Problem E. Excellent XOR Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

For an array $[b_1, b_2, \ldots, b_k]$ of integers, let's define its **weight** as the $\oplus$ of all its elements.

Here $\oplus$ denotes the **bitwise exclusive OR** operation. For example, $13 \oplus 6 = 11$, because in binary, $13 = \texttt{1101}$ and $6 = \texttt{0110}$, so their $\oplus$ is $\texttt{1011} = 11$. The weight of array $[13, 1, 4]$, for example, is 8.

You are given an array $[a_1, a_2, \ldots, a_n]$ of integers. We want to divide it into several (**more than one**) consecutive subarrays whose weights are distinct. Determine if this is possible. If it is possible, find one of such partitions.

Input

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

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$) — the elements 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, if no such partitioning exists, print `NO`.

Otherwise, print `YES`. On the following line, print a single integer k ($2 \le k \le n$) — the number of subarrays into which you are splitting a.

On the i-th of the next k lines print two numbers l_i, r_i ($1 \le l_i \le r_i \le n$), denoting that the i-th of your arrays is $[a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}]$. You can print these subarrays in any order, but each number from 1 to n must appear in **exactly one** of the segments $[l_i, r_i]$.

You can print `YES` and `NO` in any case (e.g. the strings `yEs`, `yes`, `Yes` will be taken as a positive answer).

Example

standard input	standard output
4 2 0 0 3 1 2 3 5 16 8 4 2 1 6 42 42 42 42 42 42	NO YES 3 1 1 2 2 3 3 YES 2 1 1 2 5 NO

Note

In the first test case, there is no way to split $[0, 0]$ into at least two subarrays with distinct $\oplus$s.

In the second test case, you can split array $[1, 2, 3]$ into 3 subarrays $[1], [2], [3]$ correspondingly, with $\oplus$s $1, 2, 3$ correspondingly.