

# 线段树题杂

## 线段树题杂

区间染色

区间离散化

区间子段和

SP2916 GSS5 - Can you answer these queries V

区间不重复子段和

SP1557 GSS2 - Can you answer these queries II

树链剖分+最大子段和+区间修改

动态开点

P3224 [HNOI2012]永无乡(线段树+并查集)

P3224 [HNOI2012]永无乡(splay)

线段树哈希

2020CCPC威海 G - Caesar Cipher

HDU4578 Transformation(2022威海热身B)

参考

## 区间染色

例题：POJ 2528 Mayor's posters

### 区间离散化

对于区间[1,5],[2,7],[7,100],[3,1e7],我们肯定不能直接对区间[1,1e7]进行修改，而是应该先进行排序并离散化，1->1,2->2,3->3,5->4,7->5,100->6,1e7->7，之后区间可以表示为[1,4],[2,5],[5,6],[3,7]。然而这样的表示实际上扩大了访问的区间，因此我们要在间隔大于1的两个元素中间再加数字，正确的表示方法如下：

$$x[1] = 1, x[2] = 2, x[3] = 3, x[4] = 4, x[5] = 5, x[6] = 6, x[7] = 7, x[8] = 100, x[9] = 101, x[10] = 1e7$$

发现了新增的节点x[4],x[6]和[x8]，这有什么用呢？

比如我要涂色[1,3]->颜色1,[5,7]->颜色2，加点前[1,3]->颜色1,[4,5]颜色2，可以发现最终两种颜色把[1,5]覆盖掉了，事实上中间还有一片(3,4)颜色未处理；我们增加节点后的效果是涂掉[1,3],[5,7]，那么数颜色数量的时候就能答案就修正了。

```
//Mayor's posters
#include<iostream>
#include<vector>
#include<algorithm>
#define MID int mid=(p->l + p->r)>>1
using namespace std;
struct Post{ // 海报
    int l,r;
} pst[10100];
int nodenum,cnt,ans;
int t,n,hs[1000010];
vector<int>vt;

struct Node{
    int l,r;
```

```

    bool full; // 区间[l,r]是否被完全覆盖
    Node *ls, *rs;
}tr[1000000];

void buildTree(Node *p, int l, int r){ //建树
    p->l=l;
    p->r=r;
    p->full=0; //初始化节点
    if(l==r)return; //如果是叶子, 直接返回
    nodenum++;
    p->ls=tr+nodenum; //建立左右子树
    nodenum++;
    p->rs=tr+nodenum;
    MID;
    buildTree(p->ls,l,mid); //左区间建树
    buildTree(p->rs,mid+1,r); //右区间建树
}

bool check(Node *p,int l,int r){ //判断该区间是否有覆盖
    if (p->full) return false; //已经被覆盖了, 这份海报就看不到了
    if (p->l==l&&p->r==r){
        p->full=1; //覆盖此区间
        return true;
    }
    bool res;
    MID;
    if(r<=mid) res=check(p->ls,l,r); //全在左区间, 找左子树
    else if(l>mid) res=check(p->rs,l,r);
    else{
        bool b1=check(p->ls,l,mid);
        bool b2=check(p->rs,mid+1,r);
        res=b1||b2; //其中一点没覆盖即可
    }
    if (p->ls->full&&p->rs->full){
        p->full=1; //如果左右区间都被覆盖, 那么这个区间也被覆盖了
    }
    return res;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin>>t;
    while(t--){
        cin>>n;
        nodenum=0,cnt=0,ans=0; //清空数据
        for(int i=0;i<n;i++) {
            cin>>pst[i].l>>pst[i].r;
            vt.push_back(pst[i].l); //把区间端点放入容器
            vt.push_back(pst[i].r);
        }
        sort(vt.begin(),vt.end()); //排序
        vt.erase(unique(vt.begin(),vt.end()),vt.end()); //去重
        for(int i=0;i<vt.size();i++){
            hs[vt[i]]=cnt++; //记录元素所在数的节点编号
            if(i<vt.size()-1){
                if(vt[i+1]-vt[i]>1) cnt++; //中间再插一个点
            }
        }
    }
}

```

```

    }
    buildTree(tr,0,cnt);//开始建树
    for(int i=n-1;i>=0;i--){ //这里倒序，要从没被覆盖的开始数
        if(check(tr,hs[pst[i].l],hs[pst[i].r])){
            ans++; //可见海报增加
        }
    }
    cout<<ans<<endl;
}
return 0;
}

