NOI 2021 Reference

Daniel Choo Zhenghao

January 20, 2022

Contents

1 Library Code

1.1 Treap

Refer to Animal Editor

1.2 Articulation Points

```
void dfs(int x){
    int rep=1;
    if(depth[x]==0)rep--;
    for(auto v:V[x]){
        if(depth[v]!=-1){
            low[x]=min(low[x],depth[v]);
            continue;
        }
        depth[v]=low[v]=depth[x]+1;
        dfs(v);
        low[x]=min(low[x],low[v]);
        if(low[v]>=depth[x])++rep;
    }
    ans=max(ans,rep);
}
```

1.3 Bridges

```
void dfs(int x){
    depth[x] = lowlink[x] = a++;
    stk.push(x);
    for (auto it : adjList[x]){
    if(p[it]!=-1){
      if(p[x]!=it)lowlink[x] = min(lowlink[x], lowlink[it]);
      continue;
    }
    p[it] = x;
    dfs(it);
    lowlink[x] = min(lowlink[x], lowlink[it]);
    }
    if (lowlink[x] == depth[x]){
        while (stk.top() != x){
            BCC[stk.top()] = b;
            stk.pop();
        }
        stk.pop();
        BCC[x] = b++;
    }
}
```

1.4 SCC

```
void dfs(int x){
    if(ind[x]!=-1)return;
    ind[x]=low[x]=a;++a;
    cur[x]=1;stk.push(x);
    for(auto v:A[x]){
        if(SCC[v]!=-1)continue;
        if(ind[v]==-1){
            dfs(v);
            low[x]=min(low[x],low[v]);
        }else if(cur[v]){
            low[x]=min(low[x],ind[v]);
        }
```

```
    }
    if(low[x]==ind[x]){
        while(stk.top()!=x){
            cur[stk.top()]=0;
            SCC[stk.top()]=b;
            stk.pop();
        }
        stk.pop();
        SCC[x]=b;
        ++b;
    }
}
```

1.5 Sparse Table

```
struct SparseTable{
  vector<vector<int>> ST;
  int N,K;
  SparseTable(int _N, int A[]): N(_N){
    K = MSB(N);
    ST.resize(K);
    ST[0] = vector<int>(A,A+N);
    for (int k = 1; k < K; ++k){
      for (int i = 0; i+(1<<k) <= N; ++i){
        ST[k].pb(min(ST[k-1][i], ST[k-1][i +(1<<(k-1))]));
      }
    }
  }

  inline int MSB (unsigned int x){return 32-__builtin_clz(x);}

  int query (int x, int y){
    int k = MSB (y-x+1) - 1;
    return min(ST[k][x], ST[k][y-(1<<k)+1]);
  }
}*S;
```

1.6 Fenwicks

Point Update Range Query

```
void update(ll x, ll y, ll v) { // inclusive
    x--;
    for(; y; y-=y&(-y)) fw[y] += v;
    for(; x; x-=x&(-x)) fw[x] -= v;
}

ll query(ll x) {
    ll res = 0;
    for (; x<=N; x+=x&(-x)) res += fw[x];
    return res;
}
```

BSTA Fenwick

```
int query(int x) {
  int res = 0, cur = 0;
  for (int i=19;i>=0;--i){
    if (res + (1<<i) > N)continue;
    if (fw[res+(1<<i)] + cur > x){
    cur += fw[res+(1<<i)];
      continue;
    }
```

```
    res += (1<<i);
  }
  return res;
}
```

Range Update Point query

```
void update(ll x, ll v) {
    for (; x<=N; x+=x&(-x)) fw[x] += v;
}

ll sum(ll x) {
    ll res = 0;
    for(; x; x-=x&(-x)) res += fw[x];
    return res;
}
```

Range Update Range Query

```
ll fw[100001], fw2[100001];
void update(ll x, ll y, ll v) { //inclusive
    for (ll tx=x; tx <= N; tx += tx&(-tx)) fw[tx] += v, fw2[tx] -= v*(x-1);
    for (ll ty=y+1; ty <= N; ty += ty&(-ty)) fw[ty] -= v, fw2[ty] += v*y;
}
ll sum (ll x) {
    ll res = 0;
    for (ll tx=x; tx; tx -= tx&(-tx)) res += fw[tx]*x + fw2[tx];
    return res;
}
ll range_sum(ll x, ll y) { //inclusive
    return sum(y)-sum(x-1);
}
```

1.7 Lazy Lazy LCA

```
// Use for point update-range min

// left and right (s,e) ranges will be LCA of all queries
// dfs down segment tree until find node where need to split
// binary ascent to find optimal point
// creates at most one new node

struct node{
    ll v;int s,e;
    node *l,*r;
    node(int _s,int _e):s(_s),e(_e){
        v=INF;
        l=r=nullptr;
    }
    void up(int x, ll val){
        // cerr<<"Update "<<x<<" inside "<<s<<' '<<e<<'\n';
        int m=(s+e)/2;
        if(s==e){v=min(v,val);return;}

        if(x<=m){
            if(!l){l=new node(x,x);l->up(x,val);}
            else if(l->s<=x&&l->e>=x)l->up(x,val);
            else{
                node* newl=l;
                node* newr=new node(x,x); newr->up(x,val);
                if(newr->s<newl->s)swap(newl,newr);
                // newl and newr cannot intersect
                l=nullptr;
                int as=s;int ae=e;
```

```
            while(ae>as){
                int am=(as+ae)/2;
                int lct=0;
                if(newl->s<=am)++lct;
                if(newr->s<=am)++lct;
                if(lct==1)break; //done
                if (lct==2)ae=am;
                else as=am+1;
            }
            l = new node(as,ae);
            l->l=newl; l->r=newr;
            l->v=min(newl->v,newr->v);
        }
    }else{
        if(!r){r=new node(x,x);r->up(x,val);}
        else if(r->s<=x&&r->e>=x)r->up(x,val);
        else{
            node* newl=r;
            node* newr=new node(x,x);
            newr->up(x,val);
            if(newr->s<newl->s)swap(newl,newr);
            r=nullptr;
            int as=s;int ae=e;
            while(ae>as){
                int am=(as+ae)/2;
                int lct=0;
                if(newl->s<=am)++lct;
                if(newr->s<=am)++lct;
                if(lct==1)break; //done
                if (lct==2)ae=am;
                else as=am+1;
            }
            r = new node(as,ae);
            r->l=newl;
            r->r=newr;
            r->v=min(newl->v,newr->v);
        }
    }
    v=INF;
    if(l)v=min(v,l->v);
    if(r)v=min(v,r->v);
}
ll ask(int x, int y){
    y=min(y,e);x=max(x,s);
    if(s==x&&e==y)return v;
    if(x>y)return INF;
    ll val=INF;
    if(l)val=min(val,l->ask(x,y));
    if(r)val=min(val,r->ask(x,y));
    return val;
}
}*root;
```

1.8 Convex Hull

1.8.1 Linear

```
struct CH{
    deque<pi> dq;
    double intersect(pi a,pi b){
        return (double)(b.s-a.s)/(a.f-b.f);
    }
    CH(){}
    void ins(pi a){
        while(SZ(dq)>1&&intersect(a,dq.back()) < intersect(a,dq[SZ(dq)-2])){
```

```
            dq.pop_back();
        }
        dq.pb(a);
    }
    ll ask(ll x){
        while(SZ(dq)>1&&dq[0].f*x+dq[0].s < dq[1].f*x+dq[1].s)dq.pop_front();
        return dq[0].f*x+dq[0].s;
    }
}*root;
```

1.8.2 General

```
struct ch{
    vector<pi> dq;
    ch(){}
    double intersect(pi a,pi b){
        return (double)(b.s-a.s)/(a.f-b.f);
    }
    void ins(pi x){
        while(SZ(dq)>1){
            if(intersect(dq[SZ(dq)-1],x) < intersect(dq[SZ(dq)-2],x))dq.pop_back();
            else break;
        }
        dq.pb(x);
    }
    ll ask(ll x){
        if(SZ(dq)==0)return -INF;
        int s=0;
        int e=SZ(dq)-1;
        while(e>s){
            int m=(s+e)/2;
            ll y=eval(dq[m],x);
            ll z=eval(dq[m+1],x);
            if(z>y)s=m+1;
            else e=m;
        }
        return max(eval(dq[e],x),eval(dq[s],x));
    }
};
```

1.8.3 Li Chao

```
struct node{
    pi line;
    node *l, *r;
    node(){
        line=mp(0,-INF);
        l=r=0;
    }
    void add_line(int s,int e, pi x){
        ll m=(s+e)/2;
        bool lef=ft(x,s)>ft(line,s);
        bool mid=ft(x,m)>ft(line,m);
        if(mid)swap(x,line);
        if(s==e)return;
        else if(lef!=mid){
            if(!l)l=new node();
            l->add_line(s,m,x);
        }else{
            if(!r)r=new node();
            r->add_line(m+1,e,x);
        }
    }
```

```
    ll get(int s,int e,int x){
        ll m=(s+e)/2;
        if(s==e)return ft(line,x);
        else if (x<m){
            if(!l)return ft(line,x);
            return max(ft(line,x),l->get(s,m,x));
        }
        else {
            if(!r)return ft(line,x);
            return max(ft(line,x),r->get(m+1,e,x));
        }
    }
}*root;
```

