Notebook - Maratona de Programação

Na Base do O(u)

Contents

1 Algoritmos

1.1 Mo

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

#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define int long long
#define ld long double
#define ll long long
#define pb push_back
#define ff first
#define ss second
#define vi vector<int>
#define pii pair<int, int>
#define all(x) x.begin(), x.end()
#define rall(x) x.rbegin(), x.rend()
const int MAXN = 101;
const int INF = INT64_MAX;
const int MOD = 1e9+7;
const int LOG = 60;
const ld PI = acos(-1);

struct QueryMo{

    int l, r, sec, ord;

    QueryMo(int inL, int inR, int inSec, int inOrd){

        l = inL, r = inR, sec = inL / inSec, ord =
    inOrd;
    }

    bool operator<(QueryMo &compa){

        return make_pair(sec, r) < make_pair(compa.sec
    , compa.r);
    }

};
void solve(){

    int n, q; cin >> n >> q;

    vi v(n);
    map<int, int> id;

    for(int i = 0; i < n; i++){

        int x; cin >> x;

        if(!id.count(x)) id[x] = i+1;

        v[i] = id[x];
    }

    vector<QueryMo> auxQueries;

    int rt = min((int)200, (int)(sqrt(q)));

    for(int i = 0; i < q; i++){

        int l, r; cin >> l >> r;

        QueryMo aux = QueryMo(--l, --r, rt, i);

        auxQueries.pb(aux);
    }

    vi resp(q);

    sort(all(auxQueries));

    int actL = 0, actR = 0, ans = 0;
    vector<int> freqs(2e5+2, 0);

    ans++;
    freqs[v[actL]]++;

    for(auto query : auxQueries){
            while(actR < query.r){

                actR++;
                if(!freqs[v[actR]]) ans++;
                freqs[v[actR]]++;
            }

            while(actL > query.l){

                actL--;
                if(!freqs[v[actL]]) ans++;
                freqs[v[actL]]++;
            }

            while(actL < query.l){

                freqs[v[actL]]--;
                if(!freqs[v[actL]]) ans--;
                actL++;
            }

            while(actR > query.r){

                freqs[v[actR]]--;
                if(!freqs[v[actR]]) ans--;
                actR--;
            }


            resp[query.ord] = ans;
    }

    for(int i = 0; i < q; i++) cout << resp[i] << '\n'
    ;

    return;
}
int32_t main(){
    sws;

    int t = 1;
    // cin >> t;

    while(t--)
        solve();

    return 0;
}
```

1.2 Ternary Search

```
// Uma busca em uma curva, avaliando dois pontos
    diferentes
// Complexidade: O(Nlog3N)

double check(vector<int> v, vector<int> t, double x){
  double ans = 0;
  for(int i=0; i<v.size(); i++){
    ans = max(ans, (double)(abs(v[i]-x) + t[i]));
  }
  return ans;
}

int32_t main(){ sws;

  int t; cin>>t;
  while(t--){
    int n; cin>>n;
    vector<int> v(n);
    vector<int> t(n);
    input(v);
    input(t);

    double ans = 0.0;
    double l=0.0, r=1e9;
    while(r-l >= EPS){

      double mid1 = (double) l + (r - l) / 3;
      double mid2 = (double) r - (r - l) / 3;

      double x1 = check(v, t, mid1);
      double x2 = check(v, t, mid2);

      if(x1 < x2){
        r = mid2;
      }else{
        l = mid1;
        ans = l;
      }
    }
    cout<<fixed<<setprecision(7);
    cout<<ans<<endl;
  }
    return 0;
}
```

2 DP

2.1 Dp

```
// DP - Dynamic Programming

#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const int MAX = 110;

int n;
int tab[MAX];
vector<int> v;

ll dp(int i){
    if(i>=n) return 0;
    if(tab[i] != -1) return tab[i];

    int pega = v[i] + dp(i+2);
    int npega = dp(i+1);

    tab[i] = max(pega, npega);
    return tab[i];
}

int main(){
    memset(tab, -1, sizeof(tab));
    cin>>n;

    v.assign(n, 0);

    cout<<dp(0)<<endl;

    return 0;
}
```

2.2 Knapsack

```
int n, t;
int tab[N][N];
bool pegou[N][N];
vector<pair<int,int>> v;

vector<int> resposta;

int dp(int idx, int dias){
    if(idx >= n) return 0;
    if(tab[idx][dias] != -1) return tab[idx][dias];

    int pega=0;
    if(dias+v[idx].first <= t){
        pega = dp(idx+1, dias+v[idx].first)+v[idx].
    second;
    }

    int npega = dp(idx+1, dias);

    if(pega>npega) pegou[idx][dias] = true;

    return tab[idx][dias] = max(pega, npega);
}

int32_t main(){
    memset(tab, -1, sizeof(tab));
    cin>>n>>t;
    for(int i=0; i<n; i++){
        int ti, di;
        cin>>ti>>di;
        v.push_back({ti, di});
    }
    dp(0, 0);
    int i = 0, j =0;
    vector<int> ans;
    // retornar os valores
    while(i < n){
        if(pegou[i][j]){
            j += v[i].first;
            ans.push_back(i+1);
        }
        i++;
    }
    cout<<ans.size()<<endl;
    for(int i=0; i<ans.size(); i++){
        cout<<ans[i]<<" ";
    }

}
```

2.3 Lis

```
// Longest increase sequence
// O(nlogn)
multiset<int> S;
for(int i=0;i<n;i++){
    auto it = S.upper_bound(vet[i]); // upper -
    longest strictly increase sequence
    if(it != S.end())
        S.erase(it);
    S.insert(vet[i]);
}
// size of the lis
int ans = S.size();

// return the elements in LIS
/////////////////////////// see that later
// https://codeforces.com/blog/entry/13225?#comment
    -180208

vi LIS(const vi &elements){
    auto compare = [&](int x, int y) {
        return elements[x] < elements[y];
    };
    set< int, decltype(compare) > S(compare);

    vi previous( elements.size(), -1 );
    for(int i=0; i<int( elements.size() ); ++i){
        auto it = S.insert(i).first;
        if(it != S.begin())
            previous[i] = *prev(it);
        if(*it == i and next(it) != S.end())
            S.erase(next(it));
    }

    vi answer;
    answer.push_back( *S.rbegin() );
    while ( previous[answer.back()] != -1 )
        answer.push_back( previous[answer.back()] );
    reverse( answer.begin(), answer.end() );
    return answer;
}
```

3 ED

3.1 Dsu

```
struct DSU {
    int n;
    vector<int> parent, size;

    DSU(int n): n(n) {
        parent.resize(n, 0);
        size.assign(n, 1);

        for(int i=0;i<n;i++)
            parent[i] = i;
    }

    int find(int a) {
        if(a == parent[a]) return a;
        return parent[a] = find(parent[a]);
    }
    void join(int a, int b) {
        a = find(a); b = find(b);
        if(a != b) {
            if(size[a] < size[b]) swap(a, b);
            parent[b] = a;
            size[a] += size[b];
        }
    }
};
```

3.2 Min Queue

