1.可持久化并查集
2.可撤回并查集
3.可并堆
4.ST表
5.线段树各种操作
    1.维护区间加，区间推平，历史最大值
    2.线段树合并
    3标记永久化
    4.维护矩阵
    5.主席树
    6.扫描线
6.分治
    点分
    边分
    树剖
    LCT
7.回滚莫队
8.kd_tree
9.fhq_treap
10.带修第k大
11.splay
12.块状链表
13。可持久化trie
14.树套树

1.可持久化并查集

```
int ls[200005*30],rs[200005*30],dep[200005],lst[200005*30];
int n,m,cnt,root[200005*30];
#define mid ((l+r)>>1)
void built(int &rt,int l,int r){
      rt=++cnt;
      if(l==r){
          lst[rt]=l;
          return ;
      }
     built(ls[rt],l,mid);
     built(rs[rt],mid+1,r);
}
void modify(int &rt,int pre,int l,int r,int x,int val){
   rt=++cnt;
   ls[rt]=ls[pre];
   rs[rt]=rs[pre];
   if(l==r){
       lst[rt]=val;
       return ;
   }
   if(x<=mid)modify(ls[rt],ls[pre],l,mid,x,val);
   else modify(rs[rt],rs[pre],mid+1,r,x,val);
}
int query(int rt,int l,int r,int x){
   if(l==r)return lst[rt];
   if(x<=mid)return query(ls[rt],l,mid,x);
   else return query(rs[rt],mid+1,r,x);
}
int find(int x,int id){
    int ff=query(root[id],1,n,x);
    if(x==ff)return x;
    else return find(ff,id);
}
void merge(int x,int y,int id){
    int xx=find(x,id),yy=find(y,id);
    if(xx==yy)return ;
    if(dep[xx]<dep[yy])swap(xx,yy);
    modify(root[id],root[id-1],1,n,yy,xx);
    dep[xx]++;
}
int main(){
#ifndef ONLINE_JUDGE
    file("3402");
#endif
     n=read();
     m=read();
     built(root[0],1,n);
     F(i,1,n)dep[i]=1;
//   F(i,1,n)printf("%d\n",query(root[0],1,n,i));
     F(i,1,m){
         int opt=read();
      root[i]=root[i-1];
```

```
      if(opt==1){
        int a=read(),b=read();
        merge(a,b,i);
      }
      if(opt==2){
          int x=read();
         root[i]=root[x];
      }
      if(opt==3){
        int a=read(),b=read();
        a=find(a,i);
        b=find(b,i);
        printf("%d\n",a==b);
      }
     }
    return 0;
}
```

2.可撤回并查集

```
namespace ufs {
    int f[N], s[N], v[N], t;

    void clr(int n) {
        t = 0;
        fill(s + 1, s + n + 1, 1);
        fill(f + 1, f + n + 1, 0);
    }

    int find(int x) {
        while (f[x]) x = f[x];
        return x;
    }
    void unite(int x, int y) {
        x = find(x); y = find(y);
        if (x == y) return;
        if (s[x] > s[y]) swap(x, y);
        v[++t] = x;
        f[x] = y; s[y] += s[x];
    }

    void rollback(int w) {
        while (t != w) {
            int x = v[t], y = f[x];
            f[x] = 0; s[y] -= s[x];
            --t;
        }
    }
}
```

3.可并堆

```
int fa[100005];
int ls[100005],rs[100005];
int dep[100005];
int val[100005];
int n,m;
int find(int x){
   while(fa[x])x=fa[x];
   return x;
}
int merge(int x,int y){
     if(!x||!y)return x|y;
      if(val[x]>val[y]||(val[x]==val[y]&&x>y))swap(x,y);
     rs[x]=merge(rs[x],y);
     fa[rs[x]]=x;
     if(dep[ls[x]]<dep[rs[x]])swap(ls[x],rs[x]);
     dep[x]=dep[rs[x]]+1;
     return x;
}
int main () {
#ifndef ONLINE_JUDGE
file("3377");
#endif
    n=read();
    m=read();
    F(i,1,n)val[i]=read();
    dep[0]=-1;
    while(m--){
       int opt=read(),x=read();
       if(opt==1){
          int y=read();
          if(val[x]==-1||val[y]==-1)continue;
          x=find(x),y=find(y);
          if(x==y)continue;
          merge(x,y);
       }
       else{
          if(val[x]==-1){
             puts("-1");
             continue;
          }
          x=find(x);
          printf("%d\n",val[x]);
          val[x]=-1;
          fa[ls[x]]=fa[rs[x]]=0;
          merge(ls[x],rs[x]);
       }
    }
    return 0;
}
```

可持久化

```
int merge(int x, int y) {
  if (!x || !y) return x + y;
  if (v[x] > v[y]) swap(x, y);
  int p = ++cnt;
  lc[p] = lc[x];
  v[p] = v[x];
  rc[p] = merge(rc[x], y);
  if (dist[lc[p]] < dist[rc[p]]) swap(lc[p], rc[p]);
  dist[p] = dist[rc[p]] + 1;
  return p;
}
```

4.ST表

```
int lg2[N], st[18][N];
void build(int n) {
    for (int i = 2; i <= n; ++i) lg2[i] = lg2[i >> 1] + 1;
    for (int i = 1; i <= n; ++i) st[0][i] = a[i];
    for (int j = 1; (1 << j) <= n; ++j)
        for (int i = 1; i + (1 << j) - 1 <= n; ++i)
            st[j][i] = max(st[j - 1][i], st[j - 1][i + (1 << j - 1)]);
}

int query(int l, int r) {
    int w = lg2[r - l];
    return max(st[w][l], st[w][r - (1 << w)]);
}
```

5.线段树各种操作

1.维护区间加，区间推平，历史最大值

```
int n,m;
const long long inf=1e18;
struct node{
  long long maxn,sum,lazymax,lazysum;
  void operator +=(node &rhs){
      lazymax=max(lazymax,maxn+rhs.lazymax);
      lazysum=max(lazysum,max(sum+rhs.lazymax,rhs.lazysum));
        maxn=max(maxn+rhs.maxn,-inf);
        sum=max(sum+rhs.maxn,rhs.sum);
  }
}tree[2000005],val;
long long a[500005];
#define mid ((l+r)>>1)
void built(int rt,int l,int r){
    tree[rt]=(node){0,-inf,0,-inf};
   if(l==r){
      tree[rt].maxn=a[l];
      return ;
   }
   built(rt<<1,l,mid);
   built(rt<<1|1,mid+1,r);
}
void pushdown(int rt){
    tree[rt<<1]+=tree[rt];
    tree[rt<<1|1]+=tree[rt];
    tree[rt]=(node){0,-inf,0,-inf};
}
void update(int rt,int l,int r,int L,int R){
   if(l==L&&r==R){tree[rt]+=val;return ;}
    pushdown(rt);
    if(R<=mid)update(rt<<1,l,mid,L,R);
    else if(L>mid)update(rt<<1|1,mid+1,r,L,R);
    else update(rt<<1,l,mid,L,mid),update(rt<<1|1,mid+1,r,mid+1,R);
}
void query(int rt,int l,int r,int pos){
     if(l==r){val=tree[rt];return ;}
     pushdown(rt);
     if(pos<=mid)query(rt<<1,l,mid,pos);
     else query(rt<<1|1,mid+1,r,pos);
}
int main () {
#ifndef ONLINE_JUDGE
file("164");
#endif
     n=read();
     m=read();
     F(i,1,n)a[i]=read();
     built(1,1,n);
     while(m--){
        int opt=read();
        if(opt<=3){
           int l=read(),r=read(),x=read();
```

```
            if(opt==3){
               val=(node){-inf,x,-inf,x};
            }
            if(opt==2){
              val=(node){-x,0,-x,0};
            }
            if(opt==1){
              val=(node){x,-inf,x,-inf};
            }
            update(1,1,n,l,r);
        }
        else{
            int x=read();
            query(1,1,n,x);
            if(opt==4)printf("%lld\n",max(val.maxn,val.sum));
            else printf("%lld\n",max(val.lazymax,val.lazysum));
        }
     }
    return 0;
}
```

