) — the sentence. Notice that the condition $|s| = a+b+2c+2d$ (here $|s|$ denotes the length of the string s) is equivalent to the fact that s is as long as the concatenation of the $a+b+c+d$ words.

The sum of the lengths of s over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case output $\mathtt{YES}$ if it is possible that the sentence s consists of precisely a words $\mathtt{A}$, b words $\mathtt{B}$, c words $\mathtt{AB}$, and d words $\mathtt{BA}$, and $\mathtt{NO}$ otherwise. You can output each letter in any case.

Example

input

```
8
1 0 0 0
B
0 0 1 0
AB
1 1 0 1
ABAB
1 0 1 1
ABAAB
1 1 2 2
BAABBABBAA
1 1 2 3
ABABABBAABAB
2 3 5 4
AABAABBABAAABABBABBBABB
1 3 3 10
BBABABABABBBABABABABABABAABABA
```

output

```
NO
YES
YES
YES
YES
YES
NO
YES
```

Note

In the first test case, the sentence s is $\mathtt{B}$. Clearly, it can't consist of a single word $\mathtt{A}$, so the answer is $\mathtt{NO}$.

In the second test case, the sentence s is $\mathtt{AB}$, and it's possible that it consists of a single word $\mathtt{AB}$, so the answer is $\mathtt{YES}$.

In the third test case, the sentence s is $\mathtt{ABAB}$, and it's possible that it consists of one word $\mathtt{A}$, one word $\mathtt{B}$, and one word $\mathtt{BA}$, as $\mathtt{A} + \mathtt{BA} + \mathtt{B} = \mathtt{ABAB}$.

In the fourth test case, the sentence s is $\mathtt{ABAAB}$, and it's possible that it consists of one word $\mathtt{A}$, one word $\mathtt{AB}$, and one word $\mathtt{BA}$, as $\mathtt{A} + \mathtt{BA} + \mathtt{AB} = \mathtt{ABAAB}$.

In the fifth test case, the sentence s is $\mathtt{BAABBABBAA}$, and it's possible that it consists of one word $\mathtt{A}$, one word $\mathtt{B}$, two words $\mathtt{AB}$, and two words $\mathtt{BA}$, as $\mathtt{BA} + \mathtt{AB} + \mathtt{B} + \mathtt{AB} + \mathtt{BA} + \mathtt{A} = \mathtt{BAABBABBAA}$.

C. Bring Balance

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

Alina has a bracket sequence s of length $2n$, consisting of n opening brackets '(' and n closing brackets ')'. As she likes balance, she wants to turn this bracket sequence into a balanced bracket sequence.

In one operation, she can reverse any substring of s.

What's the smallest number of operations that she needs to turn s into a balanced bracket sequence? It can be shown that it's always possible in at most n operations.

As a reminder, a sequence of brackets is called balanced if one can turn it into a valid math expression by adding characters + and 1. For example, sequences (( )) ( ), ( ), and (( ) (( ))) are balanced, while) (, (( ), and (( ))) (are not.

Input

The first line of the input contains a single integer t $(1 \le t \le 2 \cdot 10^4)$ — the number of test cases. The description of the test cases follows.

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

The second line of each test case contains a string s of length $2n$, consisting of n opening and n closing brackets.

The sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, in the first line output a single integer k $(0 \le k \le n)$ — the smallest number of operations required.

The i-th of the next k lines should contain two integers l_i, r_i $(1 \le l_i \le r_i \le 2n)$, indicating that in the i-th operation, Alina will reverse the substring $s_l s_{l+1} \dots s_{r-1} s_r$. Here the numeration starts from 1.

If there are multiple sequences of operations with the smallest length which transform the sequence into a balanced one, you can output any of them.

Example

input

```
3
2
(())
5
())((()))(
6
())((()))(()
```

output

```
0
2
3 4
9 10
1
2 11
```

Note

In the first test case, the string is already balanced.

In the second test case, the string will be transformed as follows: ( )) ((( ))) ($\rightarrow$ ( ) ( ) (( ))) ($\rightarrow$ ( ) ( ) (( )) ( ), where the last string is balanced.

In the third test case, the string will be transformed to ((( ))) ((( ))), which is balanced.

D1. Permutation Weight (Easy Version)

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

This is an easy version of the problem. The difference between the easy and hard versions is that in this version, you can output any permutation with the smallest weight.

You are given a permutation $p_1, p_2, \dots, p_n$ of integers from 1 to n.

Let's define the weight of the permutation $q_1, q_2, \dots, q_n$ of integers from 1 to n as

$$|q_1 - p_{q_2}| + |q_2 - p_{q_3}| + \dots + |q_{n-1} - p_{q_n}| + |q_n - p_{q_1}|$$

You want your permutation to be as lightweight as possible. Find any permutation q with the smallest possible weight.

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 the test cases follows.

The first line of each test case contains a single integer n $(2 \le n \le 200)$ — the size of the permutation.

The second line of each test case contains n integers $p_1, p_2, \dots, p_n$ $(1 \le p_i \le n$, all p_i are distinct$)$ — the elements of the permutation.

The sum of n over all test cases doesn't exceed 400.

Output

For each test case, output n integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$, all q_i are distinct) — one of the permutations with the smallest weight.

Example

input

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

output

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

Note

In the first test case, there are two permutations of length 2: $(1, 2)$ and $(2, 1)$. Permutation $(1, 2)$ has weight $|1 - p_2| + |2 - p_1| = 0$, and permutation $(2, 1)$ has the same weight: $|2 - p_1| + |1 - p_2| = 0$. You can output any of these permutations in this version.

