

优先队列

P0J 3614: Sunscreen 题解链接 <https://www.jianshu.com/p/dfb78e06c19d> 代码如下

```
/*

*/

#define method_1
#ifdef method_1
/*

*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
using namespace std;
typedef long long ll;
const int maxn=2500+5;
const int INF=0x3f3f3f3f;
int c,l;
struct node{
    int x,y;
}p[maxn],q[maxn];
int ans=0;
bool operator<(const node &a,const node &b){
    return a.x>b.x;
}
int main() {
    ios::sync_with_stdio(false);
    //freopen("Sunscreen.in","r",stdin);
    cin>>c>>l;
    for(int i=1;i<=c;i++){
        cin>>p[i].x>>p[i].y;
    }
    for(int i=1;i<=l;i++){
        cin>>q[i].x>>q[i].y;
    }
    sort(p+1,p+c+1);
    // for(int i=1;i<=c;i++){
    //     cout<<p[i].x<<endl;
    // }
```

```

// }
    for(int i=1;i<=c;i++){
        int temp=-1,id=-1;
        for(int j=1;j<=l;j++){
            if(q[j].y>0&&(p[i].x<=q[j].x)&&(p[i].y>=q[j].x)){
                if(q[j].x>temp){
                    temp=q[j].x;
                    id=j;
                }
            }
        }
        if(temp!=-1){
            ans++;
            q[id].y--;
        }
    }
    cout<<ans;
    return 0;
}
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

#endif

```

POJ 2010: Moo University - Financial Aid

将所有奶牛按照score升序排序后，从大向小一次尝试作为中位数是否可行即可。

为了提高对于每头奶牛判断的效率，用sum1[i]表示i左侧c/2头牛的最小aid和，可用一个大小固定为n/2的大根堆来求，每次与堆顶元素比较。同理可以对称的定义sum2[i]。

代码如下

```

/*

*/
#define method_1
#ifdef method_1
/*

```

将所有奶牛按照score升序排序后，从大向小一次尝试作为中位数是否可行即可。

为了提高对于每头奶牛判断的效率，用`sum1[i]`表示`i`左侧`c/2`头牛的最小`aid`和，可用一个大小固定为`n/2`的大根堆来求，每次与堆顶元素比较。

同理可以对称的定义`sum2[i]`。

```
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<#x<<" "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=100000+5;
const int INF=0x3f3f3f3f;
int n,c,f,sum1[maxn],sum2[maxn];//sum1[i]表示i左侧c/2头牛的最小aid和
用一个大小固定为n/2的大根堆来求 每次与堆顶元素比较
struct node{
    int s,aid;
    bool operator<(const node& h)const{return s<h.s;}
}cow[maxn];
priority_queue<int>q;
void pre1(){
    while(q.size()) q.pop();
    int sum=0,nowf; //堆中所有元素和
    for(int i=1;i<=c;i++){
        sum1[i]=sum;
        if(q.size()) nowf=q.top();
        if(q.size()<n/2){
            q.push(cow[i].aid);
            sum+=cow[i].aid;
        }
        else{
            if(cow[i].aid<nowf){
                q.pop();
                q.push(cow[i].aid);
                sum=sum-nowf+cow[i].aid;
            }
        }
    }
}
```

```

        }
    }
    //    D(sum1[i]);
}
}
void pre2(){
    while(q.size()) q.pop();
    int sum=0,nowf; //堆中所有元素和
    for(int i=c;i>=1;i--){
        sum2[i]=sum;
        if(q.size()) nowf=q.top();
        if(q.size()<n/2){
            q.push(cow[i].aid);
            sum+=cow[i].aid;
        }
        else{
            if(cow[i].aid<nowf){
                q.pop();
                q.push(cow[i].aid);
                sum=sum-nowf+cow[i].aid;
            }
        }
    }
    //    D(sum2[i]);
}
}
int main() {
    // ios::sync_with_stdio(false);
    //freopen("Moo University - Financial Aid.in","r",stdin);
    scanf("%d%d%d",&n,&c,&f);
    for(int i=1;i<=c;i++) scanf("%d",&cow[i].s,&cow[i].aid);
    sort(cow+1,cow+c+1);
    pre1();
    pre2();
    bool flag=false;
    for(int i=c-n/2;i>=n/2+1;i--){ //从大到小依次枚举当前值能否作为中位数
        if(cow[i].aid+sum1[i]+sum2[i]<=f){
            flag=true;
            cout<<cow[i].s;
            break;
        }
    }
    if(flag==false) cout<<-1;
    return 0;
}
#endif
#ifdef method_2
/*
*/
*/