1.9 HLD

Generation

```
int p[100100], depth[100100], heavy[100100], head[100100], pos[100100];
int dfs(int x){
  int size = 1;
  int mchild = 0;
  for (auto i : adjList[x]){
    if (i == p[x])continue;
    p[i] = x;
    depth[i] = depth[x] + 1;
    int cs = dfs(i);
    size += cs;
    if (cs > mchild){
      mchild = cs;
      heavy[x] = i;
    }
  }
  return size;
}

void decompose(int x, int h){
  head[x] = h; pos[x] = cur++;
  if (heavy[x] != -1)decompose(heavy[x], h);
  for (auto i : adjList[x]){
    if (i == p[x])continue;
    if (i == heavy[x])continue;
    decompose(i,i);
  }
}
memset(heavy,-1,sizeof(heavy));
cur = 1;
dfs(1);
decompose(1,1);
```

Querying

```
if (depth[a] > depth[b])swap(a,b);
for (;head[a] != head[b]; b = p[head[b]]){
  if (depth[head[a]] > depth[head[b]])swap(a,b);
  update(pos[head[b]], pos[b], 1);
}
if (depth[a] > depth[b])swap(a,b);
update(pos[a]+1, pos[b], 1);
```

Binary Search on path from root

```
ll ask(ll x){
  vpi V;
  V.clear();
```

```
  while (x > 1){
    V.pb(mp(pos[head[x]], pos[x]));
    x = p[head[x]];
  }
  reverse(ALL(V)); // V has a length less than log(N)
  for (auto i : V){
    int cur = query(i.f,i.s);
    if (cur){ // Means that that edge has a positive so the rest of the branches are unimportant
      int lower = i.f;
      int upper = i.s;
     while (lower + 1 < upper){
        int m = (lower+upper)/2;
        int x = query(i.f, m);
       if (query(i.f, m))upper = m;
        else lower = m;
      }
      if (query(i.f, lower))return rev[lower]; // rev is an array to reverse the pos array
      return rev[upper];
    }
  }
  return -1;
}
```

1.10 Centroid

```
struct CentDecomp{
  vector<vi> V;
  vi par;
  vi sub;
  vi ban;
  int dst[100100][18];
  int N,c;

  CentDecomp(int _N){
    N = _N;
    sub.resize(N+1,0);
    par.resize(N+1,0);
    ban.resize(N+1,-1);
    V.resize(N+1);
  }

  void build(int u, int p, int l){
    if(l)dst[u][l-1] = 1;
    int n = dfs1(u,p,l); //Find the size of each subtree
    int cent = dfs2(u, p, n); // Find the Centroid
    if (p == -1)p = cent; // Parent of root is urself
    par[cent] = p;
    ban[cent] = l;
    for (auto v : V[cent]){
      if (ban[v]!=-1)continue;
      build(v, cent,l+1);
    }
  }

  int dfs1(int u, int p, int l){
    sub[u] = 1;
    for (auto v : V[u]){
      if (ban[v]!=-1)continue;
      if (v == p)continue;
      if (l)dst[v][l-1] = dst[u][l-1] + 1;
      sub[u] += dfs1(v,u,l);
    }
    return sub[u];
  }
```

```
  int dfs2(int u, int p, int n){
    for (auto v : V[u]){
      if (ban[v]!=-1)continue;
      if (v != p && sub[v] > n/2){
        ++c;
        return dfs2(v,u,n);
      }
    }
    return u;
  }
}*G;
```

1.11 MCBM

1.11.1 Normal MCBM

```
struct AugPath {
    int A, B;  //size of left, right groups
    vector<vector<int> > G; //size A
    vector<bool> visited; //size A
    vector<int> P;        //size B

    AugPath(int _A, int _B): A(_A), B(_B), G(_A), P(_B, -1) {}

    void AddEdge(int a, int b) { //a from left, b from right
        G[a].push_back(b);
    }

    bool Aug(int x) {
        if (visited[x]) return 0;
        visited[x] = 1;
        /* Greedy heuristic */
        for (auto it:G[x]) {
            if (P[it] == -1) {
                P[it] = x;
                return 1;
            }
        }
        for (auto it:G[x]) {
            if (Aug(P[it])) {
                P[it] = x;
                return 1;
            }
        }
        return 0;
    }
    int MCBM() {
        int matchings = 0;
        for (int i = 0; i < A; ++i) {
            visited.resize(A, 0);
            matchings += Aug(i);
            visited.clear();
        }
        return matchings;
    }
    vpi GetMatchings() {
        vpi matchings;
        for (int i = 0; i < B; ++i) {
            if (P[i] != -1) matchings.emplace_back(P[i], i);
        }
        return matchings;
    }
}*root;
```

1.11.2 DAG MCBM

Assuming transitive closure, DAG MIS is equal to the MVC of corresponding bipartite graph A, A' where edges (x, y) are drawn directly as (A_x, A'_y)

```
for(int k=1;k<=N;++k)for(int i=1;i<=N;++i)for(int j=1;j<=N;++j){
    if(A[i][k]&&A[k][j])A[i][j]=1;
}
root=new AugPath(N,N);
for(int i=1;i<=N;++i)for(int j=1;j<=N;++j){
    if(A[i][j]){
        root->AddEdge(i-1,j-1);
    }
}
```

1.12 Mo's Algorithm

```
bool cmp(pii a, pii b){
  if (a.s.f / BSIZ == b.s.f / BSIZ)return a.s.s < b.s.s;
  return a.s.f / BSIZ < b.s.f / BSIZ;
}
```

1.13 Trie

```
struct trie{
    trie* children[26];
    int tot;

    trie(){
        for (int i=0;i<26;++i)children[i] = nullptr;
        tot = 0;
    }

    trie* insert(int ind, int ptr){
        if (ptr == SZ(A[ind])){
            tot++;
            return this;
        }else{
            int t = A[ind][ptr] - 'a';
            if (children[t] == nullptr)children[t] = new trie();
            return children[t] -> insert(ind, ptr+1);
        }
    }

    void recalc(){
        for (int i=0;i<26;++i){
            if (children[i] != nullptr){
                children[i]->recalc();
                tot += children[i]->tot;
            }
        }
        ans += c2(tot);
    }
};
```

1.14 PBDS (Stolen from Rar)

Basically a STL set or map with two extra functions below:

- `find_by_order(x)` returns an iterator pointing to the xth element in the set (0-indexed).
- `order_of_key(k)` gives you the order of the key k as if it was inserted into the set (0-indexed).

```cpp
#include <bits/stdc++.h>
#include <bits/extc++.h>
using namespace std;
using namespace __gnu_pbds;

template <typename T>
using pbds_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;

template <typename K, typename V>
using pbds_map = tree<K, V, less<K>, rb_tree_tag, tree_order_statistics_node_update>;

int main() {
    pbds_set<int> s;
    s.insert(7);
    s.insert(2);
    s.insert(3);
    s.insert(7);  //deduplicated
    printf("%d\n", *s.find_by_order(1));    //prints 3, order is 0-indexed
    printf("%d\n", s.order_of_key(7));      //prints 2, order is 0-indexed

    pbds_map<string, int> m;
    m["Rar"] = 3;
    m["Jacq"] = 4;
    m["Cat"] = 7;
    m["Dinosaur"] = 9;
    printf("%s %d\n", m.find_by_order(1)->first.c_str(), m.find_by_order(1)->second); //prints
        Dinosaur 9
    printf("%d\n", m.order_of_key("Jacq"));                                          //prints 2
}
```

2 Bad Imple stuff

2.1 Cake run (Amortisied range query)

```
int N;
vi des;
int A[MAXN];
vector<ppp> queries;
int Q,a,b,c,d;
map<int, pi> M;
set<pi> S[MAXN*2];

int main(){
    cin>>N;
    for (int i=1;i<=N;++i){
        cin>>A[i];
        des.pb(A[i]);
    }
    cin>>Q;
    for (int i=1;i<=Q;++i){
        cin>>a>>b>>c>>d;
        if (a ==2)des.pb(d);
        queries.pb(mp(a,b), mp(c,d));
    }
    sort(all(des));
    des.resize(unique(all(des)) - des.begin());
    for (int i=1;i<=N;++i)A[i] = lb(all(des),A[i])-des.begin();
    for (int i=0;i<Q;++i)if (queries[i].f.f==2)queries[i].s.s = lb(all(des),
        queries[i].s.s)-des.begin();
    pip cur = mp(1, mp(1,A[1]));
    for (int i=2;i<=N;++i){
        if (cur.s.s == A[i]){
            ++cur.s.f;
        }else{
            M[cur.f] = cur.s;
            cur = mp(i, mp(i, A[i]));
        }
    }

    for (int i=0;i<sz(des);++i){S[i].insert(mp(0,0)); S[i].insert(mp(MAXN,MAXN));}
    M[cur.f] = cur.s;
    for (auto v:M){
        S[v.s.s].insert(mp(v.f, v.s.f));
    }
    M[0]=mp(0,0);
    M[MAXN]=mp(MAXN,MAXN);

    for (auto i : queries){
        if (i.f.f == 1){
            int x=lb(all(des), i.s.s)-des.begin();
            // cout<<"Query "<<x<<'\n';
            // for (auto i : S[x])cout<<i.f<<' '<<i.s<<'\n';
            if (des[x] != i.s.s){cout<<"-1\n";continue;}
            auto y = (--S[x].lb(mp(i.f.s,0)));
            pi t = *y;
            // cout<<t.f<<' '<<t.s<<'\n';
            if (t.f <= i.f.s && t.s >= i.f.s){cout<<i.f.s<<'\n';continue;}
            ++y;
            t = *y;
            if (t.f > i.s.f){cout<<-1<<'\n';continue;}
            cout<<t.f<<'\n';

        }else{

            while (sz(M) > 2){
```

```
            pip cur = *(M.lb(i.f.s));
            if (cur.f > i.s.f){
                // cout<<"B\n";
                break; // smallest range is outside target
            }
            if (cur.s.f > i.s.f){ // If end after
                M.erase(cur.f);
                M[i.s.f+1] = mp(cur.s.f, cur.s.s);
                S[cur.s.s].erase(mp(cur.f, cur.s.f));
                S[cur.s.s].insert(mp(i.s.f+1, cur.s.f));
                break;
            }
            M.erase(cur.f);
            S[cur.s.s].erase(mp(cur.f, cur.s.f));
            // for (auto i : S[cur.s.s])cout<<i.f<<' '<<i.s<<'\n';
        }

        pip cur = *(--M.lb(i.f.s));
        if (cur.s.f >= i.f.s && cur.f < i.f.s){
            M[cur.f] = mp(i.f.s-1, cur.s.s);
            S[cur.s.s].erase(mp(cur.f,cur.s.f));
            S[cur.s.s].insert(mp(cur.f,i.f.s-1));
            if (i.s.f < cur.s.f){
                M[i.s.f + 1] = mp(cur.s.f, cur.s.s);
                S[cur.s.s].insert(mp(i.s.f+1, cur.s.f));
            }
        }

        M[i.f.s] = mp(i.s.f,i.s.s);
        S[i.s.s].insert(mp(i.f.s, i.s.f));
    }
  }
}
```

