```cpp
#include<cstdio>
#include<cstring>
#include<queue>
#define INF 0x3f3f3f3f
using namespace std;
const int MAXN = 100+5;
const int MAXM = 300+5;
int p, n;
int a[MAXN][MAXM];
int dis;
int cx[MAXN], cy[MAXM];
int dx[MAXN], dy[MAXM];
bool vis[MAXM];

bool bfs_findPath() {
    queue<int> q;
    memset(dx, -1, sizeof(dx));
    memset(dy, -1, sizeof(dy));
    // 使用BFS遍历对图的点进行分层，从X中找出一个未匹配点v
    // (所有v)组成第一层，接下来的层都是这样形成--每次查找
    // 匹配点(增广路性质)，直到在Y中找到未匹配点才停止查找，
    // 对X其他未匹配点同样进行查找增广路径(BFS只分层不标记
    // 是否匹配点)
    // 找出X中的所有未匹配点组成BFS的第一层
    dis = INF;
    for(int i = 1; i <= p; ++i) {
        if(cx[i] == -1) {
            q.push(i);
            dx[i] = 0;
        }
    }
    while(!q.empty()) {
        int u = q.front();
        q.pop();
        if(dx[u] > dis) break;// 该路径长度大于dis，等待下一次BFS扩充
        for(int v = 1; v <= n; ++v) {
            if(a[u][v] && dy[v] == -1) {// (u,v)之间有边且v还没有分层
                dy[v] = dx[u] + 1;
                if(cy[v] == -1) dis = dy[v];// v是未匹配点，停止延伸
（查找）,得到本次BFS的最大遍历层次
                else {// v是已匹配点，继续延伸
                    dx[cy[v]] = dy[v] + 1;
                    q.push(cy[v]);
```

```
                }
            }
        }
    }
    return dis != INF;// 若dis为INF说明Y中没有未匹配点，也就是没有增广路径了
}

bool dfs(int u) {
    for(int v = 1; v <= n; ++v) {
        if(!vis[v] && a[u][v] && dy[v] == dx[u] + 1) {
            vis[v] = 1;
            // 层次（也就是增广路径的长度）大于本次查找的dis
            // 是bfs中被break的情况，也就是还不确定是否是增广路
            // 只有等再次调用bfs再判断(每次只找最小增广路集)
            if(cy[v] != -1 && dy[v] == dis) continue;
            if(cy[v] == -1 || dfs(cy[v])) {// 是增广路径，更新匹配集
                cy[v] = u;
                cx[u] = v;
                return true;
            }
        }
    }
    return false;
}

int HK() {
    int ans = 0;
    memset(cx, -1, sizeof(cx));
    memset(cy, -1, sizeof(cy));
    while(bfs_findPath()) {// 有增广路
        memset(vis, 0, sizeof(vis));
        for(int i = 1; i <= p; ++i) {
            // 用DFS查找增广路径，增广路径一定从未匹配点开始
            // 如果查找到一个增广路径，匹配数加一
            if(cx[i] == -1 && dfs(i)) ++ans;
        }
    }
    return ans;
}

int main() {
    int T;
    scanf("%d", &T);
    while(T--) {
```

```
        scanf("%d%d", &p, &n);
        memset(a, 0, sizeof(a));

        for(int i = 1; i <= p; ++i) {
            int nm, x;
            scanf("%d", &nm);
            for(int j = 0; j != nm; ++j) {
                scanf("%d", &x);
                a[i][x] = 1;
            }
        }
        printf("%s\n", HK() == p ? "YES" : "NO");
    }
    return 0;
}
```