```

## 区间子段和

每次询问在序列的 $[x1, y1]$ 中选 $L$ ,  $[x2, y2]$ 中选 $R$ , 使得子段和 $[L, R]$ 最大

### SP2916 GSS5 - Can you answer these queries V

```

#include<iostream>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=1e4+7;
const int mod=1e9+7;

int t,n,m,x1,x2,y1,y2,a[N],s[N];

#define lc p<<1
#define rc p<<1|1

struct T{
    int l,r;
    int ls,rs,sum,ans,mx,mi;
}tr[N<<2];

T pushup(T L,T R){
    T res;
    res.sum=L.sum+R.sum;
    res.ls=max(L.ls,L.sum+R.ls);
    res.rs=max(R.rs,R.sum+L.rs);
    res.ans=max(max(L.ans,R.ans),L.rs+R.ls);
    res.mx=max(L.mx,R.mx);
    res.mi=min(L.mi,R.mi);
    res.l=L.l;res.r=R.r;
    return res;
}

void build(int p,int l,int r){
    if(l==r){
        tr[p].l=tr[p].r=l;
        tr[p].ans=tr[p].ls=tr[p].rs=tr[p].sum=a[l];
        tr[p].mx=s[l];tr[p].mi=s[l]-a[l];
        return;
    }

```

```

    int mid=l+r>>1;
    build(lc,l,mid);
    build(rc,mid+1,r);
    tr[p]=pushup(tr[lc],tr[rc]);
}

T query(int p,int ql,int qr){
    if(ql<=tr[p].l&&tr[p].r<=qr) return tr[p];
    if(tr[lc].r<ql) return query(rc,ql,qr);
    if(tr[rc].l>qr) return query(lc,ql,qr);
    return pushup(query(lc,ql,qr),query(rc,ql,qr));
}

int qmx(int p,int ql,int qr){
    if(ql<=tr[p].l&&tr[p].r<=qr) return tr[p].mx;
    if(tr[lc].r<ql) return qmx(rc,ql,qr);
    if(tr[rc].l>qr) return qmx(lc,ql,qr);
    return max(qmx(lc,ql,qr),qmx(rc,ql,qr));
}

int qmi(int p,int ql,int qr){
    if(ql<=tr[p].l&&tr[p].r<=qr) return tr[p].mi;
    if(tr[lc].r<ql) return qmi(rc,ql,qr);
    if(tr[rc].l>qr) return qmi(lc,ql,qr);
    return min(qmi(lc,ql,qr),qmi(rc,ql,qr));
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>t;
    while(t--){
        cin>>n;
        for(int i=1;i<=n;i++) cin>>a[i];
        for(int i=1;i<=n;i++) s[i]=s[i-1]+a[i];
        build(1,1,n);
        cin>>m;
        for(int i=1;i<=m;i++){
            cin>>x1>>y1>>x2>>y2;
            if(y1>x2){
                cout<<max(query(1,x2,y1).ans,max(qmx(1,x2,y2)-
qmi(1,x1,x2),qmx(1,y1,y2)-qmi(1,x1,y1)))<<"\n";
            }else cout<<(qmx(1,x2,y2)-qmi(1,x1,y1))<<"\n";
        }
    }
    return 0;
}

```

## 区间不重复子段和

给出 $n$ 个数， $q$ 次询问，求最大子段和，相同的数只算一次。

## SP1557 GSS2 - Can you answer these queries II

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N=1e5+7;

struct Node{
    int sum,mx,stag,mtag; //mx为sum的历史最大值,sum为区间和
    Node(){sum=mx=stag=mtag=0;}
    friend Node operator +(Node lf,Node rt){ //相当于pushup
        Node res;
        res.sum=max(lf.sum,rt.sum);
        res.mx=max(lf.mx,rt.mx);
        return res;
    }
}tr[N<<2];

struct Query{ //离线处理区间询问
    int l,r,id;
}q[N];

bool cmp(Query x,Query y){ //按区间右端从小到大排序
    return x.r<y.r;
}

int n,m,cur[N<<2],pre[N],ql,qr;
int a[N],ans[N],k;