2.线段树合并

```
int ls[200005*40],rs[200005*40],sum1[200005*40],sum2[200005*40],cnt;
int n,Root;
long long ans;
long long cntl,cntr;
int tot,root[400005];
struct node{
   int ls,rs,val;
}tree[400005];
void dfs(int &x){
    x=++tot;
    tree[x].val=read();
    if(tree[x].val)return ;
    dfs(tree[x].ls);
    dfs(tree[x].rs);
}
#define mid ((l+r)>>1)
void update(int &rt,int l,int r,int pos){
    if(!rt)rt=++cnt;
    sum1[rt]++;
    if(l==r)return ;
    if(pos<=mid)update(ls[rt],l,mid,pos);
    if(pos>mid)update(rs[rt],mid+1,r,pos);
    sum1[rt]=sum1[ls[rt]]+sum1[rs[rt]];
}
int merge(int x,int y,int l,int r){
    if(!x||!y)return x|y;
    if(l==r){
        sum1[x]+=sum1[y];
        return x;
    }
    cntl+=1ll*sum1[rs[x]]*sum1[ls[y]];
    cntr+=1ll*sum1[ls[x]]*sum1[rs[y]];
    ls[x]=merge(ls[x],ls[y],l,mid);
    rs[x]=merge(rs[x],rs[y],mid+1,r);
    sum1[x]=sum1[ls[x]]+sum1[rs[x]];
    return x;
}
void getans(int x){
     if(!x)return ;
     getans(tree[x].ls);
     getans(tree[x].rs);
     if(tree[x].val==0){
         root[x]=merge(root[tree[x].ls],root[tree[x].rs],1,n);
         ans+=min(cntl,cntr);
         cntl=cntr=0;
     }
}
int main () {
#ifndef ONLINE_JUDGE
file("3521");
#endif
    n=read();
    dfs(Root);
    F(i,1,tot)if(tree[i].val)update(root[i],1,n,tree[i].val);
```

```
    getans(Root);
    printf("%lld\n",ans);
    return 0;
}
```

3标记永久化

区间加/区间和，左闭右开。

```
typedef long long ll;
ll st[1<<18], tg[1<<18]; int a[N];

void build(int p, int lb, int rb) {
    st[p] = tg[p] = 0;
    if (lb + 1 == rb) st[p] = a[lb];
    else {
        int mid = (lb + rb) >> 1;
        build(p << 1, lb, mid);
        build(p << 1 | 1, mid, rb);
        st[p] = st[p << 1] + st[p << 1 | 1];
    }
}

void modify(int l, int r, ll t, int p, int lb, int rb) {
    st[p] += (r - l) * t;
    if (l <= lb && rb <= r) tg[p] += t;
    else {
        int mid = (lb + rb) >> 1;
        if (l < mid) modify(l, min(r, mid), t, p << 1, lb, mid);
        if (r > mid) modify(max(mid, l), r, t, p << 1 | 1, mid, rb);
    }
}

ll query(int l, int r, ll t, int p, int lb, int rb) {
    if (l <= lb && rb <= r) return st[p] + (r - l) * t;
    else {
        int mid = (lb + rb) >> 1; ll res = 0; t += tg[p];
        if (l < mid) res += query(l, min(r, mid), t, p << 1, lb, mid);
        if (r > mid) res += query(max(mid, l), r, t, p << 1 | 1, mid, rb);
        return res;
    }
}
//  Example:
build(1, 1, n + 1); //  建树，初值为a[1~n]
modify(l, r, v, 1, 1, n + 1);   //  区间[l,r)加v
query(l, r, 0, 1, 1, n + 1);    //  区间[l,r)的和
```

4.维护矩阵

```
int n,m;
int dp[100005][2];
int w[100005];
int he[200005],to[200005],ne[200005],e;
int
cnt,pos[100005],dfn[100005],top[100005],son[100005],sz[100005],dep[100005],fa[10
0005];
struct node{
   int g[2][2];
   node(){Set(g,0);}
}sum[400005],val[100005];
int t[400005];
node operator * (node a,node b){
    node c;
    F(i,0,1)
        F(j,0,1)
           F(k,0,1)
        chkmax(c.g[i][k],a.g[i][j]+b.g[j][k]);
    return c;
}
void add(int x,int y){
    to[++e]=y;
    ne[e]=he[x];
    he[x]=e;
}
void DP(int x){
   int v;
   dp[x][1]=w[x];
   for(int i=he[x];i;i=ne[i]){
         v=to[i];
        if(v^fa[x]){
          DP(v);
          dp[x][0]+=max(dp[v][0],dp[v][1]);
          dp[x][1]+=dp[v][0];
        }
   }
}
void dfs(int x,int ff){
    fa[x]=ff;
    dep[x]=dep[ff]+1;
    for(int i=he[x];i;i=ne[i]){
        int v=to[i];
        if(v==ff)continue;
        dfs(v,x);
        if(son[x]==0||sz[son[x]]<sz[v]){
           son[x]=v;
        }
        sz[x]+=sz[v];
     }
    sz[x]++;
}
void dfs2(int x,int ff){
     dfn[x]=++cnt;
     pos[cnt]=x;
```

```
     if(son[x]){
        top[son[x]]=top[x];
        dfs2(son[x],x);
        t[x]=t[son[x]];
     }
     else{
        t[x]=x;
        return ;
     }
     for(int i=he[x];i;i=ne[i]){
         int v=to[i];
         if(v==ff||v==son[x])continue;
         top[v]=v;
         dfs2(v,x);
     }
}
#define mid ((l+r)>>1)
void built(int rt,int l,int r){
    if(l==r){
       int u=pos[l],sum1=w[u],sum0=0;
       for(int i=he[u];i;i=ne[i]){
           int v=to[i];
           if(v^fa[u]&&v^son[u]){
               sum1+=dp[v][0];
               sum0+=max(dp[v][0],dp[v][1]);
           }
       }
       val[u].g[0][0]=val[u].g[0][1]=sum0;
       val[u].g[1][0]=sum1;
       sum[rt]=val[u];
    }
    else{
      built(rt<<1,l,mid);
      built(rt<<1|1,mid+1,r);
      sum[rt]=sum[rt<<1]*sum[rt<<1|1];
    }
}
void update(int rt,int l,int r,int poss){
    if(l==r)sum[rt]=val[pos[l]];
    else{
       if(poss<=mid)update(rt<<1,l,mid,poss);
       else update(rt<<1|1,mid+1,r,poss);
       sum[rt]=sum[rt<<1]*sum[rt<<1|1];
    }
}
node query(int rt,int l,int r,int L,int R){
    if(l>=L&&r<=R)return sum[rt];
    if(R<=mid)return query(rt<<1,l,mid,L,R);
    else if(L>mid)return query(rt<<1|1,mid+1,r,L,R);
    else return query(rt<<1,l,mid,L,R)*query(rt<<1|1,mid+1,r,L,R);
}
```

