并查集

P3367 【模板】并查集

```
#include<bits/stdc++.h>
 2
    #define FOR(i,a,b) for(int i=(a); i \le (b); ++i)
 3
    using namespace std;
 4
    int f[10007];
 5
 6
 7
    int find(int x){
8
        if(f[x]==x) return x;
9
        return f[x]=find(f[x]);
10
    }
11
12
    int main(){
13
        ios_base::sync_with_stdio(0);
14
        cin.tie(0);
15
        int n,m;
16
        cin>>n>>m;
17
        FOR(i,1,n)
             f[i]=i;
18
19
20
        int z,x,y;
        FOR(i,1,m){
21
22
             cin>>z>>x>>y;
23
            if(z==1) f[find(x)]=find(y);//合并
             if(z==2){//查找
24
25
                 if(find(x)==find(y)) cout<<"Y\n";</pre>
                 else cout<<"N\n";</pre>
26
27
             }
28
29
        return 0;
30
    }
```

ST表

P3865 【模板】ST表

参考1

参考2

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
using namespace std;
#include<bits/stdc++.h>
4
```

```
const int MAXJ=log2((int)(1e5))+1;//17
    const int MAXN=1e6+10;
 6
 7
    inline int IntRead(){//快速读入比关闭同步的cin快得多
8
9
        char ch=getchar();
10
        int s=0, w=1;
11
        while(ch<'0' | ch>'9'){
            if(ch=='-') w=-1;
12
            ch=getchar();
13
14
        }
        while(ch>='0' && ch<='9'){
15
16
            s=s*10+ch-'0',
17
            ch=getchar();
18
        }
19
        return s*w;
20
    }
21
22
    int Max[MAXN][17];
23
24
    int Query(int 1,int r){
25
        int k=log2(r-l+1);
        return max(Max[1][k],Max[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
26
27
    }
2.8
29
    int main(){
30
        int n=IntRead(),m=IntRead();
31
        FOR(i,1,n)
32
            Max[i][0]=IntRead();
33
        FOR(j,1,MAXJ)
34
            for(int i=1;i+(1<<j)-1<=n;i++)//注意这里要控制边界
35
                Max[i][j]=max(Max[i][j-1],Max[i+(1<<(j-1))][j-1]);
36
        FOR(i,1,m){
37
            int l=IntRead(),r=IntRead();
38
            printf("%d\n",Query(1,r));
39
40
        return 0;
41
    }
```

树状数组

单点修改、区间查询

P3374 【模板】树状数组 1

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
#define elif else if
using namespace std;</pre>
```

```
5
 6
    int a[500100],c[500100],n,m;
 7
8
    int lowbit(int x){
9
        return x&-x;
10
11
    void add(int x,int v){
12
        while(x \le n)
13
            c[x] += v, x += lowbit(x);
14
15
    }
16
17
    int sum(int x){
18
        int ans=0;
19
        while(x \ge 1)
20
            ans+=c[x],x==lowbit(x);
        return ans;
21
22
23
24
    int SUM(int 1,int r){
25
        return sum(r)-sum(l-1);
26
27
28
    int main(){
29
        ios_base::sync_with_stdio(0);
30
        cin.tie(0);
31
        int i,cmd,x,y;
32
        cin>>n>>m;
33
        FOR(i,1,n)
34
            cin>>a[i],add(i,a[i]);
35
        while(m--){
36
            cin>>cmd>>x>>y;
37
            if(cmd==1) add(x,y);
38
            elif(cmd==2) cout << SUM(x,y) << '\n';
39
40
        return 0;
41
    }
```

区间修改、单点查询

P3368 【模板】树状数组 2

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
#define elif else if
using namespace std;