```
struct MinQ {
    stack<pair<ll,ll>> in;
    stack<pair<ll,ll>> out;

    void add(ll val) {
        ll minimum = in.empty() ? val : min(val, in.
    top().ss);
        in.push({val, minimum});
    }

    ll pop() {
        if(out.empty()) {
            while(!in.empty()) {
                ll val = in.top().ff;
                in.pop();
                ll minimum = out.empty() ? val : min(
    val, out.top().ss);
                out.push({val, minimum});
            }
        }
        ll res = out.top().ff;
        out.pop();
        return res;
    }

    ll minn() {
        ll minimum = LLINF;
        if(in.empty() || out.empty())
            minimum = in.empty() ? (ll)out.top().ss :
    (ll)in.top().ss;
        else
            minimum = min((ll)in.top().ss, (ll)out.top
    ().ss);

        return minimum;
    }

    ll size() {
        return in.size() + out.size();
    }
};
```

3.3 Mo

```
//Distinct values queries
#include <bits/stdc++.h>

using namespace std;
//#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define ff first
#define ss second

const long double PI = acos(-1);
int atual = 1;
int freq[200002];
struct Q
{
    int l,r, idx, block;
    Q(int p1, int p2, int i, int b)
    {
        l = p1;
        r = p2;
        idx = i;
        block = l/b;
    }
    bool operator < (Q& query2)
    {
        if(block == query2.block) return r < query2.r;
        return block < query2.block;
    }

};

void add(int x)
{
    if(!freq[x])
    {
        atual++;
    }
    freq[x]++;
}
void rem(int x)
{
    freq[x]--;
    if(!freq[x])
    {
        atual--;
    }
}
void solve()
{
    int n,q;
    cin >> n >> q;
    int b = sqrt(q) + 1;
    b = n/b + 1;
    vector <int> v(n);
    map<int,int> compress;
    vector<Q> queries;
    int aux = 1;
    for(int i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        if(compress.count(x))
        {
            v[i] = compress[x];
        }
        else
        {
            compress[x] = aux;
            v[i] = aux;
            aux++;
        }
    }

    for(int i = 0; i < q; i++)
    {
        int x,y;
        cin >> x >> y;
        queries.pb(Q(x-1,y-1,i,b));
    }

    sort(queries.begin(), queries.end());
    vi ans(q,0);
    int curl = 0, curr = 0;
    freq[v[0]]++;

    for(auto query : queries)
    {
        //cout << query.l << ' ' << query.r << '\n';
        while(curl > query.l)
        {
            curl--;
            add(v[curl]);
        }
        while(curr < query.r)
        {
            curr++;
            add(v[curr]);
        }
        while(curl < query.l)
        {
            rem(v[curl]);
            curl++;
        }
        while(curr > query.r)
        {
            rem(v[curr]);
            curr--;
        }
        ans[query.idx] = atual;
    }

    for(auto resp : ans) cout << resp << '\n';


    return;
}
int32_t main()
{   sws
    int t = 1;
    while (t--)
    {
        solve();
    }

    return 0;
}
```

3.4 Ordered Set

```
// disable define int long long
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template <class T>
  using ord_set = tree<T, null_type, less<T>,
    rb_tree_tag,
  tree_order_statistics_node_update>;

// k-th maior elemento - O(logN) - idx em 0
s.find_by_order(k)

// qtd elementos < k - O(logN)
s.order_of_key(k)

ord_set<int> s;
```

3.5 Prefixsum 2d

```
ll find_sum(vector<vi> &mat, int x1, int y1, int x2,
    int y2){
    // superior-esq(x1,y1) (x2,y2)inferior-dir
    return mat[x2][y2]-mat[x2][y1-1]-mat[x1-1][y2]+mat
    [x1-1][y1-1];
}

int main(){

    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            mat[i][j]+=mat[i-1][j]+mat[i][j-1]-mat[i
    -1][j-1];

}
```

3.6 Rmq

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

// codigo de Range Minimum query; Spt = sparse table.

#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define int long long
#define endl "\n"
#define pb push_back
#define ff first
#define ss second
#define all(x) x.begin(), x.end()
#define rall(x) x.rbegin(), x.rend()
const int MAXN = 1e7+1;
const int INF = INT64_MAX;
const int MOD = 1e9+7;

void solve(){
    int n; cin >> n;

    int aux = n, log = 0;

    while(aux / 2){

        log++;
        aux /= 2;
    }

    vector<vector<int>> spt(n, vector<int> (log+1));
    vector<int> v(n);

    for(int i = 0; i < n; i++) {cin >> v[i]; spt[i][0]
     = v[i];}

    for(int j = 1; j < log+1; j++){
        for(int i = 0; i + (1 << j) -1 < n; i++){

            spt[i][j] = min(spt[i][j-1], spt[i + (1 <<
     j-1)][j-1]);
        }
    }

    int q; cin >> q;

    while(q--){

        int l, r; cin >> l >> r;

        aux = r-l+1; log = 0;

        while(aux / 2){log++; aux /= 2;}

        cout << min(spt[l][log], spt[r - (1 << log) +
    1][log]) << '\n';
    }
    return;
}
int32_t main(){
    sws;

    int t = 1;
    // cin >> t;

    while(t--)
        solve();

    return 0;
}
```

3.7 Segtree Lazy

```
vector<int> v(MAXN), t(4*MAXN), lazy(4*MAXN);

int merge(int x, int y){

    return x + y;
}
void prop(int id, int il, int ir){

    if(!lazy[id]) return;

    if(il != ir){

        lazy[2*id] += lazy[id];
        lazy[2*id+1] += lazy[id];
    }

    t[id] += (ir - il + 1) * lazy[id];
    lazy[id] = 0;

    return;
}

void build(int id, int il, int ir){

    if(il == ir){

        t[id] = v[il];
        return;
    }
    int im = (il + ir) >> 1;

    build(2*id, il, im);
    build(2*id+1, im+1, ir);

    t[id] = merge(t[2*id], t[2*id+1]);

    return;
}

void update(int id, int il, int ir, int l, int r, int
    x){

    prop(id, il, ir);
    if(l <= il && ir <= r){

        lazy[id] += x;
        prop(id, il, ir);
        return;
    }
    if(l > ir || il > r) return;

    int im = (ir+il) >> 1;

    update(2*id, il, im, l, r, x);
    update(2*id+1, im+1, ir, l, r, x);

    t[id] = merge(t[2*id+1], t[2*id]);
}

int query(int id, int il, int ir, int l, int r){

    prop(id, il, ir);
    if(l <= il && ir <= r) return t[id];
    if(l > ir || il > r) return 0;

    int im = (ir+il) >> 1;

    int esq = query(2*id, il, im, l, r);
    int dir = query(2*id+1, im+1, ir, l, r);

    return merge(esq, dir);
}
```

