

```
/Courier findfont 12 scalefont setfont
72 720 moveto
(Table of Contents) show
72 690 moveto
/lineheight 14 def
0 1 58 {
  /n exch def
  n 1 sub dup
  72 690 n lineheight mul sub moveto
  (File ) show n 48 string cvs show ( ..... Page ) show
  n 48 string cvs show
} for
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: int BIT[1100][1100];
5: int s;
6:
7: void init(int S) {
8:     for(int i=0 ; i<=S ; i++)
9:         for(int j=0 ; j<=S ; j++)
10:             BIT[i][j] = 0;
11:
12:     s = S;
13: }
14:
15: // update BIT
16: void update(int i , int j , int v){
17:     i++;
18:     j++;
19:     for( ; i<=s ; i += i&-i){ // row
20:         for(int k = j ; k<=s ; k+=k&-k){ // column
21:             BIT[i][k] += v; // update value
22:         }
23:     }
24: }
25:
26: // query BIT
27: int que(int i , int j){
28:     if(i <=0 || j <= 0 )return 0; // avoid runtime err
29:     int res = 0;
30:
31:     for( ; i ; i-=i&-i){ // row
32:         for(int k = j ; k ; k-=k&-k){ // column
33:             res += BIT[i][k]; // val
34:         }
35:     }
36:
37:     return res;
38: }
39:
40: // sum of range
41: int sum(int i , int j , int x , int y){
42:     i++;j++;x++;y++;
43:     return que(x,y) - que(x , j-1) - que(i-1 , y) + que(i-1 , j-1); // 2d prefix sum
44: }
45:
46:
47: // range update
48: // build difference array d[i][j]
49: const int N = 1010;
50: int t1[N][N], t2[N][N], t3[N][N], t4[N][N];
51:
52: void add(int x, int y, int z) {
53:     for (int X = x; X < N; X += X&-X)
54:         for (int Y = y; Y < N; Y += Y&-Y) {
55:             t1[X][Y] += z;
56:             t2[X][Y] += z * x; // æ³˜æ\204\217æ\230ˆ z * x è\200\214ä,\215æ\230ˆ z *
xi4\214ä\220\216é\235çâ\220\214ç\220\206
57:             t3[X][Y] += z * y;
58:             t4[X][Y] += z * x * y;
59:         }
60: }
61:
62: void range_add(int xa, int ya, int xb, int yb,
63:                 int z) { //(xa, ya) â\210° (xb, yb) â-\220ç\237@é\230µ
64:     add(xa, ya, z);
65:     add(xa, yb + 1, -z);
66:     add(xb + 1, ya, -z);
67:     add(xb + 1, yb + 1, z);
68: }
69:
70: int ask(int x, int y) {
71:     int res = 0;
72:     for (int i = x; i ; i -= i&-i)
73:         for (int j = y; j; j -= j&-j)
74:             res += (x + 1) * (y + 1) * t1[i][j] - (y + 1) * t2[i][j] -
75:                 (x + 1) * t3[i][j] + t4[i][j];
76:     return res;
77: }
78:
79: int range_ask(int xa, int ya, int xb, int yb) {
80:     return ask(xb, yb) - ask(xb, ya - 1) - ask(xa - 1, yb) + ask(xa - 1, ya - 1);
81: }

```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3: const int N = 1e6 + 10;
4:
5: int BIT[N], tag[N], Tag = 0;
6: void reset(){ ++Tag; }
7:
8: void add(int x, int v){
9:     // if tag[i] != Tag: tag[i] = Tag, BIT[i] = 0
10:    for(int i = x; i < N; i += i&-i) BIT[i] += v;
11: }
12: int qry(int x){
13:     // if tag[i] != Tag: continue
14:     int res = 0; for(int i = x; i > 0; i -= i&-i) res += BIT[i]; return res;
15: }
16:
17: int kth(int k){
18:     int sum = 0, x = 0;
19:     for(int i = __lg(N); i >= 0; i--){
20:         x += 1 << i;
21:         if(x >= N || sum + BIT[x] >= k)
22:             x -= 1 << i;
23:         else
24:             sum += BIT[x];
25:     }
26:     return x + 1;
27: }
28:
29: // BIT range add, range query
30: int t1[N], t2[N];
31: void add(int x, int v){
32:     int v1 = x * v;
33:     for(int i = x; i < N; i += i&-i) t1[i] += v, t2[i] += v1;
34: }
35:
36: int getsum(int *t, int x){
37:     int res = 0; for(int i = x; x > 0; i -= i&-i) res += t[i]; return res;
38: }
39:
40: void range_add(int l, int r, int v){
41:     add(l, v), add(r + 1, -v);
42: }
43:
44: int range_sum(int l, int r) {
45:     return (r + 1ll) * getsum(t1, r) - 1ll * l * getsum(t1, l - 1) -
46:         (getsum(t2, r) - getsum(t2, l - 1));
47: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: const int N = 1e5+10;
6:
7: struct node{
8:     int idx, val, par, ch[2];
9:     friend bool operator<(node a, node b) { return a.idx < b.idx; }
10:    void init(int _idx, int _val, int _par){
11:        idx = _idx, val = _val, par = _par, ch[0] = ch[1] = 0;
12:    }
13: }tree[N];
14:
15: int root, top, stk[N];
16:
17: int build(int n){
18:     for(int i=1; i<=n; i++){
19:         int k = i-1;
20:         while(tree[k].val > tree[i].val)k = tree[k].par; // push upward
21:         tree[i].ch[0] = tree[k].ch[1];
22:         tree[k].ch[1] = i;
23:         tree[i].par = k;
24:         tree[tree[i].ch[0]].par = i;
25:     }
26:     return tree[0].ch[1];
27: }
28:
29: int main(){
30:     int n;
31:     cin >> n;
32:     tree[0].init(0, 0, 0);
33:     for(int i=1; i<=n; i++){
34:         int w;
35:         cin >> w;
36:         tree[i].init(i,w,0);
37:     }
38:     root = build(n);
39: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: const int N = 3e5+1;
6: typedef pair<int, int> pii;
7: vector<ll> p(2*N), sz(2*N, 1), val(2*N);
8: vector<vector<int>> G2(2*N);
9:
10: struct edge{int u, v, w;};
11: bool cmp(edge a, edge b){return a.w < b.w;}
12: int find(int v){return (p[v] == v) ? v : p[v] = find(p[v]);}
13:
14: int main(){
15:     vector<edge> E;
16:     for(int i=0; i<m; i++) E.push_back({u[i],v[i],t[i]});
17:     sort(E.begin(), E.end(), cmp);
18:     iota(p.begin(), p.end(), 0);
19:
20:     int cnt = n-1;
21:     for(auto [a,b,w] : E){
22:         int x = find(a), y = find(b);
23:         if(x==y) continue;
24:         p[x] = p[y] = ++cnt;
25:         sz[cnt] = sz[x]+sz[y];
26:         val[cnt] = w;
27:         G2[cnt].push_back(x);
28:         G2[cnt].push_back(y);
29:     }
30: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: // lazy tag pq
5: struct que{
6:     priority_queue<int>x, y;
7:     inline void push(int a){x.push(a);}
8:     inline void del(int a){y.push(a);}
9:     inline int size(){return x.size()-y.size();}
10:    inline void pop(){
11:        while(y.size() && x.top() == y.top()) x.pop(), y.pop();
12:        x.pop();
13:    }
14:    inline int top(){
15:        while(y.size() && x.top() == y.top()) x.pop(), y.pop();
16:        return x.top();
17:    }
18:    inline int top2(){
19:        int a = top();
20:        pop();
21:        int b = top();
22:        push(a);
23:        return b;
24:    }
25: };
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5:
6: struct lctree{
7:     const double EPS = 1e-8;
8:     static const int N = 1e5+10; // no.of line
9:     int tot = 0;
10:    int rt = 0;
11:    int cnt = 0;
12:    int L = 0, R = 4e4+5; // bound of x
13:
14:    struct line{
15:        double a, b;
16:    }li[N];
17:
18:    struct node{
19:        int l, r;
20:        int id;
21:    }seg[N<<6];
22:
23:    double f(int id, int x){
24:        return li[id].a*x + li[id].b;
25:    }
26:
27:    bool fcmp(double a){
28:        return fabs(a) <= EPS ? 0 : a < 0 ? -1 : 1;
29:    }
30:
31:    bool cmp(int id1, int id2, int x){ // 1 = id2 better (larger x/smaller id)
32:        double f1 = f(id1,x), f2 = f(id2,x);
33:        return fcmp(f1-f2) ? f1 < f2 : id1 > id2;
34:    }
35:
36:    void add(int &v, int l, int r, int ql, int qr, int id){
37:        if(!v)v = ++tot;
38:        if(ql <= l && r <= qr){
39:            if(cmp(id, seg[v].id, l) && cmp(id, seg[v].id, r))return ;
40:            if(cmp(seg[v].id, id, l) && cmp(seg[v].id, id, r)){
41:                seg[v].id = id;
42:                return ;
43:            }
44:            int m = (l+r)/2;
45:            if(cmp(seg[v].id, id, m))swap(seg[v].id, id);
