Educational Codeforces Round 105 (Rated for Div. 2)

A. ABC String

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

You are given a string a, consisting of n characters, n is even. For each i from 1 to n a_i is one of 'A', 'B' or 'C'.

A bracket sequence is a string containing only characters "`(`" and "`)`". A regular bracket sequence is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters "1" and "+" between the original characters of the sequence. For example, bracket sequences "`()()`" and "`(())`" are regular (the resulting expressions are: "`(1)+(1)`" and "`((1+1)+1)`"), and "`)(`", "`(`" and "`)`" are not.

You want to find a string b that consists of n characters such that:

- b is a regular bracket sequence;
- if for some i and j ($1 \le i, j \le n$) $a_i = a_j$, then $b_i = b_j$.

In other words, you want to replace all occurrences of 'A' with the same type of bracket, then all occurrences of 'B' with the same type of bracket and all occurrences of 'C' with the same type of bracket.

Your task is to determine if such a string b exists.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

Then the descriptions of t testcases follow.

The only line of each testcase contains a string a. a consists only of uppercase letters 'A', 'B' and 'C'. Let n be the length of a. It is guaranteed that n is even and $2 \le n \le 50$.

Output

For each testcase print "YES" if there exists such a string b that:

- b is a regular bracket sequence;
- if for some i and j ($1 \le i, j \le n$) $a_i = a_j$, then $b_i = b_j$.

Otherwise, print "NO".

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES are all recognized as positive answer).

Example

input

```
4
AABBAC
CACA
BBBBAC
ABCA
```

output

```
YES
YES
NO
NO
```

Note

In the first testcase one of the possible strings b is "`(())()`".

In the second testcase one of the possible strings b is "`()()`".

B. Berland Crossword

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

Berland crossword is a puzzle that is solved on a square grid with n rows and n columns. Initially all the cells are white.

To solve the puzzle one has to color some cells on the border of the grid black in such a way that:

- exactly U cells in the top row are black;
- exactly R cells in the rightmost column are black;
- exactly D cells in the bottom row are black;
- exactly L cells in the leftmost column are black.

Note that you can color zero cells black and leave every cell white.

Your task is to check if there exists a solution to the given puzzle.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

Then the descriptions of t testcases follow.

The only line of each testcase contains 5 integers n, U, R, D, L ($2 \le n \le 100$; $0 \le U, R, D, L \le n$).

Output

For each testcase print "YES" if the solution exists and "NO" otherwise.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES are all recognized as positive answer).

Example

input

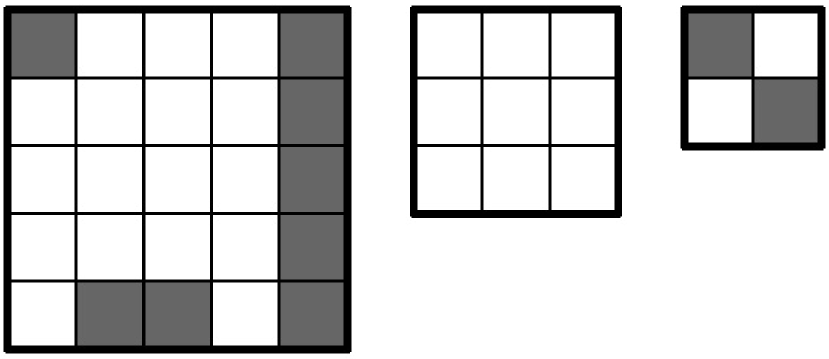
```
4
5 2 5 3 1
3 0 0 0 0
4 4 1 4 0
2 1 1 1 1
```

output

```
YES
YES
NO
YES
```

Note

Here are possible solutions to testcases 1, 2 and 4:



C. 1D Sokoban

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

You are playing a game similar to Sokoban on an infinite number line. The game is discrete, so you only consider integer positions on the line.

You start on a position 0. There are n boxes, the i-th box is on a position a_i. All positions of the boxes are distinct. There are also m special positions, the j-th position is b_j. All the special positions are also distinct.

In one move you can go one position to the left or to the right. If there is a box in the direction of your move, then you push the box to the next position in that direction. If the next position is taken by another box, then that box is also pushed to the next position, and so on. **You can't go through the boxes**. **You can't pull the boxes towards you**.