#define mid ((l+r)>>1)
#define ls p<<1
#define rs p<<1|1

void pushdown(int p){
    tr[ls].mx=max(tr[ls].mx,tr[ls].sum+tr[p].mtag);
    tr[rs].mx=max(tr[rs].mx,tr[rs].sum+tr[p].mtag);
    tr[ls].sum+=tr[p].stag;
    tr[rs].sum+=tr[p].stag;
    tr[ls].mtag=max(tr[ls].mtag,tr[ls].stag+tr[p].mtag);
    tr[rs].mtag=max(tr[rs].mtag,tr[rs].stag+tr[p].mtag);
    tr[ls].stag+=tr[p].stag;
    tr[rs].stag+=tr[p].stag;
    tr[p].stag=tr[p].mtag=0;
}

void update(int p,int l,int r){ //修改前面的区间
    if(ql<=l&&r<=qr){
        tr[p].sum+=k;
        tr[p].mx=max(tr[p].mx,tr[p].sum);
        tr[p].stag+=k;
        tr[p].mtag=max(tr[p].mtag,tr[p].stag);
        return;
    }
    pushdown(p);
    if(ql<=mid) update(ls,l,mid);
    if(qr>mid) update(rs,mid+1,r);
    tr[p]=tr[ls]+tr[rs];
}
```

```

Node query(int p,int l,int r){
    if(q1<=l&&r<=qr) return tr[p];
    pushdown(p);
    if(mid<q1) return query(rs,mid+1,r);
    else if(mid>=qr) return query(ls,l,mid);
    else return query(ls,l,mid)+query(rs,mid+1,r);
}

signed main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        pre[i]=cur[a[i]+(int)1e5]; //记录这个位置的数的前一个位置
        cur[a[i]+(int)1e5]=i; //+1e5处理负数的情况
    }
    cin>>m;
    for(int i=1;i<=m;i++){
        cin>>q[i].l>>q[i].r;
        q[i].id=i;
    }
    sort(q+1,q+1+m,cmp); //区间排序
    int j=1;
    for(int i=1;i<=n;i++){ //从前往后插入
        q1=pre[i]+1,qr=i,k=a[i];
        update(1,1,n); //更新线段树，对于出现多个的元素，每次添加元素到[pre[i]+1,i]即可；
        for(;j<=m&&q[j].r<=i;j++){ //这个数以前的区间
            q1=q[j].l,qr=q[j].r;
            ans[q[j].id]=query(1,1,n).mx; //查询区间不重复子段和
        }
    }
    for(int i=1;i<=m;i++) cout<<ans[i]<<"\n";
    return 0;
}

```

## 树链剖分+最大子段和+区间修改

支持将树中①u到v路径的点区间赋值，②查询u到v路径上的最大子段和

```

#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#define lson rt<<1
#define rson rt<<1|1

const int N=1e5+5,M=2e5+5;
int n,q,tot,x[N],lnk[N],ter[M],nxt[M];
int idx,dfn[N],seq[N],sz[N],dep[N],hvy[N],top[N],fa[N];
struct Node {
    int sum,lmx,rmx,ret,tag;
    bool cov;
    Node() {sum=lmx=rmx=ret=0;}
}seg[N<<2];

void add(int u,int v) {

```

```

        ter[++tot]=v,nxt[tot]=lnk[u],lnk[u]=tot;
    }
    void dfs1(int u,int f) {
        dep[u]=dep[f]+1,fa[u]=f,sz[u]=1;
        for(int i=lnk[u];i;i=nxt[i]) {
            int v=ter[i];
            if(v==f) continue;
            dfs1(v,u);
            sz[u]+=sz[v];
            if(sz[v]>sz[hvy[u]]) hvy[u]=v;
        }
    }
    void dfs2(int u,int tp) {
        dfn[u]=++idx,seq[idx]=u,top[u]=tp;
        if(!hvy[u]) return;
        dfs2(hvy[u],tp);
        for(int i=lnk[u];i;i=nxt[i]) {
            int v=ter[i];
            if(v==fa[u]||v==hvy[u]) continue;
            dfs2(v,v);
        }
    }
    Node merge(Node x,Node y) {
        Node ans;
        ans.sum=x.sum+y.sum;
        ans.lmx=std::max(x.lmx,x.sum+y.lmx);
        ans.rmx=std::max(y.rmx,y.sum+x.rmx);
        ans.ret=std::max(std::max(x.ret,y.ret),x.rmx+y.lmx);
        ans.tag=ans.cov=0;
        return ans;
    }
    void build(int rt,int l,int r) {
        if(l==r) {
            seg[rt].sum=x[seq[l]];
            seg[rt].lmx=seg[rt].rmx=seg[rt].ret=std::max(seg[rt].sum,0);
            seg[rt].cov=0;
            return;
        }
        int mid=(l+r)>>1;
        build(lson,l,mid);
        build(rson,mid+1,r);
        seg[rt]=merge(seg[lson],seg[rson]);
    }
    void update(int rt,int l,int r,int k) {
        seg[rt].sum=(r-l+1)*k;
        seg[rt].lmx=seg[rt].rmx=seg[rt].ret=std::max(seg[rt].sum,0);
        seg[rt].cov=1,seg[rt].tag=k;
    }
    void pushdown(int rt,int l,int r) {
        if(!seg[rt].cov) return;
        int mid=(l+r)>>1;
        update(lson,l,mid,seg[rt].tag);
        update(rson,mid+1,r,seg[rt].tag);
        seg[rt].tag=seg[rt].cov=0;
    }
    void modify(int x,int y,int rt,int l,int r,int k) {
        if(x<=l&&r<=y) {
            update(rt,l,r,k);

```