2.2 Invasion (HLD with BSTA)

```
int p[300100], depth[300100], heavy[300100], head[300100], pos[300100], rpos[300100];
vi adjList[300100];
int N,M,a,b;
int cur;

int fw[300100];
void update(int x, int y, int v) { // incluse
    x--;
    for(; y; y-=y&(-y)) fw[y] += v;
    for(; x; x-=x&(-x)) fw[x] -= v;
}

int query(int x) {
    if(x==0)return INF;
    int res = 0;
    for (; x<=N; x+=x&(-x)) res += fw[x];
    return res;
}

int dfs(int x){
  int size = 1;
  int mchild = 0;
  for (auto i : adjList[x]){
    if (i == p[x])continue;
    p[i] = x;
    depth[i] = depth[x] + 1;
    int cs = dfs(i);
    size += cs;
    if (cs > mchild){
```

```
      mchild = cs;
      heavy[x] = i;
    }
  }
  return size;
}

void decompose(int x, int h){
  head[x] = h; pos[x] = cur++;
  if (heavy[x] != -1)decompose(heavy[x], h);
  for (auto i : adjList[x]){
    if (i == p[x])continue;
    if (i == heavy[x])continue;
    decompose(i,i);
  }
}

int wgt, ans, curtop, lows, prevtop, lastadded;

int find_top(int x){
  // for (int i=1;i<=N;++i)cout<<query(i)<<' ';cout<<'\n';
  for (;x;x=p[head[x]]){
    // cout << "CURRENT " << x << ' ' << query(pos[p[head[x]]]) << '\n';
    if (query(pos[head[x]]) != lows){
      if (query(pos[p[head[x]]]) == lows){
        return p[head[x]];
      }
      continue;
    }
    int low = pos[head[x]];
    int upp = pos[x];
    // cout<<low<<' '<<upp<<'\n';
    while(upp-low > 1){
      int mid = (low+upp)/2;
      if (query(mid) == lows)low = mid;
      else upp = mid;
    }
    if (query(upp) != lows)upp = low;
    // cout << "Query " << upp << '\n';
    return rpos[upp];
  }
  return -INF;
}

void add(int x){
  ++lows;
  lastadded = x;
  if (!query(pos[x])){
    for (int y=x;y;y=p[head[y]]){
      if(!query(pos[head[y]])){
        wgt += (pos[y] - pos[head[y]] + 1);
        if (query(pos[p[head[y]]])){
          break;
        }
        continue;
      }
      int low = pos[head[y]];
      int upp = pos[y];
      // cout << low << ' ' << upp << '\n';
      // for (int i=1;i<=N;++i)cout<<query(i)<<' ';cout<<'\n';
      while(upp-low > 1){
        int mid = (low+upp)/2;
        if (query(mid))low = mid;
        else upp = mid;
      }
      if (query(upp))upp = low;
      wgt += (pos[y] - upp + 1);
```

```
      // cout << "Stop adding at " << upp << '\n';
      break;
    }
  }
  // cout << "Weight is now " << wgt << '\n';
  for(int y=x;y;y=p[head[y]]){
    // cout<<x<<'\n';
    update(pos[head[y]], pos[y], 1);
    // cout << "Increase " << pos[head[y]] << ' ' << pos[y] << '\n';
  }
  int top = find_top(x);
  wgt += (depth[prevtop] - depth[top]);
  prevtop = top;
  // cout << wgt << '\n';
}

void remove(int x){
  --lows;
  for(int y=x;y;y=p[head[y]]){
    update(pos[head[y]], pos[y], -1);
    // cout << "Decrease " << pos[head[y]] << ' ' << pos[y] << '\n';
  }
  if (query(pos[x]) == 0)for (int y=x;y;y=p[head[y]]){
    if(!query(pos[head[y]])){
      wgt -= (pos[y] - pos[head[y]] + 1);
      // cout<<"GG\n";
      // cout << "REMOVE " << (pos[y] - pos[head[y]] + 1) << '\n';
      if (query(pos[p[head[y]]])){
        break;
      }
      continue;
    }
    int low = pos[head[y]];
    int upp = pos[y];
    // cout<<low<<' '<<upp<<'\n';
    while(upp-low > 1){
      int mid = (low+upp)/2;
      if (query(mid))low = mid;
      else upp = mid;
    }
    if (query(upp))upp = low;
    wgt -= (pos[y] - upp + 1);
    // cout << "Stop adding at " << upp << '(' << y << ')' << '\n';
    // cout << "REMOVE " << pos[y] - upp + 1<< '\n';
    break;
  }
  int top = find_top(lastadded);
  // cout << "TOP " << top <<' ' << depth[top]<< ' ' << depth[prevtop] << '\n';
  wgt += (depth[prevtop] - depth[top]);
  prevtop = top;
  // cout << wgt << '\n';
}

int main(){
  ios_base::sync_with_stdio(0);cin.tie(0);
  cin>>N>>M;
  for (int i=0;i<N-1;++i){
    cin>>a>>b;
    ++a;++b;
    adjList[a].pb(b);
    adjList[b].pb(a);
  }
  memset(heavy,-1,sizeof(heavy));
  cur = 1;
  dfs(1);
  decompose(1,1);
  int startind = 1;
```

```
  for (int i=1;i<=N;++i)rpos[pos[i]] = i;
  for (int i=1;i<=N;++i){
    add(i);
    // cout<<"Added " << i << " weight is now " << wgt << '\n';
    while(wgt > M){
      remove(startind);
      // cout << "Removed " << startind << " weight is now " << wgt << '\n';
      ++startind;
    }
    ans = max(ans, i-startind+1);
  }
  cout<<ans;
}
```

2.3 RMI Colours (UFDS with reverse)

```
struct DSU{
    int p[MAXN];
    int sz[MAXN];
    stack<pi> stk;
    int N;
    int par(int x){return (p[x]==x)?x:par(p[x]);}
    DSU(){
        for(int i=1;i<MAXN;++i){
            p[i]=i;
            sz[i]=1;
        }
    }
    void merge(int a,int b){
        // cerr<<"Merge "<<a<<' '<<b<<'\n';
        a=par(a);b=par(b);
        if(a==b){
            stk.push(mp(-1,-1));
            return;
        }
        if(sz[a]<sz[b])swap(a,b);
        p[b]=a;
        sz[a]=sz[a]+sz[b];
        stk.push(mp(b, sz[b]));// merging b into a
    }
    void reverse(){
        // cerr<<"Split "<<edges.back().f<<' '<<edges.back().s<<'\n';
        pi x=stk.top();stk.pop();
        if(x.f==-1)return;
        int b=x.f;
        int a=p[b];
        int bsiz=x.s;
        p[b]=b;
        sz[a]-=bsiz;
    }
}*ufds;

int T,N,E,a,b;
int A[MAXN];
int B[MAXN];
int ans;
vi starts[MAXN];
vi ee[MAXN];
int easy[MAXN];

struct node{
    int s,e,m;
    node *l,*r;
    vpi V;
    node(int _s,int _e):s(_s),e(_e){
```

```
        m=(s+e)/2;
        if(s!=e){l=new node(s,m);r=new node(m+1,e);}
    }
    void addedge(int x,int y,pi z){
        if(s==x&&e==y){
            V.pb(z);return;
        }
        if(y<=m)l->addedge(x,y,z);
        else if(x>m)r->addedge(x,y,z);
        else{
            l->addedge(x,m,z);
            r->addedge(m+1,y,z);
        }
    }
    void solve(int N){
        for(auto i:V){
            ufds->merge(i.f,i.s);
        }
        if(s!=e){
            l->solve(N);
            if(m<N)r->solve(N);
        }else{
            if(SZ(ee[s])){
                for(auto i:starts[s])easy[ufds->par(i)]=1;
                for(auto i:ee[s])if(!easy[ufds->par(i)])ans=0;
                for(auto i:starts[s])easy[ufds->par(i)]=0;
            }
        }
        for(auto i:V)ufds->reverse();
        V.clear();
    }
}*root;

vpi egdeList;

int main(){
    cin>>T;
    ufds=new DSU();
    root=new node(1,MAXN);

    while(T--){
        cin>>N>>E;
        ans=1;
        for(int i=1;i<=N;++i){starts[i].clear();ee[i].clear();}
        for(int i=1;i<=N;++i)cin>>A[i];
        for(int i=1;i<=N;++i){
            cin>>B[i];
            ee[B[i]].pb(i);
            starts[A[i]].pb(i);
            if(B[i]>A[i])ans=0;
        }
        egdeList.clear();
        for(int i=0;i<E;++i){cin>>a>>b;egdeList.pb(a,b);}
        if(!ans){cout<<"0\n";continue;}
        for(auto i:egdeList){
            a=i.f;b=i.s;
            int latestart=max(B[a],B[b]);
            int firstend=min(A[a],A[b]);
            // cerr<<"Edge "<<a<<' '<<b<<" from "<<latestart<<' '<<firstend<<'\n';
            if(latestart>firstend)continue;
            root->addedge(latestart,firstend,mp(a,b));
        }
        root->solve(N);
        cout<<ans<<'\n';
    }
}
```