int a[500100],c[500100],n,m;</pre>
```

```
8
    int lowbit(int x){
 9
         return x&-x;
10
    }
11
    void add(int x,int v){
12
         while(x \le n)
13
14
             c[x] += v, x += lowbit(x);
15
    }
16
17
    int query(int x){
18
         int ans=0;
19
         while(x \ge 1)
20
             ans+=c[x],x-=lowbit(x);
21
         return ans;
22
    }
23
24
    int main(){
25
         ios_base::sync_with_stdio(0);
26
         cin.tie(0);
27
         int cmd, x, y, k;
28
         cin>>n>>m;
29
         int past=0,now;
30
         FOR(i,1,n){
31
             cin>>now;
32
             add(i,now-past);
33
             past=now;
34
35
         while(m--){
             cin>>cmd;
36
37
             if(cmd==1){
38
                 cin>>x>>y>>k;
39
                 add(x,k);
40
                 add(y+1,-k);
41
             }
42
             elif(cmd==2){
43
                 cin>>x;
44
                 cout<<query(x)<<'\n';</pre>
45
             }
46
         }
47
         return 0;
48
    }
```

线段树

P3372 【模板】线段树 1

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)</pre>
```

```
using namespace std;
 3
 4
    #define maxn 100017//元素总个数
 5
    #define elif else if
 6
 7
    #define LL long long
 8
9
    struct SegmentTree{
10
        #define ls (rt<<1)//lson,左子树
        #define rs (rt<<1|1)</pre>
11
        LL Sum[maxn<<2],Add[maxn<<2];//Sum求和,Add为懒惰标记
12
        LL A[maxn],n;//存原数组数据下标[1,n]
13
14
15
       void PushUp(int rt){
            //更新节点信息 , 这里是求和
16
            Sum[rt]=Sum[ls]+Sum[rs];
17
        }
18
19
20
        void Build(int l,int r,int rt){
            //1,r:当前节点区间, rt:当前节点编号
21
22
            if(l==r) {//若到达叶节点
23
                Sum[rt]=A[1];//储存数组值
2.4
               return;
25
            }
            int m=(1+r)>>1;
2.6
            //左右递归
27
28
            Build(1,m,ls);
29
            Build(m+1,r,rs);
30
            PushUp(rt);
31
        }
32
33
        void Update(int L,LL C,int l,int r,int rt){
            //1,r:当前节点区间,rt:当前节点编号
34
35
            if(l==r){//到叶节点,修改
36
               Sum[rt]+=C;
37
               return;
38
            }
39
            int m=(1+r)>>1;
            //根据条件判断往左子树调用还是往右
40
            if(L<=m) Update(L,C,l,m,ls);</pre>
41
            else Update(L,C,m+1,r,rs);
42
            PushUp(rt);//子节点更新了,所以本节点也需要更新信息
43
44
        }
45
        void PushDown(int rt,int ln,int rn){
46
            //ln,rn为左子树,右子树的数字数量。
47
            if(Add[rt]){
48
                //下推标记
49
50
               Add[ls]+=Add[rt];
51
               Add[rs]+=Add[rt];
```

```
52
                //修改子节点的Sum使之与对应的Add相对应
53
                Sum[ls]+=Add[rt]*ln;
54
                Sum[rs]+=Add[rt]*rn;
                //清除本节点标记
55
56
               Add[rt]=0;
57
            }
58
        }
59
        void Update(int L,int R,LL C,int l,int r,int rt){
60
            //L,R:操作区间, 1,r:当前节点区间, rt:当前节点编号
61
            if(L<=l and r<=R){//如果本区间完全在操作区间[L,R]以内
62
                Sum[rt]+=C*(r-l+1);//更新数字和,向上保持正确
63
               Add[rt]+=C;//增加Add标记,表示本区间的Sum正确,子区间的Sum仍需要根据Add的值来调
64
    整
65
                return ;
            }
66
            int mid=(l+r)>>1;
67
            PushDown(rt,mid-l+1,r-mid);//下推标记
68
            //这里判断左右子树跟[L,R]有无交集,有交集才递归
69
70
            if(L<=mid) Update(L,R,C,l,mid,ls);</pre>
71
            if(R>mid) Update(L,R,C,mid+1,r,rs);
            PushUp(rt);//更新本节点信息
72
73
        }
74
75
        LL Query(int L, int R, int l, int r, int rt){
            //L,R:操作区间,1,r:当前节点区间, rt:当前节点编号
76
            if(L \le 1 \text{ and } r \le R)
77
                //在区间内,直接返回
78
79
               return Sum[rt];
80
            }
81
            int mid=(l+r)>>1;
            //下推标记,否则Sum可能不正确
82
83
            PushDown(rt,mid-l+1,r-mid);
84
            //累计答案
85
            LL ANS=0;
86
            if(L<=mid) ANS+=Query(L,R,l,mid,ls);</pre>
87
            if(R>mid) ANS+=Query(L,R,mid+1,r,rs);
88
            return ANS;
89
90
        }
91
    };
92
    int main(){
93
        ios_base::sync_with_stdio(0);
94
95
        cin.tie(0);
        SegmentTree st;
96
97
        int n,m;
98
        cin>>n>>m;
99
        FOR(i,1,n)
```

```
100
             cin>>st.A[i];
         st.Build(1,n,1);
101
102
         int o,1,r,k;
103
         FOR(i,1,m){
104
             cin>>o;
105
             if(o==1){
106
                 cin>>1>>r>>k;//[1,r]+=k
107
                 st.Update(1,r,k,1,n,1);
108
109
             elif(o==2){
110
                 cin>>l>>r;
111
                 cout << st.Query(l,r,l,n,l) << endl;//sum[l->r]
112
             }
113
         }
114
         return 0;
115 }
```