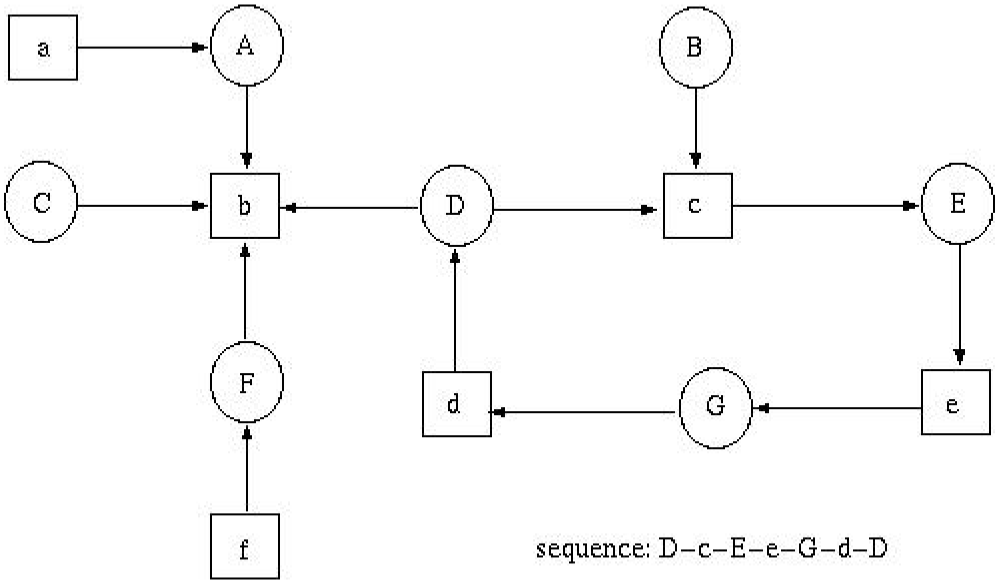
In Operating Systems a special resource-allocation graph algorithm can be used to detect whether there is any deadlock in the system. A resource-allocation graph is a directed graph consisting of two different types of nodes $P = P_1, P_2, \ldots, P_n$, the set consisting of all active processes in the system, and $R = R_1, R_2, \ldots, R_m$, the set consisting of all resource types in the system.

A directed edge from process P_i to resource R_j is denoted by $P_i \longrightarrow R_j$ and means that process P_i requested an instance resource type R_j, and is currently waiting for that resource. A directed edge from resource type R_j to process P_i, is denoted by $R_j \longrightarrow P_i$ and means that an instance of resource type R_j has been allocated to process P_i.

The following figure illustrates a resource-allocation graph where processes are denoted by circles and resources by squares. Notice that if there is a circular wait among the processes, then it implies that a deadlock has occurred.



Given a resource allocation graph in which each resource type has exactly one instance, your job is to determine whether there is a deadlock in the system. In case a deadlock exists, you must also show the sequence of processes and resources involved.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

We will assume that processes are named by capital letters and resources by small letters, so we limit to 26 the number of processes and/or resources. Therefore, the first line of input consists of three numbers N, M and E, respectively, the number of processes, the number of resources and the number of edges. The edges are given in the following lines as pairs of letters linked by a '-' character. Edges are separated by spaces or newlines.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output must be 'NO' if no deadlock is detected. In case a deadlock is detected, the output must be 'YES' followed by the sequence or sequences of circular waits detected, one per line. If more then one sequence is found, they should all be output in increasing order of their length.

Sample Input

```
1

2 2 4
A-b B-a
a-A b-B
```

Sample Output

```
YES
A-b-B-a-A
```