```

        return;
    }
    pushdown(rt,l,r);
    int mid=(l+r)>>1;
    if(x<=mid) modify(x,y,lson,l,mid,k);
    if(mid<y) modify(x,y,rson,mid+1,r,k);
    seg[rt]=merge(seg[lson],seg[rson]);
}

Node query(int x,int y,int rt,int l,int r) {
    if(x<=l&&r<=y) return seg[rt];
    pushdown(rt,l,r);
    int mid=(l+r)>>1;
    Node L,R;
    if(x<=mid) L=query(x,y,lson,l,mid);
    if(mid<y) R=query(x,y,rson,mid+1,r);
    return merge(L,R);
}

void chainModify(int u,int v,int k) {
    for(int fu=top[u],fv=top[v];fu^fv;u=fa[fu],fu=top[u]) {
        if(dep[fu]<dep[fv]) std::swap(u,v),std::swap(fu,fv);
        modify(dfn[fu],dfn[u],1,1,n,k);
    }
    if(dep[u]>dep[v]) std::swap(u,v);
    modify(dfn[u],dfn[v],1,1,n,k);
}

Node chainQuery(int u,int v) {
    Node L,R;
    for(int fu=top[u],fv=top[v];fu^fv;) {
        if(dep[fu]<dep[fv]) {
            R=merge(query(dfn[fv],dfn[v],1,1,n),R);
            v=fa[fv],fv=top[v];
        } else {
            L=merge(query(dfn[fu],dfn[u],1,1,n),L);
            u=fa[fu],fu=top[u];
        }
    }
    if(dep[u]>dep[v]) {
        L=merge(query(dfn[v],dfn[u],1,1,n),L);
    } else {
        R=merge(query(dfn[u],dfn[v],1,1,n),R);
    }
    std::swap(L.lmx,L.rmx);
    return merge(L,R);
}

int main() {
    scanf("%d",&n);
    for(int i=1;i<=n;++i) scanf("%d",&x[i]);
    for(int i=1;i<=n;++i) {
        int u,v;
        scanf("%d%d",&u,&v);
        add(u,v),add(v,u);
    }
    dfs1(1,0),dfs2(1,1),build(1,1,n);
    scanf("%d",&q);
    while(q--) {
        int opt;
        scanf("%d",&opt);
        if(opt==1) {

```



```

        int l,r;
        scanf("%d%d",&l,&r);
        printf("%d\n",chainQuery(l,r).ret);
    } else {
        int l,r,k;
        scanf("%d%d%d",&l,&r,&k);
        chainModify(l,r,k);
    }
}
return 0;
}

```

## 动态开点

给定长度为 $n$ 的序列 $\{a\}$ , 选出 $[l,r]$ 使得 $L \leq \sum_{i=1}^r a_i \leq R$ 的子序列个数

数据范围 $1 \leq n \leq 1e5$