2.4 COCI Dzumbus (Distribution DP)

```
vi dp1[MAXN]; // off
vi dp2[MAXN]; // on
vi dp3[MAXN]; // island
ll sub[MAXN];
ll A[MAXN];
vi V[MAXN];
ll N,M,a,b;
ll dpx[MAXN][2];
ll dpy[MAXN][2];
ll dpz[MAXN][2];

void dfs(ll x,ll p){
    // cerr<<"Dfs "<<x<<' '<<p<<'\n';
    sub[x]=1;
    if(SZ(V[x])==1&&p!=-1){
        dp1[x].pb(0);dp1[x].pb(INF);
        dp2[x].pb(INF);dp2[x].pb(A[x]);
        dp3[x].pb(INF);dp3[x].pb(INF);
        db(x);
        return;
    }
    vi child;
    for(auto v:V[x])if(v!=p)child.pb(v);
    for(auto i:child){dfs(i,x);sub[x]+=sub[i];}

    // first we dp to find the optimal for OFF
    for(ll i=0;i<=sub[x];++i)dpx[i][0]=dpx[i][1]=INF;
    ll T=0;
    dpx[0][0]=0;
    ll t=1;
    for(auto i:child){
        // cerr<<"Child "<<i<<' '<<sub[i]<<' '<<SZ(dp3[i])<<'\n';
        for(ll i=0;i<=T+sub[i];++i)dpx[i][t]=INF;
        for(ll c=0;c<=sub[i];++c){
            ll val=c;
            ll cst=min({dp1[i][c],dp3[i][c]});
            for(ll i=0;i<=T;++i){
                ll n=i+val;
                ll w=cst+dpx[i][1-t];
                dpx[n][t]=min(w,dpx[n][t]);
            }
        }
        T+=sub[i];
        t=1^t;
    }
    for(ll i=0;i<=T;++i)dp1[x].pb(dpx[i][t^1]);
    dp1[x].pb(INF);

    // dp to find the optimal for ON
    for(ll i=0;i<=sub[x];++i)dpx[i][0]=dpx[i][1]=INF;
    for(ll i=0;i<=sub[x];++i)dpy[i][0]=dpy[i][1]=INF;
    T=0;t=1;
    dpx[0][0]=0; // dpx[0] means not on

    for(auto i:child){
        for(ll i=0;i<=T+sub[i];++i)dpx[i][t]=dpy[i][t]=INF;
        for(ll c=0;c<=sub[i];++c){
            ll val=c;
            ll csa=dp1[i][c];
            ll csb=min(dp2[i][c],dp3[i][c]);
            for(ll i=0;i<=T;++i){ // OFF to OFF
                ll n=i+val;
                ll w=csa+dpx[i][1^t];
                dpx[n][t]=min(w,dpx[n][t]);
```

```
            }
            for(ll i=0;i<=T;++i){ // OFF to ON
                ll n=i+val;
                ll w=csb+dpx[i][1^t];
                dpy[n][t]=min(w,dpy[n][t]);
            }
            for(ll i=0;i<=T;++i){ // ON to ON
                ll n=i+val;
                ll w=min(csa,csb)+dpy[i][1^t];
                // cerr<<i<<" to "<<n<<' '<<dpy[i][1-t]<<' '<<w<<'\n';
                dpy[n][t]=min(w,dpy[n][t]);
            }
        }
        T+=sub[i];
        t=1^t;
    }
    dp2[x].pb(INF);dp3[x].pb(INF);
    for(ll i=0;i<=T;++i)dp3[x].pb(dpy[i][t^1]+A[x]); // ON
    for(ll i=0;i<=T;++i)dp2[x].pb(dpx[i][t^1]+A[x]); // ISLAND
    db(x);
}

ll res[MAXN];

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>N>>M;
    for(ll i=1;i<=N;++i)cin>>A[i];
    for(ll i=0;i<M;++i){
        cin>>a>>b;V[a].pb(b);V[b].pb(a);
    }
    for(ll i=0;i<=N;++i)dpz[i][0]=dpz[i][1]=INF;
    ll T=0;
    dpz[0][0]=0;
    ll t=1;

    for(int x=1;x<=N;++x){
        if(SZ(dp1[x]))continue;

        dfs(x,-1);
        vi tm;

        for(int i=0;i<=sub[x];++i)tm.pb(min(dp1[x][i],dp3[x][i]));
        for(int i=sub[x]-1;i>=0;--i)tm[i]=min(tm[i],tm[i+1]);

        for(ll i=0;i<=T+sub[x];++i)dpz[i][t]=INF;
        for(ll c=0;c<=sub[x];++c){
            ll val=c;
            ll cst=tm[c];
            for(ll i=0;i<=T;++i){
                ll n=i+val;
                ll w=cst+dpz[i][1^t];
                dpz[n][t]=min(w,dpz[n][t]);
            }
        }
        T+=sub[x];
        // cerr<<t<<'\n';
        // for(int i=0;i<=T;++i)cout<<dpz[i][t]<<' ';cout<<'\n';
        t=1^t;
    }
    for(int i=0;i<=N;++i)res[i]=dpz[i][t^1];

    cin>>a;while(a--){
        cin>>b;
        ll t=ub(res,res+N+1,b)-res-1;
        cout<<t<<'\n';
    }
```

```
}
```

