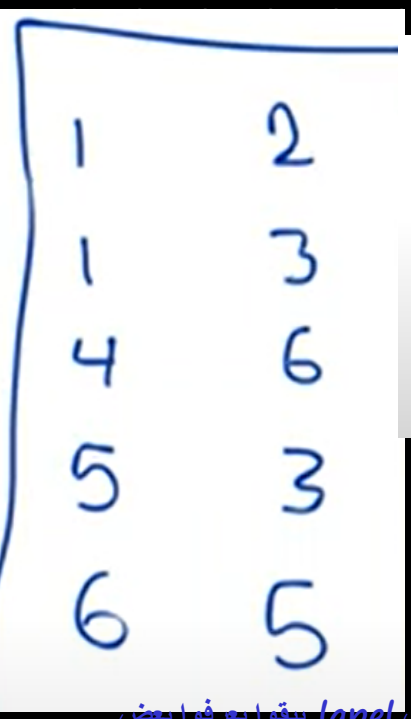
DSU, MST, and Kruskal Algorithm

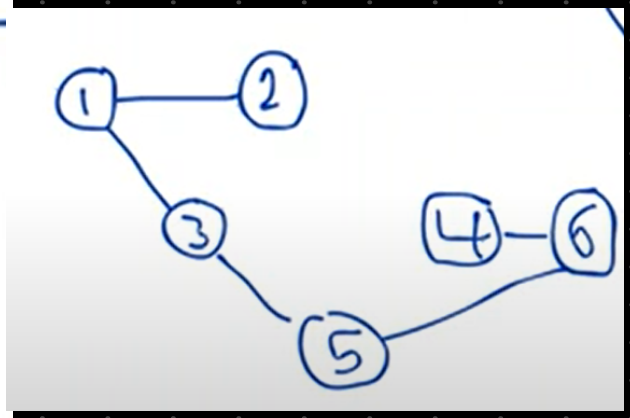
1. Dis Joint Set Unit (DSU):-

نوع داتا استراكتشر

هديك جراف لشوية اشخاص وهقولك فلان يعرف فلان وفلان يعرف فلان
وهسألك هل فلان وفلان يعرفوا بعض

يعني الجراف دا سألتك هل 1 يعرف 6 ؟؟

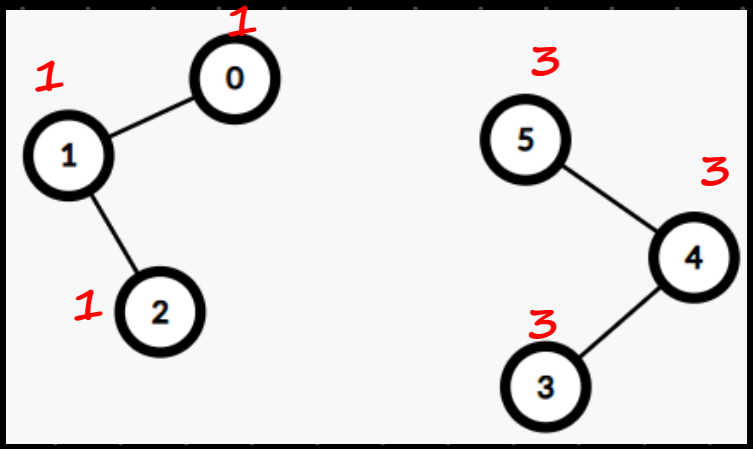




هنعمل DFS من عند ال 1 مثلا و
لما يخلصا اشوف علي هل ال 6
visited ولا لأ

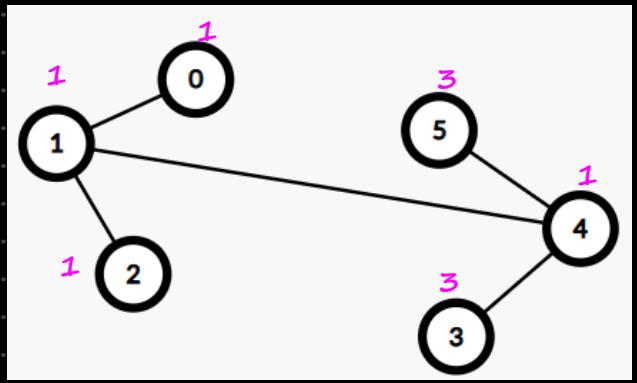
بس لو جيت تحسب ال time هنا هتلاقي انه لو كديك quires فانت كدا كل
مره هتعمل dfs يعني ال time هيبقي O((n+m)*q)