4 Geometria

4.1 Convex Hull

```
#include <bits/stdc++.h>

using namespace std;
#define int long long
typedef int cod;

struct point
{
    cod x,y;
    point(cod x = 0, cod y = 0): x(x), y(y)
    {}

    double modulo()
    {
        return sqrt(x*x + y*y);
    }

    point operator+(point o)
    {
        return point(x+o.x, y+o.y);
    }
    point operator-(point o)
    {
        return point(x - o.x , y - o.y);
    }
    point operator*(cod t)
    {
        return point(x*t, y*t);
    }
    point operator/(cod t)
    {
        return point(x/t, y/t);
    }

    cod operator*(point o)
    {
        return x*o.x + y*o.y;
    }
    cod operator^(point o)
    {
        return x*o.y - y * o.x;
    }
    bool operator<(point o)
    {
        if( x != o.x) return x < o.x;
        return y < o.y;
    }

};

int ccw(point p1, point p2, point p3)
{
    cod cross = (p2-p1) ^ (p3-p1);
    if(cross == 0) return 0;
    else if(cross < 0) return -1;
    else return 1;
}

vector <point> convex_hull(vector<point> p)
{
    sort(p.begin(), p.end());
    vector<point> L,U;

    //Lower
    for(auto pp : p)
    {
        while(L.size() >= 2 and ccw(L[L.size() - 2], L
    .back(), pp) == -1)
        {
            // é -1 pq eu ãno quero excluir os
    colineares
            L.pop_back();
        }
        L.push_back(pp);
    }

    reverse(p.begin(), p.end());

    //Upper
    for(auto pp : p)
    {
        while(U.size() >= 2 and ccw(U[U.size()-2], U .
    back(), pp) == -1)
        {
            U.pop_back();
        }
        U.push_back(pp);
    }

    L.pop_back();
    L.insert(L.end(), U.begin(), U.end()-1);
    return L;
}

cod area(vector<point> v)
{
    int ans = 0;
    int aux = (int)v.size();
    for(int i = 2; i < aux; i++)
    {
        ans += ((v[i] - v[0])^(v[i-1] - v[0]))/2;
    }
    ans = abs(ans);
    return ans;
}

int bound(point p1 , point p2)
{
    return __gcd(abs(p1.x-p2.x), abs(p1.y-p2.y));
}
//teorema de pick [pontos = A - (bound+points)/2 + 1]

int32_t main()
{

    int n;
    cin >> n;

    vector<point> v(n);
    for(int i = 0; i < n; i++)
    {
        cin >> v[i].x >> v[i].y;
    }

    vector <point> ch = convex_hull(v);

    cout << ch.size() <<  '\n';
    for(auto p : ch) cout << p.x << " " << p.y << "\n"
    ;

    return 0;
}
```

4.2 Inside Polygon

```
// Convex O(logn)

bool insideT(point a, point b, point c, point e){
    int x = ccw(a, b, e);
    int y = ccw(b, c, e);
    int z = ccw(c, a, e);
    return !((x==1 or y==1 or z==1) and (x==-1 or y
    ==-1 or z==-1));
}

bool inside(vp &p, point e){ // ccw
    int l=2, r=(int)p.size()-1;
    while(l<r){
        int mid = (l+r)/2;
        if(ccw(p[0], p[mid], e) == 1)
            l=mid+1;
        else{
            r=mid;
        }
    }
    // bordo
    // if(r==(int)p.size()-1 and ccw(p[0], p[r], e)
    ==0) return false;
    // if(r==2 and ccw(p[0], p[1], e)==0) return false
    ;
    // if(ccw(p[r], p[r-1], e)==0) return false;
    return insideT(p[0], p[r-1], p[r], e);
}


// Any O(n)

int inside(vp &p, point pp){
    // 1 - inside / 0 - boundary / -1 - outside
    int n = p.size();
    for(int i=0;i<n;i++){
        int j = (i+1)%n;
        if(line({p[i], p[j]}).inside_seg(pp))
            return 0;
    }
    int inter = 0;
    for(int i=0;i<n;i++){
        int j = (i+1)%n;
        if(p[i].x <= pp.x and pp.x < p[j].x and ccw(p[
    i], p[j], pp)==1)
            inter++; // up
        else if(p[j].x <= pp.x and pp.x < p[i].x and
    ccw(p[i], p[j], pp)==-1)
            inter++; // down
    }

    if(inter%2==0) return -1; // outside
    else return 1; // inside
}
```

4.3 Point Location

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

#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define int long long

#define pb push_back
#define ff first
#define ss second

const int MOD = 1e9+7;
const int MAX = 2e5+1;

int32_t main(){
    sws;

    int t; cin >> t;

    while(t--){

        int x1, y1, x2, y2, x3, y3; cin >> x1 >> y1 >>
     x2 >> y2 >> x3 >> y3;

        int deltax1 = (x1-x2), deltay1 = (y1-y2);

        int compx = (x1-x3), compy = (y1-y3);

        int ans = (deltax1*compy) - (compx*deltay1);

        if(ans == 0){cout << "TOUCH\n"; continue;}
        if(ans < 0){cout << "RIGHT\n"; continue;}
        if(ans > 0){cout << "LEFT\n"; continue;}
    }
    return 0;
}
```

5 Grafos

5.1 Articulation Point

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

#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define endl "\n"
#define int long long
#define ld long double
#define pb push_back
#define ff first
#define ss second
#define all(x) x.begin(), x.end()
#define rall(x) x.rbegin(), x.rend()
const int MAXN = 1e5+1;
const int INF = INT32_MAX;
const int MOD = 998244353;
const int LOG = 18;

vector<bool> vis(MAXN);
vector<vector<int>> g(MAXN);
vector<int> tin(MAXN, -1), low(MAXN, -1);
int t = 0;
set<int> ans;

void AP(int u, int p = -1){

    int qtdfilhos = 0;
    low[u] = tin[u] = t++;
    vis[u] = true;

    for(auto v: g[u]){

        if(v == p) continue;

        if(!vis[v]){

            qtdfilhos++;

            AP(v, u);

            low[u] = min(low[u], low[v]);

            if(low[v] >= tin[u] && u != 1) ans.insert(
    u);
        } else{

            low[u] = min(low[u], tin[v]);
        }
    }

    if(u == 1 && qtdfilhos >= 2) ans.insert(u);
}
```

```
void solve(){

    int n, m; cin >> n >> m;

    for(int i = 0; i < m; i++){

        int u, v; cin >> u >> v;

        g[u].pb(v);
        g[v].pb(u);
    }

    AP(1);

    cout << ans.size() << '\n';

    for(auto x : ans) cout << x << ' ';

    cout << '\n';

    return;
}
int32_t main(){
    sws;
    int t = 1;
    // cin >> t;

    while(t--)
        solve();

    return 0;
}
```

5.2 Bellman Ford

```
#include <bits/stdc++.h>

using namespace std;
#define int long long
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int MAXN = 2e5 + 1;
const int INF = 1e18+1;

vector<int> d(MAXN, -INF), points(MAXN, -1);
vector<bool> vis(MAXN);
vector<vector<int>> g(MAXN);
int n;

struct edge{

    int x, y, c;
};

vector<edge> edges;
void bf(int u){

    d[u] = 0;

    for(int i = 0; i < n - 1; i++){

        for(auto e : edges){

            if(d[e.x] > -INF){
                if(d[e.y] < d[e.x] + e.c){

                    d[e.y] = d[e.x] + e.c;
                }
            }
        }
    }
}

int dfs(int u){

    if(u == n) return points[u] = 1;

    vis[u] = true;

    int aux = -1;
    for(auto v : g[u]){

        if(!vis[v]) aux = dfs(v);
    }

    return points[u] = aux;
}

bool find(){

   for(int i = 0; i <= n - 1; i++){

        for(auto e : edges){

                if(d[e.y] < d[e.x] + e.c){

                    if(points[e.y] == 1) return true;
                    d[e.y] = d[e.x] + e.c;
                }
        }
    }

    return false;
}


int32_t main(){
    sws;

    int m; cin >> n >> m;

    for(int i = 0; i < m; i++){

        edge e; cin >> e.x >> e.y >> e.c;

        g[e.x].push_back(e.y);
        edges.push_back(e);
    }

    bf(1);
    dfs(1);
    if(find()){cout << -1 << '\n'; return 0;}

    cout << d[n] << '\n';
    return 0;
}
```

