并查集

P3367 【模板】并查集

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

int f[10007];

int find(int x){
    if(f[x]==x) return x;
    return f[x]=find(f[x]);
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n,m;
    cin>>n>>m;
    FOR(i,1,n)
        f[i]=i;

    int z,x,y;
    FOR(i,1,m){
        cin>>z>>x>>y;
        if(z==1) f[find(x)]=find(y);//合并
        if(z==2){//查找
            if(find(x)==find(y)) cout<<"Y\n";
            else cout<<"N\n";
        }
    }
    return 0;
}
```

ST表

P3865 【模板】ST 表

参考1

参考2

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

```

```
const int MAXJ=log2((int)(1e5))+1;//17
const int MAXN=1e6+10;

inline int IntRead(){//快速读入比关闭同步的cin快得多
    char ch=getchar();
    int s=0,w=1;
    while(ch<'0' || ch>'9'){
        if(ch=='-') w=-1;
        ch=getchar();
    }
    while(ch>='0' && ch<='9'){
        s=s*10+ch-'0',
        ch=getchar();
    }
    return s*w;
}

int Max[MAXN][17];

int Query(int l,int r){
    int k=log2(r-l+1);
    return max(Max[l][k],Max[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
}

int main(){
    int n=IntRead(),m=IntRead();
    FOR(i,1,n)
        Max[i][0]=IntRead();
    FOR(j,1,MAXJ)
        for(int i=1;i+(1<<j)-1<=n;i++)//注意这里要控制边界
            Max[i][j]=max(Max[i][j-1],Max[i+(1<<(j-1))][j-1]);
    FOR(i,1,m){
        int l=IntRead(),r=IntRead();
        printf("%d\n",Query(l,r));
    }
    return 0;
}
```

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;
```

```

int a[500100],c[500100],n,m;

int lowbit(int x){
    return x&-x;
}

void add(int x,int v){
    while(x<=n)
        c[x]+=v,x+=lowbit(x);
}

int sum(int x){
    int ans=0;
    while(x>=1)
        ans+=c[x],x-=lowbit(x);
    return ans;
}

int SUM(int l,int r){
    return sum(r)-sum(l-1);
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int i,cmd,x,y;
    cin>>n>>m;
    FOR(i,1,n)
        cin>>a[i],add(i,a[i]);
    while(m--){
        cin>>cmd>>x>>y;
        if(cmd==1) add(x,y);
        elif(cmd==2) cout<<SUM(x,y)<<'\n';
    }
    return 0;
}
```

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;

```

```cpp
int lowbit(int x){
    return x&-x;
}

void add(int x,int v){
    while(x<=n)
        c[x]+=v,x+=lowbit(x);
}

int query(int x){
    int ans=0;
    while(x>=1)
        ans+=c[x],x-=lowbit(x);
    return ans;
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int cmd,x,y,k;
    cin>>n>>m;
    int past=0,now;
    FOR(i,1,n){
        cin>>now;
        add(i,now-past);
        past=now;
    }
    while(m--){
        cin>>cmd;
        if(cmd==1){
            cin>>x>>y>>k;
            add(x,k);
            add(y+1,-k);
        }
        elif(cmd==2){
            cin>>x;
            cout<<query(x)<<'\n';
        }
    }
    return 0;
}
```

线段树

P3372 【模板】线段树 1

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

```cpp
using namespace std;

#define maxn 100017//元素总个数
#define elif else if
#define LL long long

struct SegmentTree{
    #define ls (rt<<1)//lson,左子树
    #define rs (rt<<1|1)
    LL Sum[maxn<<2],Add[maxn<<2];//Sum求和，Add为懒惰标记
    LL A[maxn],n;//存原数组数据下标[1,n]

    void PushUp(int rt){
        //更新节点信息 ，这里是求和
        Sum[rt]=Sum[ls]+Sum[rs];
    }

    void Build(int l,int r,int rt){
        //l,r:当前节点区间，rt:当前节点编号
        if(l==r) {//若到达叶节点
            Sum[rt]=A[l];//储存数组值
            return;
        }
        int m=(l+r)>>1;
        //左右递归
        Build(l,m,ls);
        Build(m+1,r,rs);
        PushUp(rt);
    }

    void Update(int L,LL C,int l,int r,int rt){
        //l,r:当前节点区间,rt:当前节点编号
        if(l==r){//到叶节点，修改
            Sum[rt]+=C;
            return;
        }
        int m=(l+r)>>1;
        //根据条件判断往左子树调用还是往右
        if(L<=m) Update(L,C,l,m,ls);
        else Update(L,C,m+1,r,rs);
        PushUp(rt);//子节点更新了，所以本节点也需要更新信息
    }

    void PushDown(int rt,int ln,int rn){
        //ln,rn为左子树，右子树的数字数量。
        if(Add[rt]){
            //下推标记
            Add[ls]+=Add[rt];
            Add[rs]+=Add[rt];
```

```
            //修改子节点的Sum使之与对应的Add相对应
            Sum[ls]+=Add[rt]*ln;
            Sum[rs]+=Add[rt]*rn;
            //清除本节点标记
            Add[rt]=0;
        }
    }

    void Update(int L,int R,LL C,int l,int r,int rt){
        //L,R:操作区间，l,r:当前节点区间，rt:当前节点编号
        if(L<=l and r<=R){//如果本区间完全在操作区间[L,R]以内
            Sum[rt]+=C*(r-l+1);//更新数字和，向上保持正确
            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的值来调整
            return ;
        }
        int mid=(l+r)>>1;
        PushDown(rt,mid-l+1,r-mid);//下推标记
        //这里判断左右子树跟[L,R]有无交集，有交集才递归
        if(L<=mid) Update(L,R,C,l,mid,ls);
        if(R>mid) Update(L,R,C,mid+1,r,rs);
        PushUp(rt);//更新本节点信息
    }

    LL Query(int L,int R,int l,int r,int rt){
        //L,R:操作区间,l,r:当前节点区间，rt:当前节点编号
        if(L<=l and r<=R){
            //在区间内，直接返回
            return Sum[rt];
        }
        int mid=(l+r)>>1;
        //下推标记，否则Sum可能不正确
        PushDown(rt,mid-l+1,r-mid);

        //累计答案
        LL ANS=0;
        if(L<=mid) ANS+=Query(L,R,l,mid,ls);
        if(R>mid) ANS+=Query(L,R,mid+1,r,rs);
        return ANS;
    }
};

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    SegmentTree st;
    int n,m;
    cin>>n>>m;
    FOR(i,1,n)
```

```
        cin>>st.A[i];
    st.Build(1,n,1);
    int o,l,r,k;
    FOR(i,1,m){
        cin>>o;
        if(o==1){
            cin>>l>>r>>k;//[l,r]+=k
            st.Update(l,r,k,1,n,1);
        }
        elif(o==2){
            cin>>l>>r;
            cout<<st.Query(l,r,1,n,1)<<endl;//sum[l->r]
        }
    }
    return 0;
}
```