In the second test case, the weight of the permutation $(1, 3, 4, 2)$ is $|1 - p_3| + |3 - p_4| + |4 - p_2| + |2 - p_1| = |1 - 1| + |3 - 4| + |4 - 3| + |2 - 2| = 2$. There are no permutations with smaller weights.

In the third test case, the weight of the permutation $(1, 4, 2, 3, 5)$ is $|1 - p_4| + |4 - p_2| + |2 - p_3| + |3 - p_5| + |5 - p_1| = |1 - 2| + |4 - 4| + |2 - 3| + |3 - 1| + |5 - 5| = 4$. There are no permutations with smaller weights.

D2. Permutation Weight (Hard Version)

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

This is a hard version of the problem. The difference between the easy and hard versions is that in this version, you have to output the lexicographically smallest permutation with the smallest weight.

You are given a permutation $p_1, p_2, \ldots, p_n$ of integers from 1 to n.

Let's define the weight of the permutation $q_1, q_2, \ldots, q_n$ of integers from 1 to n as

$$|q_1 - p_{q_2}| + |q_2 - p_{q_3}| + \ldots + |q_{n-1} - p_{q_n}| + |q_n - p_{q_1}|$$

You want your permutation to be as lightweight as possible. Among the permutations q with the smallest possible weight, find the lexicographically smallest.

Permutation $a_1, a_2, \ldots, a_n$ is lexicographically smaller than permutation $b_1, b_2, \ldots, b_n$, if there exists some $1 \le i \le n$ such that $a_j = b_j$ for all $1 \le j < i$ and $a_i < b_i$.

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 the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 200$) — the size of the permutation.

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$, all p_i are distinct) — the elements of the permutation.

The sum of n over all test cases doesn't exceed 400.

Output

For each test case, output n integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$, all q_i are distinct) — the lexicographically smallest permutation with the smallest weight.

Example

input

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

```
5 4 3 2 1
```

output

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

Note

In the first test case, there are two permutations of length 2: $(1, 2)$ and $(2, 1)$. Permutation $(1, 2)$ has weight $|1 - p_2| + |2 - p_1| = 0$, and the permutation $(2, 1)$ has the same weight: $|2 - p_1| + |1 - p_2| = 0$. In this version, you have to output the lexicographically smaller of them — $(1, 2)$.

In the second test case, the weight of the permutation $(1, 3, 4, 2)$ is $|1 - p_3| + |3 - p_4| + |4 - p_2| + |2 - p_1| = |1 - 1| + |3 - 4| + |4 - 3| + |2 - 2| = 2$. There are no permutations with smaller weights.

In the third test case, the weight of the permutation $(1, 3, 4, 2, 5)$ is $|1 - p_3| + |3 - p_4| + |4 - p_2| + |2 - p_5| + |5 - p_1| = |1 - 3| + |3 - 2| + |4 - 4| + |2 - 1| + |5 - 5| = 4$. There are no permutations with smaller weights.

E. The Ultimate LIS Problem

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

It turns out that this is exactly the 100-th problem of mine that appears in some programming competition. So it has to be special! And what can be more special than another problem about LIS...

You are given a permutation $p_1, p_2, \ldots, p_{2n+1}$ of integers from 1 to $2n + 1$. You will have to process q updates, where the i-th update consists in swapping p_{u_i}, p_{v_i}.

After each update, find any cyclic shift of p with $LIS \le n$, or determine that there is no such shift. (Refer to the output section for details).

Here $LIS(a)$ denotes the length of longest strictly increasing subsequence of a.

Hacks are disabled in this problem. Don't ask why.

Input

The first line of the input contains two integers n, q ($2 \le n \le 10^5$, $1 \le q \le 10^5$).

The second line of the input contains $2n + 1$ integers $p_1, p_2, \ldots, p_{2n+1}$ ($1 \le p_i \le 2n + 1$, all p_i are distinct) — the elements of p.

The i-th of the next q lines contains two integers u_i, v_i ($1 \le u_i, v_i \le 2n + 1$, $u_i \neq v_i$) — indicating that you have to swap elements p_{u_i}, p_{v_i} in the i-th update.

Output

After each update, output **any** k $(0 \le k \le 2n)$, such that the length of the longest increasing subsequence of $(p_{k+1}, p_{k+2}, \ldots, p_{2n+1}, p_1, \ldots, p_k)$ doesn't exceed n, or -1, if there is no such k.

Example

input

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

output

```
-1
-1
2
-1
4
0
```

Note

After the first update, our permutation becomes $(5, 2, 3, 4, 1)$. We can show that all its cyclic shifts have $LIS \ge 3$.

After the second update, our permutation becomes $(1, 2, 3, 4, 5)$. We can show that all its cyclic shifts have $LIS \ge 3$.

After the third update, our permutation becomes $(1, 2, 3, 5, 4)$. Its shift by 2 is $(3, 5, 4, 1, 2)$, and its $LIS = 2$.

After the fourth update, our permutation becomes $(1, 2, 3, 4, 5)$. We can show that all its cyclic shifts have $LIS \ge 3$.

After the fifth update, our permutation becomes $(4, 2, 3, 1, 5)$. Its shift by 4 is $(5, 4, 2, 3, 1)$, and its $LIS = 2$.

After the fifth update, our permutation becomes $(4, 5, 3, 1, 2)$. Its shift by 0 is $(4, 5, 3, 1, 2)$, and its $LIS = 2$.