An educational institute has ordered it’s courses according to some criteria. The ordering tells a student which all courses are the pre-requisites for a particular course. An ordering is said to be consistent only if there are no cyclic dependencies in it.

You are supposed to write a program that will take an ordering as input and find out whether or not it is consistent.

Assume that the course names will be less than 10 characters in length and that the maximum number of courses will be 50.

input

The first line will contain a number N (the number of courses may be more than N). This line will be followed by N lines, each having a course name, followed by a number M, giving the number of prerequisites that course has, followed by the names of those pre-requisite courses, one space apart.

Note that the input lines or the list of pre-requisites are not in any specific order.

output

Print "YES" if the ordering is consistent, and "NO", otherwise.

**Input**

5

abc 3 asd sdf dfg

sdf 2 lkj kjh

lkj 2 poi kjh

kjh 1 xyz

xyz 1 sdf

**Output (for the Input shown):**

NO

**Explanation**

In this example, sdf-kjh-xyz-sdf is a cycle, so that sdf becomes a prerequisite for itself!

Test case

course-a 1 course-d

course-b 2 course-d course-a

course-c 1 course-e

course-d 1 course-f

course-e 1 course-f

testcase 2

4

aa 1 bb

dd 1 bb

bb 1 cc

cc 3 aa dd ee

tescase 3

8

aa 1 ee

bb 1 ff

cc 1 gg

dd 1 hh

ee 1 ii

ff 1 ii

gg 1 ii

hh 1 ii

#include <stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX\_COURSES 50

#define MAX\_NAME\_LEN 10

int graph[MAX\_COURSES][MAX\_COURSES];

int Visited[MAX\_COURSES];

char course\_array[MAX\_COURSES][MAX\_NAME\_LEN];

int INDEX = 0;

int find\_index(char string\_array[][MAX\_NAME\_LEN], char \*str, int num);

int find\_cycle(int init\_node, int curr\_node, int graph[][MAX\_COURSES],

int num\_nodes);

int main()

{

int i, j, num\_courses, dep\_count, ind1, ind2, lines;

char str[MAX\_NAME\_LEN];

for(i = 0; i < MAX\_COURSES; i++){

strcpy(course\_array[i], "");

for(j = 0; j < MAX\_COURSES; j++)

graph[i][j] = 0;

}

/\* scanf("%d", &num\_courses); \*/

/\* for(i = 0; i < num\_courses; i++){

scanf("%s", str);

strcpy(course\_array[i], str);

} \*/

scanf("%d", &lines);

for(i = 0; i < lines; i++){

scanf("%s", str);

ind1 = find\_index(course\_array, str, MAX\_COURSES);

if(ind1 == -1){

strcpy(course\_array[INDEX], str);

ind1 = INDEX;

INDEX++;

}

scanf("%d", &dep\_count);

for(j = 0; j < dep\_count; j++){

scanf("%s", str);

ind2 = find\_index(course\_array, str, MAX\_COURSES);

if(ind2 == -1){

strcpy(course\_array[INDEX], str);

ind2 = INDEX;

INDEX++;

}

graph[ind2][ind1] = 1;

}

}

for(i = 0; i < INDEX; i++){

if(find\_cycle(i, i, graph, MAX\_COURSES)){

printf("NO\n");

return 0;

}

}

printf("YES\n");

return 0;

}

int find\_index(char string\_array[][MAX\_NAME\_LEN], char \*str, int num){

int i;

for(i = 0; i < num; i++)

if(strcmp(string\_array[i], str) == 0){

return i;

}

return -1;

}

int find\_cycle(int init\_node, int curr\_node, int graph[][MAX\_COURSES],

int num\_nodes){

int j;

for(j = 0; j < num\_nodes; j++){

if(graph[curr\_node][j] == 1){

if((j == init\_node) || find\_cycle(init\_node, j, graph, num\_nodes))

return 1;

}

}

return 0;

}