2.5 Cake Eats Cake (Treap)

```
struct node {
    int prior,cnt=0,value;
    node *l,*r;
    node() { }
    node(int value) : prior(rand()),value(value), l(NULL), r(NULL) { }
};

int cnt(node* &x){
    if (!x)return 0;
    return x->cnt;
}

void prop(node* &t){}
vi oddres,evenres;

void output(node* &t, int flag){
    if(!t)return;
    prop(t);
    output(t->l,flag);
    if (flag)oddres.pb(t->value);
    else evenres.pb(t->value);
    output(t->r,flag);
}

void upd(node* &t){
    if(!t)return;
    // cerr<<!(t->l)<<' '<<!(t->r)<<'\n';
    t->cnt=1;
    if(t->l)t->cnt+=t->l->cnt;
    if(t->r)t->cnt+=t->r->cnt;
}

void merge(node* &t, node* &l, node* &r){
    prop(l);prop(r);
    if (!l)t=r;
    else if (!r)t=l;
    else if(l->prior>r->prior){
        merge(l->r,l->r,r);
        t=l;
    }else{
        merge(r->l,l,r->l);
        t=r;
    }
    upd(t);
}

void split(node* t, node* &l, node* &r, int key){
    // You want key number of things in the left subtree
    if (!t){
        l=r=NULL;
        return;
    }
    prop(t);
    if (key<=cnt(t->l)){
        split(t->l,l,t->l,key);
        upd(t);
        r=t;
    }else if (key==cnt(t->l)+1){
        r=t->r;
        t->r=NULL;
```

```
        upd(t);
        l=t;
    }else{
        split(t->r,t->r,r,key-1-cnt(t->l));
        upd(t);
        l=t;
    }
}

void insert(node* &t, int pos, int x){
    // Split open, then merge twice to put the new element inside
    node* item = new node(x);
    node* T1=NULL;
    node* T2=NULL;
    split(t,T1,T2,pos);
    merge(T1,T1,item);
    merge(t,T1,T2);
}

node* odds;
node* evens;
int N,C,a,b;
ll K,M;
int A[MAXN];
vpi V;

void specswap(){
    node* T1=NULL;
    node* T2=NULL;
    split(odds,T1,odds,1);
    if (N%2==0){
        split(evens,evens,T2,cnt(evens)-1);
        merge(odds,T2,odds);
        merge(evens,evens,T1);
    }else{
        split(odds,odds,T2,cnt(odds)-1);
        merge(odds,T2,odds);
        merge(odds,odds,T1);
    }
}

void res(int a, int b){
    if (a>b)return;
    node* T1=NULL;
    node* T2=NULL;
    node* T3=NULL;
    node* T4=NULL;
    node* T5=NULL;
    node* T6=NULL;
    // cout<<a<<' '<<b<<'\n';
    int af = (a)/2;
    int ae = (b+1)/2;
    split(odds,odds,T3,ae);
    split(odds,T1,T2,af);
    int bf = (a+1)/2-1;
    int be = (b)/2;
    split(evens,evens,T6,be);
    split(evens,T4,T5,bf);
    merge(T1,T1,T5);
    merge(odds,T1,T3);
    merge(T4,T4,T2);
    merge(evens,T4,T6);
    return;
}

void run(pi x){
    int a=x.f;
```

```
    int b=x.s;
    if (b >= a)res(a,b);
    else{
        int endpt = N;
        if (a%2==N%2){
            res(a,endpt-1);
            res(2,b);
            specswap();
        }
        else{
            res(a,endpt);
            res(1,b);
        }
    }
}

int B[MAXN];
int D[MAXN];

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin>>N>>C>>M>>K;
    for (int i=0;i<C;++i){
        cin>>a;
        A[a]=1;
    }

    for (int i=0;i<M;++i){
        cin>>a>>b;
        V.pb(a,b);
    }

    odds = NULL;
    evens = NULL;

    for (int i=1;i<=N;++i){
        if (i%2==1)insert(odds,cnt(odds),i);
        else insert(evens,cnt(evens),i);
    }
    for(int i=0;i<M;++i){
        run(V[i]);
    }

    output(odds,1);
    output(evens,0);
    for (int i=0;i<SZ(oddres);++i){
        B[2*i+1] = oddres[i];
    }
    for (int i=0;i<SZ(evenres);++i){
        B[2*i+2] = evenres[i];
    }
    // for (int i=1;i<=N;++i)cout<<B[i]<<' ';cout<<'\n';
    ll loop = K/M;
    memset(D,-1,sizeof(D));
    for (int i=1;i<=N;++i)if(D[i]==-1){
        int t=B[i];
        vi v;
        while (t!=i){
            v.pb(t);
            t = B[t];
        }
        v.pb(i);
        for (int j=0;j<SZ(v);++j){
            D[v[j]] = A[v[(j+loop)%SZ(v)]];
        }
        // for (auto x:v)cout<<x<<' ';cout<<'\n';
    }
```

```
    // for (int i=1;i<=N;++i)cout<<D[i]<<' ';cout<<'\n';
    odds = NULL;
    evens = NULL;

    for (int i=1;i<=N;++i){
        if (i%2==1)insert(odds,cnt(odds),D[i]);
        else insert(evens,cnt(evens),D[i]);
    }
    for(int i=0;i<K%M;++i){
        run(V[i]);
    }

    oddres.clear();evenres.clear();
    output(odds,1);
    output(evens,0);
    vi out;
    for (int i=0;i<SZ(oddres);++i)if(oddres[i]){
        out.pb(2*i+1);
    }
    for (int i=0;i<SZ(evenres);++i)if(evenres[i]){
        out.pb(2*i+2);
    }
    sort(ALL(out));
    for(auto t:out)cout<<t<<' ';
}
```

2.6 Werewolf (KRT)

```
vpi edges;
vector<vi> V[2];
vi pre[2];
vi ipre[2];
vi post[2];
vi ord;
int p[MAXN][MAXL][2];

int up[MAXN];
int par(int x){return(up[x]==x)?x:up[x] = par(up[x]);}
int gx;
int N;

void dfs(int x,int pa,int a){
    pre[a][x]=gx;
    ipre[a][gx]=x;
    ++gx;
    p[x][0][a] = pa;
    for(int i=1;i<MAXL;++i){
        if(p[x][i-1][a] == -1)continue;
        int b=p[x][i-1][a];
        p[x][i][a] = p[b][i-1][a];
    }

    for(auto t:V[a][x])if(t!=pa)dfs(t,x,a);
    post[a][x]=gx-1;
}

void generate_tree(int a){
    for(int i=0;i<N;++i){
        up[i]=i;
    }
    pre[a].resize(N,0);
    ipre[a].resize(N,0);
    post[a].resize(N,0);

    V[a].resize(N);
```

```
    gx=0;
    for(auto i:edges){
        if(par(i.f) == par(i.s))continue;
        i.s=par(i.s);
        assert(up[i.f] == i.f);
        if(a==0)assert(i.f>i.s);
        else assert(i.f<i.s);
        V[a][i.f].pb(i.s);
        up[i.s]=par(i.f);
    }
    if(a==0)dfs(N-1,-1,a);
    else dfs(0,-1,a);
}

bool B[MAXN];

typedef pair<pi,pi> pp;
vector<pp> sweep[MAXN]; // V[preA] = {index, postA, preB, postB}

struct node{
    int s,e,m,v;
    node *l,*r;
    node(int _s,int _e):s(_s),e(_e){
        m=(s+e)/2;
        v=1e9;
        if(s!=e){
            l=new node(s,m);r=new node(m+1,e);
        }
    }
    void up(int x,int va){
        if(s==e){v=va;return;}
        if(x<=m)l->up(x,va);
        else r->up(x,va);
        v=min(l->v,r->v);
    }
    int rmin(int x,int y){
        if(s==x&&e==y)return v;
        if(y<=m)return l->rmin(x,y);
        else if (x>m)return r->rmin(x,y);
        return min(l->rmin(x,m),r->rmin(m+1,y));
    }
}*root;

std::vector<int> check_validity(int _N, std::vector<int> X, std::vector<int> Y,
                                std::vector<int> S, std::vector<int> E,
                                std::vector<int> L, std::vector<int> R) {
    N=_N;
    memset(p,-1,sizeof(p));
    for(int i=0;i<SZ(X);++i){
        if(Y[i] > X[i])swap(X[i],Y[i]);
        edges.pb(X[i],Y[i]);
    }
    sort(ALL(edges));
    for(int i=0;i<N;++i)ord.pb(i);
    generate_tree(0);


    for(auto &i:edges)swap(i.f,i.s);
    sort(ALL(edges));reverse(ALL(edges));
    generate_tree(1);
    // for(int i=0;i<N;++i)cout<<pre[1][i]<<' '<<post[1][i]<<'\n';

    int Q = S.size();
    std::vector<int> A(Q);
    for (int i = 0; i < Q; ++i) {
        A[i] = 0;
    }
```

```
    for(int t=0;t<SZ(S);++t){
        int st=S[t];

        for(int i=MAXL-1;i>=0;--i){
            if(p[st][i][1] >= L[t]){
                st=p[st][i][1];
            }
        }

        int en=E[t];
        for(int i=MAXL-1;i>=0;--i){
            if(p[en][i][0] != -1 && p[en][i][0] <= R[t]){
                en=p[en][i][0];
            }
        }
        sweep[pre[1][st]].pb(mp(post[1][st], t), mp(pre[0][en], post[0][en]));
    }
    // In the seg tree pairs are (preB, preA)
    root=new node(0,N-1);
    for(int i=N-1;i>=0;--i){
        int node = ipre[1][i];
        root->up(pre[0][node], pre[1][node]);
        for(auto x:sweep[i]){
            int ind=x.f.s;
            int m=root->rmin(x.s.f,x.s.s);
            if(x.f.f >= m)A[ind]=1;
            // cerr<<"Query "<<ind<<' '<<x.f.f<<' '<<m<<'\n';
        }
    }

    return A;
}
```

2.7 Animal Editor (Treap)

```
vector<char> out;
struct node {
    int prior,cnt=0,value,off=0,rev=0;
    node *l,*r;
    node() { }
    node(int value) : prior(rand()),value(value), l(NULL), r(NULL) { }
};

int cnt(node* &x){
    if (!x)return 0;
    return x->cnt;
}

void prop(node* &t){
    if (!t)return;
    if (t->rev){
        swap(t->l,t->r);
        if (t->l)t->l->rev^=1;
        if(t->r)t->r->rev^=1;
        t->rev=0;
    }
    t->value = (t->value+t->off)%26;
    if (t->l)t->l->off+=t->off;
    if(t->r)t->r->off+=t->off;
    t->off=0;
}

void output(node* &t){
    if(!t)return;
```

```
    prop(t);
    assert(!t->rev);
    output(t->l);
    cout<<(char)('a'+t->value);
    output(t->r);
}

void upd(node* &t){
    if(!t)return;
    t->cnt=1+cnt(t->l)+cnt(t->r);
}

void merge(node* &t, node* l, node* r){
    prop(l);prop(r);
    if (!l)t=r;
    else if (!r)t=l;
    else if(l->prior>r->prior){
        merge(l->r,l->r,r);
        t=l;
    }else{
        merge(r->l,l,r->l);
        t=r;
    }
    upd(t);
}

void split(node* t, node* &l, node* &r, int key){
    // You want key number of things in the left subtree
    if (!t){
        l=r=NULL;
        return;
    }
    prop(t);
    if (key<=cnt(t->l)){
        split(t->l,l,t->l,key);
        upd(t);
        r=t;
    }else if (key==cnt(t->l)+1){
        r=t->r;
        t->r=NULL;
        upd(t);
        l=t;
    }else{
        split(t->r,t->r,r,key-1-cnt(t->l));
        upd(t);
        l=t;
    }
}

void insert(node* &t, int pos, int x){
    node* item = new node(x);
    node* T1=NULL;
    node* T2=NULL;
    split(t,T1,T2,pos);
    merge(T1,T1,item);
    merge(t,T1,T2);
}

void up(node* &t, int x, int y, int v){
    if (!t)return;
    prop(t);
    int l = cnt(t);
    if (x==1&&y==l){t->off=(v+t->off)%26;return;}
    int m = cnt(t->l);
    if (x<=m)up(t->l,x,min(y,m),v);
    if (x<=m+1&&y>=m+1)t->value=(t->value+v)%26;//middle taken
    if (y>m+1)up(t->r,max(1,x-m-1),y-m-1,v);
```

```
}

string A;
int a,b,c,d,N;
vi X;

int main(){
    cin>>A>>N;
    X.pb(0);
    for (auto i:A)X.pb(i-'a');
    node* root = new node(A[0] - 'a');
    for (int i=1;i<SZ(A);++i){
        insert(root, i, A[i]-'a');
    }
    for (int i=0;i<N;++i){
        cin>>a;
        if (a == 2){
            cin>>b>>c>>d;
            d%=26;
            up(root,b,c,d);
        }else if (a==1){
            cin>>b>>c>>d;
            node* t1=NULL;
            node* t2=NULL;
            node* t3=NULL;
            split(root,root,t3,c);
            split(root,t1,t2,b-1);
            merge(root,t1,t3);
            t1=NULL;t3=NULL;
            split(root,t1,t3,d);
            merge(root,t1,t2);
            merge(root,root,t3);
        }else{
            cin>>b>>c;
            node* t1=NULL;
            node* t2=NULL;
            node* t3=NULL;
            split(root,root,t3,c);
            split(root,t1,t2,b-1);
            t2->rev^=1;
            merge(root,t1,t2);
            merge(root,root,t3);
        }
    }
    output(root);cout<<'\n';
}
```