5.主席树

```
int tot;
int n,m;
int size;
int
a[200005],b[200005],rt[200005*40],ls[200005*40],rs[200005*40],sum[200005*40];
#define mid ((l+r)>>1)
void built(int &rt,int l,int r){
     rt=++tot;
     sum[rt]=0;
     if(l==r)return ;
     built(ls[rt],l,mid);
     built(rs[rt],mid+1,r);
}
void update(int &rt,int l,int r,int lst,int p){
         rt=++tot;
         ls[rt]=ls[lst];
         rs[rt]=rs[lst];
         sum[rt]=sum[lst]+1;
         if(l==r)return ;
         if(p<=mid)update(ls[rt],l,mid,ls[lst],p);
         else update(rs[rt],mid+1,r,rs[lst],p);
}
int query(int s,int t,int l,int r,int k){
   if(l==r)return l;
   int cnt=sum[ls[t]]-sum[ls[s]];
   if(k<=cnt)return query(ls[s],ls[t],l,mid,k);
   else return query(rs[s],rs[t],mid+1,r,k-cnt);
}
int main () {
#ifndef ONLINE_JUDGE
freopen("p3834.in","r",stdin);
freopen("p3834.out","w",stdout);
#endif
     n=read();
     m=read();
     F(i,1,n)a[i]=read(),b[i]=a[i];
     sort(b+1,b+n+1);
     size=unique(b+1,b+n+1)-(b+1);
     tot=0;
     built(rt[0],1,size);
     F(i,1,n)a[i]=lower_bound(b+1,b+size+1,a[i])-b;
     F(i,1,n)update(rt[i],1,size,rt[i-1],a[i]);
     while(m--){
        int ll=read(),rr=read(),k=read();
       int res=query(rt[ll-1],rt[rr],1,size,k);
       printf("%d\n",b[res]);
     }
    return 0;
}
```

6.扫描线

```
int n,cnt,cnt2;
struct node{
   int l,r,h,opt;
   bool operator<(const node &rhs)const{
        return h<rhs.h;
   }
}pos[400005];
int poi[400005];
struct seg{
    long long len,xorlen,lazy;
    bool now;
}tree[1600005];
#define mid (l+r)/2
void pushdown(int rt,int l,int r){
    if(tree[rt].now){
        tree[rt<<1].now^=1;
        tree[rt<<1|1].now^=1;
        tree[rt<<1].xorlen=poi[mid]-poi[l]-tree[rt<<1].xorlen;
        tree[rt<<1|1].xorlen=poi[r]-poi[mid]-tree[rt<<1|1].xorlen;
        tree[rt].now=0;
    }
    if(tree[rt].lazy){
       tree[rt].len=poi[r]-poi[l];
    }
    else if(l==r)tree[rt].len=0;
    else tree[rt].len=tree[rt<<1].len+tree[rt<<1|1].len;
}
void modify(int rt,int l,int r,int L,int R){
     if(L<=l&&r<=R){
            tree[rt].xorlen=poi[r]-poi[l]-tree[rt].xorlen;
            tree[rt].now^=1;
            return ;
     }
     pushdown(rt,l,r);
     if(L<mid)modify(rt<<1,l,mid,L,R);
     if(R>mid)modify(rt<<1|1,mid,r,L,R);
     tree[rt].xorlen=tree[rt<<1].xorlen+tree[rt<<1|1].xorlen;
}
void update(int rt,int l,int r,int L,int R,int val){
     if(L<=l&&r<=R){
        tree[rt].lazy+=val;
        pushdown(rt,l,r);
        return ;
     }
     if(L<mid)update(rt<<1,l,mid,L,R,val);
     if(R>mid)update(rt<<1|1,mid,r,L,R,val);
     pushdown(rt,l,r);
}
int main () {
#ifndef ONLINE_JUDGE
    File("i");
#endif
    n=read();
    F(i,1,n){
```

```
        int x1=read(),y1=read(),x2=read(),y2=read();
        x1+=1e9;
        x2+=1e9;
        y1+=1e9;
        y2+=1e9;
        if(x1>x2)swap(x1,x2);
        if(y1>y2)swap(y1,y2);
        pos[++cnt].l=x1,pos[cnt].r=x2,pos[cnt].h=y1,poi[cnt]=x1,pos[cnt].opt=1;
        pos[++cnt].l=x1,pos[cnt].r=x2,pos[cnt].h=y2,poi[cnt]=x2,pos[cnt].opt=-1;
    }
    //cout<<cnt<<endl;
    sort(poi+1,poi+cnt+1);
    sort(pos+1,pos+cnt+1);
    cnt2=unique(poi+1,poi+cnt+1)-poi-1;
    long long ans1=0,ans2=0;
    pos[0].h=1e9;
    F(i,1,cnt){
        pos[i].l=lower_bound(poi+1,poi+cnt2+1,pos[i].l)-poi;
        pos[i].r=lower_bound(poi+1,poi+cnt2+1,pos[i].r)-poi;
        ans1+=1ll*tree[1].len*(pos[i].h-pos[i-1].h);
        ans2+=1ll*tree[1].xorlen*(pos[i].h-pos[i-1].h);
        modify(1,1,cnt2,pos[i].l,pos[i].r);
        update(1,1,cnt2,pos[i].l,pos[i].r,pos[i].opt);
    }
//  printf("%lld %lld\n",ans1,ans2);
    printf("%lld\n",ans1-ans2);
    return 0;
}
```

6.分治

点分

```
vector<int> g[N];
bool vis[N]; int sz[N], msz[N];
int dfs_sz(int u, int f, int s) {
    int res = msz[u] = 0; sz[u] = 1;
    for (int v : g[u]) {
        if (vis[v] || v == f) continue;
        int r = dfs_sz(v, u, s); sz[u] += sz[v];
        msz[u] = max(msz[u], sz[v]);
        if (!res || msz[r] < msz[res]) res = r;
    }
    msz[u] = max(msz[u], s - sz[u]);
    if (!res || msz[u] < msz[res]) res = u;
    return res;
}

int vdc(int u, int s) {
    vis[u = dfs_sz(u, 0, s)] = 1;

    for (int v : g[u]) {
        if (vis[v]) continue;
        int w = vdc(v, sz[v] < sz[u] ? sz[v] : s - sz[u]);

    }
    return u;
}
```