طب بص اي رأيك اخلي الحل بطريقه تانيه واني خلي كل جروب باسم او ب lapel معين ولو اتنين ليهم نفس ال lapel يبقوا يعرفوا بعض
اولا في اضافه ان ال quereمتغيره يعني دلوقت بسألك هل دا ودا صحاب بعدها هقولك طب خلي فلان وعلان صحاب وهكذا فاول حاجه انا هخلي
لكل جروب ليدر . فقبل ما اخد اي qurere كل نود الليدر بتاعها هي بردو اول ما يقولي خلي فلان وعلان صحاب هاخد واحد منهم واخليه هو الليدر
بتاع الجروب



هتظهر عندي مشكله صغيره لو عندي جروبين منفصلين
عن بعض كل واحد ليه ليدر وجيت اقول خلي واحد من
هنا وواحد من هنا صحاب كدا بقيت الجروب مش عارفين
بقيت الجروب التاني

يعني في المثال دا بعد ما اخلي 2 و4 هتبقي كدا



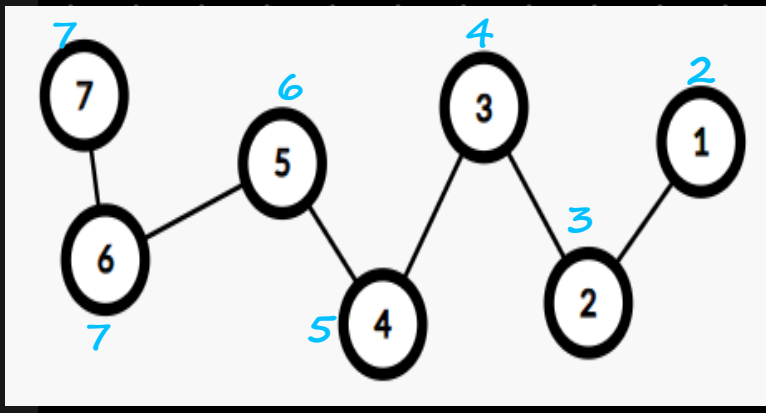
دلوقت لو جيت اسأل هل 5 و 2 بعروفة بعض هيقولي لأ فالحل ان لما يجي يسألني فلان
وفلان صحاب اقوله هاتالليدو بتاع فلان ولو له ليدر هاته يعني هات اكبر راس هنا واكبر
راس هنا وشوف هما نفس الشخص ولا لأ

CODE

```cpp
const int N = 1e4+5,M = 2e5 + 5;

int n, q, queryType, u, v;

int leader[N];
void init() { // called first thing in the main (once)
    for (int i=0; i<=n; ++i)
        leader[i] = i;
}

int get_leader(int u) {
    if( u == leader[u])
        return u;
    return get_leader(leader[u]);
}

bool are_friends(int u,int v) {
    return get_leader(u) == get_leader(v);
}

void make_friends(int u,int v) {
    u = get_leader(u);
    v = get_leader(v);

    leader[u] = v;
}
```