46:            if(cmp(seg[v].id, id, l))add(seg[v].l, l, m, ql, qr, id);
47:            if(cmp(seg[v].id, id, r))add(seg[v].r, m+1, r, ql, qr, id);
48:        }else{
49:            int m = (l+r)/2;
50:            if(ql<=m)add(seg[v].l, l, m, ql, qr, id);
51:            if(qr>m)add(seg[v].r, m+1, r, ql, qr, id);
52:        }
53:    }
54:
55:    int qry(int v, int l, int r, int x){
56:        if(l == x && x == r)return seg[v].id;
57:        else{
58:            int m = (l+r)/2;
59:            int res = (x<=m) ? qry(seg[v].l, l, m, x) : qry(seg[v].r, m+1, r, x);
60:            if(cmp(res, seg[v].id, x))res = seg[v].id;
61:            return res;
62:        }
63:    }
64:
65:    void insl(int a, int b, int id){ // insert line based on slope
66:        li[id] = {a,b};
67:        add(rt, L, R, L, R, id);
68:    }
69:
70:    void ins2(int x0, int y0, int x1, int y1, int id){ // insert line based on 2 coor
71:        if(x0 > x1)swap(x0,x1), swap(y0, y1);
72:        if(x0 != x1){
73:            double s = (double)(y1-y0)/(x1-x0);
74:            li[id] = {s, y1-s*x1};
75:        }else li[id] = {0, max(y0,y1)};
76:        add(rt, L, R, x0, x1, id);
77:    }
78: }lct;
79:
80: const int M = 39989;
81: const int M2 = 1e9;
82:
83: int main(){
84:     int n;
85:     cin >> n;
86:

```

```
87:     int lst = 0;
88:
89:     auto f = [&](int v, int m){
90:         return (v+lst-1)%m+1;
91:     };
92:
93:     while(n--){
94:         int op;
95:         cin >> op;
96:
97:         if(!op){
98:             int k;
99:             cin >> k;
100:            k = f(k, M);
101:            lst = lc.qry(lc.rt, lc.L, lc.R, k);
102:            cout << lst << "\n";
103:        }else{
104:            int x0, x1, y0, y1;
105:            cin >> x0 >> y0 >> x1 >> y1;
106:            x0 = f(x0, M), x1 = f(x1, M), y0 = f(y0, M2), y1 = f(y1, M2);
107:            lc.cnt++;
108:            lc.ins2(x0,y0,x1,y1,lc.cnt);
109:        }
110:    }
111: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: #pragma comment(linker, "/stack:200000000")
5: #pragma GCC optimize("Ofast,no-stack-protector")
6: #pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")
7: #pragma GCC optimize("unroll-loops")
8: #pragma GCC optimize(2)
9: #pragma GCC optimize(3)
10: #pragma G++ optimize(2)
11: #pragma G++ optimize(3)
12:
13: typedef long long ll;
14: typedef tuple<int,int,int> tii;
15:
16: const int N = 5e4+10;
17: const ll M = 1e9+7;
18: vector<int>a(N), ans(N);
19: vector<tii>upd(N), qry1[N], qry2[N];
20:
21: inline int Mod(ll x, ll y){return (x+y) > M ? x+y-M : x+y;} // why negative : (
22:
23: struct mat{ // matrix
24:     int a[4][4];
25:     mat(){memset(a,0,sizeof a);}
26:     inline void reset(){
27:         memset(a,0, sizeof a);
28:         for(int i=0; i<=3; i++)a[i][i] = 1;
29:     }
30:     inline mat operator*(const mat &rhs)const{ // multiply
31:         mat r;
32:         for(int k=0; k<=3; k++){
33:             for(int i=0; i<=3; i++){
34:                 for(int j=0; j<=3; j++){
35:                     r.a[i][j] = Mod(r.a[i][j], (1ll*a[i][k]*rhs.a[k][j])%M);
36:                 }
37:             }
38:         }
39:         return r;
40:     }
41:     inline mat operator+(const mat &rhs)const{ // addition
42:         mat r;
43:         for(int i=0; i<=3; i++){
44:             for(int j=0; j<=3; j++){
45:                 r.a[i][j] = Mod(a[i][j] , rhs.a[i][j]);
46:             }
47:         }
48:         return r;
49:     }
50: };
51:
52: struct Node{
53:     mat x, lazy;
54: };
55:
56: vector<Node>seg(4*N);
57:
58: void merge(int v){
59:     seg[v].x = seg[v*2].x+seg[v*2+1].x;
60: }
61:
62: void apply(int v, mat &k){
63:     seg[v].x = seg[v].x*k;
64:     seg[v].lazy = seg[v].lazy*k;
65: }
66:
67: void push(int v){
68:     apply(v*2, seg[v].lazy);
69:     apply(v*2+1, seg[v].lazy);
70:     seg[v].lazy.reset();
71: }
72:
73: void build(int v, int l, int r){
74:     seg[v].lazy.reset();
75:     if(l==r){
76:         seg[v].x.a[0][0] = Mod(a[l],M); // range sum
77:         seg[v].x.a[0][1] = (1LL*a[l]*a[l])%M; // range square sum
78:         seg[v].x.a[0][2] = 1; // range len
79:         return ;
80:     }
81:     int m = (l+r)/2;
82:     build(v*2,l,m);
83:     build(v*2+1,m+1,r);
84:     merge(v);
85: }
86:

```

```

87: void update(int v, int l, int r, int ql, int qr, mat k){
88:     if(ql > r || qr < l)return ;
89:     if(ql <= l && r <= qr){
90:         apply(v, k);
91:         return ;
92:     }
93:     push(v);
94:     int m = (l+r)/2;
95:     update(v*2, l, m, ql, qr, k);
96:     update(v*2+1, m+1, r, ql, qr, k);
97:     merge(v);
98: }
99:
100: int query(int v, int l, int r, int ql, int qr){
101:     if(ql > r || qr < l)return 0;
102:     if(ql <= l && r <= qr)return seg[v].x.a[0][3]; // history square sum
103:     push(v);
104:     int m = (l+r)/2;
105:     return Mod(query(v*2,l,m,ql,qr) , query(v*2+1,m+1,r,ql,qr));
106: }
107:
108: int main(){
109:     ios::sync_with_stdio(0);
110:     cin.tie(0);
111:
112:     int n, m, q;
113:     cin >> n >> m >> q;
114:     m++;
115:     for(int i=1; i<=n; i++)cin >> a[i];
116:     for(int i=2; i<=m; i++){
117:         int l, r, x;
118:         cin >> l >> r >> x;
119:         upd[i] = {l,r,x};
120:     }
121:     for(int i=1; i<=q; i++){
122:         int l, r, x, y;
123:         cin >> l >> r >> x >> y;
124:         qry1[x].push_back({l,r,i});
125:         qry2[y+1].push_back({l,r,i});
126:     }
127:     build(1,1,n);
128:     upd[1] = {1,1,0};
129:     mat tmp;
130:     for(int i=1; i<=m; i++){
131:         tmp.reset();
132:         auto [l,r,x] = upd[i];
133:         tmp.a[0][0] = 1;
134:         tmp.a[0][1] = Mod((2LL*x)%M,M);
135:         tmp.a[1][1] = 1;
136:         tmp.a[2][0] = Mod(x,M);
137:         tmp.a[2][1] = (1LL*x*x)%M;
138:         tmp.a[2][2] = 1;
139:         tmp.a[3][3] = 1;
140:         update(1,1,n,l,r,tmp); // update range
141:         tmp.reset();
142:         tmp.a[1][3] = 1;
143:         apply(1,tmp); // update everything
144:         for(auto [l,r,j] : qry1[i])ans[j] = Mod(ans[j],Mod(M,-query(1,1,n,l,r)));
145:         for(auto [l,r,j] : qry2[i])ans[j] = Mod(ans[j],Mod(M,query(1,1,n,l,r)));
146:     }
147:     for(int i=1; i<=q; i++)cout << ans[i] << "\n";
148: }

```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: typedef pair<ll,ll> pii;
6: const int N = 50005;
7: int c[N], cnt[N];
8: ll sum = 0;
9:
10: struct qry{
11:     ll l, r, blk, id;
12:     inline bool operator<(const qry&b)const{
13:         return (blk^b.blk)?blk<b.blk:((blk&1)?r<b.r:r>b.r);
14:     }
15: }q[N];
16:
17: void add(int i){
18:     sum += cnt[c[i]];
19:     cnt[c[i]]++;
20: }
21:
22: void del(int i){
23:     cnt[c[i]]--;
24:     sum -= cnt[c[i]];
25: }
26:
27: int main(){
28:     ios::sync_with_stdio(0);
29:     cin.tie(0);
30:
31:     int n, m;
32:     cin >> n >> m;
33:
34:     int mxn = sqrt(n);
35:     for(int i=1; i<=n; i++)cin >> c[i];
36:     for(int i=0; i<m; i++)cin >> q[i].l >> q[i].r, q[i].id = i, q[i].blk = q[i].l/mxn;
37:     sort(q,q+m);
38:
39:     vector<pii>ans(m);
40:
41:     for(int i=0, l=1, r=0; i<m; i++){
42:         if(q[i].l == q[i].r){
43:             ans[q[i].id] = {0,1};
44:             continue;
45:         }
46:         while(l > q[i].l)add(--l);
47:         while(r < q[i].r)add(++r);
48:         while(l < q[i].l)del(l++);
49:         while(r > q[i].r)del(r--);
50:         ll len = r-l+1;