2.8 Prefix Enlightenment (2SAT)

Given K subsets where intersection of 3 subsets is zero. 1 operation can flip state of entire subset. Find minimum number of operations to do first i for $i \leq N$.

```
int p[MAXN];
int flip[MAXN];
pi sizes[MAXN];
vi V[MAXN];
pi A[MAXN];
int N,K,a,b;
int ans;
string S;
ll THROW = 300001;
ll WANT = 300002;

int par(int x){return (x == p[x])?x:p[x] = par(p[x]);}
```

```
int val(int x){
    if (x < THROW)return min(sizes[x].f, sizes[x].s);
    return sizes[x].f;
}

void merge(int a, int b, int f){
    if (par(a) == par(b))return;
    // cout<<"Merging "<<a<<" and "<<b<<'\n';
    // Merging A into B
    if (f == 0){ // No need to flip
        // cout<<"No flip\n";
        ans -= val(a)+val(b);
        p[a] = b;
        for (auto i : V[a]){
            // no change to flip
            V[b].pb(i);
        }
        sizes[b].f += sizes[a].f;
        sizes[b].s += sizes[a].s;
    }else{ // Needs to flip
        // cout<<"Flip\n";
        ans -= val(a)+val(b);
        p[a] = b;
        for (auto i : V[a]){
            // flip
            V[b].pb(i);
            flip[i] ^= 1;
        }
        sizes[b].f += sizes[a].s;
        sizes[b].s += sizes[a].f;
    }
    ans += val(b);
    // cout<<sizes[b].f<<' '<<sizes[b].s<<'\n';
}

int main(){
    cin>>N>>K;
    cin>>S;
    S = "0" + S;
    for (int i=1;i<=K;++i){
        cin>>a;
        for (int j=0;j<a;++j){
            cin>>b;
            if (A[b].f == 0)A[b].f = i;
            else A[b].s = i;
        }
    }
    // for (int i=1;i<=N;++i)cout<<A[i].f<<' '<<A[i].s<<'\n';
    for (int i=1;i<=K;++i){
        p[i] = i;
        sizes[i] = mp(1,0);
        V[i].pb(i);
    }

    p[THROW] = THROW;
    p[WANT] = WANT;

    for (int i=1;i<=N;++i){
        // cout<<"At "<<i<<'\n';
        int a = A[i].f;
        int b = A[i].s;
        // cout<<a<<' '<<b<<'\n';
        if (a == 0 && S[i] == '0'){
            assert(0);
        }
        if (a == 0 && S[i] == '1'){
            cout<<ans<<'\n';
```

```
            continue;
        }
        // cout<<(b == 0)<<'\n';
        if (b == 0 && S[i] == '1'){ // Make sure a is on
            // cout<<"Don't want "<<a<<'\n';
            int aflip = flip[a]^1;
            merge(par(a), THROW, aflip);
            // cout<<flip[a]<<'\n';
            cout<<ans<<'\n';
            continue;
        }
        if (b == 0 && S[i] == '0'){
            // cout<<"Want "<<a<<'\n';
            int aflip = flip[a];
            merge(par(a), WANT, aflip);
            // cout<<flip[a]<<'\n';
            cout<<ans<<'\n';
            continue;
        }
        if (par(a) == par(b)){
            // cout<<"OK\n";
            cout<<ans<<'\n';
            continue;
        }

        if (SZ(V[par(a)]) > SZ(V[par(b)]) && par(b) != WANT && par(b) != THROW)swap(a,b);
        if (par(a) == WANT || par(a) == THROW)swap(a,b);

        int target_same = (S[i] == '0');
        int aflip = flip[a];
        int bflip = flip[b];
        if ((aflip == bflip) == target_same)merge(par(a), par(b), 1);
        else merge(par(a), par(b), 0);
        cout<<ans<<'\n';
    }
}
```

2.9 Frisbee (Slope Trick)

```
pi A[1002000];

// ll dist(pi a, pi b){
  // return abs(a.f - b.f) + abs(a.s - b.s);
// }

map<ll,ll> hori, vert;
ll N, hoffset, voffset;
ll hmin, vmin;

ll findhori(ll x){
  auto above = hori.lb(x);
  ll ans = INF;
  ans = above->s + (above->f - x);
  --above;
  ans = min(ans, above->s + x - above->f);
  // cout<< "Horizontal at " << x << ": " << ans+hoffset << '\n';
  return ans + hoffset;
}

ll findvert(ll x){
  auto above = vert.lb(x);
  ll ans = INF;
  ans = above->s + (above->f - x);
  --above;
  ans = min(ans, above->s + x - above->f);
```

```
  // cout<< "Vertical at " << x << ": " << ans+voffset << '\n';
  return ans + voffset;
}

void hinsert(ll x, ll y){
  // cout<<"Horizontal insert " << x << ' ' << y << '\n';
  y -= hoffset;
  hmin = min(y, hmin);
  hori[x] = y;
  while(1){
    auto above = hori.ub(x);
    if (above->f == INF)break;
    if (above->s > y + above->f - x)hori.erase(above);
    else break;
  }
  while(1){
    auto below = --(hori.lb(x));
    if (below->f == -INF)break;
    if (below->s > y + x - below ->f)hori.erase(below);
    else break;
  }
}

void vinsert(ll x, ll y){
  // cout<<"Vertical insert " << x << ' ' << y << '\n';
  y -= voffset;
  vmin = min(y, vmin);
  vert[x] = y;
  while(1){
    auto above = vert.ub(x);
    if (above->f == INF)break;
    if (above->s > y + above->f - x)vert.erase(above);
    else break;
  }
  while(1){
    auto below = --(vert.lb(x));
    if (below->f == -INF)break;
    if (below->s > y + x - below ->f)vert.erase(below);
    else break;
  }
}


ll h,q;

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    // freopen("input.txt","r",stdin);
    cin >> N;
    for (ll i=1;i<=N;++i)cin>>A[i].f>> A[i].s;
    hori[-INF] = 2*INF;
    vert[-INF] = 2*INF;
    hori[0] = 0;
    vert[0] = 0;
    hori[INF] = 2*INF;
    vert[INF] = 2*INF;

    for (ll i=1;i<=N;++i){
      ll hcost = findhori(A[i].s);
      ll vcost = findvert(A[i].f);
      hoffset = hoffset + abs(A[i].f - A[i-1].f);
      voffset = voffset + abs(A[i].s - A[i-1].s);
      if (hcost < findvert(A[i-1].f)){
        vinsert(A[i-1].f, hcost);
      }
      if (vcost < findhori(A[i-1].s)){
        hinsert(A[i-1].s, vcost);
```

```
      }
      h = findhori(A[i].s);
      q = findvert(A[i].f);
      // cout<<hoffset<<' '<<voffset<<'\n';
    }
    cout<<min(vmin+voffset, hmin+hoffset)<<'\n';
}
```

