CodeTON Round 1 (Div. 1 + Div. 2, Rated, Prizes!)

A. Good Pairs

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

You are given an array $a_1, a_2, \ldots, a_n$ of positive integers. A *good pair* is a pair of indices (i, j) with $1 \leq i, j \leq n$ such that, for all $1 \leq k \leq n$, the following equality holds:

$$|a_i - a_k| + |a_k - a_j| = |a_i - a_j|,$$

where $|x|$ denotes the absolute value of x.

Find a good pair. Note that i can be equal to j.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. Description of the test cases follows.

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) where a_i is the i-th element of the array.

The sum of n for all test cases is at most $2 \cdot 10^5$.

Output

For each test case, print a single line with two space-separated indices i and j which form a good pair of the array. The case $i = j$ is allowed. It can be shown that such a pair always exists. If there are multiple good pairs, print any of them.

Example

input

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

output

```
2 3
1 2
1 1
```

Note

In the first case, for $i = 2$ and $j = 3$ the equality holds true for all k:

- $k = 1$: $|a_2 - a_1| + |a_1 - a_3| = |2 - 5| + |5 - 7| = 5 = |2 - 7| = |a_2 - a_3|$,
- $k = 2$: $|a_2 - a_2| + |a_2 - a_3| = |2 - 2| + |2 - 7| = 5 = |2 - 7| = |a_2 - a_3|$,
- $k = 3$: $|a_2 - a_3| + |a_3 - a_3| = |2 - 7| + |7 - 7| = 5 = |2 - 7| = |a_2 - a_3|$.

B. Subtract Operation

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

You are given a list of n integers. You can perform the following operation: you choose an element x from the list, erase x from the list, and subtract the value of x from all the remaining elements. Thus, in one operation, the length of the list is decreased by exactly 1.

Given an integer k ($k > 0$), find if there is some sequence of $n - 1$ operations such that, after applying the operations, the only remaining element of the list is equal to k.

Input

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

The first line of each test case contains two integers n and k ($2 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^9$), the number of integers in the list, and the target value, respectively.

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

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

Output

For each test case, print YES if you can achieve k with a sequence of $n - 1$ operations. Otherwise, print NO.

You may print each letter in any case (for example, "YES", "Yes", "yes", "yEs" will all be recognized as a positive answer).

Example

input

```
4
4 5
4 2 2 7
5 4
1 9 1 3 4
2 17
17 0
2 17
18 18
```

output

```
YES
NO
YES
NO
```

Note

In the first example we have the list $\{4, 2, 2, 7\}$, and we have the target $k = 5$. One way to achieve it is the following: first we choose the third element, obtaining the list $\{2, 0, 5\}$. Next we choose the first element, obtaining the list $\{-2, 3\}$. Finally, we choose the first element, obtaining the list $\{5\}$.

C. Make Equal With Mod

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

You are given an array of n non-negative integers $a_1, a_2, \ldots, a_n$. You can make the following operation: choose an integer $x \ge 2$ and replace each number of the array by the remainder when dividing that number by x, that is, for all $1 \le i \le n$ set a_i to $a_i \bmod x$.

Determine if it is possible to make all the elements of the array equal by applying the operation zero or more times.

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 an integer n ($1 \le n \le 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) where a_i is the i-th element of the array.

The sum of n for all test cases is at most $2 \cdot 10^5$.

Output

For each test case, print a line with YES if you can make all elements of the list equal by applying the operation. Otherwise, print NO.

You may print each letter in any case (for example, "YES", "Yes", "yes", "yEs" will all be recognized as a positive answer).

Example

input

```
4
4
2 5 6 8
3
1 1 1
5
4 1 7 0 8
4
5 9 17 5
```

output

```
YES
YES
NO
YES
```

Note

In the first test case, one can apply the operation with $x = 3$ to obtain the array $[2, 2, 0, 2]$, and then apply the operation with $x = 2$ to obtain $[0, 0, 0, 0]$.

In the second test case, all numbers are already equal.

In the fourth test case, applying the operation with $x = 4$ results in the array $[1, 1, 1, 1]$.

D. K-good

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We say that a positive integer n is k-good for some positive integer k if n can be expressed as a sum of k positive integers which give k distinct remainders when divided by k.