51:         ans[q[i].id] = {sum,len*(len-1)/2};
52:     }
53:
54:     for(auto [x,y] : ans){
55:         if(x){
56:             ll t = __gcd(x,y);
57:             x = x/t;
58:             y = y/t;
59:         }else y = 1;
60:         cout << x << "/" << y << "\n";
61:     }
62: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: struct edge{
5:     int u;
6:     int v;
7:     int w;
8: };
9:
10: // DSU : parent
11: vector<int>parent;
12:
13: // sort 2 edges
14: bool cmp(edge a , edge b){
15:     return a.w < b.w;
16: }
17:
18: // DSU : find
19: int find(int v){
20:     return ((parent[v] == v) ? v : parent[v] = find(parent[v]));
21: }
22:
23: // DSU : merge
24: void merge(int a , int b){
25:     if(find(a) != find(b))parent[find(a)] = find(b);
26: }
27:
28: int main() {
29:     int n , m;
30:     cin >> n >> m;
31:
32:     // add edges
33:     edge edges[m];
34:     for(int i=0 ; i<m ; i++)cin >> edges[i].u >> edges[i].v >> edges[i].w;
35:     sort(edges , edges+m , cmp);
36:
37:     // set parent of DSU
38:     for(int i=0 ; i<=n ; i++)parent.push_back(i);
39:
40:     //kruskal algorithm
41:     int sum = 0;
42:     for(int i=0 ; i<m ; i++){
43:         if(find(edges[i].u) != find(edges[i].v)){
44:             sum += edges[i].w;
45:             merge(edges[i].u , edges[i].v);
46:         }
47:     }
48:
49:     cout << sum;
50:     return 0;
51: }
```

```
1: #include <bits/stdc++.h>
2: #include <ext/pb_ds/assoc_container.hpp>
3: #include <ext/pb_ds/tree_policy.hpp>
4:
5: using namespace std;
6: using namespace __gnu_pbds;
7:
8: template <class T>
9: using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
10:
11: int main() {
12:     int n;
13:     cin >> n;
14:     vector<int> a(n);
15:     for (int i=0; i<n; i++) cin >> a[i];
16:     vector<int> b(n, a[0]);
17:     ordered_set<pair<int, int>> os;
18:     os.insert(pair(a[0], 0));
19:     function<int(int)> rank;
20:     rank = [&] (int x) {
21:         return 1LL + os.order_of_key(pair(x, 0));
22:     };
23:
24:     cout << "1\n";
25:     for (int i=1; i<n; i++) {
26:         b[i] = a[i] + rank(b[i-1]);
27:         os.insert(pair(b[i], i));
28:         cout << rank(b[i]) << "\n";
29:     }
30: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: const int N = 1e6+5;
6: ll c[N], cnt[N];
7: ll sum = 0;
8: int mxn, qcnt = 0, rcnt = 0;
9:
10: struct qry{
11:     ll l, r, t, blk, id;
12:     inline bool operator<(const qry&b)const{
13:         if(blk^b.blk)return blk<b.blk;
14:         if((r/mxn)^(b.r/mxn))return ((blk&1)?r<b.r>b.r);
15:         return t < b.t;
16:     }
17: }q[N];
18:
19: struct upd{
20:     ll p, x;
21: }R[N];
22:
23: void add(int i){
24:     if(!cnt[i])sum++;
25:     cnt[i]++;
26: }
27:
28: void del(int i){
29:     cnt[i]--;
30:     if(!cnt[i])sum--;
31: }
32:
33: int main(){
34:     ios::sync_with_stdio(0);
35:     cin.tie(0);
36:
37:     int n, m;
38:     cin >> n >> m;
39:
40:     memset(cnt,0,sizeof cnt);
41:     mxn = pow(n, 2.0/3.0);
42:     for(int i=1; i<=n; i++)cin >> c[i];
43:     for(int i=0; i<m; i++){
44:         char op;
45:         int x, y;
46:         cin >> op >> x >> y;
47:         if(op == 'Q'){
48:             q[qcnt] = {x,y,rcnt,x/mxn,qcnt};
49:             qcnt++;
50:         }else{
51:             rcnt++;
52:             R[rcnt] = {x,y};
53:         }
54:     }
55:     sort(q,q+qcnt);
56:
57:     vector<int>ans(qcnt);
58:     for(int i=0, l=1, r=0,t=0; i<qcnt; i++){
59:         while(l > q[i].l)add(c[--l]);
60:         while(r < q[i].r)add(c[++r]);
61:         while(l < q[i].l)del(c[l++]);
62:         while(r > q[i].r)del(c[r--]);
63:         while(t < q[i].t){
64:             t++;
65:             if(l <= R[t].p && R[t].p <= r){
66:                 add(R[t].x);
67:                 del(c[R[t].p]);
68:             }
69:             swap(c[R[t].p],R[t].x);
70:         }
71:         while(t > q[i].t){
72:             if(l <= R[t].p && R[t].p <= r){
73:                 add(R[t].x);
74:                 del(c[R[t].p]);
75:             }
76:             swap(c[R[t].p],R[t].x);
77:             t--;
78:         }
79:         ans[q[i].id] = sum;
80:     }
81:
82:     for(auto e : ans)cout << e << "\n";
83: }
84:

```

```

1: template <class Info, class Tag> class LazySegtree {
2:     private:
3:         const int n;
4:         vector<Info> tree;
5:         vector<Tag> lazy;
6:
7:         /** builds the segtree values in O(N) time */
8:         void build(int v, int l, int r, const vector<Info> &a) {
9:             if (l == r) {
10:                 tree[v] = a[l];
11:             } else {
12:                 int m = (l + r) / 2;
13:                 build(2 * v, l, m, a);
14:                 build(2 * v + 1, m + 1, r, a);
15:                 tree[v] = tree[2 * v] + tree[2 * v + 1];
16:             }
17:         }
18:
19:         /** applies update x to lazy[v] and tree[v] */
20:         void apply(int v, int l, int r, const Tag &x) {
21:             tree[v].apply(x, l, r);
22:             lazy[v].apply(x);
23:         }
24:
25:         /** pushes lazy updates down to the children of v */
26:         void push_down(int v, int l, int r) {
27:             int m = (l + r) / 2;
28:             apply(2 * v, l, m, lazy[v]);
29:             apply(2 * v + 1, m + 1, r, lazy[v]);
30:             lazy[v] = Tag();
31:         }
32:
33:         void range_update(int v, int l, int r, int ql, int qr, const Tag &x) {
34:             if (qr < l || ql > r) { return; }
35:             if (ql <= l && r <= qr) {
36:                 apply(v, l, r, x);
37:             } else {
38:                 push_down(v, l, r);
39:                 int m = (l + r) / 2;
40:                 range_update(2 * v, l, m, ql, qr, x);
41:                 range_update(2 * v + 1, m + 1, r, ql, qr, x);
42:                 tree[v] = tree[2 * v] + tree[2 * v + 1];
43:             }
44:         }
45:
46:         Info range_query(int v, int l, int r, int ql, int qr) {
47:             if (qr < l || ql > r) { return Info(); }
48:             if (l >= ql && r <= qr) { return tree[v]; }
49:             push_down(v, l, r);
50:             int m = (l + r) / 2;
51:             return range_query(2 * v, l, m, ql, qr) +
52:                 range_query(2 * v + 1, m + 1, r, ql, qr);
53:         }
54:
55:     public:
56:         LazySegtree() {}
57:
58:         LazySegtree(int n) : n(n) {
59:             tree.assign(4 << __lg(n), Info());
60:             lazy.assign(4 << __lg(n), Tag());
61:         }
62:
63:         LazySegtree(const vector<Info> &a) : n(a.size()) {
64:             tree.assign(4 << __lg(n), Info());
65:             lazy.assign(4 << __lg(n), Tag());
66:             build(1, 0, n - 1, a);
67:         }
68:
69:         /** updates [ql, qr] with the arbitrary update chosen */
70:         void range_update(int ql, int qr, const Tag &x) {
71:             range_update(1, 0, n - 1, ql, qr, x);
72:         }
73:
74:         /** @return result of range query on [ql, qr] */
75:         Info range_query(int ql, int qr) { return range_query(1, 0, n - 1, ql, qr); }
76: };
77:
78: enum QueryType { ADD, SET, NONE };
79:
80: struct Tag {
81:     QueryType type = NONE;
82:     int val = 0;
83:     void apply(const Tag &t) {
84:         if (t.type == ADD) {
85:             val += t.val;
86:             if (type != SET) { type = ADD; }

```

```
87:         } else if (t.type == SET) {
88:             type = SET;
