原創/不分類題庫 - a445

這題會用到 Disjoint Set Union (DSU) 的概念，那什麼是 DSU 呢

「並查集 DSU 是一種樹形的資料結構，用於處理不交集的 合併 (union) 及查詢 (find) 問題。」

我個人認為用圖的方式理解會比較容易。以下用電腦連接網路來舉例

一開始都是自己連自己

| **編號** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| 連接 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

連接 3 和 5，就把 3 號改成 5 號

| **編號** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| 連接 | 1 | 2 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

接著就可以用遞迴的方法找到他們是否一組了

例如查 3 和 7 是否連接，那就可以看 3 的朋友是 5，而 5 的朋友是 5，那代表已經查完了。3 和 7 並沒有連接。

但是利用這個邏輯的話，再極端例子中，可能會花費許多時間在改動上，那有沒有更省時間的想法呢?

有，而且省力很多。

其實最重要的是找到源頭，因為假如兩台電腦都在同一個源頭裡的話，就代表兩台電腦有連結。而若是源頭要和其他節點連接的話就只要改動源頭的數值就好，不需要改變原有子節點的數值。

那有了這些之後就可以開始寫程式啦

```
//用遞迴找到老大是誰
int find_root(int x){
    if(r[x]==x) return x; //如果他自己就是他那群的源頭，那代表他是自己的朋友
    return find_root(r[x]);
}
```

```
//讓老大代表全部的小弟
void attach(int x, int y){
    int root_x=find_root(x); //找到 x 的源頭
    int root_y=find_root(y); //找到 y 的源頭
    if(root_x!=root_y){
        r[root_x]=root_y; //讓 x 的源頭歸順於 y 的源頭 (若要反過來也可以)
    }
}
```

```
int main(){
    //I/O 加速
    ios::sync_with_stdio(false);
    cin.tie(0);
    int N, M, Q, A, B;
    N=M=Q=A=B=0;
    cin>>N>>M>>Q;
    for(int i=1;i<=N;i++) r[i]=i;
    for(int i=1;i<=M;i++){
        cin>>A>>B;
        attach(A, B);
    }
    while(Q--){
        cin>>A>>B;
        //如果 A 和 B 老大一樣，代表 A 和 B 是朋友
        if(find_root(A)==find_root(B)){
            cout<<":)"<<'\n';
        }else{
            cout<<":("<<'\n';
        }
    }
}
```