Given a positive integer n, find some $k \geq 2$ so that n is k-good or tell that such a k does not exist.

Input

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

Each test case consists of one line with an integer n ($2 \leq n \leq 10^{18}$).

Output

For each test case, print a line with a value of k such that n is k-good ($k \geq 2$), or -1 if n is not k-good for any k. If there are multiple valid values of k, you can print any of them.

Example

input

```
5
2
4
6
15
20
```

output

```
-1
-1
3
3
5
```

Note

6 is a 3-good number since it can be expressed as a sum of 3 numbers which give different remainders when divided by 3: $6 = 1 + 2 + 3$.

15 is also a 3-good number since $15 = 1 + 5 + 9$ and $1, 5, 9$ give different remainders when divided by 3.

20 is a 5-good number since $20 = 2 + 3 + 4 + 5 + 6$ and $2, 3, 4, 5, 6$ give different remainders when divided by 5.

E. Equal Tree Sums

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

You are given an undirected unrooted tree, i.e. a connected undirected graph without cycles.

You must assign a **nonzero** integer weight to each vertex so that the following is satisfied: if any vertex of the tree is removed, then each of the remaining connected components has the same sum of weights in its vertices.

Input

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

The first line of each test case contains an integer n ($3 \leq n \leq 10^5$) — the number of vertices of the tree.

The next $n - 1$ lines of each case contain each two integers u, v ($1 \leq u, v \leq n$) denoting that there is an edge between vertices u and v. It is guaranteed that the given edges form a tree.

The sum of n for all test cases is at most 10^5.

Output

For each test case, you must output one line with n space separated integers $a_1, a_2, \dots, a_n$, where a_i is the weight assigned to vertex i. **The weights must satisfy $-10^5 \le a_i \le 10^5$ and $a_i \neq 0$.**

It can be shown that there always exists a solution satisfying these constraints. If there are multiple possible solutions, output any of them.

Example

input

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

output

```
-3 5 1 2 2
1 1 1
```

Note

In the first case, when removing vertex 1 all remaining connected components have sum 5 and when removing vertex 3 all remaining connected components have sum 2. When removing other vertices, there is only one remaining connected component so all remaining connected components have the same sum.

F. Parametric MST

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

You are given n integers $a_1, a_2, \dots, a_n$. For any real number t, consider the complete weighted graph on n vertices $K_n(t)$ with weight of the edge between vertices i and j equal to $w_{ij}(t) = a_i \cdot a_j + t \cdot (a_i + a_j)$.

Let $f(t)$ be the cost of the minimum spanning tree of $K_n(t)$. Determine whether $f(t)$ is bounded above and, if so, output the maximum value it attains.

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 an integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices of the graph.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($-10^6 \le a_i \le 10^6$).

The sum of n for all test cases is at most $2 \cdot 10^5$.

Output

For each test case, print a single line with the maximum value of $f(t)$ (it can be shown that it is an integer), or `INF` if $f(t)$ is not bounded above.

Example

input

```
5
2
1 0
2
-1 1
3
1 -1 -2
3
3 -1 -2
4
1 2 3 -4
```

output

```
INF
-1
INF
-6
-18
```

G. Cycle Palindrome

time limit per test: 1 second

memory limit per test: 256 megabytes
input: standard input
output: standard output

We say that a sequence of n integers $a_1, a_2, \ldots, a_n$ is a palindrome if for all $1 \leq i \leq n$, $a_i = a_{n-i+1}$. You are given a sequence of n integers $a_1, a_2, \ldots, a_n$ and you have to find, if it exists, a *cycle permutation* σ so that the sequence $a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(n)}$ is a palindrome.

A permutation of $1, 2, \ldots, n$ is a bijective function from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$. We say that a permutation σ is a cycle permutation if $1, \sigma(1), \sigma^2(1), \ldots, \sigma^{n-1}(1)$ are pairwise different numbers. Here $\sigma^m(1)$ denotes $\underbrace{\sigma(\sigma(\ldots\sigma}_{m \text{ times}}(1)\ldots))$.

Input

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

The first line of each test case contains an integer n ($2 \leq n \leq 2 \cdot 10^5$) — the size of the sequence.

The second line of each test case contains n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq n$).

The sum of n for all test cases is at most $2 \cdot 10^5$.