2.10 Bridge CERC (SQRT Decomposition)

Bridges added and deleted between islands with weight between 0 and 10. Find minimax along path between nodes X and Y in queries.

```
#define BSIZ 1000

struct edge{
    ll st,en,w,beg,stop;
    edge(ll a,ll b,ll c,ll d, ll e): st(a),en(b),w(c),beg(d),stop(e) {}
};

vector<edge> E;
// For each edge we store start, end, wgt, starttime,endtime

ll p[MAXN];
ll par(ll x){return p[x] == x ? x : p[x] = par(p[x]); }

vector<pair<pi,pi>> V,query;
ll N,Q;
ll a,b,c,d;
ll out[MAXN];
unordered_map<ll, ll> edges;
unordered_map<ll, pi> M;

ll edgetoint(pi x){
    return x.f * MAXN + x.s;
}

pi inttoedge(ll x){
    return mp(x/MAXN,x%MAXN);
}

vi aList[MAXN];

stack<ll> st;
bool vis[MAXN];
unordered_map<ll,bool> trash;

void dfs(ll x){
    st.push(x);
    for (auto v : aList[x])if (vis[v] == 0){
        vis[v]=1;
        dfs(v);
    }
}

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    // freopen("in.txt","r",stdin);
    // freopen("o2.txt","w",stdout);
    cin>>N>>Q;
    ll T=0;
    for (ll i=1;i<=Q;++i){
        cin>>a;
        if (a == 0){
            cin>>b>>c>>d;
```

```
        ++b;++c;
        if (b > c)swap(b,c);
        assert(d<10);
        ++d;
        V.pb(mp(mp(a,b), mp(c,d)));
    }else{
        cin>>b>>c;
        ++b;++c;
        if (b > c)swap(b,c);
        V.pb(mp(mp(a,T), mp(b,c)));
        // 2 is a query
        if (a==2)++T;
    }
}

ll NUM_BCK = (Q+BSIZ-1)/BSIZ;
memset(out,-1,sizeof(out));

for (ll i=0;i<NUM_BCK;++i){
    E.clear();
    query.clear();
    // cout<<"\nBucket number "<<i+1<<'\n';
    M.clear();

    for (ll j=0;j<i*BSIZ;++j){
        ll x = j;
        if (V[x].f.f == 0){
            M[edgetoint(mp(V[x].f.s, V[x].s.f))] = mp(V[x].s.s,-1);
        }else if (V[x].f.f == 1){
            pi pp = M[edgetoint(mp(V[x].s.f, V[x].s.s))];
            assert(pp.f!=0&&pp.s!=0);
            M.erase(edgetoint(mp(V[x].s.f, V[x].s.s)));
        }
    }

    for (ll j=0;j<BSIZ && i*BSIZ + j < Q;++j){
        ll x = i*BSIZ + j;
        // Consider the op x.
        if (V[x].f.f == 0){
            // cout<<"Add Edge "<<V[x].f.s<<' '<<V[x].s.f<<'\n';
            M[edgetoint(mp(V[x].f.s, V[x].s.f))] = mp(V[x].s.s,x+1);
        }else if (V[x].f.f == 1){
            ll a=V[x].s.f;
            ll b=V[x].s.s;
            // Delete
            pi p = M[edgetoint(mp(a,b))];
            M.erase(edgetoint(mp(a,b)));
            if(p.s == -1){
                E.pb({a,b,p.f,0,x+1});
            }else{
                E.pb({a,b,p.f,p.s,x+1});
            }
        }else{
            query.pb(mp(mp(V[x].s.f, V[x].s.s), mp(V[x].f.s,x+1)));
        }
    }

    ll b = i*BSIZ;
    ll t = b + BSIZ-1;
    t=min(t,Q-1);
    trash.clear();

    for(int j=t; j>=b;--j){
        ll x = j;
        if (V[x].f.f == 0){
            ll l = edgetoint(mp(V[x].f.s, V[x].s.f));
            if (trash[l])continue;
```

```
            // cout<<x+1<<' '<<Q+1<<'\n';
            trash[l]=1;

            pi t = M[l];
            M.erase(l);
            if (t.s==0)continue; //deleted already
            E.pb({V[x].f.s, V[x].s.f, V[x].s.s, x+1, Q+1});
        }
    }

    // cout<<"Current edges \n";
    // for (auto i : M)cout<<inttoedge(i.f).f<<' '<<inttoedge(i.f).s<<' '<<i.s.f<<'\n';

    // cout<<"Consider edges\n";
    // for (auto i : E)cout<<"Bewteen "<<i.st<<' '<<i.en<<" wgt "<<i.w<<" during "<<i.beg<<'
        '<<i.stop<<'\n';

    // cout<<"Queries \n";
    // for (auto i : query)cout<<"From "<<i.f.f<<' '<<i.f.s<<" write to "<<i.s.f<<" at time
        "<<i.s.s<<'\n';

    for (ll R=1;R<=10;++R){
        for (ll i=1;i<=N;++i)p[i]=i;
        for (auto i : M){
            if (i.s.f > R)continue;
            pi x = inttoedge(i.f);
            p[par(x.f)]=par(x.s);
        }
        // Add all the outside edges first
        for (auto q : query){
            if (out[q.s.f] != -1)continue;
            ll tim = q.s.s;

            for (auto i : E){
                if (i.w > R)continue;
                if (tim > i.stop || tim < i.beg)continue;
                // Add the edge
                ll a=i.st;ll b=i.en;
                a=par(a);b=par(b);
                aList[a].pb(b);aList[b].pb(a);
            }

            // for (ll i=1;i<=N;++i)cout<<par(i)<<' ';cout<<'\n';
            // cout<<"From "<<par(q.f.f)<<' '<<par(q.f.s)<<'\n';
            vis[par(q.f.f)]=1;
            dfs(par(q.f.f));
            if (vis[par(q.f.s)]){
                out[q.s.f] = R;
                // cout<<"SET "<<q.s.f<<' '<<R<<'\n';
            }else if (R==10){
                out[q.s.f]=0;
            }
            for (auto i : E){
                ll a=i.st;ll b=i.en;
                a=par(a);b=par(b);
                aList[a].clear();aList[b].clear();
            }
            while(SZ(st)){vis[st.top()]=0;st.pop();}
        }
    }
}

for (ll i=0;i<T;++i){
    cout<<out[i]-1<<'\n';
}
}
```

2.11 Progression (Just Die) (segment tree)

```
ll N,Q;
ll A[MAXN];
ll D[MAXN];
ll K;

struct node{
    ll s,e,m,len,lz,t,O;
    pi rt,lt,bs; // (len,val)
    node *l,*r;
    node(ll _s,ll _e):s(_s),e(_e){
        m=(s+e)/2;lz=0;len=(e-s+1);O=-INF;
        l=r=nullptr;
        if(s==e){
            lt=mp(1,D[s]);
            rt=mp(1,D[s]);
            bs=mp(1,D[s]);
            t+=D[s];
        }else{
            l=new node(s,m);r=new node(m+1,e);
            build(l,r);
        }
    }
    void build(node* &l, node* r){
        s=l->s;e=r->e;m=(s+e)/2;len=(e-s+1);lz=0;
        // cerr<<"Build "<<s<<' '<<e<<'\n';
        propo();
        if(l)l->propo();
        if(r)r->propo();

        lt=l->lt;
        if(lt.f == l->len && r->lt.s==lt.s){
            lt.f+=r->lt.f;
        }
        rt=r->rt;
        if(rt.f==r->len&&l->rt.s==rt.s){
            rt.f+=l->rt.f;
        }
        bs=max({l->bs,r->bs,lt,rt});
        if(l->rt.s==r->lt.s){
            bs=max(bs, mp(l->rt.f+r->lt.f,l->rt.s));
        }
        t=l->t+r->t;
    }

    void query(node* &X,ll x,ll y){
        propo();
        if(s==x&&e==y){
            if(X==nullptr){
                X=new node(0,0);
                X->s=s;X->e=e;X->m=m;X->len=len;
                X->rt=rt;X->lt=lt;X->bs=bs;
            }else{
                node* Y = new node(0,0);
                Y->build(X,this);
                swap(X,Y);
            }
            return;
        }
        if(y<=m)l->query(X,x,y);
        else if(x>m)r->query(X,x,y);
        else{
            l->query(X,x,m);
            r->query(X,m+1,y);
        }
```

```
    }

    void upd(ll x,ll y,ll c){
        propo();
        if(s==x&&e==y){lz+=c;return;}
        if(y<=m)l->upd(x,y,c);
        else if(x>m)r->upd(x,y,c);
        else{
            l->upd(x,m,c);r->upd(m+1,y,c);
        }
        build(l,r);
    }

    void po(){
        if(O==-INF)return;
        lz=0;
        rt=lt=bs=mp(len,O);
        t=O*len;
        if(s!=e){l->O=O;r->O=O;}
        O=-INF;
    }

    void propo(){
        po();
        rt.s+=lz;lt.s+=lz;bs.s+=lz;
        t+=len*lz;
        if(l){l->po();l->lz+=lz;}
        if(r){r->po();r->lz+=lz;}
        lz=0;
    }

    ll get(ll x){
        propo();
        if(s==e)return t;
        l->propo();r->propo();
        if(x<=m)return l->get(x);
        else return l->t+r->get(x);
    }

    void rs(ll x,ll y,ll v){
        propo();
        if(s==x&&e==y){O=v;return;}
        if(y<=m)l->rs(x,y,v);
        else if(x>m)r->rs(x,y,v);
        else{
            l->rs(x,m,v);
            r->rs(m+1,y,v);
        }
        build(l,r);
    }

    void op(){
        for(ll i=s+1;i<=e;++i)cerr<<get(i-1)<<' ';
        cerr<<'\n';
    }
}*root;

node* X;
ll a,b,c,d,e;

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>N>>Q;
    for(ll i=1;i<=N;++i){
        cin>>A[i];
    }
    for(ll i=0;i<=N;++i)D[i]=A[i+1]-A[i];
```

```
    root=new node(0,N);

    while(Q--){
        cin>>a;
        if(a==1){
            cin>>b>>c>>d>>e;
            root->upd(b-1,b-1,d);
            if(b!=c)root->upd(b,c-1,e);
            // c changes by
            ll l=(c-b)*e+d;
            // cerr<<"Last change by "<<l<<'\n';
            root->upd(c,c,-l);

            // root->op();
        }
        else if(a==2){
            cin>>b>>c>>d>>e;

            ll f2=root->get(c);
            if(b!=c)root->rs(b,c-1,e);

            ll fs=d;
            ll cur=root->get(b-1);
            // root->rs(b-1,b-1,fs);
            root->upd(b-1,b-1,fs-cur);

            cur=root->get(c);
            root->upd(c,c,f2-cur);
            // root->rs(c,c,f2);

            // root->op();
        }
        else{
            cin>>b>>c;
            X=nullptr;
            if(b==c){
                cout<<1<<'\n';
                continue;
            }
            root->query(X,b,c-1);
            cout<<X->bs.f+1<<'\n';
        }
    }
}
```