边分

`eu/ev` 为边的两端

`ex=eu^ev`

`ew` 为边权

`ef` 表示该边是否为虚边

`flg` 表示该点是否为虚点

```
struct edge { int v, w; };
vector<edge> g0[N];
vector<int> g[N]; int nc; bool flg[N];
int eu[N], ev[N], ex[N], ew[N], ec; bool ef[N];
bool vis[N]; int dep[N], msz[N], sz[N];

int adde(int u, int v, int w, int f) {
    int i = ++ec;
    eu[i] = u; ev[i] = v; ex[i] = u ^ v;
    ew[i] = w; ef[i] = f; vis[i] = 0;
    g[u].push_back(i);
    g[v].push_back(i);
    return i;
}

void dfs_rec(int u, int f) {
    int p = u, c = 0;
    for (edge e : g0[u]) {
        int v = e.v; if (v == f) continue;
        adde(p, v, e.w, 0); dfs_rec(v, u);
        if (c + 2 + !!f >= g0[u].size()) continue;
        int q = ++nc; ++c; flg[q] = 1;
        adde(p, q, 0, 1); p = q;
    }
}

int dfs_sz(int i, int f, int s) {
    int res = 0, u = ex[i] ^ f;
    dep[u] = dep[f] + 1; sz[i] = 1; msz[i] = 0;
    for (int j : g[u]) {
        if (j == i || vis[j]) continue;
        int r = dfs_sz(j, u, s); sz[i] += sz[j];
        if (!res || msz[r] < msz[res]) res = r;
    }
    msz[i] = max(sz[i], s - sz[i]);
    if (!res || msz[i] < msz[res] && !vis[i]) res = i;
    return res;
}

void dfs_cnt(int u, int f, int d, int* c) {
    if (!flg[u]) c[d]++;
    for (int i : g[u])
        if (!vis[i] && (ex[i] ^ u) != f)
            dfs_cnt(ex[i] ^ u, u, (d + ew[i]) % 3, c);
}
```

```
ll edc(int i, int f, int s) {
    i = dfs_sz(i, f, s);
    if (vis[i]) return 0; else vis[i] = 1;
    int u = eu[i], v = ev[i]; ll res = cal(i);
    res += edc(i, u, dep[u] > dep[v] ? s - sz[i] : sz[i]);
    res += edc(i, v, dep[v] > dep[u] ? s - sz[i] : sz[i]);
    return res;
}

ll edc(int n) {
    ec = 0; nc = n; dfs_rec(1, 0);
    int i = adde(0, 1, 0, 1);
    vis[i] = 1;
    return edc(i, 0, nc);
}
```

树剖

```
int n,m,mod,root;
int he[100005],to[200005],ne[200005],e;
long long val[100005];
int
dfn[100005],son[100055],sz[100005],cnt,pos[100050],top[100005],dep[100005],fa[10
0005];
void add(int x,int y){
    to[++e]=y;
    ne[e]=he[x];
    he[x]=e;
}
void dfs(int x,int ff){
   fa[x]=ff;
   dep[x]=dep[ff]+1;
   for(int i=he[x];i;i=ne[i]){
      int v=to[i];
      if(v==ff)continue;
      dfs(v,x);
      sz[x]+=sz[v];
      if(son[x]==0||sz[son[x]]<sz[v])son[x]=v;
   }
   sz[x]++;
}
void dfs2(int x,int ff){
    dfn[x]=++cnt;
    pos[cnt]=x;
    if(son[x]){
        top[son[x]]=top[x];
        dfs2(son[x],x);
    }
    for(int i=he[x];i;i=ne[i]){
       int v=to[i];
       if(v==ff||v==son[x])continue;
       top[v]=v;
       dfs2(v,x);
    }
}
long long  tree[400005],lazy[400005];
#define mid ((l+r)>>1)
void built(int rt,int l,int r){
    if(l==r){
       tree[rt]=val[pos[l]];
       return ;
    }
    built(rt<<1,l,mid);
    built(rt<<1|1,mid+1,r);
    tree[rt]=(tree[rt<<1]+tree[rt<<1|1])%mod;
}
void pushdown(int rt,int l,int r){
    lazy[rt<<1]+=lazy[rt];
    lazy[rt<<1|1]+=lazy[rt];
    tree[rt<<1]=(tree[rt<<1]+1ll*(mid-l+1)*lazy[rt]%mod)%mod;
    tree[rt<<1|1]=(tree[rt<<1|1]+1ll*(r-mid)*lazy[rt]%mod)%mod;
    lazy[rt]=0;
```

```
}
void update(int rt,int l,int r,int L,int R,long long  val){
    if(L<=l&&r<=R){
       lazy[rt]+=val;
       tree[rt]=(tree[rt]+1ll*(r-l+1)*val%mod)%mod;
       return ;
    }
    if(lazy[rt])pushdown(rt,l,r);
    if(L<=mid)update(rt<<1,l,mid,L,R,val);
    if(R>mid)update(rt<<1|1,mid+1,r,L,R,val);
    tree[rt]=(tree[rt<<1]+tree[rt<<1|1])%mod;
}
long long query(int rt,int l,int r,int L,int R){
    if(L<=l&&r<=R)return tree[rt];
    if(lazy[rt])pushdown(rt,l,r);
    long long res=0;
    if(L<=mid)res=query(rt<<1,l,mid,L,R);
    if(R>mid)res=(res+query(rt<<1|1,mid+1,r,L,R))%mod;
    return res%mod;
}

void addlca(int x,int y,long long z){
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]]){
            swap(x,y);
        }
        update(1,1,n,dfn[top[x]],dfn[x],z);
        x=fa[top[x]];
    }
    if(dfn[x]>dfn[y])swap(x,y);
    update(1,1,n,dfn[x],dfn[y],z);
}
long long querylca(int x,int y){
   long long res=0;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]]){
            swap(x,y);
        }
        res=(res+query(1,1,n,dfn[top[x]],dfn[x]))%mod;
        x=fa[top[x]];
    }
    if(dfn[x]>dfn[y])swap(x,y);
    res=(res+query(1,1,n,dfn[x],dfn[y]))%mod;
    return res;
}
int main () {
#ifndef ONLINE_JUDGE
file("a");
#endif
    n=read();
    m=read();
    root=read();
    mod=read();
    F(i,1,n)val[i]=read();
    F(i,1,n-1){
       int x=read(),y=read();
       add(x,y);
       add(y,x);
```

```
    }
    dfs(root,0);
    top[root]=root;
    dfs2(root,0);
    built(1,1,n);
    while(m--){
       int opt=read();
       if(opt==1){
         int x=read(),y=read();
         long long z=read();
         addlca(x,y,z);
       }
       if(opt==2){
          int x=read(),y=read();
          printf("%lld\n",querylca(x,y));
       }
       if(opt==3){
           int x=read();
           long long z=read();
         update(1,1,n,dfn[x],dfn[x]+sz[x]-1,z);
       }
       if(opt==4){
          int x=read();
          printf("%lld\n",query(1,1,n,dfn[x],dfn[x]+sz[x]-1));
       }
    }
    return 0;
}
```