89:             val = t.val;
90:         }
91:     }
92: };
93:
94: struct Info {
95:     int sum = 0;
96:     void apply(const Tag &t, int l, int r) {
97:         if (t.type == SET) {
98:             sum = t.val * (r - l + 1);
99:         } else if (t.type == ADD) {
100:             sum += t.val * (r - l + 1);
101:         }
102:     }
103: };
104:
105: /** @return result of joining nodes a and b together */
106: Info operator+(const Info &a, const Info &b) { return {a.sum + b.sum}; }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: typedef pair<ll, ll> pii;
6:
7: struct node{
8:     ll l, r; // range
9:     ll ls, rs; // child
10: };
11:
12: const ll N = 2e5;
13: vector<node> seg(4*N);
14: ll cnt = 0; // id of node
15:
16: void add_edge(ll u, ll v, ll id, ll w){
17:     if(id) adj[u].push_back({v,w}); // child -> parent
18:     else adj[v].push_back({u,w}); // parent -> child
19: }
20:
21: ll build(ll l, ll r, ll id){
22:     if(l==r){
23:         seg[l] = {l,r};
24:         return l; // leaf only use [1..n]
25:     }
26:     ll u = ++cnt;
27:     ll m = (l+r)/2;
28:     seg[u] = {l, r, build(l,m,id), build(m+1,r,id)};
29:     add_edge(u, seg[u].ls, id, 0);
30:     add_edge(u, seg[u].rs, id, 0);
31:     return u; // return child idx to parent
32: }
33:
34: void add(ll v, ll ql, ll qr, ll w, ll u, ll id){
35:     // 0 : u -> [l,R]
36:     // 1 : [L,R] -> u
37:     if(ql > seg[v].r || qr < seg[v].l) return; // not related
38:     if(ql <= seg[v].l && seg[v].r <= qr) add_edge(u, v, id, w); // correct range -> add edge
39:     else{ // go to visit child
40:         add(seg[v].ls, ql, qr, w, u, id);
41:         add(seg[v].rs, ql, qr, w, u, id);
42:     }
43: }
44:
45: int main(){
46:     ll n, q, s;
47:     cin >> n >> q >> s;
48:     cnt = n;
49:     ll in = build(1,n,1);
50:     ll out = build(1,n,0);
51:
52:     for(ll i=0; i<q; i++){
53:         ll t, l, r, w, v, u;
54:         cin >> t;
55:         if(t == 1){
56:             cin >> u >> v >> w;
57:             add_edge(u,v,1,w);
58:         }else if(t == 2){
59:             cin >> u >> l >> r >> w;
60:             add(in,l,r,w,u,1);
61:         }else{
62:             cin >> u >> l >> r >> w;
63:             add(out,l,r,w,u,0);
64:         }
65:     }
66: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 250005;
5: vector<int>a(N), ans(N), L(N), R(N), qry[N];
6:
7: struct T1{ // range add/ min
8:     struct Node{
9:         int mn, lazy;
10:    }seg[4*N];
11:
12:    void clear(int n){
13:        for(int i=1; i<=4*n; i++)seg[i] = {0,0};
14:    }
15:
16:
17:    void merge(int v){
18:        seg[v].mn = min(seg[v*2].mn, seg[v*2+1].mn);
19:    }
20:
21:    void apply(int v, int val){
22:        seg[v].mn += val;
23:        seg[v].lazy += val;
24:    }
25:
26:    void push(int v, int l, int r){
27:        if(seg[v].lazy != 0){
28:            apply(v*2, seg[v].lazy);
29:            apply(v*2+1, seg[v].lazy);
30:            seg[v].lazy = 0;
31:        }
32:    }
33:
34:    void build(int v, int l, int r){
35:        if(l==r){
36:            seg[v] = {a[l],0};
37:        }else{
38:            int m = (l+r)/2;
39:            build(v*2, l, m);
40:            build(v*2+1, m+1, r);
41:            seg[v].lazy = 0;
42:            merge(v);
43:        }
44:    }
45:
46:    void upd(int v, int l, int r, int ql, int qr, int val){
47:        if(ql > r || qr < l)return ;
48:        if(ql <= l && r <= qr){
49:            apply(v,val);
50:            return ;
51:        }
52:        int m = (l+r)/2;
53:        push(v,l,r);
54:        upd(v*2,l,m,ql,qr,val);
55:        upd(v*2+1,m+1,r,ql,qr,val);
56:        merge(v);
57:    }
58:
59:    int qry(int v, int l, int r, int ql, int qr){
60:        if(ql > r || qr < l)return 1e9;
61:        if(ql <= l && r <= qr)return seg[v].mn;
62:        int m = (l+r)/2;
63:        push(v,l,r);
64:        return min(qry(v*2,l,m,ql,qr), qry(v*2+1,m+1,r,ql,qr));
65:    }
66: }mnT;
67:
68: struct T2{ // query max node
69:     struct Node{
70:         int id, mx;
71:         friend Node operator + (Node A, Node B){
72:             if(A.mx > B.mx)return A;
73:             else if(A.mx < B.mx)return B;
74:             else return ((Node){min(A.id, B.id), A.mx});
75:         }
76:     }seg[4*N];
77:
78:    void merge(int v){
79:        seg[v] = seg[v*2]+seg[v*2+1];
80:    }
81:
82:    void build(int v, int l, int r){
83:        if(l==r){
84:            seg[v] = {l, a[l]};
85:            return ;
86:        }

```

```

87:         int m = (l+r)/2;
88:         build(v*2, l, m);
89:         build(v*2+1, m+1, r);
90:         merge(v);
91:     }
92:
93:     Node qry(int v, int l, int r, int ql, int qr){
94:         if(ql > r || qr < l) return {0, (int)-1e9};
95:         if(ql <= l && r <= qr) return seg[v];
96:         int m = (l+r)/2;
97:         return qry(v*2, l, m, ql, qr) + qry(v*2+1, m+1, r, ql, qr);
98:     }
99: }mxT;
100:
101: void f(int n, int q){
102:     mnT.clear(n);
103:     for(int i=1; i<=n; i++) qry[i].clear();
104:     for(int i=1; i<=q; i++) qry[R[i]].push_back(i);
105:     vector<int> sk(n+1);
106:     sk[0] = 1;
107:     int top = 0;
108:
109:     for(int i=1; i<=n; i++){
110:         while(top && a[i] >= a[sk[top]]){
111:             mnT.upd(1, 1, n, sk[top-1], sk[top]-1, a[i]-a[sk[top]]);
112:             top--;
113:         }
114:         sk[++top] = i;
115:         if(i>1) mnT.upd(1, 1, n, i-1, i-1, a[i-1]+a[i]);
116:         for(int e : qry[i]){
117:             auto [id, mx] = mxT.qry(1, 1, n, L[e], R[e]);
118:             if(id+1<=R[e]-1) ans[e] = min(ans[e], mx+mnT.qry(1, 1, n, id+1, R[e]-1));
119:         }
120:     }
121: }
122:
123: int main(){
124:     ios::sync_with_stdio(0);
125:     cin.tie(0);
126:
127:     int n, q;
128:     cin >> n >> q;
129:
130:     for(int i=1; i<=n; i++) cin >> a[i];
131:     for(int i=1; i<=q; i++){
132:         cin >> L[i] >> R[i];
133:         ans[i] = 1e9;
134:     }
135:
136:     mxT.build(1, 1, n);
137:
138:     // front
139:     f(n, q);
140:
141:     // mid
142:     for(int i=1; i<=q; i++) ans[i] = min(ans[i], a[L[i]]+a[R[i]]+mxT.qry(1, 1, n, L[i]+1, R[i]-1).mx);
143:
144:     // back
145:     reverse(a.begin()+1, a.begin()+n+1);
146:     for(int i=1; i<=q; i++){
147:         int l = L[i], r = R[i];
148:         L[i] = n-r+1;
149:         R[i] = n-l+1;
150:     }
151:     mxT.build(1, 1, n);
152:     f(n, q);
153:
154:     for(int i=1; i<=q; i++) cout << ans[i] << "\n";
155: }

```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 1e6 + 10;
5: struct SparseTable{
6:     int st[N][20];
7:     void build(const vector<int>& sa){
8:         int n = sa.size();
9:         for(int i = 0; i < n; i++) st[i][0] = sa[i];
10:        for(int j = 1; j < 20; j++){
11:            for(int i = 0; i <= n - (1 << j); i++){
12:                st[i][j] = min(st[i][j - 1], st[i + (1 << j - 1)][j - 1]);
13:            }
14:        }
15:    }
16:    int query(int l, int r){
17:        // cout << "L, R: " << l << " " << r << endl;
18:        int d = __lg(r - l + 1);
19:        return min(st[l][d], st[r - (1 << d) + 1][d]);
20:    }
21: } ST;
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int maxN = 10000, sqrtN = 100;
5: int arr[maxN];
6: int block[sqrtN];
7: int sz;
8:
9: void preprocess(int n){
10:     int idx = -1;
11:     sz = sqrt(n);
12:
13:     for(int i=0 ; i<n ; i++){
14:         if(i % sz == 0){
15:             idx++;
