Lab8 Solutions

Note: ในเฉลยของแลปนี้ กราฟที่เป็น Adjacency list จะ implement โดยใช้ vector แทนการ implement linked list ขึ้นมาใช้เอง เพื่อให้โค้ดเฉลยเข้าใจง่ายและกระชับ vector มีลักษณะคล้าย array เข้าถึงโดยใช้เลข index ได้เหมือนกัน แต่จะมีความยืดหยุ่นมากกว่า สามารถเพิ่มและลบข้อมูลได้ง่าย มีคำสั่งที่ใช้บ่อยประมาณนี้

- vector<variable type> name ประกาศ vector
- name.push_back(data) นำข้อมูลไปเพิ่มตรงท้ายสุดของ vector
- name.size() ใช้หาขนาดของ vector

ศึกษาเพิ่มเติมได้ที่ https://cplusplus.com/reference/vector/vector/

Degree of Vertex

เขียนกราฟโดยใช้ adjacency list รับข้อมูลและเพิ่ม edge ตามปกติ degree ของ vertex k ที่ต้องการหา คือขนาด vector ใน index ที่ k ของ graph

Example Code

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n,m,k,co=0,a,b;
    cin >> n >> m >> k;
    vector<int> graph[n];
    for(int i=0;i<m;i++)
    {
        cin >> a >> b;
        graph[a].push_back(b);
        graph[b].push_back(a);
    }
    cout << graph[k].size();
}
```

$time\ complexity : O(m)$

Road Construction

การทำ Breath First Search(BFS) หรือ Depth First Search(DFS) ที่ vertex ใด ๆ จะมีการเข้าถึง vertex ทั้งหมดที่เชื่อมต่อกับ vertex นั้นได้ ดังนั้นสิ่งที่ต้องทำคือ หาว่าต้อง *Search* ทั้งหมดที่ครั้งจึงจะเข้าถึงทุก *vertex* แล้วจะได้จำนวนกลุ่มของเมืองที่เชื่อมกัน ถ้าหากจะสร้างถนนให้เข้าถึงกันได้ทั้งหมดต้องสร้างถนนจำนวนกลุ่ม -1 เส้น

Example Code

```cpp
#include <bits/stdc++.h>
using namespace std;

void dfs(int u,vector<int>* graph,bool* visited)
{
    if(visited[u]) return;

    visited[u] = true;
    for(int i=0;i<graph[u].size();i++) dfs(graph[u][i],graph,visited);
}

int main()
{
    int n,m,u,v,count=0;
    cin >> n >> m;
    vector<int> graph[n];
    bool visited[n] = {false};
    for(int i=0;i<m;i++)
    {
        cin >> u >> v;
        graph[u-1].push_back(v-1);
        graph[v-1].push_back(u-1);
    }
    for(int i=0;i<n;i++) if(!visited[i]) dfs(i,graph,visited),count++;
    cout << count-1;
}
```

$time\ complexity : O(n+m)$

Choir

พิจารณาเลือกนักเรียนที่เป็นเพื่อนกันมาทีละ 2 คู่ผ่าน adjacency list(เพราะว่าการ loop ใช้เวลาน้อยกว่า) แล้วตรวจสอบว่านักเรียนทั้ง 4 คนนี้เป็นเพื่อนกันเกิน 5 คู่หรือไม่ผ่าน adjacency matrix (เพราะว่าการเข้าถึง edge ใช้เวลาน้อยกว่า) เนื่องจากการพิจารณาแบบนี้อาจจะทำให้เกิดวงประสานเสียงที่มีสมาชิกซ้ำกันทั้งหมดได้ โดยวงประสานเสียงที่มีนักเรียนเป็นเพื่อนกัน 5 คู่หนึ่งวงสามารถเกิดจากการเลือกนักเรียนที่เป็นเพื่อนกันสองคู่ได้ทั้งหมด 4 แบบ ในขณะที่วงประสานเสียงที่มีนักเรียนเป็นเพื่อนกัน 6 คู่มีได้ถึง 6 แบบ ดังนั้นควรนับวงประสานเสียงแยกเป็นแบบ 5 คู่และ 6 คู่ แล้วนำมาหารกับ 4 และ 6 ตามลำดับ แล้วค่อยรวมเป็นคำตอบ

Example Code

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n,e;
    cin >> n >> e;
    vector<int> graph1[n];
    int graph2[n][n] = {0};
    for(int i=0;i<e;i++)
    {
        int x,y;
        cin >> x >> y;
        graph1[x-1].push_back(y-1);
        graph1[y-1].push_back(x-1);
        graph2[x-1][y-1] = 1;
        graph2[y-1][x-1] = 1;
    }
```

```
    int count5=0,count6=0;
    for(int i=0;i<n;i++)
    {
        for(int ii=0;ii<graph1[i].size();ii++)
        {
            for(int j=0;j<n;j++)
            {
                for(int jj=0;jj<graph1[j].size();jj++)
                {
                    int member[] = {i,graph1[i][ii],j,graph1[j][jj]};
                    if(member[0]>member[1] || member[2]>member[3]) continue;
                    if(member[0]==member[2] || member[0]==member[3]) continue;
                    if(member[1]==member[2] || member[1]==member[3]) continue;

                    int friends = 0;
                    for(int i=0;i<4;i++)
                        for(int j=i+1;j<4;j++)
                            friends += graph2[member[i]][member[j]];
                    if(friends == 5) count5++;
                    if(friends == 6) count6++;
                }
            }
        }
    }
    cout << count5/4 + count6/6;
}
```

$time\ complexity : O(\mathrm{e}^2)$