```

//luoguP5459 [BJOI2016]回转寿司
#include<bits/stdc++.h>
#define inf 1e10
// #define int long long
using namespace std;
typedef long long ll;
const int N=1e5+7;
const int mod=1e9+7;
const int LOG=34;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0' || ch>'9'){if(ch=='-')f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}

struct Seg{
    int rt,tot,sum[N*LOG*4],ls[N*LOG*4],rs[N*LOG*4];
    inline void pushup(int p){
        sum[p]=sum[ls[p]]+sum[rs[p]];
    }
    void update(int &p,ll l,ll r,ll pos,ll val){
        if(!p) p=++tot;
        if(l==r){
            sum[p]+=val;
            return;
        }
        ll mid=(l+r)>>1;
        if(pos<=mid) update(ls[p],l,mid,pos,val);
        else update(rs[p],mid+1,r,pos,val);
        pushup(p);
    }
    int query(int &p,ll l,ll r,ll ql,ll qr){
        if(!p) p=++tot;
        if(ql<=l&&r<=qr) return sum[p];
        ll res=0,mid=(l+r)>>1;
        if(ql<=mid) res+=query(ls[p],l,mid,ql,qr);
        if(qr>mid) res+=query(rs[p],mid+1,r,ql,qr);
        return res;
    }
}

```

```

}tr;

int n,L,R,x;
ll pre[N],ans;

signed main(){
    n=read();L=read();R=read();
    for(int i=1;i<=n;i++){
        x=read();
        pre[i]=pre[i-1]+x;
    }
    tr.update(tr.rt,-inf,inf,pre[0],1); //插入pre[0]
    for(int i=1;i<=n;i++){
        ans+=tr.query(tr.rt,-inf,inf,pre[i]-R,pre[i]-L);
        tr.update(tr.rt,-inf,inf,pre[i],1);
    }
    printf("%lld",ans);
    return 0;
}

```

这道题还有神仙CDQ分治的做法。

```

#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=2e5+7;
const int mod=1e9+7;
int n,L,R,s[N],ans,x;

void cdq(int l,int r){
    if(l==r) return;
    int mid=(l+r)/2;
    cdq(l,mid);cdq(mid+1,r);
    int head=l,tail=l-1;
    for(int i=mid+1;i<=r;i++){
        while(tail+1<=mid&& s[i]>=s[tail+1]+L) tail++;
        while(head<=mid&& s[i]>s[head]+R) head++;
        ans+=tail-head+1;
    }
    sort(s+l,s+r+1);
}

signed main(){
    cin>>n>>L>>R;
    for(int i=1;i<=n;i++) cin>>x,s[i]=s[i-1]+x;
    cdq(0,n);
    cout<<ans<<"\n";
    return 0;
}

```

### P3224 [HNOI2012]永无乡(线段树+并查集)

给定每个点、一开始的连边和每个点的排名，支持两个操作：

- Q x y: 连接两个点x和y
- B x y: 求x所在联通块内第y重要的点

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N=1e5+7,M=N*32;

int n,m,q,tot,rt[N],fa[N];
int ls[M],rs[M],id[M],sum[M];
char op;

inline int find(int x){ //并查集查找祖先
    if(x==fa[x]) return x;
    return fa[x]=find(fa[x]);
}

inline void update(int x){ //更新树的大小
    sum[x]=sum[ls[x]]+sum[rs[x]];
}

inline int add(int p,int l,int r,int pos,int idx){ //权值线段树加点
    if(!p) p=++tot;
    if(l==r){
        id[p]=idx;
        sum[p]++;
        return p;
    }
    int mid=((l+r)>>1);
    if(pos<=mid) ls[p]=add(ls[p],l,mid,pos,idx);
    else rs[p]=add(rs[p],mid+1,r,pos,idx);
    update(p);
    return p;
}

inline int merge(int p,int o,int l,int r){ //相对于两棵权值线段树合并
    if(!p||!o) return p+o;
    if(l==r) return p;
    int mid=((l+r)>>1);
    ls[p]=merge(ls[p],ls[o],l,mid);
    rs[p]=merge(rs[p],rs[o],mid+1,r);
    update(p);
    return p;
}

inline int query(int p,int k,int l,int r){ //查询第k大
    if(sum[p]<k||!p) return 0;
    if(l==r) return id[p];
    int mid=((l+r)>>1);
    if(k<=sum[ls[p]]) return query(ls[p],k,l,mid);
    else return query(rs[p],k-sum[ls[p]],mid+1,r);
}

signed main(){
```

```

ios::sync_with_stdio(0);
cin.tie(0);cout.tie(0);
cin>>n>>m;
for(int i=1,x;i<=n;++i){ //初始化并查集维护树根
    fa[i]=i;
    cin>>x;
    rt[i]=add(rt[i],1,n,x,i);
}
for(int i=1,x,y;i<=m;++i){ //并查集合并
    cin>>x>>y;
    x=find(x);y=find(y);
    fa[y]=x;
    rt[x]=merge(rt[x],rt[y],1,n); //更新树根
}
cin>>q;
for(int i=1,x,y;i<=q;++i){
    cin>>op>>x>>y;
    if(op=='B'){ //连接两个点
        x=find(x);y=find(y);
        if(x==y) continue;
        fa[y]=x;
        rt[x]=merge(rt[x],rt[y],1,n);
    }else{ //问与x联通的点中第y大的点
        x=find(x);
        int ans=query(rt[x],y,1,n);
        if(!ans) cout<<"-1\n";
        else cout<<ans<<"\n";
    }
}
return 0;
}