2.12 Inequality (Well Yes)

```
ll N,a,ans;
ll A[MAXN], B[MAXN];
deque<pi> dq;
pi V[MAXN];
pi V2[MAXN];
ll out[MAXN];
ll coeff[MAXN];
ll left_vals[MAXN];
vi des;

struct node{
    ll s,e,m,v;
    node *l, *r;

    node (ll _s, ll _e): s(_s), e(_e), v(A[s]), m((_s + _e)/2){
        if (s != e){
            l = new node(s,m);
```

```
            r = new node(m+1,e);
            v = max(l->v, r->v);
        }
    }

    ll query(ll x,ll val) {
        if(v<val) return -1;
        if(s==e) return s;
        if(s==x) {
            if(l->v > val) return l->query(x,val);
            else return r->query(m+1,val);
        }
        if(x>m) return r->query(x,val);
        else {
            ll lp=l->query(x,val);
            if(lp != -1) return lp; else return r->query(m+1,val);
        }
    }
}*root;

struct n2{
    ll s,e,m,v,sum,lazy,tot;
    n2 *l, *r;

    n2 (ll _s, ll _e): s(_s), e(_e), v(A[s]), sum(0), tot(0), lazy(0), m((_s + _e)/2){
        if (s != e){
            l = new n2(s,m);
            r = new n2(m+1,e);
            tot = l->tot + r->tot;
            sum = l->sum + r->sum;
        }
    }

    void update_range(ll x, ll y, ll val){
        if (s == x && e == y){lazy = val;return;}
        prop();
        if (y <= m)l->update_range(x,y,val);
        else if (x > m)r->update_range(x,y,val);
        else {l->update_range(x,m,val); r->update_range(m+1,y,val);}
        l->prop(); r->prop();
        tot = l->tot + r->tot;
        sum = l->sum + r->sum;
    }

    void update_coeff(ll x, ll val){
        prop();
        if (s == e){
            sum = val;
            if (lazy != 0)tot = lazy*val;
            else tot = left_vals[s]*val;
            return;
        }
        if (x <= m)l->update_coeff(x,val);
        else r->update_coeff(x,val);
        l->prop();r->prop();
        tot = l->tot + r->tot;
        sum = l->sum + r->sum;
    }

    void prop(){
        if (lazy == 0)return;
        tot = sum * lazy;
        if (s == e)return;
        l->lazy = r-> lazy = lazy;
        r->lazy = lazy;
        lazy = 0;
    }
```

```
}*r2;

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>N;
    for (ll i=1;i<=N;++i)cin>>A[i];
    A[N+1] = 1e9;

    dq.pb(0,1e9);
    for (ll i=1;i<=N;++i){
        while (SZ(dq) > 1 && dq.back().s < A[i]){
            V[dq[SZ(dq) - 1].f] = mp(dq[SZ(dq) - 2].f,i);
            dq.pop_back();
        }
        dq.pb(i,A[i]);
    }

    for (;SZ(dq) > 1;dq.pop_back())V[dq[SZ(dq) - 1].f] = mp(dq[SZ(dq) - 2].f,N+1);

    for (ll i=1;i<=N;++i){
        cin>>B[i];
        B[i] += A[i];
    }

    for (ll i=1;i<=N;++i){
        while (SZ(dq) > 1 && dq.back().s < B[i]){
            V2[dq[SZ(dq) - 1].f] = mp(dq[SZ(dq) - 2].f,i);
            dq.pop_back();
        }
        dq.pb(i,B[i]);
    }

    for (;SZ(dq)>1;dq.pop_back())V2[dq[SZ(dq) - 1].f] = mp(dq[SZ(dq) - 2].f,N+1);

    root = new node(1,N+1);

    for (ll i=1;i<=N;++i){
        ll llen = i - V2[i].f;
        ll final_right = V2[i].s;
        ll original_right = root->query(i+1, B[i]);
        ll original_value = (original_right - i) * llen * B[i];
        ll final_value = (final_right - i) * llen * B[i];
        out[final_right] += final_value - original_value;
        out[i] += original_value;
        out[i] += out[i-1];
    }
    vector<pair<pi, pi>> include_coefficient;

    for (ll i=1;i<=N;++i)des.pb(A[i]);
    sort(ALL(des));
    ll cur = 0;
    for (ll i=1;i<=N;++i){
        ll seg_ind = lb(ALL(des), A[i]) - des.begin() + 1;
        ll rlen = V[i].s - i;
        ll inital_left = V[i].f;
        ll original_value = A[i] * (i - inital_left) * rlen;
        ll changed_value = A[i] * i * rlen;
        cur += original_value;
        left_vals[seg_ind] = inital_left;
        include_coefficient.pb(mp(mp(inital_left, changed_value - original_value), mp(seg_ind, A[i] *
            rlen)));
        pair<pi,pi> t = include_coefficient.back();
        coeff[i] = changed_value;
    }

    sort(ALL(include_coefficient));reverse(ALL(include_coefficient));
    r2 = new n2(1,N);
```

```
    out[0] = cur;

    for (ll i=1;i<=N;++i){
        ll seg_ind = lb(ALL(des), A[i]) - des.begin() + 1;
        while (SZ(include_coefficient) && include_coefficient.back().f.f == i-1){
            pair<pi,pi> t = include_coefficient.back();
            cur += t.f.s;
            r2->update_coeff(t.s.f, t.s.s);
            r2->update_range(t.s.f, t.s.f, t.f.f);
            include_coefficient.pop_back();
        }
        ll update_des = lb(ALL(des), B[i]) - des.begin();
        if (update_des > N)update_des=N;
        if (update_des >= 1)r2->update_range(1,update_des,i);
        r2->update_coeff(seg_ind,0);
        cur -= coeff[i];
        r2->prop(); out[i] += cur - r2->tot;
    }
    for (ll i=0;i<=N;++i)cout<<out[i]<<'\n';
}
```

2.13 Bananafarm (Wavelet)

```
int N,Q;
int A[MAXN];
vi des;

struct node{
    int s,e,m;
    vi values;
    vi leftcnt;
    node *l,*r;
    node(int _s,int _e):s(_s),e(_e){
        m=(s+e)/2;
        l=r=0;
    }
    inline int ask(int x){
        if(x==-1)return 0;
        return leftcnt[x];
    }
    void cons(){
        // cerr<<"Node "<<s<<' '<<e<<' '<<"len "<<SZ(values)<<'\n';
        if(s==e)return;
        int lc=0;
        int rc=0;
        leftcnt.resize(SZ(values));
        for(int i=0;i<SZ(values);++i){
            if(values[i]<=m){++lc;leftcnt[i]=1;}
            else {++rc;leftcnt[i]=0;}
        }
        if(lc){
            l=new node(s,m);
            l->values.resize(lc);
        }
        if(rc){
            r=new node(m+1,e);
            r->values.resize(rc);
        }
        int le=0;
        int re=0;
        for(int i=0;i<SZ(values);++i){
            if(values[i]<=m)l->values[le++]=values[i];
            else r->values[re++]=values[i];
        }
        for(int i=1;i<SZ(leftcnt);++i)leftcnt[i]+=leftcnt[i-1];
        if(l)l->cons();
        if(r)r->cons();
    }
    int query(int x,int y,int k){
        if(s==e)return s;
        // cerr<<"Askin "<<x<<' '<<y<<' '<<k<<'\n';
        // for(auto i:values)cout<<i<<' ';cerr<<'\n';
        // for(auto i:leftcnt)cout<<i<<' ';cerr<<'\n';
        // count left side number in range
        int lft=ask(y)-ask(x-1);
        if(lft>=k){
            // cerr<<"Go left\n";
            // go to the left side
            return l->query(ask(x-1),ask(y)-1,k);
        }else{
            // cerr<<"Go reft\n";
            k-=lft;
            return r->query(x-ask(x-1),y-ask(y),k);
        }
        return -1;
    }
}*root;
```

```
int main(){
    cin>>N>>Q;
    for(int i=1;i<=N;++i)cin>>A[i];
    for(int i=1;i<=N;++i)des.pb(A[i]);
    sort(ALL(des));des.resize(unique(ALL(des))-des.begin());
    for(int i=1;i<=N;++i)A[i]=lb(ALL(des),A[i])-des.begin();

    root = new node(0,SZ(des));
    for(int i=1;i<=N;++i){
        root->values.pb(A[i]);
    }
    root->cons();
    while(Q--){
        int a,b,c;
        cin>>a>>b>>c;
        --a;--b;
        c=b-a-c+2;
        // cerr<<a<<' '<<b<<' '<<c<<'\n';
        cout<<des[root->query(a,b,c)]<<'\n';
    }
}
```