LCT

```
int n,m;
int fa[300005],son[300005][2],st[300005],v[300005];
int stk[300005];
bool root[300005];
#define lson(x) son[x][0]
#define rson(x) son[x][1]
bool isroot(int x){
     return lson(fa[x])==x||rson(fa[x])==x;
}
void pushup(int x){
    st[x]=st[lson(x)]^st[rson(x)]^v[x];
}
void rot(int x){
    swap(lson(x),rson(x));
    root[x]^=1;
}
void pushdown(int x){
     if(root[x]){
         if(lson(x))rot(lson(x));
         if(rson(x))rot(rson(x));
         root[x]=0;
     }
}
void rotate(int x){
     int f=fa[x],ef=fa[f],o=rson(f)==x,w=son[x][!o];
     if(isroot(f))son[ef][rson(ef)==f]=x;
     son[x][!o]=f;
     son[f][o]=w;
     if(w)fa[w]=f;
     fa[f]=x;
     fa[x]=ef;
     pushup(f);
}
void splay(int x){
     int f=x,ef=0;
      stk[++ef]=f;
      while(isroot(f))stk[++ef]=fa[f],f=fa[f];
      while(ef)pushdown(stk[ef--]);
      while(isroot(x)){
         f=fa[x];
         ef=fa[f];
         if(isroot(f)){
            rotate((lson(f)==x)^(lson(ef)==f)?x:f);
         }
         rotate(x);
    }
      pushup(x);
}
void access(int x){
    for(int i=0;x;x=fa[i=x]){
        splay(x);
        rson(x)=i;
        pushup(x);
    }
```

```
}
void makeroot(int x){
    access(x);
    splay(x);
    rot(x);
}
int findroot(int x){
     access(x);
     splay(x);
     while(lson(x))pushdown(x),x=lson(x);
     return x;
}
void link(int x,int y){
   makeroot(x);
   if(findroot(x)!=y)fa[x]=y;
}
void cut(int x,int y){
    makeroot(x);
    if(findroot(y)==x&&fa[x]==y&&!rson(x)){
       fa[x]=lson(y)=0;
       pushup(y);
    }
}
int main () {
#ifndef ONLINE_JUDGE
file("a");
#endif
    n=read();
    m=read();
    F(i,1,n)v[i]=read();
    while(m--){
       int opt=read(),x=read(),y=read();
       if(opt==0){
          makeroot(x);
          access(y);
          splay(y);
          printf("%d\n",st[y]);
       }
       if(opt==1){
           link(x,y);
       }
       if(opt==2){
           cut(x,y);
       }
       if(opt==3){
          splay(x);
          v[x]=y;
       }
    }
    return 0;
}
```

7.回滚莫队

```
int cnt,belongs[400005],blksz,blkcnt;
int a[400005],b[400005];
int n,m;
int lst[400005],rst[400005];
int ls[400005],rs[400005];
int bf[400005];
int appear[400005],top;
int Ans[400005];
struct node{
   int l,r,id;
   bool operator<(const node &rhs)const{
         if(belongs[l]==belongs[rhs.l])return r<rhs.r;
         return l<rhs.l;
   }
}que[400005];
struct opt{
    int l,r,id;
}stk[400005];
int stkcnt;
int main(){
#ifndef ONLINE_JUDGE
    file("t");
#endif
     n=read();
     blksz=(int)sqrt(n);
     F(i,1,n)belongs[i]=(i-1)/blksz+1;
     blkcnt=belongs[n];
     F(i,1,blkcnt)lst[i]=min((i-1)*blksz+1,n),rst[i]=min(i*blksz,n);
     F(i,1,n)a[i]=read(),b[i]=a[i];
//   F(i,1,n)printf("%d\n",belongs[i]);
 //F(i,1,blkcnt)cout<<lst[i]<<" "<<rst[i]<<endl;
     sort(b+1,b+n+1);
     cnt=unique(b+1,b+n+1)-b-1;
     F(i,1,n)a[i]=lower_bound(b+1,b+cnt+1,a[i])-b;
//   F(i,1,n)cout<<a[i]<<endl;
     m=read();
     F(i,1,m)que[i].l=read(),que[i].r=read(),que[i].id=i;
     sort(que+1,que+m+1);
     F(i,1,n)rs[i]=0,ls[i]=0x3f3f3f3f;
     int st=1,ed=0;
     int ans=0;
     int nowpos=1;
//F(i,1,m)cout<<que[i].id<<" "<<que[i].l<<" "<<que[i].r<<endl;
     F(i,1,blkcnt){
         st=rst[i]+1;
         ed=rst[i];
         ans=0;
         for(;belongs[que[nowpos].l]==i&&nowpos<=m;nowpos++){
        //   cout<<nowpos<<" "<<i<<endl;
               if(belongs[que[nowpos].l]==belongs[que[nowpos].r]){
                   F(j,que[nowpos].l,que[nowpos].r){
                   bf[a[j]]=0;
               }
               F(j,que[nowpos].l,que[nowpos].r){
```

```
                    if(!bf[a[j]])bf[a[j]]=j;
                    chkmax(Ans[que[nowpos].id],j-bf[a[j]]);
                }
                F(j,que[nowpos].l,que[nowpos].r){
                    bf[a[j]]=0;
                }
            }
                else{
                  while(ed<que[nowpos].r){
                        ed++;
                        stk[++stkcnt].l=ls[a[ed]];
                        stk[stkcnt].r=rs[a[ed]];
                        stk[stkcnt].id=ed;
                        chkmin(ls[a[ed]],ed);
                        chkmax(rs[a[ed]],ed);
                        chkmax(ans,rs[a[ed]]-ls[a[ed]]);
                  }
                  int lstans=ans;
                  while(st>que[nowpos].l){
                        st--;
                        stk[++stkcnt].l=ls[a[st]];
                        stk[stkcnt].r=rs[a[st]];
                        stk[stkcnt].id=st;
                        chkmin(ls[a[st]],st);
                        chkmax(rs[a[st]],st);
                        chkmax(ans,rs[a[st]]-ls[a[st]]);
                  }
                  Ans[que[nowpos].id]=ans;
                   while(st<=rst[i]){
                        ls[a[st]]=stk[stkcnt].l;
                        rs[a[st]]=stk[stkcnt].r;
                        stkcnt--;
                        st++;
                  }
                   ans=lstans;
                }
          }
          while(ed>=st){
              //cout<<ed<<" "<<stk[stkcnt].id<<endl;
             ls[a[ed]]=stk[stkcnt].l;
             rs[a[ed]]=stk[stkcnt].r;
             stkcnt--;
             ed--;
          }
      }
      F(i,1,m)printf("%d\n",Ans[i]);
      return 0;
}
```

