Balanced Course Assignment (BCA)

- At the beginning of the semester, the head of a computer science department D have to assign courses to teachers in a balanced way. The department D has m teachers T = {1,2,...,m} and n courses C = {1,2,...,n}. Each teacher t ∈T has a preference list which is a list of courses he/she can teach depending on his/her specialization. We known a list of pairs of conflicting two courses that cannot be assigned to the same teacher as these courses have been already scheduled in the same slot of the timetable. The load of a teacher is the number of courses assigned to her/him. How to assign n courses to m teacher such that each course assigned to a teacher is in his/her preference list, no two conflicting courses are assigned to the same teacher, and the maximal load is minimal.
- Input
  - The input consists of following lines
  - Line 1: contains two integer m and n ($1 \leq m \leq 10$, $1 \leq n \leq 30$)
  - Line i+1: contains an positive integer k and k positive integers indicating the courses that teacher i can teach ($\forall i=1,\ldots,m$)
  - Line m+2: contains an integer k
  - Line i+m+2: contains two integer i and j indicating two conflicting courses ($\forall i=1,\ldots,k$)
- Output
  - The output contains a unique number which is the maximal load of the teachers in the solution found and the value -1 if not solution found.

Balanced Course Assignment (BCA)

- Example

stdin	stdout

4 12
5 1 3 5 10 12
5 9 3 4 8 12
6 1 2 3 4 9 7
7 1 2 3 5 6 10 11
5
1 2
1 3
1 5
2 4
2 5
	3

Balanced Course Assignment (BCA): Hint

- Sử dụng backtracking để duyệt qua tất cả các phương án, kết hợp Branch and Bound để bỏ qua các tính toán dư thừa
- Biểu diễn lời giải: *x*[1…*n*] trong đó *x*[*i*] là giáo viên được phân công dạy môn *i* (*i* = 1, 2…, *n*)
- res: giá trị hàm mục tiêu tối ưu
- Các cấu trúc dữ liệu phụ trợ:
    - load[*t*]: số môn được phân công cho giáo viên *t* (*t* = 1, …, *m*).
    - load[*t*]: được tích lũy dần trong quá trình duyệt
- Branch and Bound:
    - Try(i): thử tất cả các giá trị (giáo viên) cho x[*i*]
        - Với mỗi giá trị (giáo viên) *t* được gán cho *x*[*i*], thực hiện:
            - Cập nhật: load[*t*] = load[*t*] + 1
            - Nếu load[*t*] < res thì gọi tiếp Try(*i*+1)
            - Ngược lại thì thuật toán quay lui để duyệt giá trị khác cho *x*[*i*]

Implementation

```
#include <bits/stdc++.h>
using namespace std;
#define N 50
vector<int> T[N];// T[i] is the list of teachers that can be assigned to course i
int m,n;
bool conflict[N][N];
int x[N];
int load[N];
int res;
```

Implementation

```
void input(){
    cin >> m >> n;
    for(int t = 1; t <= m; t++){
        int k; cin >> k;
        for(int j = 1; j <= k; j++){
            int c;    cin >> c;  T[c].push_back(t);
        }
    }
    int K;
    for(int i = 1; i <= n; i++)
        for(int j =1 ; j <= n; j++)
            conflict[i][j] = false;
    cin >> K;
    for(int k =1 ; k <= K; k++){
        int i,j;  cin >> i >> j;
        conflict[i][j] = true;   conflict[j][i] = true;
    }
}
```

Implementation

```
bool check(int t, int k){
    for(int i =1; i <= k-1; i++){
        if(conflict[i][k] && x[i] == t) return false;
    }
    return true;
}
void solution(){
    int maxLoad = 0;
    for(int t = 1; t <= m; t++){
        maxLoad = max(maxLoad, load[t]);
    }
    if(maxLoad < res) res = maxLoad;
}
```

Implementation

```
void Try(int k){
    for(int i = 0; i < T[k].size(); i++){
        int t = T[k][i];
        if(check(t,k)){
            x[k] = t; // assign course k to teacher t
            load[t] += 1;
            if(k == n)solution();
            else{
                    if(load[t] < res)
                        Try(k+1);
            }
            load[t] -= 1;
        }
    }
}
int main(){
    input();
    for(int t = 1; t <= m; t++) load[t] = 0;
    res = 1e9;
    Try(1);
    cout << res;
    return 0;
}
```