5.3 Bridgetree

```
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
#define int long long
#define sws ios::sync_with_stdio(false);cin.tie(
    nullptr);
typedef pair<int, int> ii;
#define INF INT64_MAX
const int MAX = 2e5+1;
const int MOD = 1e9+7;
const int LOG = 30;

vector<bool> vis;
vector<int> tin, low, comp;
vector<vector<int>> g(MAX), bt(MAX);
map<pair<int, int>, bool> ponteh;
int time = 0;

void buildBt(int u, int c){

    comp[u] = c;
    vis[u] = true;

    for(auto v : g[u]){

        if(vis[v] || ponteh[{u, v}]) continue;

        buildBt(v, c);
    }
}

void findBridge(int u, int p = -1){

    vis[u] = true;
    tin[u] = low[u] = time++;

    for(auto v : g[u]){
        if(v == p) continue;

        if(vis[v]){

            low[u] = min(low[u], tin[u]);
        } else{

            dfs(v, u);
            low[u] = min(low[u], low[v]);

            if(low[v] > tin[u])
                éPonte(u, v);
        }
    }
}

int32_t main(){
    sws;

    for(auto[u, v]: ponteh){

        if(v){

            bt[comp[u.ff]].pb(comp[u.ss]);
            bt[comp[u.ss]].pb(comp[u.ff]);
        }
    }
    return 0;
}
```

5.4 Dfs Tree

```
int desce[N], sobe[N], vis[N], h[N];
int backedges[N], pai[N];

// backedges[u] = backedges que comecam embaixo de (ou
     =) u e sobem pra cima de u; backedges[u] == 0 =>
    u eh ponte
void dfs(int u, int p) {
    if(vis[u]) return;
    pai[u] = p;
    h[u] = h[p]+1;
    vis[u] = 1;

    for(auto v : g[u]) {
        if(p == v or vis[v]) continue;
        dfs(v, u);
        backedges[u] += backedges[v];
    }
    for(auto v : g[u]) {
        if(h[v] > h[u]+1)
            desce[u]++;
        else if(h[v] < h[u]-1)
            sobe[u]++;
    }
    backedges[u] += sobe[u] - desce[u];
}
```

5.5 Dijktra

```
#define pii pair<int, int>
vector<vector<pii>> g(N);
vector<bool> used(N);
vector<ll> d(N, LLINF);
priority_queue< pii, vector<pii>, greater<pii> > fila;

void dijkstra(int k) {
    d[k] = 0;
    fila.push({0, k});

    while (!fila.empty()) {
        auto [w, u] = fila.top();
        fila.pop();
        if (used[u]) continue;
        used[u] = true;

        for (auto [v, w]: g[u]) {
            if (d[v] > d[u] + w) {
                d[v] = d[u] + w;
                fila.push({d[v], v});
            }
        }
    }
}
```

5.6 Dinic

```
const int N = 300;

struct Dinic {
    struct Edge{
        int from, to; ll flow, cap;
    };
    vector<Edge> edge;

    vector<int> g[N];
    int ne = 0;
    int lvl[N], vis[N], pass;
    int qu[N], px[N], qt;

    ll run(int s, int sink, ll minE) {
        if(s == sink) return minE;

        ll ans = 0;

        for(; px[s] < (int)g[s].size(); px[s]++) {
            int e = g[s][ px[s] ];
            auto &v = edge[e], &rev = edge[e^1];
            if(lvl[v.to] != lvl[s]+1 || v.flow >= v.
    cap)
                continue;           // v.cap - v.flow
    < lim
            ll tmp = run(v.to, sink,min(minE, v.cap-v.
    flow));
            v.flow += tmp, rev.flow -= tmp;
            ans += tmp, minE -= tmp;
            if(minE == 0) break;
        }
        return ans;
    }
    bool bfs(int source, int sink) {
        qt = 0;
        qu[qt++] = source;
        lvl[source] = 1;
        vis[source] = ++pass;
        for(int i = 0; i < qt; i++) {
            int u = qu[i];
            px[u] = 0;
            if(u == sink) return true;
            for(auto& ed : g[u]) {
                auto v = edge[ed];
                if(v.flow >= v.cap || vis[v.to] ==
    pass)
                    continue; // v.cap - v.flow < lim
                vis[v.to] = pass;
                lvl[v.to] = lvl[u]+1;
                qu[qt++] = v.to;
            }
        }
        return false;
    }
    ll flow(int source, int sink) {
        reset_flow();
        ll ans = 0;
        //for(lim = (1LL << 62); lim >= 1; lim /= 2)
        while(bfs(source, sink))
            ans += run(source, sink, LLINF);
        return ans;
    }
    void addEdge(int u, int v, ll c, ll rc) {
        Edge e = {u, v, 0, c};
        edge.pb(e);
        g[u].push_back(ne++);

        e = {v, u, 0, rc};
        edge.pb(e);
        g[v].push_back(ne++);
    }
    void reset_flow() {
        for(int i = 0; i < ne; i++)
            edge[i].flow = 0;
        memset(lvl, 0, sizeof(lvl));
        memset(vis, 0, sizeof(vis));
        memset(qu, 0, sizeof(qu));
        memset(px, 0, sizeof(px));
        qt = 0; pass = 0;
    }
    vector<pair<int, int>> cut() {
        vector<pair<int, int>> cuts;
```

```cpp
        for (auto [from, to, flow, cap]: edge) {
            if (flow == cap and vis[from] == pass and
    vis[to] < pass and cap>0) {
                cuts.pb({from, to});
            }
        }
        return cuts;
    }
};
```

5.7 Floyd

```cpp
// Floyd Warshall

int dist[N][N];

for(int k = 1; k <= n; k++)
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            dist[i][j] = min(dist[i][j], dist[i][k] +
    dist[k][j]);
```

5.8 Ford Fulkerson Isa

```cpp
#include <bits/stdc++.h>

using namespace std;
#define int long long
#define pb push_back

// Description:
// Obtains the maximum possible flow rate given a
    network. A network is a graph with a single source
     vertex and a single sink vertex in which each
    edge has a capacity

// Complexity:
// O(V * E^2) where V is the number of vertex and E is
     the number of edges
const int MAXN = 501;
const int MAXE = 1001;
const int INF = INT64_MAX;

// represents the capacities of the edges
int capacity[MAXN][MAXE];
// represents the graph and it may contain negative
    edges
vector<int> adj[MAXN];
int n, e;


int bfs(int s, int t, vector<int>& parent) {
    fill(parent.begin(), parent.end(), -1);
    parent[s] = -2;
    queue<pair<int, int>> q;
    q.push({s, INF});

    while (!q.empty()) {
        int cur = q.front().first;
        int flow = q.front().second;
        q.pop();

        for (int next : adj[cur])
        {
            //cout << "cur next " << cur << ' ' <<
    next << ' ' << parent[next] << ' ' << capacity[
    cur][next] << endl;
            if (parent[next] == -1 && capacity[cur][
    next])
            {
                parent[next] = cur;
                int new_flow = min(flow, capacity[cur
    ][next]);
                if (next == t)
                {
                    //cout << new_flow << endl;
                    return new_flow;
                }
                q.push({next, new_flow});
            }
        }
    }

    return 0;
}

int maxflow(int s, int t) {
    int flow = 0;
    vector<int> parent(n+1);
    int new_flow;

    while (new_flow = bfs(s, t, parent)) {
        flow += new_flow;
        int cur = t;
        while (cur != s) {
            int prev = parent[cur];
            capacity[prev][cur] -= new_flow;
            capacity[cur][prev] += new_flow;
            cur = prev;
        }
    }

    return flow;
}

int32_t main()
{
    cin>>n>>e;
    int s = 1, t = n;
    //cin>>s>>t;

    for(int i = 0; i < e; i++)
    {
        int from, to, cap;
        cin>>from>>to>>cap;

        capacity[from][to] += cap;
        adj[from].push_back(to);
        //adding the negative edges
        adj[to].push_back(from);
    }

    // for(int i = 1; i <= n; i++)
    // {   cout << i << " : ";
    //     for(auto x : graph[i]) cout << x << ' ';
    //     cout << endl;
    // }

    int maxFlow = maxflow(s, t);

    cout<<maxFlow<<endl;

    return 0;
}
```