8.kd_tree

```
int n,m,rt,d;
struct node{
    int d[2],minn[2],maxn[2],l,r,sz;
    int & operator[](int x){return d[x];}
    node(int x=0,int y=0){
        l=0,r=0;
        d[0]=x;
        d[1]=y;
    }
}p[1000005];
const double alpha=0.75;
int stk[1000005];
int newnode(){
    if(stk[0])return stk[stk[0]--];
    else return ++n;
}
bool operator<(node a,node b){
   return a[d]<b[d];
}
int dis(node a,node b){
    return abs(a[0]-b[0])+abs(a[1]-b[1]);
}
struct kdtree{
   int ans;
   node t[1000005],T;
   void update(int k){
      int l=t[k].l,r=t[k].r;
      F(i,0,1){
          t[k].minn[i]=t[k].maxn[i]=t[k][i];
         if(l){
             chkmin(t[k].minn[i],t[l].minn[i]);
              chkmax(t[k].maxn[i],t[l].maxn[i]);
         }
         if(r){
            chkmin(t[k].minn[i],t[r].minn[i]);
            chkmax(t[k].maxn[i],t[r].maxn[i]);
         }
      }
      t[k].sz=t[l].sz+t[r].sz+1;
   }
#define mid ((l+r)>>1)
   int built(int l,int r,int now){
       if(l>r)return 0;
       int  k=newnode();
       d=now;
       nth_element(p+l,p+mid,p+r+1);
       t[k].d[1]=p[mid].d[1];
       t[k].d[0]=p[mid].d[0];
          t[k].l=built(l,mid-1,now^1);
          t[k].r=built(mid+1,r,now^1);
          update(k);
          return k;
   }
   int getans(int k,node ps){
```

```
      int tmp=0;
      F(i,0,1){
         tmp+=max(0,t[k].minn[i]-ps[i]);
      }
      F(i,0,1){
        tmp+=max(0,ps[i]-t[k].maxn[i]);
      }
      return tmp;
}
void pushdown(int k,int sum){
   if(t[k].l)pushdown(t[k].l,sum);
   p[sum+t[t[k].l].sz+1].d[1]=t[k].d[1];
   p[sum+t[t[k].l].sz+1].d[0]=t[k].d[0];
   stk[++stk[0]]=k;
   if(t[k].r)pushdown(t[k].r,sum+t[t[k].l].sz+1);
}
void insert(int &k,int now){
  if(!k){
     k=newnode();
     t[k].d[1]=T.d[1];
     t[k].d[0]=T.d[0];
     t[k].l=t[k].r=0;
     update(k);
     return ;
  }
  if(t[k][now]<T.d[now])insert(t[k].r,now^1);
  else insert(t[k].l,now^1);
  update(k);
  if(alpha*t[k].sz<t[t[k].l].sz||alpha*t[k].sz<t[t[k].r].sz){
  pushdown(k,0);
  k=built(1,t[k].sz,now);
}
}
void query(int k){
   int d,dl=2147483647,dr=2147483647;
   d=dis(t[k],T);
   ans=min(ans,d);
   if(t[k].l)dl=getans(t[k].l,T);
   if(t[k].r)dr=getans(t[k].r,T);
   if(dl<dr){
      if(dl<ans)query(t[k].l);
      if(dr<ans)query(t[k].r);
   }
   else{
      if(dr<ans)query(t[k].r);
      if(dl<ans)query(t[k].l);
   }
}
int queryans(node ps){
  ans=2147483647;
  T=ps;
  query(rt);
  return ans;
}
void insertnode(node p){
   T=p;
   insert(rt,0);
}
```

```
}Tree;
int main () {
#ifndef ONLINE_JUDGE
file("4169");
#endif
    n=read();
    m=read();
    F(i,1,n)p[i][0]=read(),p[i][1]=read();
    rt=Tree.built(1,n,0);
    while(m--){
       int opt=read(),x=read(),y=read();
       if(opt==1)Tree.insertnode(node(x,y));
       else printf("%d\n",Tree.queryans(node(x,y)));
    }
    return 0;
}
```

9.fhq_treap

```
int n;
struct node{
  int ra,sum,sz,ls,rs;
}tree[500005];
int tot,rt;
int tmp;
void built(int &x,int xx){
    tree[x=++tot].ra=rand()<<15|rand();
    tree[x].sum=xx;
    tree[x].sz=1;
}
void pushup(int x)
{if(!x)return;tree[x].sz=tree[tree[x].ls].sz+tree[tree[x].rs].sz+1;}
void merge(int &x,int l,int r){
   if(!l||!r)x=l+r;
   else if(tree[l].ra<tree[r].ra)x=l,merge(tree[x].rs,tree[x].rs,r),pushup(x);
   else x=r,merge(tree[x].ls,l,tree[x].ls),pushup(x);
}
void spilt(int x,int &l,int &r,int k){
      if(!k)l=0,r=x;
      else if(k==tree[x].sz)l=x,r=0;
      else
if(k<=tree[tree[x].ls].sz)r=x,spilt(tree[x].ls,l,tree[x].ls,k),pushup(x);
      else l=x,spilt(tree[x].rs,tree[x].rs,r,k-tree[tree[x].ls].sz-1),pushup(x);
}
int ran(int x,int w){
  if(!x)return 0;
  if(tree[x].sum>=w)return ran(tree[x].ls,w);
  else return ran(tree[x].rs,w)+tree[tree[x].ls].sz+1;
}
void insert(int xx){
    int x,y,rk=ran(rt,xx);
    spilt(rt,x,y,rk);
    built(tmp,xx);
    merge(x,x,tmp);
    merge(rt,x,y);
}
void del(int xx){
   int x,y,z,rk=ran(rt,xx)+1;
   spilt(rt,x,y,rk);
   spilt(x,x,z,rk-1);
   merge(rt,x,y);
}
int find(int xx){
   int x,y,z,ans;
   spilt(rt,x,y,xx);
   spilt(x,z,x,xx-1);
   ans=tree[x].sum;
   merge(x,z,x);
   merge(rt,x,y);
   return ans;
}
int pre(int xx){
   int x,y,z,ans,rk=ran(rt,xx);
```

```
   spilt(rt,x,y,rk);
   spilt(x,z,x,rk-1);
   ans=tree[x].sum;
   merge(x,z,x);
   merge(rt,x,y);
   return ans;
}
int nxt(int xx){
    int x,y,z,ans,rk=ran(rt,xx+1);
    spilt(rt,x,y,rk+1);
    spilt(x,z,x,rk);
    ans=tree[x].sum;
    merge(x,z,x);
    merge(rt,x,y);
    return ans;
}
int main () {
#ifndef ONLINE_JUDGE
file("fhqtreap");
#endif
    srand(19260817);
    n=read();
    tree[0].ra=tree[0].sum=1e9+7;
    while(n--){
        int opt=read(),x=read();
        if(opt==1)insert(x);
        else if(opt==2)del(x);
        else if(opt==3)printf("%d\n",ran(rt,x)+1);
        else if(opt==4)printf("%d\n",find(x));
        else if(opt==5)printf("%d\n",pre(x));
        else printf("%d\n",nxt(x));
    }
    return 0;
}
```

10.带修第k大

```
int n,m;
int cnt;
int lowbit(int x){return x&(-x);}
int root[40000005],ls[40000005],rs[40000005],sz[40000005];
#define mid ((l+r)/2)
void update(int &rt,int lst,int l,int r,int pos,int val){
      rt=++cnt;
      sz[rt]=sz[lst]+val;
      if(l==r)return ;
      if(pos<=mid){
         rs[rt]=rs[lst];
         update(ls[rt],ls[lst],l,mid,pos,val);
      }
      else{
         ls[rt]=ls[lst];
         update(rs[rt],rs[lst],mid+1,r,pos,val);
      }
}
struct node{
   int opt,x,y,z;
}q[200005];
int Size;
int a[400005],b[400005];
int tl,tr;
int rig[4000005],lef[4000005];
int query(int l,int r,int k){
    if(l==r)return l;
    static int sum;
    sum=0;
    F(i,1,tl)sum-=sz[rs[lef[i]]];
    F(i,1,tr)sum+=sz[rs[rig[i]]];
    if(sum<k){
        F(i,1,tl)lef[i]=ls[lef[i]];
        F(i,1,tr)rig[i]=ls[rig[i]];
        return query(l,mid,k-sum);
    }
    else{
        F(i,1,tl)lef[i]=rs[lef[i]];
        F(i,1,tr)rig[i]=rs[rig[i]];
        return query(mid+1,r,k);
    }
}
void add(int x,int val){
    int k=lower_bound(b+1,b+Size+1,a[x])-b;
    while(x<=n){
       update(root[x],root[x],1,Size,k,val);
       x+=lowbit(x);
    }
}
int main () {
#ifndef ONLINE_JUDGE
file("z");
#endif
   int t=read();
```

```
    while(t--){
        F(i,1,cnt)root[i]=ls[i]=rs[i]=sz[i]=0;
        F(i,1,tl)lef[i]=0;
        F(i,1,tr)rig[i]=0;
        cnt=0;
        n=read();
        m=read();
        F(i,1,n)b[i]=a[i]=read();
        Size=n;
        F(i,1,m){
                q[i].opt=read();
                if(q[i].opt==2){
                q[i].x=read(),q[i].y=read(),q[i].z=read();
            }
            else{
                q[i].x=read(),q[i].y=read();
                b[++Size]=q[i].y;
            }
        }
    sort(b+1,b+Size+1);
    Size=unique(b+1,b+Size+1)-(b+1);
    F(i,1,n)add(i,1);
    F(i,1,m){
       if(q[i].opt==2){
           tl=tr=0;
           int l=q[i].x-1,r=q[i].y;
           while(l){
              lef[++tl]=root[l];
              l-=lowbit(l);
           }
           while(r){
               rig[++tr]=root[r];
               r-=lowbit(r);
           }
           printf("%d\n",b[query(1,Size,q[i].z)]);
       }
        if(q[i].opt==1){
                add(q[i].x,-1);
                a[q[i].x]=q[i].y;
                add(q[i].x,1);
            }
        }
    }
    return 0;
}
```