16:             block[idx] = 0;
17:         }
18:         block[idx] += arr[i];
19:     }
20: }
21:
22: // time complexity : O(1)
23: void update(int idx , int val){
24:     block[idx/sz] += val - arr[idx];
25:     arr[idx] = val;
26: }
27:
28:
29: // time complexity : O(sqrt(N))
30: int query(int l , int r){
31:     int sum = 0;
32:
33:     while(l<r && l%sz != 0 && l != 0){
34:         sum += arr[l];
35:         l++;
36:     }
37:
38:     while(l + sz -1 <= r){
39:         sum += block[l/sz];
40:         l += sz;
41:     }
42:
43:     while(l<=r){
44:         sum += arr[r];
45:         l++;
46:     }
47:
48:     return sum;
49: }
50:
51: int main() {
52:     int n;
53:     cin >> n;
54:     for(int i=0 ; i<n ; i++)cin >> arr[i];
55:     preprocess(n);
56:
57:     int m,l,r;
58:     cin >> m;
59:     for(int i=0 ; i<m ; i++){
60:         int v;
61:         cin >> v >> l >> r;
62:         if(v==1){
63:             cout << query(l,r) << "\n";
64:         }else{
65:             update(l,r);
66:         }
67:     }
68:
69:     return 0;
70: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5:
6: struct line{
7:     ll l, r, y, v;
8: };
9:
10: struct node{
11:     ll sum, lazy;
12: };
13:
14: vector<ll>arr; // idx -> val
15:
16: const ll N = 200000;
17: vector<node>seg(4*N);
18:
19: void update(ll v, ll l, ll r, ll ql, ll qr, ll val){
20:     if(ql > r || qr < l) return ;
21:     if(ql <= l && r <= qr){
22:         seg[v].lazy += val;
23:     }else{
24:         ll m = (l+r)>>1;
25:         update(v*2, l, m, ql, qr, val);
26:         update(v*2+1, m+1, r, ql, qr, val);
27:     }
28:     if(seg[v].lazy > 0){
29:         // cout << arr[r+1] << ' ' << arr[l] << " " << l << ' ' << r << "\n";
30:         seg[v].sum = arr[r+1] - arr[l];
31:     }else{
32:         if(l==r){
33:             seg[v].sum = 0;
34:         }else{
35:             seg[v].sum = seg[v*2].sum + seg[v*2+1].sum;
36:         }
37:     }
38:     // cout << seg[v].sum << ' ' << l << " " << r << "\n";
39: }
40:
41: bool cmp(line a, line b){
42:     if(a.y == b.y) return a.v > b.v;
43:     else return a.y < b.y;
44: }
45:
46: int main(){
47:     ll n;
48:     cin >> n;
49:
50:     // discretization how???
51:     ll lx[n], ly[n], hx[n], hy[n];
52:
53:     for(ll i=0; i<n; i++){
54:         cin >> lx[i] >> ly[i] >> hx[i] >> hy[i];
55:         arr.push_back(lx[i]);
56:         arr.push_back(hx[i]);
57:     }
58:
59:     sort(arr.begin(), arr.end());
60:     ll cnt = unique(arr.begin(), arr.end())-arr.begin();
61:
62:     unordered_map<ll,ll>mp; // val -> idx
63:     for(ll i=0; i<cnt; i++){
64:         mp[arr[i]] = i;
65:     }
66:
67:     vector<line>lines(2*n);
68:     for(ll i=0; i<n; i++){
69:         lines[2*i] = {mp[lx[i]], mp[hx[i]], ly[i], 1};
70:         lines[2*i+1] = {mp[lx[i]], mp[hx[i]], hy[i], -1};
71:     }
72:
73:     sort(lines.begin(), lines.end(), cmp);
74:     ll ans = 0, y = 0;
75:
76:     for(ll i=0; i<2*n; i++){
77:         // cout << seg[1].sum << " " << lines[i].y-y << "\n";
78:         ans += seg[1].sum*(lines[i].y-y);
79:         update(1, 0, cnt-1, lines[i].l, lines[i].r-1, lines[i].v);
80:         y = lines[i].y;
81:     }
82:
83:     cout << ans;
84: }

```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: const ll N = 2e5+1;
6:
7: vector<ll>p(N), val(N,0);
8:
9: int find(int v){
10:     if(v == p[v]){
11:         return v;
12:     }else{
13:         int t = p[v]; // old parent
14:         p[v] = find(p[v]); // compression
15:         val[v] += val[t]; // val + parent val
16:         return p[v]; // return new parent
17:     }
18: }
19:
20: int main(){
21:     int n, q;
22:     cin >> n >> q;
23:
24:     for(int i=1; i<=n; i++)p[i] = i;
25:
26:     vector<int>ans;
27:     for(int i=0; i<q; i++){
28:         ll a, b, d;
29:         cin >> a >> b >> d;
30:
31:         if(a == b){
32:             if(d == 0)ans.push_back(i+1);
33:             continue;
34:         }
35:
36:         int x = find(a);
37:         int y = find(b);
38:
39:         if(x == y){
40:             if(val[a] - val[b] == d)ans.push_back(i+1);
41:         }else{
42:             p[x] = y;
43:             val[x] = val[b] + d - val[a];
44:             ans.push_back(i+1);
45:         }
46:     }
47:
48:     for(int e : ans)cout << e << " ";
49: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5:
6: int main(){
7:     ll n, k;
8:     cin >> n >> k;
9:
10:    vector<ll>a(n+1), b(n+1);
11:    for(ll i=1; i<=n; i++)cin >> a[i];
12:    for(ll i=1; i<=n; i++)cin >> b[i];
13:
14:    ll dp[n+1][20100];
15:    ll off = 10000;
16:    for(ll i=0; i<=n; i++)for(ll j=0; j<20100; j++)dp[i][j] = -1e9;
17:    dp[0][off] = 0;
18:    for(ll i=1; i<=n; i++){
19:        ll w = a[i]-k*b[i];
20:        for(ll j=0; j<20100; j++){ // from j -> ?
21:            if(dp[i-1][j] < 0)continue;
22:            dp[i][j] = max(dp[i][j], dp[i-1][j]);
23:            ll t = j+w;
24:            if(t < 0 || t > 20100)continue;
25:            dp[i][t] = max(dp[i][t], dp[i-1][j]+a[i]);
26:        }
27:    }
28:
29:    if(dp[n][off])cout << dp[n][off];
30:    else cout << -1;
31: }
32:
33: /*
34: sum(a)/sum(b) = k , find maximum sum(a)
35: -> sum(a) = sum(b)*k
36:
37: dp[i][j] = use i element, diff between sum(a)-sum(b)*k = j
38: ans : dp[n][0]
39:
40: * maybe -ve, require shifting
41: */
```

```

1: /*
2:
3: Knapsack 9 talk
4:
5: */
6:
7: /*
8: 1. 0-1 knapsack
9:
10: Problem statement:
11: - N item V size knapsack
12: - c[i] cost w[i] value
13: - put 1 max
14:
15: use rolling array for O(n) space complexity
16:
17: Trick:
18: - must fillin -> set f[1..v] as -inf, f[0] as 0
19: - not mist fillin -> set[0..v] as 0
20: - set bound : max(v - c[i+1..n], c[i])
21:   -> can't contribute ans below bound
22:
23: Formula:
24: for(int i=1; i<=n; i++){ // item 1 -> item n
25:     for(int j=v; j>=0; j--){ // j-c[i] is still in prev state
26:         f[j] = max(f[j], f[j-c[i]]+w[i]); // from 2 subproblem
27:     }
28: }
29: */
30:
31:
32:
33:
34: /*
35: 2. Complete knapsack
36:
37: Problem statement:
38: - N item V size knapsack
39: - c[i] cost w[i] value
40: - no max limit
41:
42: Trick:
43: - maintain monotonic (remove price larger but value smaller)
44: - use pow to represent (1,2,4,8) -> O(V) to O(log(V))
45:
46: Formula:
47: for(int i=1; i<=n; i++){
48:     for(int j=c[i]; j<=v; j++){
49:         f[j] = max(f[j], f[j-c[i]]+w[i]); // best f[j-c[i]] with item i is found already
50:     }
51: }
52: */
53:
54:
55:
56:
57: /*
58:
59: 3. Multiple knapsack(â€\232é\207\215è\203\214â€\214\205)
60:
61: Problem statement:
62: - N item V size knapsack
63: - C[i] cost W[i] value
64: - N[i] max amount
65:
66: 1. ith item -> N[i] item
67: - 0-1 knapsack
68: - 1 item -> N[i] item
69:
70: 2. ith item -> 1,2,4...2^(k-1), N[i]-2^(k-1)+1
71: - 1 item -> log(N[i]) item
72:
73: Formula:
74: for(int i=1; i<=n; i++){
75:     int c = 1, k = N[i];
76:     while(k > c){
77:         k -= c; // divide item
78:         // 0-1 knapsack
79:         c *= 2;
80:     }
81:
82:     if(k > 0){ // remain item
83:         // 0-1 knapsack
84:     }
85: }
86:

```

```

87: 3. Montonic optimization
88: - ref : https://zhuanlan.zhihu.com/p/379510583
89: -  $F[i] = \max(F[i], F[i-C[i]]+W[i], \dots, F[i-C[i]*N[i]]+W[i]*N[i])$ 
90: - E.g.  $V = 10, C = 2, N = 3$ 
91: -  $F[10] \leftarrow \max(F[8], F[6], F[4])$ 
92: -  $F[8] \leftarrow (F[6], F[4], F[2])$ 
93: - Sliding window!