Output

For each test case, output one line with YES if a cycle permutation exists, otherwise output one line with NO.

If the answer is YES, output one additional line with n integers $\sigma(1), \sigma(2), \ldots, \sigma(n)$, the permutation. If there is more than one permutation, you may print any.

Example

input

```
3
4
1 2 2 1
3
1 2 1
7
1 3 3 3 1 2 2
```

output

```
YES
3 1 4 2
NO
YES
5 3 7 2 6 4 1
```

H. Equal LCM Subsets

time limit per test: 10 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given two sets of positive integers A and B. You have to find two non-empty subsets $S_A \subseteq A$, $S_B \subseteq B$ so that the least common multiple (LCM) of the elements of S_A is equal to the least common multiple (LCM) of the elements of S_B.

Input

The input consists of multiple test cases. The first line of the input contains one integer t ($1 \leq t \leq 200$), the number of test cases.

For each test case, there is one line containing two integers n, m ($1 \leq n, m \leq 1000$), the sizes of the sets A and B, respectively.

The next line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 4 \cdot 10^{36}$), the elements of A.

The next line contains m distinct integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 4 \cdot 10^{36}$), the elements of B.

The sum of n for all test cases and the sum of m for all test cases is at most 1000.

Output

For each test case, if there do not exist two subsets with equal least common multiple, output one line with NO.

Otherwise, output one line with YES, followed by a line with two integers $|S_A|, |S_B|$ ($1 \leq |S_A| \leq n$, $1 \leq |S_B| \leq m$), the sizes of the subsets S_A and S_B

The next line should contain $|S_A|$ integers $x_1, x_2, \ldots, x_{|S_A|}$, the elements of S_A, followed by a line with $|S_B|$ integers $y_1, y_2, \ldots, y_{|S_B|}$, the elements of S_B. If there are multiple possible pairs of subsets, you can print any.

Example

input

```
4
3 4
5 6 7
2 8 9 10
4 4
5 6 7 8
2 3 4 9
1 3
1
1 2 3
5 6
3 4 9 7 8
2 15 11 14 20 12
```

output

```
NO
YES
1 2
6
2 3
YES
1 1
1
1
YES
3 2
3 7 4
12 14
```

I. Neighbour Ordering

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Given an undirected graph G, we say that a *neighbour ordering* is an ordered list of all the neighbours of a vertex for each of the vertices of G. Consider a given neighbour ordering of G and three vertices u, v and w, such that v is a neighbor of u and w. We write $u <_v w$ if u comes after w in v's neighbor list.

A neighbour ordering is said to be *good* if, for each simple cycle $v_1, v_2, \ldots, v_c$ of the graph, one of the following is satisfied:

- $v_1 <_{v_2} v_3, v_2 <_{v_3} v_4, \ldots, v_{c-2} <_{v_{c-1}} v_c, v_{c-1} <_{v_c} v_1, v_c <_{v_1} v_2$.
- $v_1 >_{v_2} v_3, v_2 >_{v_3} v_4, \ldots, v_{c-2} >_{v_{c-1}} v_c, v_{c-1} >_{v_c} v_1, v_c >_{v_1} v_2$.

Given a graph G, determine whether there exists a good neighbour ordering for it and construct one if it does.

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 two integers n and m ($2 \le n \le 3 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$), the number of vertices and the number of edges of the graph.

The next m lines each contain two integers u, v ($0 \le u, v < n$), denoting that there is an edge connecting vertices u and v. It is guaranteed that the graph is connected and there are no loops or multiple edges between the same vertices.

The sum of n and the sum of m for all test cases are at most $3 \cdot 10^5$.

Output

For each test case, output one line with YES if there is a good neighbour ordering, otherwise output one line with NO. You can print each letter in any case (upper or lower).

If the answer is YES, additionally output n lines describing a good neighbour ordering. In the i-th line, output the neighbours of vertex i in order.

If there are multiple good neigbour orderings, print any.

Example

input

```
3
5 6
0 1
0 2
1 2
2 3
3 4
4 1
2 1
0 1
6 10
0 1
2 0
```

```
0 3
0 4
1 2
1 4
2 3
2 5
3 5
4 5
```

output

```
YES
1 2
4 2 0
0 1 3
2 4
3 1
YES
1
0
NO
```