5.9 Kosaraju

```cpp
vector<int> g[N], gi[N]; // grafo invertido
int vis[N], comp[N]; // componente conexo de cada
    vertice
stack<int> S;

void dfs(int u){
    vis[u] = 1;
    for(auto v: g[u]) if(!vis[v]) dfs(v);
    S.push(u);
}

void scc(int u, int c){
    vis[u] = 1; comp[u] = c;
    for(auto v: gi[u]) if(!vis[v]) scc(v, c);
}

void kosaraju(int n){
    for(int i=0;i<n;i++) vis[i] = 0;
    for(int i=0;i<n;i++) if(!vis[i]) dfs(i);
    for(int i=0;i<n;i++) vis[i] = 0;
    while(S.size()){
        int u = S.top();
        S.pop();
        if(!vis[u]) scc(u, u);
    }
}
```

5.10 Two Sat

```
#include <bits/stdc++.h>

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define ff first
#define ss second
 const int INF = 0x3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 1004;

int n;
int componente[MAX];
vector<int> adj[MAX];
vector<int> adj2[MAX];
vector<int> saida;
int vis[MAX];
bool ans[MAX];

void dfs(int u)
{
    vis[u] = 1;
    for(auto v : adj[u])
    {
        if(!vis[v])
        {
            dfs(v);
        }
    }
    saida.pb(u);
}

void dfs2(int u, int c)
{
    vis[u] = 2;
    componente[u] = c;
    for(auto v : adj2[u])
    {
        if(vis[v] == 1) dfs2(v, c);
    }
}
void add(int a, bool na, int b, bool nb)
{
    a = 2*(abs(a)-1) ^ na;
    b = 2*(abs(b)-1) ^ nb;
    int neg_a = a ^ 1;
    int neg_b = b ^ 1;
    adj[neg_a].pb(b);
    adj2[b].pb(neg_a);

}
bool possible()
{
    for(int i = 0; i < n; i++)
    {
        if(componente[2*i] == componente[2*i+1])
    return false;
        ans[i] = componente[2*i + 1] < componente[2*i
    ];
    }
    return true;
}
void solve()
{
    cin >> n;
    vector<vector<int>> m(3, vector<int>(n));
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < n; j++)
        {
            int x;
            cin >> x;
            m[i][j] = x;
        }
    }
   for(int i = 0; i < n; i++)
   {
        add(m[0][i], m[0][i] > 0, m[1][i], m[1][i] >
    0);
        add(m[0][i], m[0][i] > 0, m[2][i], m[2][i] >
    0);

        add(m[1][i], m[1][i] > 0, m[0][i], m[0][i] >
    0);
        add(m[1][i], m[1][i] > 0, m[2][i], m[2][i] >
    0);

        add(m[2][i], m[2][i] > 0, m[0][i], m[0][i] >
    0);
        add(m[2][i], m[2][i] > 0, m[1][i], m[1][i] >
    0);
   }

    // for(int i = 0; i < 2*n +2; i++)
    // {
    //     cout << i << ": ";
    //     for(auto x : adj[i])
    //     {
    //         cout << x << " ";
    //     }
    //     cout << endl;
    // }
    //  return;

    for(int i = 0; i < 2*n; i++)
    {
        if(!vis[i])
        {
            dfs(i);
        }
    }

    int c = 0;
    for(int i = saida.size() - 1; i >= 0; i--)
    {
        if(vis[saida[i]] == 1)
        {
            c++;
            dfs2(saida[i], c);
        }
    }

    bool resp = possible();

    cout << (resp? "YES\n" : "NO\n");

    return;
}
int32_t main()
{   sws

    int t;
    //t = 1;
    cin >> t;

    while(t--)
    {
        memset(vis,0,sizeof(vis));
        memset(componente, 0 , sizeof(componente));
        for(int i = 0; i < MAX; i++)adj[i].clear();
        for(int i = 0; i < MAX; i++)adj2[i].clear();
        saida.clear();
        solve();
    }
    return 0;
}
```

6 Math

6.1 Fastexp

```
// recursivo
int fast_exp(int base, int e, int m){
    if(!e) return 1;
    int ans = fast_exp(base * base % m, e/2, m);
    if(e % 2) return base * ans % m;
    else return ans;
}
//iterativo
int fast_exp(int base, int e, int m) {
  int ret = 1;
  while (e) {
    if (e & 1) ret = (ret * base) % m;
    e >>= 1;
    base = (base * base) % m;
  }
  return ret;
}
```

6.2 Fft

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define vvi vector<vector<int>>
#define pb push_back
#define all(x) x.begin(), x.end()
#define endl "\n"
#define ff first
#define ss second
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = INT64_MAX;
const long double PI = acos(-1);
const int MAX = (1e6) + 1;
const int MOD = 998244353;
const int LOG = 30;

using cd = complex<double>;

// FFT (usei na H da mineira de 2024 de contar os
    quadrados)

void fft(vector<cd> &A, bool invert) {
  int N = size(A);

  for (int i = 1, j = 0; i < N; i++) {
    int bit = N >> 1;
    for (; j & bit; bit >>= 1)
      j ^= bit;
    j ^= bit;

    if (i < j)
      swap(A[i], A[j]);
  }

  for (int len = 2; len <= N; len <<= 1) {
    double ang = 2 * PI / len * (invert ? -1 : 1);
    cd wlen(cos(ang), sin(ang));
    for (int i = 0; i < N; i += len) {
      cd w(1);
      for (int j = 0; j < len/2; j++) {
        cd u = A[i+j], v = A[i+j+len/2] * w;
        A[i+j] = u + v;
        A[i+j+len/2] = u-v;
        w *= wlen;
      }
    }
  }

  if (invert) {
    for (auto &x : A)
      x /= N;
  }
}

vector<int> multiply(vector<int> const& A, vector<int>
     const& B) {
  vector<cd> fa(begin(A), end(A)), fb(begin(B), end(B)
    );
  int N = 1;
  while (N < size(A) + size(B))
    N <<= 1;
  fa.resize(N);
  fb.resize(N);

  fft(fa, false);
  fft(fb, false);
  for (int i = 0; i < N; i++)
    fa[i] *= fb[i];
  fft(fa, true);

  vector<int> result(N);
  for (int i = 0; i < N; i++)
    result[i] = round(fa[i].real());
  return result;
}



void solve()
{
  vector<int> A(MAX,0);
  vector<int> B(MAX,0);

  int n;
  cin >> n;

  for(int i = 0; i < n; i++)
  {
    int x;
    cin >> x;
    // A com os expoentes positivos e B com os
    expoentes negativos
    A[x] = 1;
    //A[i] é o coeficiente de z^i
    B[MAX-1-x] = 1;
  }

  // MAX-1 é o novo "0"

  //multiply me da o resultado da çãmultiplicao desses
     dois ôpolinmios
  // C[i] é o coeficiente de x^i

  vector<int> C = multiply(A,B);

  return;
}



int32_t main()
{
    sws

    int t = 1;
    //cin >> t;
    while(t--)
    {
        solve();
    }
    return 0;
}
```

6.3 Inverso Mult