```

```

#endif
#ifdef method_3
/*

*/

#endif

```

并查集

[POJ 2236: Wireless Network](#) 这题的复杂度是 $O(N^2 + M)$ ，因为修理操作最多执行 N 次。每次检测 $N - 1$ 个计算机，剩下的只能是判断操作，每一次判断可以看作 $O(1)$ 。代码如下

```

/*

*/
#define method_1
#ifdef method_1
/*
这题的复杂度是 $O(N^2 + M)$ ，因为修理操作最多执行 $N$ 次。
每次检测 $N - 1$ 个计算机，剩下的只能是判断操作，每一次判断可以看作 $O(1)$ 。
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"  "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=1000+5;
const int INF=0x3f3f3f3f;
const string s1="SUCCESS";
const string s2="FAIL";
int f[maxn],n,d,x[maxn],y[maxn],vis[maxn]={0};

```

```

void init(){
    for(int i=1;i<=n;i++) f[i]=i;
}
int find(int x){
    return f[x]==x?x:f[x]=find(f[x]);
}
bool check(int i,int j){ //判断i和j两个电脑能否直接连接
    int dis=(x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]);
    return dis<=d*d;
}
int main() {
    // ios::sync_with_stdio(false);
    //freopen("Wireless Network.in","r",stdin);
    scanf("%d%d",&n,&d);
    init();
    for(int i=1;i<=n;i++) scanf("%d%d",&x[i],&y[i]);
    getchar();
    int p,q;
    char op;
    while(~scanf("%c",&op)){
//        cout<<op<<endl;
        if(op=='O'){
            scanf("%d",&p);
            if(vis[p]) continue;
            vis[p]=1;
            int fp=find(p);
            for(int i=1;i<=n;i++){
                if(i==p) continue;
                if(vis[i]&&check(p,i)){
                    int fi=find(i);
                    f[fi]=fp;
                }
            }
        }
        else{
            scanf("%d%d",&p,&q);
            if(find(p)==find(q)) printf("%s\n",s1.c_str());
            else printf("%s\n",s2.c_str());
        }
        getchar();
    }
    return 0;
}
#endif
#ifdef method_2
/*
*/
*/

```

```

#endif
#ifdef method_3
/*

*/

#endif

```

[POJ 1703: Find them, Catch them](#) 拓展域并查集模板题。 375msAC。 用cin会超时。

代码如下

```

/*

*/
#define method_1
#ifdef method_1
/*
拓展域并查集模板题。
375msAC。
用cin会超时。
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<" "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=2e5+5;
const int INF=0x3f3f3f3f;
const string s1="Not sure yet.";
const string s2="In different gangs.";
const string s3="In the same gang.";
int n,m,T,f[maxn];
void init() {

```

```

        for(int i=1; i<=2*n; i++) f[i]=i;
    }
    int find(int x) {
        return f[x]==x?x:f[x]=find(f[x]);
    }
    int main() {
        // ios::sync_with_stdio(false);
        //freopen("Find them, Catch them.in","r",stdin);
        // cin>>T;
        scanf("%d",&T);
        while(T-->0) {
            scanf("%d%d",&n,&m);
            // cin>>n>>m;
            init();
            char op;
            int x,y;
            while(m-->0) {
                getchar(); //读入换行符
                // cin>>op>>x>>y;
                scanf("%c%d%d",&op,&x,&y);
                // cout<<x<<" "<<y<<endl;
                int x1=find(x),y1=find(y),x2=find(x+n),y2=find(y+n);
                if(op=='A') {
                    /*
                    if(x1==y1) cout<<s3<<endl;
                    else if(x1==y2) cout<<s2<<endl;
                    else cout<<s1<<endl;
                    */
                    if(x1==y1) printf("%s\n",s3.c_str());
                    else if(x1==y2) printf("%s\n",s2.c_str());
                    else printf("%s\n",s1.c_str()); //printf不能直接输出string
                } else {
                    f[x1]=y2,f[x2]=y1;
                }
            }
        }
        return 0;
    }
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

```

#endif