94: - for each  $i\%C[i]$  same, we can maintain a queue to find max
95: - why need 2 array?
96: - otherwise use more then N item (E.g.  $F[2] \rightarrow F[8] \rightarrow F[10]$ , but  $F[2]$  can't reach  $F[10]!$ )
97:
98: Formula:
99: for(int i=0; i<c[i]; i++){ // mod
100:     deque<int>dq;
101:     for(int k=i; k<=V; k+=c[i]){
102:         dp[k] = f[k];
103:         if(!dq.empty() && dq.front < k-c[i]*n[i])dq.pop_front(); // out of bound (E.g.  $F[2]$  can't transfer
to F[10])
104:         if(!dq.empty()){
105:             dp[k] = max(dp[k], f[dq.front()]+(k-dq.front())/c[i]*w[i]); // current max in queue
106:         }
107:
108:         while(!dq.empty() && f[dq.back()]- (dq.back()-j)/c[i]*w[i] <= f[k]- (k-j)/c[i]*w[i]) dq.pop_back();
// ( $F[k]$  is larger  $\rightarrow$  pop useless out)
109:         dq.push(k); // add  $F[k]$  into queue
110:     }
111: }
112: */
113:
114:
115:
116:
117: /*
118:
119: 4. Mix knapsack
120:
121: Problem statement:
122: Some items only take once
123: some items inf take
124: some items limit take
125:
126: mix 0-1 knapsack , Complete knapsack and mutiple knapsack together
127:
128: Formula:
129: for(int i=0; i<n; i++){
130:     int w,v,p;
131:     cin >> w >> v >> p;
132:     if(p==0){ // complete
133:         for(int j=w; j<=t; j++)dp[j] = max(dp[j], dp[j-w]+v);
134:     }else if(p == 1){ // 0-1
135:         for(int j=t; j>=w; j--)dp[j] = max(dp[j], dp[j-w]+v);
136:     }else{ // multiple
137:         int cnt = p;
138:         int c = 1;
139:         while(cnt > c){
140:             cnt -= c;
141:             for(int j=t; j>=c*w; j--)dp[j] = max(dp[j], dp[j-c*w]+c*v);
142:             c *= 2;
143:         }
144:         if(cnt != 0){
145:             for(int j=t; j>=cnt*w; j--)dp[j] = max(dp[j], dp[j-cnt*w]+cnt*v);
146:         }
147:     }
148: }
149: */
150:
151:
152:
153:
154:
155: /*
156:
157: 5. 2 dimension cost knapsack (ä°\214ç»è¹¸\224"è\203\214â\214\205)
158:
159: Problem statement:
160: - Each item has 2 different cost ( $C[i], G[i]$ )
161: - each cost has its own maximum ( $V, M$ )
162:
163: Add one more dimension for cost
164: The second dimension maybe hidden (E.g. Limit total amount of item can take  $\rightarrow$  cost)
165:
166: Formula:
167: for(int i=0; i<n; i++){
168:     for(int j=v; j>=c[i]; j--){
169:         for(int k=m; k>=g[i]; k--){
170:             dp[j][k] = max(dp[j][k], dp[j-c[i]][k-g[i]]+w[i]);

```

```

171:     }
172: }
173: }
174:
175: */
176:
177:
178:
179:
180:
181: /*
182:
183: 6. Group knapsack (â\210\206ç»\204è\203\214â\214\205)
184:
185: Problem statement:
186: - Items are divided into groups
187: - Each group only can pick 1 item
188:
189: For each gp , either we choose 1 or no choose
190:
191: F[k][j] = max(F[k][j], F[k-1][j-w[i]]+v[i]) // item i belongs to gp k
192:
193: Group : choose the 0,1,2...th item in gp
194: Multiple : choose 0,1,2... item
195:
196: limit max item -> 1 gp
197:
198: Formula:
199:
200: for(int i=1 ; i<=n; i++){
201:     cin >> s; // no.of item in gp
202:     for(int j=0; j<s; j++){cin >> w[j] >> v[j];
203:         for(int j=v; j>=0; j--){ // set 0 to min w for optimization
204:             for(int k=0; k<s; k++){
205:                 dp[j] = max(dp[j] , dp[j-w[k]]+v[k]);
206:             }
207:         }
208:     }
209:
210:
211: */
212:
213:
214:
215:
216:
217: /*
218:
219: 7. Dependency knapsack(æ\234\211ä¾\235èµ\226ç\232\204è\203\214â\214\205)
220:
221: Problem statement:
222: - item has some relationship
223: - if we choose item i, we must choose item j
224:
225:
226: Assume one parent , with n child
227: -> we have 2^n + 1 options
228:
229: We can only choose 1 option -> transfer it to a group knapsack
230: For options with same value -> only leave largest value one
231:
232: 0-1 knapsack -> get F[0...V-c[i]]
233: -> change it to V - c[i] +1 item -> c[i] + k item value = F'[k] + w[i]
234:
235:
236: Formula:
237: for(int p : gp[0]){
238:     memcpy(dp2,dp,sizeof dp2);
239:     for(int j : gp[p]){
240:         for(int i=n-w[p]; i>=w[j];i--)dp2[i] = max(dp2[i], dp2[i-w[j]]+w[j]*v[j]); // knapsack on gp
241:     }
242:     for(int i= w[p]; i<=n; i++)dp[i] = max(dp[i], dp2[i-w[p]]+w[p]*v[p]); // + parent item
243: }
244:
245:
246: Tree:
247: void dfs(int u){
248:     for(int v : adj[u]){ // each child in v
249:         dfs(v); // transfer it into item
250:
251:         for(int i=m; i>=0; i--){// 0-1 knapsack
252:             for(int j=i; j>=1; j--){ // subtree v pick how many
253:                 dp[u][i] = max(dp[u][i], dp[v][j]+dp[u][i-j]);
254:             }
255:         }
256:     }

```

```
257:
258: // dp[u][i] contain max with only child -> parent also needed!