بس لو جيت تشوف هنا احنا مقللناش التايم اوي بردو كدا كل كويري هتاخد $O(n)$

في بقا optimaization خطير



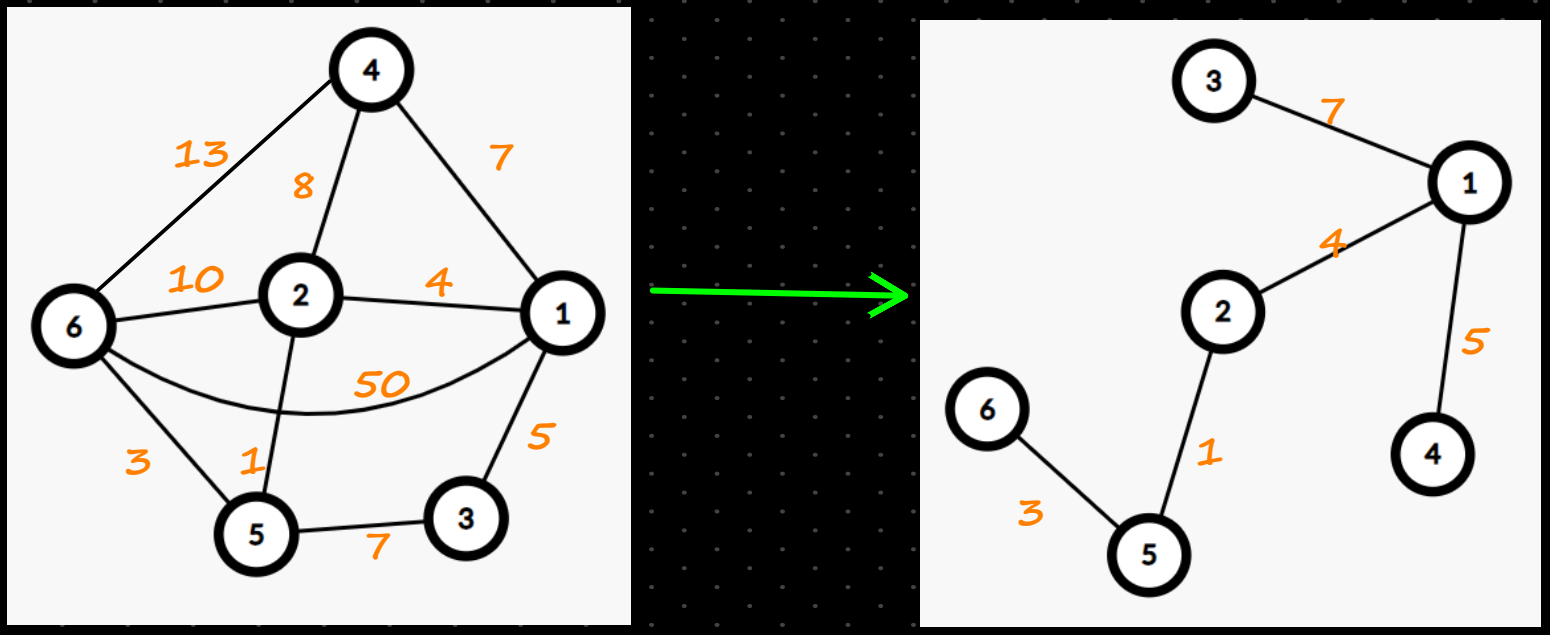
هنا لو سألتك هل ال 1 و 2 صحاب ولا لأ في ال1 هيروح وبعدها بردو في ال2 طب مش وانا في الواحد مش كل اللي في الطريق دول نفس الليدر $O(N)$ بتاع الواحد طب مونا راجع بال recursion اقولهم الليدر الاصلي بتاعهم الفكره دي اسمها path compresion

```cpp
int get_leader(int u) {
    if( u == leader[u])
        return u;
    return leader[u] = get_leader(leader[u]);
}
```

2. Minmum Spanning Tree (MST) :- نوع بروبليم

عايزك تلاقيلي شوية نقط في weighted graph يكونوا tree و مجموع ال edge weight يكون اقل ما يمكن

عندك شوية مدن في بينهم شوية طرق ومعاك تكلفة تجديد الطرق سنويا كل طريق له التكلفه الخاصه بتاعه والسنادي الدوله مفلسه وعايزين يوقفوا بعض الطرق بشرط ان كل البلاد تقدر توصل لبعضها وتكون التكلفه للتجديد اقل مايمكن



عايز اجيب اقل تكلفه

3. الالجورزم اللي بيحل النوع دا من المسائل اسمه Kruskal

اول حاجه هاخد ال edges وال weight وارتبهم حسب ال weight اللي هو التكلفه وبعدين هاخد الاقل تكلفه واشوفه هل الedge بتاعه هتعمل خلل في شرط ال tree ولا لأ وهمشي عليهم اللي هلاقيه هيعمل cicle يعني هيبوظ ال tree مش هرسمه وهخش علي اللي بعده عشان اعرف هل الedge دي هتضر بمفهوم ال tree ولا لا اشوف هل ال 2nodes دول يعرفوا بعض ولا لا عن طريق DSU

CODE

```cpp
#define C first
#define U second.first
#define V second.second
typedef pair<int, pair<int,int>> edge;
edge edgelist[M];
int main(){     fast
    cin >> n>> m;

    for (int i =0 ; i< m; ++i) {
        cin>> u>>v>>c;
        edgelist[i] = {c, b: { x: [&] u, y: [&] v}};
    }

    //kruskal
    sort( first: edgelist, last: edgelist + m);
    init();
    int cost = 0;
    for (int i=0 ; i < m;++i) {
        if(! are_friends( u: edgelist[i].U, v: edgelist[i].V)) {
            make_friends( u: edgelist[i].U, v: edgelist[i].V);
            cost += edgelist[i].C;
        }
    }

    cout<<cost;
}
```