结构体排序

```
#include<iostream>
#include<algorithm>
using namespace std;
struct student{
    int a,b;
}

//该方法返回false代表要交换，true代表不需要交换，科根据实际灵活变通
bool map_a(student x,student y){
    return x.a<y.a; //根据属性 a 递增排序，递减是 x.a>y.a
}

bool map_b(student x,student y){
    return x.b<y.b; //根据属性 b 递增排序，递减是 x.b>y.b
}

bool map_a_b(student x,student y){
    if(x.a==y.a){
        return x.b>y.b; //当属性 a 相等时，根据 b 递增排序
    }
    return x.a<y.a; //根据属性 a 递增排序
}

int main()
{
    int n;
    cin>>n;
    student s[n];
    for(int i=0;i<n;i++){
        s[n].a=i;
        s[n].b=i;
    }
    sort(s,s+n,map_a);   //根据 a 递增
    sort(s,s+n,map_b);   //根据 b 递增
    sort(s,s+n,map_a_b);   //根据 a 递增, a 相等时根据b递增
    return 0;
}
```

并查集

```
#include<iostream>
using namespace std;
//数组范围要符合题目要求
int p[100000];

//初始化，刚开始每个点都是一个集合
void init(){
    for(int i=1;i<n;i++){
        p[i]=i;
    }
}

//查找 x 的根节点
int find(int x){
    if(p[x] != x){
        p[x]=find(p[x]);    //查找父节点+路径压缩
    }
    return p[x];
}

int main()
{
    int n;
    cin>>n;
    init(); //初始化

    int a,b;
    cin>>a>>b;
    p[find(a)]=find(b); //合并 a,b 集合

    if(find(a) == find(b)){ //判断 a,b 是否属于同一集合
        cout<<"YES"<<endl;
    }else{
        cout<<"NO"<<endl;
    }

    int count=0;
    for(int i=0;i<n;i++){   //统计根节点数，即统计集合数
        if(p[i]==i){
            count++;
        }
    }
    cout<<count;
    return 0;
}
```