259: for(int i=m; i>=1; i--)dp[u][i] = dp[u][i-1]+s[u];
260: }
261:
262: */
263:
264:
265: #include <bits/stdc++.h>
266: using namespace std;
267:
268: int main(){
269:     int n, m;
270:     cin >> n >> m;
271:
272:     vector<int>gp[m+1], w(m+1), v(m+1);
273:
274:     for(int i=1; i<=m; i++){
275:         int p;
276:         cin >> w[i] >> v[i] >> p;
277:         gp[p].push_back(i);
278:     }
279:
280:     int dp[n+1], dp2[n+1];
281:     memset(dp,0, sizeof dp);
282:
283:     for(int p : gp[0]){
284:         memcpy(dp2,dp,sizeof dp2);
285:         for(int j : gp[p]){
286:             for(int i=n-w[p]; i>=w[j];i--)dp2[i] = max(dp2[i], dp2[i-w[j]]+w[j]*v[j]);
287:         }
288:         for(int i= w[p]; i<=n; i++)dp[i] = max(dp[i], dp2[i-w[p]]+w[p]*v[p]);
289:     }
290:
291:     cout << dp[n];
292: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: const ll N = 2e5+5;
6: vector<ll>adj[N], a(N), d(N,0), sa(N);
7: ll ans = 0;
8:
9: void dfs1(int u, int p){ // Precompute ans for root
10:     sa[u] = a[u];
11:     for(int v : adj[u]){
12:         if(v==p)continue;
13:         dfs1(v,u);
14:         d[u] += d[v]+sa[v];
15:         sa[u] += sa[v];
16:     }
17: }
18:
19: void reroot(int v, int u){
20:     d[u] -= d[v]+sa[v]; // remove v from u
21:     sa[u] -= sa[v];
22:     d[v] += d[u]+sa[u]; // add u to v
23:     sa[v] += sa[u];
24: }
25:
26: void dfs2(int u, int p){
27:     ans = max(ans, d[u]);
28:     for(int v : adj[u]){
29:         if(v==p)continue;
30:         reroot(v,u);
31:         dfs2(v,u);
32:         reroot(u,v);
33:     }
34: }
35:
36: int main(){
37:     int n;
38:     cin >> n;
39:     for(int i=1; i<=n; i++)cin >> a[i];
40:     for(int i=0; i<n-1; i++){
41:         int u, v;
42:         cin >> u >> v;
43:         adj[u].push_back(v);
44:         adj[v].push_back(u);
45:     }
46:     dfs1(1,0);
47:     dfs2(1,0);
48:     cout << ans;
49: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 1e5+5;
5: vector<int>adj[N];
6: int f[N][3];
7:
8: void dfs0(int u, int p){ // compute 2 largest child
9:     for(int v : adj[u]){
10:         if(v==p)continue;
11:         dfs0(v,u);
12:         if(f[v][0] >= 0){
13:             if(f[v][0]+1 >= f[u][0]){
14:                 f[u][1] = f[u][0];
15:                 f[u][0] = f[v][0]+1;
16:             }else if(f[v][0]+1 > f[u][1]){
17:                 f[u][1] = f[v][0]+1;
18:             }
19:         }
20:     }
21: }
22:
23: void dfs1(int u, int p){ // find largest ancestor
24:     for(int v : adj[u]){
25:         if(v==p)continue;
26:         int t = 0;
27:         if(f[u][0] == f[v][0]+1)t = max(f[u][1], f[u][2]); // can't use largest
28:         else t = max(f[u][0], f[u][2]);
29:         if(t>=0)f[v][2] = t+1;
30:         dfs1(v,u);
31:     }
32: }
33:
34: int main(){
35:     memset(f, -1, sizeof f);
36:
37:     int n, m, d;
38:     cin >> n >> m >> d;
39:
40:     for(int i=0; i<m; i++){
41:         int v;
42:         cin >> v;
43:         f[v][0] = f[v][2] = 0;
44:     }
45:
46:     for(int i=0; i<n-1; i++){
47:         int u, v;
48:         cin >> u >> v;
49:         adj[u].push_back(v);
50:         adj[v].push_back(u);
51:     }
52:     dfs0(1,0);
53:     dfs1(1,0);
54:     int cnt = 0;
55:     for(int i=1; i<=n; i++)cnt += (max(f[i][0],f[i][2]) <= d);
56:     cout << cnt;
57: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: int main(){
5:     int n;
6:     cin >> n;
7:
8:     vector<int>a(1<<n);
9:     for(int i=0; i<(1<<n); i++)cin >> a[i];
10:
11:     for(int i=0; i<n; i++){ // dimension
12:         for(int j=0; j<(1<<n); j++){ // mask
13:             if(j&(1<<i))f[j] += f[j^(1<<i)];
14:         }
15:     }
16: }
17:
18: /*
19: Find subset sum
20: 1100101 <- 1000101
21: -
22: */
```

```
1: #include <bits/stdc++.h>
2: #define sz(a) (int)a.size()
3: using namespace std;
4:
5: const int N = 1e5 + 10;
6:
7: struct Point {
8:     double x, y, ang;
9:     Point operator-(const Point& p) const { return {x - p.x, y - p.y, 0}; }
10: } p[N];
11:
12: double dis(Point p1, Point p2){
13:     return sqrt((p1.x - p2.x)*(p1.x - p2.x) + (p1.y - p2.y)*(p1.y - p2.y));
14: }
15:
16: bool cmp(Point p1, Point p2){
17:     if(p1.ang == p2.ang){
18:         return dis(p1, p[1]) < dis(p2, p[1]);
19:     }
20:     return p1.ang < p2.ang;
21: }
22:
23: double cross(Point p1, Point p2){ return p1.x * p2.y - p1.y * p2.x; }
24:
25:
26:
27: int main(){
28:     int n;
29:     cin >> n;
30:
31:     for(int i = 1; i <= n; i++) cin >> p[i].x >> p[i].y;
32:     for(int i = 2; i <= n; i++){
33:         if(p[i].y < p[1].y || (p[i].y == p[1].y && p[i].x < p[1].x)){
34:             swap(p[1], p[i]);
35:         }
36:     }
37:     for (int i = 2; i <= n; ++i) {
38:         p[i].ang = atan2(p[i].y - p[1].y, p[i].x - p[1].x);
39:     }
40:     sort(p + 2, p + n + 1, cmp);
41:     vector<int> sta{1};
42:     for(int i = 2; i <= n; i++){
43:         while(sz(sta) >= 2 && cross(p[sta[sz(sta) - 1]] - p[sta[sz(sta) - 2]], p[i] - p[sta[sz(sta) -
44: 1]]) <= 0) sta.pop_back();
45:         sta.push_back(i);
46:     }
47:     sta.push_back(1);
48:     double ans = 0;
49:     for(int i = 0; i + 1 < sz(sta); i++){
50:         ans += dis(p[sta[i]], p[sta[i + 1]]);
51:     }
52:     cout << fixed << setprecision(2) << ans << '\n';
53: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5:
6: int main(){
7:     ll n, m;
8:     cin >> n >> m;
9:
10:    set<ll>horz, vert, slash, bslash;
11:    for(int i=0; i<m; i++){
12:        ll x, y;
13:        cin >> x >> y;
14:        horz.insert(x);
15:        vert.insert(y);
16:        slash.insert(x+y);
17:        bslash.insert(x-y);
18:    }
19:
20:    ll ans = (n-horz.size())*(n-vert.size());
21:
22:    for(auto d : slash){
23:        set<ll>cross;
24:        for(auto x : horz)if(1 <= d-x && d-x <= n)cross.insert(x);
25:        for(auto y : vert)if(1 <= d-y && d-y <= n)cross.insert(d-y);
26:        if(d-1 <= n)ans -= d-1-cross.size();
27:        else ans -= 2*n-(d-1)-cross.size();
28:    }
29:
30:    for(auto d : bslash){
31:        set<ll>cross;
32:        for(auto x : horz)if(1 <= x-d && x-d <= n)cross.insert(x);
33:        for(auto y : vert)if(1 <= y+d && y+d <= n)cross.insert(y+d);
34:        for(auto e : slash)if((d+e)%2 == 0 && 1 <= (d+e)/2 && (d+e)/2 <= n && 1 <= (e-d)/2 && (e-d)/2
<= n)cross.insert((d+e)/2);
35:        if(1-d >= 1)ans -= n+d-cross.size();
36:        else ans -= n-d-cross.size();
37:    }
38:
39:    cout << ans << "\n";
40: }
41:
42: /*
43: d = x-y
44: e = x+y
45: x = (d+e)/2, y = (e-d)/2
46: */
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: int main(){
5:     int k;
6:     cin >> k;
7:
8:     int dis[k];
9:     for(int i=0; i<k; i++)dis[i] = 1e9;
10:    dis[1] = 1;
11:
12:    deque<int>dq;
13:    dq.push_back(1);
14:
15:    while(!dq.empty()){
16:        int u = dq.front();
17:        dq.pop_front();
18:
19:        // +1
20:        if(dis[(u+1)%k] > dis[u]+1){
21:            dis[(u+1)%k] = dis[u]+1;
22:            dq.push_back((u+1)%k);
23:        }
24:
25:        // *10
26:        if(dis[(u*10)%k] > dis[u]){
27:            dis[(u*10)%k] = dis[u];
28:            dq.push_front((u*10)%k);
29:        }
30:    }
31:    cout << dis[0];
32: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef pair<int,int> pii;
5: const int N = 2e5+10;
6: vector<int>adj[N], st(N), low(N);
7: vector<bool>ap(N);
8: int tin = 0;
9:
10: void dfs(int u, int p){
11:     st[u] = low[u] = ++tin; // init
12:     int cnt = 0;
13:     for(int v: adj[u]){
14:         if(!st[v]){ // not visit -> (u-v) tree edge
15:             dfs(v,u);
16:             cnt++;
17:             low[u] = min(low[u], low[v]);
18:             if(low[v] >= st[u] && p!=0){ // not passing though + not root
19:                 ap[u] = 1;
20:             }
21:         }else if(v != p){
22:             low[u] = min(low[u], st[v]); // only 1 back edge
23:         }
24:     }
25:     if(p==0 && cnt > 1)ap[u] = 1; // root has more than 1 child
26: }
27:
28: /*
29: Def : vertex which del will increase component
30:
31: no back edge passing over node u
32: -> even reach u still split when del
33: */
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef pair<int,int> pii;
5: const int N = 2e5+10;
