1. หากเราต้องการสร้าง Facebook เราเก็บความสัมพันธ์ของเพื่อนๆ ไว้ใน Adjacency Metrix หรือ Adjacency List ถ้าเราอยากให้คนเชื่อมต่อกันเยอะๆ จึงต้องทำการออกแบบระบบสำหรับแนะนำเพื่อนขึ้นมา วิธีการสร้างระบบแนะนำเพื่อนก็ไม่ยากแค่เพียงหาว่าคนสองคนที่มีเพื่อนเหมือนกันมากที่สุดแล้วก็ทำการแนะนำเพื่อนที่มีเพื่อนร่วมกัน จงเขียนโปรแกรมเพื่อทำการ suggest friends of friends พร้อมกับวิเคราะห์ complexity ของโปรแกรมที่ตัวเองออกแบบ **(10 คะแนน)**

```cpp
#include<iostream>
#include<queue>
#include<cstring>
using namespace std;


struct info
{
    int i;
    string name;
};

int main()
{
    int n;
    cout << "How many people : ";
    cin >> n;
    bool matrix[n + 1][n + 1];
    info student[n + 1];
    memset(matrix, false, sizeof(matrix));

    for(int i = 0; i < n; i++)
    {
        student[i].i = i;
        cout << i << ".name : ";
        cin >> student[i].name;
    }
    cout << endl;

    int m, dp_min[n], dp_max[n] = {}; //เก็บตำแหน่ง min max ที่มีค่าเป็น True ของทุกแถวเพื่อให้ลดเวลาในการเปรียบเทียบที่ไม่จำเป็น for min - max จาก 0 -
    for(int i = 0; i < n; i++)
    {
        dp_min[i] = n;
    }

    cout << "How many relations : ";
    cin >> m;

    if(m != 0) cout << "Use a number instead of a name.\n";

    for(int i = 0; i < m; i++)
    {
        cout << i + 1 << ": ";
        int a, b;
        cin >> a >> b;
        matrix[a][b] = true;
        matrix[b][a] = true;
        if(dp_min[a] > b){dp_min[a] = b;} //เก็บตำแหน่ง min max ที่มีค่าเป็น True ของทุกแถว
        if(dp_max[a] < b){dp_max[a] = b;}
        if(dp_min[b] > a){dp_min[b] = a;}
        if(dp_max[b] < a){dp_max[b] = a;}

    }

    // relations matrix

    cout << endl << "Relations matrix" << endl;
    for(int i = 0; i < n; i++)
    {
        cout << student[i].name << "\t";
        for(int j = 0; j < n; j++)
        {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
    // for(int i = 0; i < n; i++)   // check min[i] and max[i]
    // {
    //     cout << i << " min" << dp_min[i] << " max" << dp_max[i] << endl;
    // }
    queue<int> ans;
    queue<int> q;

    int nubans = 0, ans_a, ans_b;
    //finding
    for(int i = 0; i < n - 1; i++)// o(n)
    {
        for(int j = i + 1; j < n; j++)// o(n)
        {
```

```cpp
            int nub = 0, mn = max(dp_min[i], dp_min[j]), mx = min(dp_max[i], dp_max[j]);

            if(matrix[i][j]){continue;} //หากเป็นเพื่อนกันอยู่แล้วจะข้าม
            for(int k = mn; k < mx + 1; k++) // o(max - min) ~~ o(m)
            {
                if(matrix[i][k] && matrix[j][k] && k<=n)
                {
                    nub++;
                    q.push(k);
                }
            }

            if(nub > nubans) //answer   if new > old
            {
                ans_a = i;
                ans_b = j;
                while(!ans.empty())
                {
                    ans.pop();
                }
                while(!q.empty())
                {
                    ans.push(q.front());
                    q.pop();
                }
            }
            else //clear queue
            {
                while(!q.empty())
                {
                    q.pop();
                }
            }
        }
    }

    cout << "[" << ans_a << "]"<< student[ans_a].name  << " and " << "[" << ans_b << "]" << student[ans_b].name  << " have mutual frien
    while(!ans.empty())
    {
        cout << "[" << ans.front() << "]" << student[ans.front()].name << " ";
        cout << "[" << ans.front() << "]" << student[ans.front()].name << " ";
        ans.pop();
    }

    cout << endl << endl;
}

/*
5
nut
ohm
kitti
kla
pong
5
0 1
0 2
0 3
1 2
1 3
*/
```

CODE : https://ideone.com/zqVIXw

เก็บความสัมพันธ์ไว้ในรูปแบบ Matrix ดังบรรทัดที่ 46, 47

```
Relations matrix
nut     0 1 1 1 0
ohm     1 0 1 1 0
kitti   1 1 0 0 0
kla     1 1 0 0 0
pong    0 0 0 0 0
```

For loop สองชั้น เพื่อเปรียบเทียบระหว่างสองแถว(2 คน) O(N * (N+1/2)) ~ O(N^2) บรรทัดที่ 77 – 114

ใช้ Dynamic programming ในการเก็บตำแหน่งที่ต้องการเปรียบเทียบของแต่ละแถว เพื่อลดเวลาในการเปรียบเทียบ จาก 0 – n เป็น min[i] – max[i] ซึ่ง จะลดเวลาจาก O(n) เป็น O(max - min) ~ O(m) บรรทัดที่ 48 – 51 และ บรรทัดที่ 81 – 112

OUTPUT :

```
How many people : 5
0.name : nut
1.name : ohm
2.name : kitti
3.name : kla
4.name : pong
```

```
How many relations : 5
Use a number instead of a name.
1: 0 1
2: 0 2
3: 0 3
4: 1 2
5: 1 3
```

```
Relations matrix
nut     0 1 1 1 0
ohm     1 0 1 1 0
kitti   1 1 0 0 0
kla     1 1 0 0 0
pong    0 0 0 0 0
```

```
[2]kitti and [3]kla have mutual friends
: [0]nut [1]ohm
```

วิเคราะห์การใช้ Big O

O(MN^2) ~~ O(N^2)