```
// gcd(a, m) = 1 para existir solucao
// ax + my = 1, ou a*x = 1 (mod m)
ll inv(ll a, ll m) { // com gcd
  ll x, y;
  gcd(a, m, x, y);
  return (((x % m) +m) %m);
}

ll inv(ll a, ll phim) { // com phi(m), se m for primo
    entao phi(m) = p-1
  ll e = phim-1;
  return fexp(a, e);
}
```

6.4 Matrix Exp

```
struct Matrix {
    vector<vl> m;
    int r, c;
    Matrix(vector<vl> mat) {
        m = mat;
        r = mat.size();
        c = mat[0].size();
    }

    Matrix(int row, int col, bool ident=false) {
        r = row; c = col;
        m = vector<vl>(r, vl(c, 0));
        if(ident) {
            for(int i = 0; i < min(r, c); i++) {
                m[i][i] = 1;
            }
        }
    }

    Matrix operator*(const Matrix &o) const {
        assert(c == o.r); // garantir que da pra
    multiplicar
        vector<vl> res(r, vl(o.c, 0));

        for(int i = 0; i < r; i++) {
            for(int k = 0; k < c; k++) {
                for(int j = 0; j < o.c; j++) {
                    res[i][j] = (res[i][j] + m[i][k]*o
    .m[k][j]) % MOD;
                }
            }
        }

        return Matrix(res);
    }
};

Matrix fexp(Matrix b, int e, int n) {
    if(e == 0) return Matrix(n, n, true); //
    identidade
    Matrix res = fexp(b, e/2, n);
    res = (res * res);
    if(e%2) res = (res * b);

    return res;
}
```

6.5 Mulmod

```
ll mulmod(ll a, ll b) {
    if(a == 0) {
        return 0LL;
    }
    if(a%2 == 0) {
        ll val = mulmod(a/2, b);
        return (val + val) % MOD;
    }
    else {
        ll val = mulmod((a-1)/2, b);
        val = (val + val) % MOD;
        return (val + b) % MOD;
    }
}
```

6.6 Mult Matriz

```
    for(int i=0; i<n; i++) {
        aux_ab=0, aux_ba=0;
        for (int j=0; j<n; j++){
            aux_ab+= A[i][j]*B[j][i];
            aux_ba+= B[i][j]*A[j][i];
        }
        if (aux_ab!=aux_ba){
            val = false;
            break;
        }
    }
```

7 Misc

7.1 Bitwise

```
// Least significant bit (lsb)
    int lsb(int x) { return x&-x; }
    int lsb(int x) { return __builtin_ctz(x); } // bit
    position
// Most significant bit (msb)
    int msb(int x) { return 32-1-__builtin_clz(x); }
    // bit position

// Power of two
    bool isPowerOfTwo(int x){ return x && (!(x&(x-1)))
    ; }

// floor(log2(x))
int flog2(int x)  { return 32-1-__builtin_clz(x); }
int flog2ll(ll x) { return 64-1-__builtin_clzll(x); }

// Built-in functions
// Number of bits 1
__builtin_popcount()
__builtin_popcountll()

// Number of leading zeros
__builtin_clz()
__builtin_clzll()

// Number of trailing zeros
__builtin_ctz()
__builtin_ctzll()
```

7.2 Template

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

#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define int long long int
#define float long double
#define ld long double
#define ll long long
#define pb push_back
#define ff first
#define ss second
#define vi vector<int>
#define vpii vector<pair<int, int>>
#define vvi vector<vector<int>>
#define pii pair<int, int>
#define all(x) x.begin(), x.end()
#define rall(x) x.rbegin(), x.rend()
#define in(v) for(auto & x : v) cin >> x;
#define out(v) for(auto x : v) cout << x << ' ';
#define tfii tuple<float, int, int>

const int MAXN = 31700;
const int INF = INT64_MAX;
const int MOD = 1e9+7;
const int LOG = 31;
const ld PI = acos(-1);
const int MINF = INT64_MIN;
vpii dirs = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};

void solve(){


    return;
}
int32_t main(){
    sws;

    int t = 1;
    // cin >> t;

    while(t--)
        solve();

    return 0;
}
```

8 QuestoesCSES

8.1 Bracketsequence

```
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
```

```
#define esp  ' '
#define int long long int
#define pii pair<int, int>
#define pb push_back
#define ff first
#define ss second
#define sws ios::sync_with_stdio(false);cin.tie(
    nullptr);cout.tie(nullptr);
const string YES = "YES";
const string NO = "NO";
const int MAX= 2e6+5;
const int MOD= 1e9+7;
const int INF = 0x3f3f3f3f3f3f3f3f;
int fat[MAX], C[MAX];

int fexp(int b, int e){
    if (e==0) return 1;

    int ans = fexp(b, e/2);
    if(e%2) return (((ans*ans)%MOD)*b)%MOD;
    else return (ans*ans)%MOD;
}

void fluminense(){
    int  n; cin >> n;
    int ans = 0;
    if(n%2==1) ans=0;
    else{
        n = n>>1;
        ans = C[n];
    }

    cout << ans <<endl;


}

int32_t main(){
    sws;
    fat[0]=1;
    for(int i=1; i<MAX; i++) fat[i] = (i*fat[i-1])%MOD
    ;
    for(int i=0; i<(MAX>>1)-1; i++){
        C[i] = ((((fat[2*i]*(fexp(fat[i], MOD-2)%MOD))
    %MOD*(fexp(fat[i], MOD-2)%MOD)))%MOD)*(fexp(i+1,
    MOD-2)%MOD);
        C[i]%=MOD;
    }


    int T=1;
    //cin >> T;
    while(T--)fluminense();

}
```

8.2 Editdistance

```
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
#define int long long int
#define sws ios::sync_with_stdio(false);cin.tie(
    nullptr);
typedef pair<int, int> ii;
#define INF INT64_MAX
const int MAX = 5e3+1;

vector<vector<int>> memo(MAX, vector<int> (MAX, -1));
string s, t;

int dp(int i, int j){

    if(i == -1) return j+1;
    if(j == -1) return i+1;

    if(memo[i][j] != -1) return memo[i][j];

    int ins = dp(i-1, j) + 1;
    int del = dp(i, j-1) + 1;
    int mod = dp(i-1, j-1) + (s[i] != t[j]);

    int aux = min(del, mod);
    return memo[i][j] = min(ins, aux);
}
int32_t main(){
    sws;

    cin >> s >> t;

    cout << dp(s.size()-1, t.size()-1) << '\n';
    return 0;
}
```

8.3 Prefixsumqueries

```
#include <bits/stdc++.h>

using namespace std;
#define int long long
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int MAXN = 2e5 + 1;
const int INF = 1e18+1;
vector<int> v(MAXN, 0), t(4*MAXN), lazy(4*MAXN), aux(
    MAXN);

int merge(int x, int y){

    return max(x, y);
}
void prop(int id, int il, int ir){

    if(!lazy[id]) return;

    if(il != ir){

        lazy[2*id] += lazy[id];
        lazy[2*id+1] += lazy[id];
    }

    t[id] += lazy[id];
    lazy[id] = 0;

    return;
}

void build(int id, int il, int ir){

    if(il == ir){

        t[id] = v[il];
        return;
    }

    int im = (il + ir) >> 1;

    build(2*id, il, im);
    build(2*id+1, im+1, ir);

    t[id] = merge(t[2*id], t[2*id+1]);

    return;
}

void update(int id, int il, int ir, int l, int r, int
    x){

    prop(id, il, ir);
    if(l <= il && ir <= r){

        lazy[id] += x;
        prop(id, il, ir);
        return;
    }
    if(l > ir || il > r) return;

    int im = (ir+il) >> 1;

    update(2*id, il, im, l, r, x);
    update(2*id+1, im+1, ir, l, r, x);

    t[id] = merge(t[2*id+1], t[2*id]);
}

int query(int id, int il, int ir, int l, int r){
    prop(id, il, ir);
    if(l <= il && ir <= r) return t[id];
    if(l > ir || il > r) return -INF;

    int im = (ir+il) >> 1;

    int esq = query(2*id, il, im, l, r);
    int dir = query(2*id+1, im+1, ir, l, r);

    return merge(esq, dir);
}

int32_t main(){
    sws;

    int n, q; cin >> n >> q;

    for(int i = 1; i <= n; i++){

        cin >> aux[i];

        v[i] = v[i-1] + aux[i];
    }

    build(1, 0, n);

    while(q--){

        int t, l, r; cin >> t >> l >> r;

        if(t == 2){

            cout << query(1, 0, n, l-1, r) - query(1,
    0, n, l-1, l-1) << '\n';
        } else {

            update(1, 0, n, l, n, r-aux[l]);

            aux[l] = r;
        }
    }

    return 0;
}
```