```

### P3224 [HNOI2012]永无乡(splay)

```

#include<bits/stdc++.h>
using namespace std;
const int N=5e5+7;

struct Nod{
    int ch[2],val,f,sz;
}t[N];

int n,m,q,fa[N],root[N],tot,hh[N];

inline int find(int x){return x==fa[x]?x:fa[x]=find(fa[x]);}
inline void pushup(int x){t[x].sz=t[t[x].ch[0]].sz+t[t[x].ch[1]].sz+1;}
inline void rotate(int x){
    int y=t[x].f,z=t[y].f,k=(t[y].ch[1]==x);
    t[z].ch[t[z].ch[1]==y]=x;t[x].f=z;
    t[y].ch[k]=t[x].ch[k^1];t[t[x].ch[k^1]].f=y;
    t[x].ch[k^1]=y;t[y].f=x;
    pushup(y);pushup(x);
}

inline void splay(int x,int goal){
    while(t[x].f!=goal){
        int y=t[x].f,z=t[y].f;
        if(z!=goal) (t[z].ch[0]==y)^(t[y].ch[0]==x)?rotate(x):rotate(y);
        rotate(x);
    }
}

```

```

    }
    if(goal<=n) root[goal]=x;
}

inline void insert(int x,int bh){
    int u=root[bh],ff=bh;
    while(u&& t[u].val!=x) ff=u,u=t[u].ch[x>t[u].val];
    u=++tot;
    t[u].sz=1;t[u].f=ff;
    if(ff>n) t[ff].ch[x>t[ff].val]=u;
    t[u].val=x;t[u].ch[0]=t[u].ch[1]=0;
    splay(u,bh);
}

inline void dfs(int u,int kk){ //遍历整颗splay
    if(t[u].ch[0]) dfs(t[u].ch[0],kk);
    if(t[u].ch[1]) dfs(t[u].ch[1],kk);
    insert(t[u].val,kk); //合并到另一颗splay中
}

inline void merge(int x,int y){
    x=find(x),y=find(y);
    if(x==y) return;
    if(t[root[x]].sz>t[root[y]].sz) swap(x,y);
    fa[x]=y;
    dfs(root[x],y);
}

inline int kth(int bh,int k){
    int u=root[bh];
    if(t[u].sz<k) return -1;
    while(233){
        if(t[t[u].ch[0]].sz+1<k){
            k-=t[t[u].ch[0]].sz+1;
            u=t[u].ch[1];
        }else{
            if(t[t[u].ch[0]].sz>=k) u=t[u].ch[0];
            else return t[u].val;
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    for(int i=1;i<=n;++i) root[i]=i+n,fa[i]=i;
    tot=n+n;
    for(int i=1,x;i<=n;++i){
        cin>>x;
        hh[x]=i;
        t[i+n].val=x;t[i+n].sz=1;t[i+n].f=i;
    }
    for(int i=1,x,y;i<=m;++i) cin>>x>>y,merge(x,y);
    cin>>q;
    char op;
    for(int i=1,x,y,ans;i<=q;++i){
        cin>>op>>x>>y;
        if(op=='B') merge(x,y);
        else{
            ans=kth(find(x),y);
            cout<<(ans== -1?ans:hh[ans])<<"\n";
        }
    }
}

```

```

    }
}
return 0;
}

```

## 线段树哈希

### 2020CCPC威海 G - Caesar Cipher

给定长度为 $n$ 的序列 $\{a\}$ ,  $a_i < 65536$ , 支持以下操作:

- 1 l r: 在区间 $[L, R]$ 中的所有数 $x := (x + 1) \% 65536$
- 2 l r L: 询问 $[l, l + L - 1]$ 区间是否与 $[r, r + L - 1]$ 区间相同。

```

#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=5e5+7;
const int mod=65536;
const int mod1=1e9+7,mod2=1e9+9;
const int base=31;

ll hs[N<<2],mx[N<<2],tg[N<<2];
ll pre[N],pw[N];
int n,q,a[N];

#define ls (p<<1)
#define rs (p<<1|1)
#define mid ((l+r)>>1)
void pushup(int p,int l,int r){
    hs[p]=(hs[ls]*pw[r-mid]%mod1+hs[rs])%mod1;
    mx[p]=max(mx[ls],mx[rs]);
}

void pushdown(int p,int l,int r){
    if(!tg[p]) return;
    hs[ls]=(hs[ls]+tg[p]*pre[mid-1]%mod1)%mod1;
    hs[rs]=(hs[rs]+tg[p]*pre[r-mid-1]%mod1)%mod1;
    mx[ls]+=tg[p];
    mx[rs]+=tg[p];
    tg[ls]+=tg[p];
    tg[rs]+=tg[p];
    tg[p]=0;
}

void update(int p,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr){
        hs[p]=(hs[p]+pre[r-l])%mod1;
        tg[p]++;
        mx[p]++;
        return;
    }
    pushdown(p,l,r);
    if(ql<=mid) update(ls,l,mid,ql,qr);
    if(qr>mid) update(rs,mid+1,r,ql,qr);
    pushup(p,l,r);
}

```

```

}

void update_mod(int p,int l,int r){
    if(mx[p]<mod) return;
    if(l==r){
        mx[p]-=mod;
        hs[p]-=mod;
        return;
    }
    pushdown(p,l,r);
    if(mx[ls]>=mod) update_mod(ls,l,mid);
    if(mx[rs]>=mod) update_mod(rs,mid+1,r);
    pushup(p,l,r);
}

ll query(int p,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr) return hs[p];
    pushdown(p,l,r);
    ll res=0;
    if(qr>mid) res=(res+query(rs,mid+1,r,ql,qr))%mod1;
    if(ql<=mid) res=(res+pw[max(0,min(qr,r)-
mid])*query(ls,l,mid,ql,qr)%mod1)%mod1;
    return res;
}

void build(int p,int l,int r){
    if(l==r){
        hs[p]=mx[p]=a[l];
        return;
    }
    build(ls,l,mid);
    build(rs,mid+1,r);
    pushup(p,l,r);
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    pw[0]=pre[0]=1;
    for(int i=1;i<N;++i) pw[i]=pw[i-1]*base%mod1;
    for(int i=1;i<N;++i) pre[i]=(pre[i-1]+pw[i])%mod1;
    cin>>n>>q;
    for(int i=1;i<=n;++i) cin>>a[i];
    build(1,1,n);
    for(int i=1,op,x,y,L;i<=q;++i){
        cin>>op>>x>>y;
        if(op==1){
            update(1,1,n,x,y);
            update_mod(1,1,n);
        }else{
            cin>>L;
            ll h1=query(1,1,n,x,x+L-1);
            ll h2=query(1,1,n,y,y+L-1);
            if(h1==h2) printf("yes\n");
            else printf("no\n");
        }
    }
    return 0;
}