You are allowed to perform any number of moves (possibly, zero). Your goal is to place as many boxes on special positions as possible. Note that some boxes can be initially placed on special positions.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

Then descriptions of t testcases follow.

The first line of each testcase contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of boxes and the number of special positions, respectively.

The second line of each testcase contains n distinct integers in the increasing order $a_1, a_2, \dots, a_n$ ($-10^9 \le a_1 < a_2 < \dots < a_n \le 10^9$; $a_i \ne 0$) — the initial positions of the boxes.

The third line of each testcase contains m distinct integers in the increasing order $b_1, b_2, \dots, b_m$ ($-10^9 \le b_1 < b_2 < \dots < b_m \le 10^9$; $b_i \ne 0$) — the special positions.

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

Output

For each testcase print a single integer — the maximum number of boxes that can be placed on special positions.

Example

input

```
5
5 6
-1 1 5 11 15
-4 -3 -2 6 7 15
2 2
-1 1
-1000000000 1000000000
2 2
-1000000000 1000000000
-1 1
3 5
-1 1 2
-2 -1 1 2 5
2 1
1 2
10
```

output

```
4
2
0
3
1
```

Note

In the first testcase you can go 5 to the right: the box on position 1 gets pushed to position 6 and the box on position 5 gets pushed to position 7. Then you can go 6 to the left to end up on position -1 and push a box to -2. At the end, the boxes are on positions $[-2, 6, 7, 11, 15]$, respectively. Among them positions $[-2, 6, 7, 15]$ are special, thus, the answer is 4.

In the second testcase you can push the box from -1 to -10^9, then the box from 1 to 10^9 and obtain the answer 2.

The third testcase showcases that you are not allowed to pull the boxes, thus, you can't bring them closer to special positions.

In the fourth testcase all the boxes are already on special positions, so you can do nothing and still obtain the answer 3.

In the fifth testcase there are fewer special positions than boxes. You can move either 8 or 9 to the right to have some box on position 10.

D. Dogeforces

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

The Dogeforces company has k employees. Each employee, except for lower-level employees, has at least 2 subordinates. Lower-level employees have no subordinates. Each employee, except for the head of the company, has exactly one direct supervisor. The head of the company is a direct or indirect supervisor of all employees. It is known that in Dogeforces, each supervisor receives a salary strictly more than all his subordinates.

The full structure of the company is a secret, but you know the number of lower-level employees and for each pair of lower-level employees, the salary of their common supervisor is known (if there are several such supervisors, then the supervisor with the minimum salary). You have to restore the structure of the company.

Input

The first line contains a single integer n ($2 \le n \le 500$) — the number of lower-level employees.

This is followed by n lines, where i-th line contains n integers $a_{i,1}, a_{i,2}, \dots, a_{i,n}$ ($1 \le a_{i,j} \le 5000$) — salary of the common supervisor of employees with numbers i and j. It is guaranteed that $a_{i,j} = a_{j,i}$. Note that $a_{i,i}$ is equal to the salary of the i-th employee.

Output

In the first line, print a single integer k — the number of employees in the company.

In the second line, print k integers $c_1, c_2, \ldots, c_k$, where c_i is the salary of the employee with the number i.

In the third line, print a single integer r — the number of the employee who is the head of the company.

In the following $k-1$ lines, print two integers v and u ($1 \le v, u \le k$) — the number of the employee and his direct supervisor.

Note that the lower-level employees have numbers from 1 to n, and for the rest of the employees, you have to assign numbers from $n+1$ to k. If there are several correct company structures, you can print any of them.

Example

input

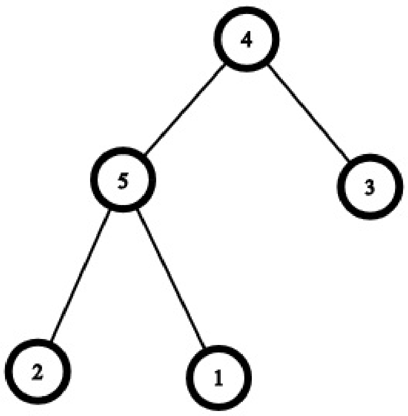
```
3
2 5 7
5 1 7
7 7 4
```

output

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

Note

One of the possible structures in the first example:



E. A-Z Graph

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

You are given a directed graph consisting of n vertices. Each directed edge (or arc) labeled with a single character. Initially, the graph is empty.