8.4 Removalgame

```
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
#define int long long int
#define sws ios::sync_with_stdio(false);cin.tie(
    nullptr);
typedef pair<int, int> ii;
#define INF INT64_MAX
const int MAX = 5e3+1;

int memo[MAX][MAX][2];
vector<int> v(MAX);

int dp(int l, int r, bool w){

    if(l > r) return 0;
    if(memo[l][r][w] != -1) return memo[l][r][w];

    if(w){

        int aux = dp(l+1, r, !w);
        int aux1 = dp(l, r-1, !w);
        return memo[l][r][w] = min(aux, aux1);
    } else{

        int aux = dp(l+1, r, !w) + v[l];
        int aux1 = dp(l, r-1, !w) + v[r];
        return memo[l][r][w] = max(aux, aux1);
    }
}
int32_t main(){
    sws;

    int n; cin >> n;

    memset(memo, -1, sizeof(memo));
    for(int i = 0; i < n; i++) cin >> v[i];

    cout << dp(0, n-1, 0) << '\n';

    return 0;
}
```

8.5 Sintaxenextperm

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
int32_t main(){

    string s; cin >> s;

    vector<char> c;

    for(int i = 0; i < s.size(); i++) c.push_back(s[i
    ]);

    set<string> se;
    sort(c.begin(), c.end());

    string resp = "";
    for(int i = 0; i < s.size(); i++) resp += c[i];

    se.insert(resp);

    while(next_permutation(c.begin(), c.end())){

        string resp = "";
        for(int i = 0; i < s.size(); i++) resp += c[i
    ];

        se.insert(resp);
    }

    cout << se.size() << '\n';

    for(auto t: se) cout << t << '\n';
    return 0;
}
```

9 Strings

9.1 Kmp

```
#include <bits/stdc++.h>

using namespace std;
const int MAX = 1e6+1;
string p;
vector<int> nbr(MAX);

int nxt(char c, int n){

    while(n != -1){
        if((n+1) < p.size() && p[n + 1] == c){

            n++;
            break;
        } else {

            n = nbr[n];
        }
    }

    if(n == -1 && p[0] == c) n++;

    return n;
}
void kmp(){

    int n = p.size();

    nbr[0] = -1;

    for(int i = 1; i < n; i++){

        nbr[i] = nbr[i-1];
        nbr[i] = nxt(p[i], nbr[i]);
    }
}
```

```
int main(){

    string s; cin >> s >> p;
    int ans = 0, lider = -1;

    kmp();

    for(int i = 0; i < s.size(); i++){

        lider = nxt(s[i], lider);

        if(lider == p.size()-1) ans++;
    }

    cout << ans;

    return 0;
}
```

9.2 Lcs

```
string LCSubStr(string X, string Y)
{
    int m = X.size();
    int n = Y.size();

    int result = 0, end;
    int len[2][n];
    int currRow = 0;

    for(int i=0;i<=m;i++){
        for(int j=0;j<=n;j++){
            if(i==0 || j==0)
                len[currRow][j] = 0;
            else if(X[i-1] == Y[j-1]){
                len[currRow][j] = len[1-currRow][j-1]
    + 1;
                if(len[currRow][j] > result){
                    result = len[currRow][j];
                    end = i - 1;
                }
            }
            else
                len[currRow][j] = 0;
        }

        currRow = 1 - currRow;
    }

    if(result==0)
        return string();

    return X.substr(end - result + 1, result);
}
```

9.3 Lcs Especial

```
void recover(int i, int j){
     if (i>=s_size || j>=t_size) return ;
     if (s[i]==t[j]){ans.push_back(s[i]);recover(i+1,
    j+1);}
     else if(lcs_size[i+1][j]>lcs_size[i][j+1]) return
     recover(i+1, j);
     else return recover(i, j+1);

 }

 int main(){
     cin >> s >> t;
     s_size = s.size();
     t_size = t.size();


     for(int i=s_size-1; i>=0; i--){
         for(int j =t_size-1;j>=0; j--){
             if(s[i]==t[j]) lcs_size[i][j] = 1+
    lcs_size[i+1][j+1];
             else lcs_size[i][j] = max(lcs_size[i+1][j
    ], lcs_size[i][j+1]);
         }
     }

     recover(0,0);

     cout << ans << endl;

 }
```

9.4 Suffix Array

```
vector<int> suffix_array(string s) {
    s += "$";
    int n = s.size(), N = max(n, 260);
    vector<int> sa(n), ra(n);
    for (int i = 0; i < n; i++) sa[i] = i, ra[i] = s[i
    ];

    for (int k = 0; k < n; k ? k *= 2 : k++) {
        vector<int> nsa(sa), nra(n), cnt(N);

        for (int i = 0; i < n; i++) nsa[i] = (nsa[i]-k
    +n)%n, cnt[ra[i]]++;
        for (int i = 1; i < N; i++) cnt[i] += cnt[i
    -1];
        for (int i = n-1; i+1; i--) sa[--cnt[ra[nsa[i
    ]]]] = nsa[i];

        for (int i = 1, r = 0; i < n; i++) nra[sa[i]]
    = r += ra[sa[i]] !=
            ra[sa[i-1]] or ra[(sa[i]+k)%n] != ra[(sa[i
    -1]+k)%n];
        ra = nra;
        if (ra[sa[n-1]] == n-1) break;
    }
    return vector<int>(sa.begin()+1, sa.end());
}

vector<int> kasai(string s, vector<int> sa) {
    int n = s.size(), k = 0;
    vector<int> ra(n), lcp(n);
    for (int i = 0; i < n; i++) ra[sa[i]] = i;

    for (int i = 0; i < n; i++, k -= !!k) {
        if (ra[i] == n-1) { k = 0; continue; }
        int j = sa[ra[i]+1];
        while (i+k < n and j+k < n and s[i+k] == s[j+k
    ]) k++;
        lcp[ra[i]] = k;
    }
    return lcp;
}


int32_t main(){
    sws;
    string s;
    cin>>s;

    vector<int> suf = suffix_array(s);
    vector<int> lcp = kasai(s, suf);

    ll ans = 0;
    for(int i=0; i<s.size(); i++){
        if(islower(s[suf[i]])){
            int sz = s.size()-suf[i];
            ans += (sz - lcp[i]);
        }
    }
    cout<<ans<<endl;
}
```

9.5 Trie

```
struct Trie{

    int trie[MAX][26];
    bool finish[MAX];
    int nxt = 1, len = 0;

    void add(string s){
        int node = 0;
        for(auto c: s){
            if(trie[node][c-'a'] == 0)
                node = trie[node][c-'a'] = nxt++;
            else
```

