Problem B. Disjoint Set Union

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Recently, Little Cyan Fish has been learning about the Disjoint Set Union (DSU) data structure. It is a powerful data structure that allows you to add edges to a graph and test whether two vertices of the graph are connected.

The DSU maintains a rooted forest structure consisting of n vertices. Each vertex x ($1 \le x \le n$) has a unique parent $f[x]$. If $x = f[x]$, then x is the root of its subtree. Initially, each vertex forms a single rooted tree. That is, $f[x] = x$ for all $1 \le x \le n$.

The basic interface of DSU consists of these two operations:

- **find** x: returns the root of the tree where x is located.
- **unite** x y: let $x' \leftarrow$ **find**(x) and $y' \leftarrow$ **find**(y). If $x' = y'$, do nothing. Otherwise, modify the parent of x' to y'.

To speed up the **unite** operation, Little Cyan Fish uses an optimization called *Path Compression*:

- If we call **find**(x) for some vertex x, we set the parent of each vertex from x to the root directly to the root.

The following pseudocode describes the details of the DSU.

Algorithm 1 An implementation of DSU with Path Compression

```
procedure FIND(f, x)
    if x = f[x] then
        return x
    end if
    f[x] ← find(f, f[x])
    return f[x]
end procedure
procedure UNITE(f, x, y)
    x ← FIND(f, x)
    y ← FIND(f, y)
    if x ≠ y then
        f[x] ← y
    end if
end procedure
```

Little Cyan Fish loves the DSU very much, so he would like to play with it. He got an array f of length n, where $f[i] = i$ in the beginning. Then, Little Cyan Fish did the following operations many times (possibly zero):

- Choose an integer $1 \le x \le n$, apply $\textsc{find}(x)$.
- Choose two integers $1 \le x \le n$ and $1 \le y \le n$, apply $\textsc{unite}(x, y)$.

He will give you the array f after all his operations. However, you would like to transform the array f into another given array g by using the DSU operations described above. You are wondering if it is possible to apply any additional operations so that $f[i] = g[i]$ for all $1 \le i \le n$.

Input

There are multiple test cases. The first line contains one integer T ($1 \le T \le 10^5$), representing the number of test cases.

For each test case, the first line contains one positive integer n ($3 \le n \le 1\,000$).

The next line contains n integers $f_1, f_2, \ldots, f_n$ denoting the array f after Little Cyan Fish's operations. It is guaranteed that the array f can be generated by using the operation above.

The following line contains n integers $g_1, g_2, \ldots, g_n$ denoting the array g that you would like to transform the array f into.

It is guaranteed that the sum of n^2 over all test cases does not exceed 5×10^6.

Output

For each test case, if it is impossible to transform the array f into the array g, print a single line `NO`.

Otherwise, the first line of the output should contain a single word `YES`.

The next line of the output should contain an integer m ($0 \le m \le 2 \cdot n^2$), indicating the number of operations you used.

The next m lines describe the operations you used. Each operation is described in the following format:

- 1 x: Call FIND(x).
- 2 x y: Call UNITE(x, y).

If there are multiple solutions, you may print any of them. It can be proved that if any solution exists, then there's a plan consisting of no more than $2 \cdot n^2$ operations.

Example

standard input	standard output
5 3 1 2 3 2 2 3 4 1 2 3 3 1 1 1 2 5 1 2 3 4 5 2 3 4 5 5 5 1 1 1 1 1 1 2 3 4 5 6 1 2 2 4 5 6 1 1 5 1 4 2	YES 1 2 1 2 YES 4 2 3 2 1 4 2 2 1 1 3 YES 4 2 1 2 2 1 3 2 2 4 2 3 5 NO YES 7 2 6 2 2 2 5 1 3 2 2 4 1 2 2 2 1 1 2