11.splay

```
int ch[3][N], sz[N], nc;
int *ls = ch[0], *rs = ch[1], *fa = ch[2];
int val[N];

int id(int x) { return ch[1][fa[x]] == x; }
bool isr(int x) { return !fa[x]; }

int gn(int v) {
    int p = ++nc;
    ls[p] = rs[p] = fa[p] = 0;
    sz[p] = 1; val[p] = v;
    //  ...
    return p;
}

void update(int x, ...) {
    //  ...
}

void push_up(int x) {
    sz[x] = sz[ls[x]] + sz[rs[x]] + 1;
    //  ...
}

void push_down(int x) {
    if (ls[x]) update(ls[x], ...);
    if (rs[x]) update(rs[x], ...);
    //  ...
}

void rot(int x) {
    int y = fa[x], z = fa[y], o = id(x), w = ch[!o][x];
    if (!isr(y)) ch[id(y)][z] = x; fa[x] = z;
    ch[o][y] = w; if (w) fa[w] = y;
    ch[!o][x] = y; fa[y] = x;
    push_up(y); push_up(x);
}

void splay(int x) {
    for (int y; !isr(x); rot(x))
        if(!isr(y=fa[x])) rot(id(x)^id(y)?x:y);
}

int build(const vector<int>& v, int lb, int rb) {
    if (lb == rb) return 0;
    int mid = (lb + rb) >> 1;
    int x = gn(v[mid]);
    ls[x] = build(v, lb, mid); if (ls[x]) fa[ls[x]] = x;
    rs[x] = build(v, mid + 1, rb); if (rs[x]) fa[rs[x]] = x;
    push_up(x);
    return x;
}

void find_kth(int& r, int k) {
```

```
    int x = r;
    while (1) {
        push_down(x); int cnt = sz[ls[x]];
        if (cnt >= k) x = ls[x];
        else if (cnt == k - 1) break;
        else x = rs[x], k -= cnt + 1;
    }
    splay(x); r = x;
}

void insert_kth(int& r, int k, int v) {
    int* p = &r, x;
    while (*p) {
        push_down(x = *p); int cnt = sz[ls[x]];
        if (k <= cnt + 1) p = &ls[x];
        else p = &rs[x], k -= cnt + 1;
    }
    int y = *p = gn(v); fa[y] = x;
    splay(y); r = y;
}

void erase(int& r) {
    int x = r; push_down(x);
    if (ls[x]) {
        int y = ls[x]; push_down(y);
        while (rs[y]) push_down(y = rs[y]);
        splay(y); r = y;
    }
    if (!fa[x]) {
        r = rs[x]; if (rs[x]) fa[rs[x]] = 0;
    }
    else {
        int w = rs[x], y = fa[x];
        ch[id(x)][y] = w; if (w) fa[w] = y;
        do push_up(y); while (!isr(y = fa[y]));
    }
}

int find_range(int& r, int lb, int rb) {
    assert(lb <= rb);
    int n = sz[r], x = r;
    if (lb != 1 && rb != n) {
        find_kth(r, rb + 1); int rp = r;
        find_kth(r, lb - 1); int lp = r;
        if (fa[rp] != lp) rot(rp); x = ls[rs[r]];
    }
    else if (rb != n) find_kth(r, rb + 1), x = ls[r];
    else if (lb != 1) find_kth(r, lb - 1), x = rs[r];
    return x;
}

void modify_range(int& r, int lb, int rb, ...) {
    int x = find_range(r, lb, rb); update(x, ...);
    while (!isr(x)) push_up(x = fa[x]);
}

void insert_range(int& r, int x, int k) {
    if (!r) r = x;
```

```
    else {
        int n = sz[r];
        if (k == 1) find_kth(r, 1), ls[r] = x, push_up(fa[x] = r);
        else if(k == n + 1) find_kth(r, n), rs[r] = x, push_up(fa[x] = r);
        else {
            find_kth(r, k); find_kth(r, k - 1);
            ls[rs[r]] = x; push_up(fa[x] = rs[r]); push_up(r);
        }
    }
}

void erase_range(int& r, int lb, int rb) {
    int x = find_range(r, lb, rb);
    if (!isr(x)) {
        ch[id(x)][fa[x]] = 0;
        while (!isr(x)) push_up(x = fa[x]);
    }
    else r = 0;
}


bool find_lwr(int& r, int v) {
    int x = r, y = -1;
    while (x) { if (val[x] >= v) y = x; x = ch[val[x] < v][x]; }
    if (y != -1) { splay(y), r = y; return true; }
    else return false;
}

bool find_upr(int& r, int v) {
    int x = r, y = -1;
    while (x) { if (val[x] > v) y = x; x = ch[val[x] <= v][x]; }
    if (y != -1) { splay(y), r = y; return true; }
    else return false;
}

void insert(int& r, int v) {
    int* p = &r, x;
    while (*p) push_down(x = *p), p = &ch[val[x] < v][x];
    int y = *p = gn(v); fa[y] = x; splay(y); r = y;
}
```

12.块状链表

```
#include <cctype>
#include <cstdio>
#include <cstring>
using namespace std;
static const int sqn = 1e3;
struct node {  //定义块状链表
  node* nxt;
  int size;
  char d[(sqn << 1) + 5];
  node() { size = 0, nxt = NULL; }
  void pb(char c) { d[size++] = c; }
}* head = NULL;
char inits[(int)1e6 + 5];
int llen, q;
void readch(char& ch) {  //读入字符
  do
    ch = getchar();
  while (!isalpha(ch));
}
void check(node* p) {  //判断，记得要分裂
  if (p->size >= (sqn << 1)) {
    node* q = new node;
    for (int i = sqn; i < p->size; i++) q->pb(p->d[i]);
    p->size = sqn, q->nxt = p->nxt, p->nxt = q;
  }
}
void insert(char c, int pos) {  //元素插入，借助链表来理解
  node* p = head;
  int tot, cnt;
  if (pos > llen++) {
    while (p->nxt != NULL) p = p->nxt;
    p->pb(c), check(p);
    return;
  }
  for (tot = head->size; p != NULL && tot < pos; p = p->nxt, tot += p->size)
    ;
  tot -= p->size, cnt = pos - tot - 1;
  for (int i = p->size - 1; i >= cnt; i--) p->d[i + 1] = p->d[i];
  p->d[cnt] = c, p->size++;
  check(p);
}
char query(int pos) {  //查询
  node* p;
  int tot, cnt;
  for (p = head, tot = head->size; p != NULL && tot < pos;
       p = p->nxt, tot += p->size)
    ;
  tot -= p->size;
  return p->d[pos - tot - 1];
}
int main() {
  scanf("%s %d", inits, &q), llen = strlen(inits);
  node* p = new node;
  head = p;
```

```
  for (int i = 0; i < llen; i++) {
    if (i % sqn == 0 && i) p->nxt = new node, p = p->nxt;
    p->pb(inits[i]);
  }
  char a;
  int k;
  while (q--) {
    readch(a);
    if (a == 'Q')
      scanf("%d", &k), printf("%c\n", query(k));
    else
      readch(a), scanf("%d", &k), insert(a, k);
  }
  return 0;
}
```

