[解题报告] ZOJ3261

[Source]

http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3261

[Description]

星球之间有相互联系的双向道路。每个星球有个防御值。星球大战时，如果一个星球遭到攻击，那么它会向它所能联系到的星球中防御值最大的星球求助，如果防御值最大的星球有多个，则选择其中编号最小的那个。星球大战过程中，会有道路被损毁。
n(n<=10000)个星球,编号 0~n-1，星球大战前有 m（m<=20000）条边。给出 q(q<=50000)次查询，查询分两种：1)destroy a b 表示 a 与 b 星球间的道路损毁;2)query a 表示查询星球 a 当前的求助对象。对于每个 query 查询，输出结果。

[Solution]

并查集。运用逆向思维，最初开始合并时，只将最终经过所有 destroy 操作后剩下的边相关联的集合进行合并。然后对查询从后前进行道路“恢复”的操作，即合并当时 destroy 的边所关联的集合。对于每个 query 查询，输出星球当前的祖先即可。注意集合合并时，应按照题目要求进行合并，即防御值小的星球将防御值大的星球作为祖先，防御值相同时编号小的为祖先。

[Code]

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 11000
#define M 65000
using namespace std;

struct Query
{
    int flag;
    int u,v;
};
```

```
Query que[M];
int father[N],p[N];
int first[N],next[M],end[M],tag[M];
int cnt_edge;
int ans[M];
char str[20];

void addEdge(int u,int v)
{
    end[cnt_edge]=v;
    next[cnt_edge]=first[u];
    first[u]=cnt_edge++;
}

void reset(int n)
{
    int i;

    for (i=0;i<=n;i++)
        father[i]=i;
}

int find(int a)
{
    if (father[a]==a)
        return a;
    else
        return father[a]=find(father[a]);
}

void merge(int a,int b)
{
    int fa=find(a);
    int fb=find(b);

    if (fa!=fb)
    {
        if (p[fa]<p[fb]||p[fa]==p[fb]&&fa>fb)
            father[fa]=fb;
        else
            father[fb]=fa;
    }
}
```

```
int main()
{
    int n,m,q,i,j,u,v,f,cnt,ys=0;

    while (scanf("%d",&n)!=EOF)
    {
        memset(first,0,sizeof(first));
        memset(tag,0,sizeof(tag));
        reset(n);
        cnt_edge=1;
        cnt=0;

        for (i=1;i<=n;i++)
            scanf("%d",&p[i]);

        scanf("%d",&m);
        for (i=0;i<m;i++)
        {
            scanf("%d%d",&u,&v);
            u++;
            v++;
            if (u>v)
                swap(u,v);

            addEdge(u,v);
        }

        scanf("%d",&q);
        for (i=0;i<q;i++)
        {
            scanf("%s",str);
            if (str[0]=='q')
            {
                scanf("%d",&u);
                u++;
                que[i].flag=0;
                que[i].u=u;
            }
            else
            {
                scanf("%d%d",&u,&v);
                u++;
                v++;
```

```
            if (u>v)
                swap(u,v);

            que[i].flag=1;
            que[i].u=u;
            que[i].v=v;

            for (j=first[u];j;j=next[j])
                if (end[j]==v)
                {
                    tag[j]=1;
                    break;
                }
        }
    }

    for (u=1;u<=n;u++)
        for (i=first[u];i;i=next[i])
        {
            v=end[i];

            if (tag[i]==0)
                merge(u,v);
        }


    for (i=q-1;i>=0;i--)
        if (que[i].flag)
            merge(que[i].u,que[i].v);
        else
        {
            f=find(que[i].u);
            if (p[f]>p[que[i].u])
                ans[cnt]=f-1;
            else
                ans[cnt]=-1;
            cnt++;
        }

    if (ys==0)
        ys=1;
    else
        printf("\n");
    for (i=cnt-1;i>=0;i--)
```

```
            printf("%d\n",ans[i]);
    }

    return 0;
}
```