```

```
}
```

## HDU4578 Transformation(2022威海热身B)

支持①区间加法、②区间乘法、③区间赋值、④询问区间各元素k次方之和。

```
#include <bits/stdc++.h>
// #define int long long
#define mid ((l+r)>>1)
#define ls (p<<1)
#define rs (p<<1|1)
using namespace std;
const int N=1e5+7;
const int mod=10007;

inline int read(){
    int x=0;char ch=getchar();
    while(!isdigit(ch)) ch=getchar();
    while(isdigit(ch)) x=x*10+ch-'0',ch=getchar();
    return x;
}

struct mint{
    int val;
    mint(): val(0) {}
    mint(int x) : val(x) {}
    mint operator+(const mint &b) const {
        return (val + b.val >= mod) ? val + b.val - mod : val + b.val;
    }
    mint operator-(const mint &b) const {
        return (val - b.val < 0) ? val - b.val + mod : val - b.val;
    }
    mint operator*(const mint &b) const {
        return (int)((long long)val * b.val % mod);
    }
    bool operator!=(const mint &b) const {
        return b.val != val;
    }
    bool operator==(const mint &b) const {
        return b.val == val;
    }
};

mint c[13][13] = {};
struct tag {
    mint add, mul;
    tag() {clear();}
    void clear() { add = 0; mul = 1; }
};

struct node {
    mint val[11];
    tag t;
    node() { clear(); }
    void clear() {
        for(int i = 0; i <= 10; i++) val[i] = 0;
        t.clear();
    }
}
```



```

};
mint Pow(mint a, int p) {
    mint ans = 1;
    for( ; p >= 1, a = a * a)
        if(p & 1)
            ans = ans * a;
    return ans;
}
struct SGT {
    node tr[N<<1];
    int *a, n;
    void update(int p) {
        for(int i = 0; i <= 10; i++) {
            tr[p].val[i] = tr[ls].val[i] + tr[rs].val[i];
        }
    }
    void build(int p, int l, int r) {
        tr[p].clear();
        if(l == r) {
            tr[p].val[0] = 1;
            for(int i = 1; i <= 10; i++) {
                tr[p].val[i] = tr[p].val[i - 1] * a[l];
            }
            return ;
        }
        build(ls, l, mid);
        build(rs, mid + 1, r);
        update(p);
    }
    void setup(int t, int *data) {
        a = data;
        n = t;
        build(1, 1, n);
    }
    void pushtag(int p, const tag &tg) {
        for(int i = 10; i >= 0; i--) {
            if(tg.mul != 1) {
                tr[p].val[i] = tr[p].val[i] * Pow(tg.mul, i);
            }
        }
        for(int i = 10; i >= 0; i--) {
            if(tg.add != 0) {
                mint now = 0;
                for(int j = 0; j <= i; j++) {
                    now = now + tr[p].val[i - j] * C[i][j] * Pow(tg.add, j);
                }
                tr[p].val[i].val = now.val;
            }
        }
        tr[p].t.mul = tr[p].t.mul * tg.mul;
        tr[p].t.add = tr[p].t.add * tg.mul;
        tr[p].t.add = tr[p].t.add + tg.add;
    }
    void pushdown(int p) {
        pushtag(ls, tr[p].t);
        pushtag(rs, tr[p].t);
        tr[p].t.clear();
    }
}

```

```

void modify(int p, int l, int r, int ql, int qr, const tag &tg) {
    if(ql <= l && r <= qr) {
        pushtag(p, tg);
        return ;
    }
    pushdown(p);
    if(ql <= mid) modify(ls, l, mid, ql, qr, tg);
    if(mid < qr) modify(rs, mid + 1, r, ql, qr, tg);
    update(p);
}

mint query(int p, int l, int r, int ql, int qr, int k) {
    if(ql <= l && r <= qr) return tr[p].val[k];
    pushdown(p);
    if(qr <= mid) return query(ls, l, mid, ql, qr, k);
    if(ql > mid) return query(rs, mid + 1, r, ql, qr, k);
    return query(ls, l, mid, ql, qr, k) + query(rs, mid + 1, r, ql, qr, k);
}

void modify(int l, int r, const tag &tg) {
    modify(1, 1, n, l, r, tg);
}

mint query(int l, int r, int k) {
    return query(1, 1, n, l, r, k);
}

} T;
int n, q, a[N];

void solve(){
    //cin >> n >> q;
    //for(int i = 1; i <= n; i++) cin >> a[i];
    T.setup(n, a);
    for(int i = 1, op, l, r, k; i <= q; i++) {
        //cin >> op >> l >> r >> k;
        op=read(); l=read(); r=read(); k=read();
        tag tg;
        if(op == 1) {
            tg.add = k;
            T.modify(l, r, tg);
        }
        if(op == 2) {
            tg.mul = k;
            T.modify(l, r, tg);
        }
        if(op == 3) {
            tg.mul = 0;
            tg.add = k;
            T.modify(l, r, tg);
        }
        if(op == 4) {
            printf("%d\n", T.query(l, r, k).val);
            //cout << T.query(l, r, k).val << endl;
        }
    }
}

}

signed main() {
    c[0][0] = 1;
    for(int i = 1; i <= 10; i++) {
        c[i][0] = 1;
        for(int j = 1; j <= i; j++) {

```

```
        c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
    }
}
int T=1;
while(1){
    n=read();q=read();
    if(!n&&!q) break;
    solve();
}
return 0;
}
```

## 参考

<https://www.luogu.com.cn/blog/MiraiMemorabilia/solution-sp1557>

<https://hydingsy.github.io/>