```
                node = trie[node][c-'a'];
        }
        if(!finish[node]){
            finish[node] = true;
            len++;
        }
    }

    bool find(string s, bool remove=false){
        int node = 0;
        for(auto c: s)
            if(trie[node][c-'a'] == 0)
                return false;
            else
                node = trie[node][c-'a'];
        if(remove and finish[node]){
            finish[node]=false;
            len--;
        }
        return finish[node];
    }
};
```

9.6 Z Func

```
vector<int> Z(string s) {
    int n = s.size();
    vector<int> z(n);
    int x = 0, y = 0;
    for (int i = 1; i < n; i++) {
        z[i] = max(0, min(z[i - x], y - i + 1));
        while (i + z[i] < n and s[z[i]] == s[i + z[i
    ]]) {
            x = i; y = i + z[i]; z[i]++;
        }
    }
    return z;
}
```

10 Tree

10.1 Binary Lifting

```
vector<int> adj[MAX];
const int LOG = 30;
int up[MAX][LOG], parent[MAX];

void process(int n){
    for(int v=1; v<=n; v++){
        up[v][0]= parent[v];
        for(int i=1; i<LOG; i++){
            up[v][i] = up[ up[v][i-1] ][i-1];
        }
    }
}

int jump(int n, int k){
    for(int i=0; i<LOG; i++){
    if(k & (1 << i)){
      n = up[n][i];
    }
  }
  if(n == 0) return -1;
  return n;
}

int32_t main(){

    int n, q; cin>>n>>q;

    parent[1] = 0;
    for(int i=1; i<=n-1; i++){
        int x;
        cin>>x;
        parent[i+1] = x;

        adj[i+1].pb(x);
        adj[x].pb(i+1);
    }
    process(n);
    for(int i=0; i<q; i++){
        int a, b;
        cin>>a>>b;
        cout<<(jump(a,b))<<endl;
    }
}
```

10.2 Eulertour Segt

```
#include <bits/stdc++.h>

using namespace std;
#define int long long
const int MOD = 1e9+7;

const int MAX = 2e5+1;

vector<int> segt(8*MAX), euler(2*MAX), in(MAX), out(
    MAX), aux(MAX);
int tempo = 0;
vector<int> t[MAX];

void dfs(int u, int p){

    euler[tempo] = u;
    in[u] = tempo;

    tempo++;

    for(auto v : t[u]){

        if(v != p) dfs(v, u);
    }

    euler[tempo] = u;
    out[u] = tempo;

    tempo++;

    return;
}

void build(int id, int il, int ir){

    if(il == ir){

        segt[id] = aux[euler[il]];
        return;
    }

    int im = (il + ir) / 2;

    build(2*id, il, im);
    build(2*id+1, im+1, ir);


    segt[id] = segt[2*id] + segt[2*id+1];
}

void update(int id, int il, int ir, int idx, int x){

    if(il == ir){

        segt[id] = x;
        aux[euler[idx]] = x;
        return;
    }

    int im = (il + ir) / 2;

    if(im < idx){

        update(2*id+1, im+1, ir, idx, x);
    } else {

        update(2*id, il, im, idx, x);
    }

    segt[id] = segt[2*id] + segt[2*id+1];

    return;
}

int query(int id, int il, int ir, int l, int r){

    if(il >= l && ir <= r){
        return segt[id];
    }
    if(l > ir || r < il) return 0;

    int im = (il + ir) / 2;

    int esq = query(2*id, il, im, l, r);
    int dir = query(2*id+1, im+1, ir, l, r);

    return esq + dir;
}

int32_t main(){

    int n, q; cin >> n >> q;

    for(int i = 1; i <= n; i++) cin >> aux[i];

    for(int i = 2; i <= n; i++){

        int u, v; cin >> u >> v;

        t[u].push_back(v);
        t[v].push_back(u);
    }

    dfs(1, 0);

    build(1, 0, 2*(n));

    while(q--){

        int t; cin >> t;

        if(t == 1){

            int v, p; cin >> p >> v;

            update(1, 0, 2*(n), in[p], v);
            update(1, 0, 2*(n), out[p], v);
        } else {

            int p; cin >> p;

            cout << query(1, 0, 2*(n), in[p], out[p])
    / 2 << '\n';
        }
    }

    return 0;
}
```

10.3 Kruskall

```
// Arvore geradora minima (arvore conexa com peso
    minimo)
// O(MlogN)

#include <bits/stdc++.h>
using namespace std;

int n;
class DSU{
    vector<int> parent, sz;
    public:
    void make(int v){
        parent[v] = v;
        sz[v] = 1;
    }

    int find(int v){
        if (v == parent[v]) return v;
        return parent[v] = find(parent[v]);
    }

    void union_(int a, int b){
        a = find(a), b = find(b);

        if(sz[b]>sz[a]) swap(a,b);
        if (a != b){
            sz[a] += sz[b];
            parent[b] = a;
        }
    }

    bool same(int a, int b){
        a = find(a), b = find(b);
        return a == b;
    }

    DSU(int n): parent(n+1), sz(n+1){
        for(int i=1; i<=n; i++) make(i);
    }
};

// {a, b, weight}
vector<tuple<int,int,int>> MST(vector<tuple<int,int,
    int>> &v){
    DSU dsu(n);
    sort(v.begin(), v.end());
    vector<tuple<int,int,int>> ans;
    for(int i=0; i<v.size(); i++){
        int w, a, b;
        tie(w, a, b) = v[i];
        if(!dsu.same(a, b)){
            dsu.union_(a, b);
            ans.push_back({a, b, w});
        }
    }
    return ans;
}

int32_t main(){
    int m;
    cin>>n>>m;
    DSU dsu(n);
    vector<tuple<int,int,int>> vt;
    for(int i=0; i<m; i++){
        int a, b, w;
        cin>>a>>b>>w;
        // {weight, a, b}
        vt.push_back({w, a, b});
    }
    vector<tuple<int,int,int>> ans = MST(vt);
    return 0;
}
```

10.4 Lca

```
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
#define int long long
#define sws ios::sync_with_stdio(false);cin.tie(
    nullptr);
typedef pair<int, int> ii;
#define INF INT64_MAX
const int MAX = 2e5+1;
const int MOD = 1e9+7;
const int LOG = 30;

int ances[MAX][LOG];
int depth[MAX];

int get_lca(int no, int no1){

    int k;

    if (depth[no1] > depth[no]) swap(no, no1);

    k = depth[no] - depth[no1];

    for(int i = LOG-1; i >= 0; i--){

        if(k & (1 << i)){

            no = ances[no][i];
        }
    }

    if(no == no1) return no;

    for(int i = LOG-1; i >= 0; i--){

        if(ances[no][i] != ances[no1][i]){

            no = ances[no][i];
            no1 = ances[no1][i];
        }
    }
    return ances[no][0];

}
int32_t main(){
    sws;

    int n, q; cin >> n >> q;

    vector<int> parents(n+1);

    for(int i = 2; i <= n; i++){

        int v; cin >> v;

        parents[i] = v;
    }

    for(int j = 1; j < LOG; j++){
```

```
        for(int i = 1; i <= n; i++){

            ances[i][0] = parents[i];

            if(i != 1) depth[i] = depth[parents[i]] +
    1;

            ances[i][j] = ances[ances[i][j-1]][j-1];
        }
    }

    while(q--){

        int no, no1; cin >> no >> no1;

        int ans = get_lca(no, no1);

        cout << ans << '\n';
    }

    return 0;
}
```