You should process m queries with it. Each query is one of three types:

- "$+\ u\ v\ c$" — add arc from u to v with label c. It's guaranteed that there is no arc (u, v) in the graph at this moment;
- "$-\ u\ v$" — erase arc from u to v. It's guaranteed that the graph contains arc (u, v) at this moment;
- "$?\ k$" — find the sequence of k vertices $v_1, v_2, \ldots, v_k$ such that there exist both routes $v_1 \to v_2 \to \cdots \to v_k$ and $v_k \to v_{k-1} \to \cdots \to v_1$ and if you write down characters along both routes you'll get the same string. You can visit the same vertices any number of times.

Input

The first line contains two integers n and m ($2 \le n \le 2 \cdot 10^5$; $1 \le m \le 2 \cdot 10^5$) — the number of vertices in the graph and the number of queries.

The next m lines contain queries — one per line. Each query is one of three types:

- "$+\ u\ v\ c$" ($1 \le u, v \le n$; $u \ne v$; c is a lowercase Latin letter);
- "$-\ u\ v$" ($1 \le u, v \le n$; $u \ne v$);
- "$?\ k$" ($2 \le k \le 10^5$).

It's guaranteed that you don't add multiple edges and erase only existing edges. Also, there is at least one query of the third type.

Output

For each query of the third type, print YES if there exist the sequence $v_1, v_2, \ldots, v_k$ described above, or NO otherwise.

Example

input

```
3 11
+ 1 2 a
+ 2 3 b
+ 3 2 a
+ 2 1 b
? 3
? 2
- 2 1
- 3 2
+ 2 1 c
+ 3 2 d
? 5
```

output

```
YES
NO
YES
```

Note

In the first query of the third type $k = 3$, we can, for example, choose a sequence $[1, 2, 3]$, since $1 \xrightarrow{\text{a}} 2 \xrightarrow{\text{b}} 3$ and $3 \xrightarrow{\text{a}} 2 \xrightarrow{\text{b}} 1$.

In the second query of the third type $k = 2$, and we can't find sequence p_1, p_2 such that arcs (p_1, p_2) and (p_2, p_1) have the same characters.

In the third query of the third type, we can, for example, choose a sequence $[1, 2, 3, 2, 1]$, where $1 \xrightarrow{\text{a}} 2 \xrightarrow{\text{b}} 3 \xrightarrow{\text{d}} 2 \xrightarrow{\text{c}} 1$.

F. Delete The Edges

time limit per test: 8 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an undirected connected graph consisting of n vertices and m edges. Your goal is to destroy all edges of the given graph.

You may choose any vertex as the starting one and begin walking from it along the edges. When you walk along an edge, you destroy it. Obviously, you cannot walk along an edge if it is destroyed.

You can perform the **mode shift** operation at most once during your walk, and this operation can only be performed when you are at some vertex (you cannot perform it while traversing an edge). After the **mode shift**, the edges you go through are deleted in the following way: the first edge after the **mode shift** is not destroyed, the second one is destroyed, the third one is not destroyed, the fourth one is destroyed, and so on. You cannot switch back to the original mode, and you don't have to perform this operation if you don't want to.

Can you destroy all the edges of the given graph?

Input

The first line contains two integers n and m ($2 \le n \le 3000$; $n - 1 \le m \le \min(\frac{n(n-1)}{2}, 3000)$) — the numbef of vertices and the number of edges in the graph.

Then m lines follow, each containing two integers x_i and y_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$) — the endpoints of the i-th edge.

These edges form a connected undirected graph without multiple edges.

Output

If it's impossible to destroy all of the edges, print 0.

Otherwise, print the sequence of your actions as follows. First, print k — the number of actions ($k \le 2m + 2$). Then, print the sequence itself, consisting of k integers. The first integer should be the index of the starting vertex. Then, each of the next integers should be either the index of the next vertex in your traversal, or -1 if you use **mode shift**. You are allowed to use **mode shift** at most once.

If there are multiple answers, print any of them.

Examples

input

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

output

```
4
```

```
1 2 3 1
```

input

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

output

```
8
2 -1 1 2 3 2 4 2
```

input

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

output

```
9
2 3 1 2 -1 4 2 5 2
```

input

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

output

```
8
5 4 2 3 1 -1 2 1
```

input

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

output

```
0
```