13。可持久化trie

```
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 600010;
int n, q, a[maxn], s[maxn], l, r, x;
char op;
struct Trie {
  int cnt, rt[maxn], ch[maxn * 33][2], val[maxn * 33];
  void insert(int o, int lst, int v) {
    for (int i = 28; i >= 0; i--) {
      val[o] = val[lst] + 1;  // 在原版本的基础上更新
      if ((v & (1 << i)) == 0) {
        if (!ch[o][0]) ch[o][0] = ++cnt;
        ch[o][1] = ch[lst][1];
        o = ch[o][0];
        lst = ch[lst][0];
      } else {
        if (!ch[o][1]) ch[o][1] = ++cnt;
        ch[o][0] = ch[lst][0];
        o = ch[o][1];
        lst = ch[lst][1];
      }
    }
    val[o] = val[lst] + 1;
    // printf("%d\n",o);
  }
  int query(int o1, int o2, int v) {
    int ret = 0;
    for (int i = 28; i >= 0; i--) {
      // printf("%d %d %d\n",o1,o2,val[o1]-val[o2]);
      int t = ((v & (1 << i)) ? 1 : 0);
      if (val[ch[o1][!t]] - val[ch[o2][!t]])
        ret += (1 << i), o1 = ch[o1][!t],
                         o2 = ch[o2][!t];  // 尽量向不同的地方跳
      else
        o1 = ch[o1][t], o2 = ch[o2][t];
    }
    return ret;
  }
} st;
int main() {
  scanf("%d%d", &n, &q);
  for (int i = 1; i <= n; i++) scanf("%d", a + i), s[i] = s[i - 1] ^ a[i];
  for (int i = 1; i <= n; i++)
    st.rt[i] = ++st.cnt, st.insert(st.rt[i], st.rt[i - 1], s[i]);
  while (q--) {
    scanf(" %c", &op);
    if (op == 'A') {
      n++;
      scanf("%d", a + n);
      s[n] = s[n - 1] ^ a[n];
      st.rt[n] = ++st.cnt;
      st.insert(st.rt[n], st.rt[n - 1], s[n]);
```

```
    }
    if (op == 'Q') {
      scanf("%d%d%d", &l, &r, &x);
      l--;
      r--;
      if (l == r && l == 0)
        printf("%d\n", s[n] ^ x);  // 记得处理 l=r=1 的情况
      else
        printf("%d\n", st.query(st.rt[r], st.rt[max(l - 1, 0)], x ^ s[n]));
    }
  }
  return 0;
}
```

14.树套树

```
#include <bits/stdc++.h>
#define F(i, l, r) for(int i = (l), _end_ = (int)(r); i <= _end_; ++i)
#define f(i, r, l) for(int i = (r), _end_ = (int)(l); i >= _end_; --i)
#define Set(a, v) memset(a, v, sizeof(a))
#define file(a) freopen(a".in","r",stdin),freopen(a".out","w",stdout)
using namespace std;

bool chkmin(int &a, int b) {return b < a ? a = b, 1 : 0;}
bool chkmax(int &a, int b) {return b > a ? a = b, 1 : 0;}

inline int read() {
 int x = 0, fh = 1; char ch = getchar();
    for (; !isdigit(ch); ch = getchar() ) if (ch == '-') fh = -1;
    for (; isdigit(ch); ch = getchar() ) x = (x<<1) + (x<<3) + (ch ^ '0');
    return x * fh;
}
int n,m;
int cnt;
int lowbit(int x){return x&(-x);}
int root[40000005],ls[40000005],rs[40000005],sz[40000005];
#define mid ((l+r)/2)
void update(int &rt,int lst,int l,int r,int pos,int val){
      rt=++cnt;
      sz[rt]=sz[lst]+val;
      if(l==r)return ;
      if(pos<=mid){
         rs[rt]=rs[lst];
         update(ls[rt],ls[lst],l,mid,pos,val);
      }
      else{
         ls[rt]=ls[lst];
         update(rs[rt],rs[lst],mid+1,r,pos,val);
      }
}
struct node{
   int opt,x,y,z;
}q[200005];
int Size;
int a[400005],b[400005];
int tl,tr;
int rig[4000005],lef[4000005];
int query(int l,int r,int k){
    if(l==r)return l;
    static int sum;
    sum=0;
    F(i,1,tl)sum-=sz[rs[lef[i]]];
    F(i,1,tr)sum+=sz[rs[rig[i]]];
    if(sum<k){
        F(i,1,tl)lef[i]=ls[lef[i]];
        F(i,1,tr)rig[i]=ls[rig[i]];
        return query(l,mid,k-sum);
    }
    else{
        F(i,1,tl)lef[i]=rs[lef[i]];
```

```
        F(i,1,tr)rig[i]=rs[rig[i]];
        return query(mid+1,r,k);
    }
}
void add(int x,int val){
    int k=lower_bound(b+1,b+Size+1,a[x])-b;
    while(x<=n){
       update(root[x],root[x],1,Size,k,val);
       x+=lowbit(x);
    }
}
int main () {
#ifndef ONLINE_JUDGE
file("z");
#endif
   int t=read();
    while(t--){
        F(i,1,cnt)root[i]=ls[i]=rs[i]=sz[i]=0;
        F(i,1,tl)lef[i]=0;
        F(i,1,tr)rig[i]=0;
        cnt=0;
        n=read();
        m=read();
        F(i,1,n)b[i]=a[i]=read();
        Size=n;
        F(i,1,m){
                q[i].opt=read();
                if(q[i].opt==2){
                q[i].x=read(),q[i].y=read(),q[i].z=read();
            }
            else{
                q[i].x=read(),q[i].y=read();
                b[++Size]=q[i].y;
            }
        }
    sort(b+1,b+Size+1);
    Size=unique(b+1,b+Size+1)-(b+1);
    F(i,1,n)add(i,1);
    F(i,1,m){
       if(q[i].opt==2){
           tl=tr=0;
           int l=q[i].x-1,r=q[i].y;
           while(l){
              lef[++tl]=root[l];
              l-=lowbit(l);
           }
           while(r){
               rig[++tr]=root[r];
               r-=lowbit(r);
           }
           printf("%d\n",b[query(1,Size,q[i].z)]);
       }
        if(q[i].opt==1){
                add(q[i].x,-1);
                a[q[i].x]=q[i].y;
                add(q[i].x,1);
            }
        }
```

```
    }
    return 0;
}
```