6: vector<int>adj[N], st(N), low(N),bcc[N], s;
7: int tin = 0, k = 0;
8:
9: void dfs(int u, int p){
10:     st[u] = low[u] = ++tin; // init
11:     s.emplace_back(s);
12:
13:     for(int v: adj[u]){
14:         if(!st[v]){ // not visit -> (u-v) tree edge
15:             dfs(v,u);
16:             low[u] = min(low[u], low[v]);
17:             if(st[u]==low[v]){ // ecc vertex are descendent
18:                 ++k;
19:                 for(int x = -1; x!=u; s.pop_back()){
20:                     x = s.back();
21:                     bcc[x].emplace_back(k);
22:                 }
23:                 bcc[u].emplace_back(k);
24:             }
25:         }
26:         low[u] = min(low[u], st[v]); // only 1 back edge
27:     }
28: }
29:
30: /*
31: Def : at least 2 path without duplicate node
32: */
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 1e5;
5: vector<int>adj[N];
6:
7: int main(){
8:     int n;
9:     cin >> n;
10:
11:     bool flag = 1;
12:     vector<int>side(n,-1);
13:     queue<int>q;
14:     for(int i=0; i<n; i++){
15:         if(side[i] != -1)continue;
16:         q.push(i);
17:         side[i] = 0;
18:         while(!q.empty()){
19:             int u = q.front();
20:             q.pop();
21:             for(int v : adj[u]){
22:                 if(side[v]==-1){
23:                     side[v] = side[u]^1;
24:                     q.push(v);
25:                 }else if(side[v] == side[u])flag = 0;
26:             }
27:         }
28:     }
29:     cout << (flag ? "YES" : "NO") << endl;
30: }
31:
32:
33:
34:
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef pair<int,int> pii;
5: const int N = 2e5+10;
6: const int M = 1e5;
7: vector<int>adj[N], st(N), low(N), s, adj2[N];
8: int tin = 0, k = M-1;
9:
10: void dfs(int u, int p){
11:     st[u] = low[u] = ++tin; // init
12:     s.emplace_back(u);
13:
14:     for(int v: adj[u]){
15:         if(!st[v]){ // not visit -> (u-v) tree edge
16:             dfs(v,u);
17:             low[u] = min(low[u], low[v]);
18:             if(st[u]==low[v]){ // ecc vertex are descendent
19:                 ++k;
20:                 for(int x = -1; x!=v; s.pop_back()){
21:                     x = s.back();
22:                     adj2[x].push_back(k);
23:                     adj2[k].push_back(x);
24:                 }
25:                 adj2[u].push_back(k);
26:                 adj2[k].push_back(u);
27:             }
28:         }
29:         low[u] = min(low[u], st[v]); // only 1 back edge
30:     }
31: }
32:
33: /*
34: Create new node to connect same bcc (biconnected)
35: -> shrink it as a tree
36: */
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef pair<int,int> pii;
5: const int N = 2e5+10;
6: vector<int>adj[N], st(N), low(N);
7: vector<pii>bridge;
8: int tin = 0;
9:
10: void dfs(int u, int p){
11:     st[u] = low[u] = ++tin; // init
12:
13:     for(int v: adj[u]){
14:         if(!st[v]){ // not visit -> (u-v) tree edge
15:             dfs(v,u);
16:             low[u] = min(low[u], low[v]);
17:             if(low[v] > st[u]){ // not passing though
18:                 bridge.emplace_back(u,v);
19:             }
20:         }else if(v != p){
21:             low[u] = min(low[u], st[v]); // only 1 back edge
22:         }
23:     }
24: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef pair<int,int> pii;
5: const int N = 2e5+10;
6: vector<int>adj[N], st(N), low(N), ecc(N), s;
7: int tin = 0, k = 0;
8:
9: void dfs(int u, int p){
10:     st[u] = low[u] = ++tin; // init
11:     s.emplace_back(s);
12:
13:     for(int v: adj[u]){
14:         if(!st[v]){ // not visit -> (u-v) tree edge
15:             dfs(v,u);
16:             low[u] = min(low[u], low[v]);
17:         }else if(v != p){
18:             low[u] = min(low[u], st[v]); // only 1 back edge
19:         }
20:     }
21:
22:     if(st[u]==low[u]){ // ecc vertex are descendent
23:         ++k;
24:         for(int x = -1; x!=u; s.pop_back()){
25:             x = s.back();
26:             ecc[x] = k;
27:         }
28:     }
29: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: typedef pair<ll, ll> pii;
6:
7: const ll N = 2e5+1;
8:
9: vector<int>sub(N), G[N];
10: vector<bool>vis(N,0);
11: vector<pii>adj[N];
12:
13: vector<bool>isP(N,1);
14: vector<int>pri;
15: vector<int>facts[N];
16:
17: void init(){
18:     for(ll i=2; i<N;i++){
19:         if(isP[i]){
20:             pri.push_back(i);
21:             for(ll j=i*i; j<N; j+=i)isP[j] = 0;
22:         }
23:     }
24:     for(int p : pri)for(int i=p; i<N; i+= p)facts[i].push_back(p);
25: }
26:
27:
28: ll dfs(ll u, ll p){
29:     sub[u] = 1;
30:     for(auto [v,w] : adj[u]){
31:         if(v==p || vis[v])continue;
32:         sub[u] += dfs(v,u);
33:     }
34:     return sub[u];
35: }
36:
37: ll centroid(ll u, ll p, ll sz){
38:     for(auto [v,w] : adj[u]){
39:         if(v==p || vis[v])continue;
40:         if(sub[v] > sz/2)return centroid(v,u,sz);
41:     }
42:     return u;
43: }
44:
45: ll build(ll u, ll p){
46:     ll sz = dfs(u, p); // size of subtree
47:     ll c = centroid(u,p,sz); // centroid of subtree
48:     if(p!=-1)G[p].push_back(c);
49:     vis[c] = 1;
50:     for(auto [v,w] : adj[c]){ // split
51:         if(vis[v])continue;
52:         build(v,c);
53:     }
54:     return c;
55: }
56:
57: vector<ll>a(N);
58: vector<bool>vis2(N,0);
59: ll ans = 0, len2 = 0;
60:
61: void add(int u, int p, int f, ll dis){
62:     len2 = max(len2, dis);
63:     for(auto [v,w] : adj[u]){
64:         if(vis2[v] || v == p || a[v]%f)continue;
65:         add(v,u,f,dis+1);
66:     }
67: }
68:
69: bool cmp(pii a, pii b){
70:     return a.first < b.first;
71: }
72:
73: void calc(int u){
74:     ans = max(ans, (ll)(a[u]!=1));
75:     for(int f : facts[a[u]]){
76:         ll len1 = 0;
77:         for(auto [v,w] : adj[u]){
78:             if(vis2[v] || a[v]%f)continue;
79:             len2 = 0;
80:             add(v,u,f,1);
81:             ans = max(len1+len2+1, ans);
82:             len1 = max(len1, len2);
83:         }
84:     }
85: }
86:

```

```
87: void solve(int u){
88:     vis2[u] = 1;
89:     calc(u);
90:     for(auto v : G[u]) solve(v);
91: }
92:
93: int main(){
94:     init();
95:     ios::sync_with_stdio(0);
96:     cin.tie(0);
97:     int n;
98:     cin >> n;
99:
100:    for(int i=1; i<=n; i++) cin >> a[i];
101:    for(int i=1; i<n; i++){
102:        int u, v;
103:        cin >> u >> v;
104:        adj[u].push_back({v,1});
105:        adj[v].push_back({u,1});
106:    }
107:    int rt = build(1,-1);
108:    solve(rt);
109:    cout << ans;
110: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: typedef pair<ll, ll> pii;
6:
7: const ll N = 100001;
8: const ll LG = 23;
9: vector<int>sub(N), G[N], fa(N);
10: vector<bool>vis(N,0);
11: vector<pii>adj[N];
12:
13: // lazy tag pq
14: struct que{
15:     priority_queue<int>x, y;
16:     inline void push(int a){x.push(a);}
17:     inline void del(int a){y.push(a);}
18:     inline int size(){return x.size()-y.size();}
19:     inline void pop(){
20:         while(y.size() && x.top() == y.top()) x.pop(), y.pop();
21:         x.pop();
22:     }
23:     inline int top(){
24:         while(y.size() && x.top() == y.top()) x.pop(), y.pop();
25:         return x.top();
26:     }
27:     inline int top2(){
28:         int a = top();
29:         pop();
30:         int b = top();
31:         push(a);
32:         return b;
33:     }
34: }A, B[N], C[N];
35:
36: void pusha(int x){
37:     if(B[x].size() >= 2) A.push(B[x].top()+B[x].top2());
38: }
39:
40: void dela(int x){
41:     if(B[x].size() >= 2) A.del(B[x].top()+B[x].top2());
42: }
43:
44: // Build centroid decomposition tree
45: ll dfs(ll u, ll p){
46:     sub[u] = 1;
47:     for(auto [v,w] : adj[u]){
48:         if(v==p || vis[v]) continue;
49:         sub[u] += dfs(v,u);
50:     }
51:     return sub[u];
52: }
53:
54: ll centroid(ll u, ll p, ll sz){
55:     for(auto [v,w] : adj[u]){
56:         if(v==p || vis[v]) continue;
57:         if(sub[v] > sz/2) return centroid(v,u,sz);
58:     }
59:     return u;
60: }
61:
62: ll build(ll u, ll p){
63:     ll sz = dfs(u, p); // size of subtree
64:     ll c = centroid(u,p,sz); // centroid of subtree
65:     if(p) G[p].push_back(c);
66:     fa[c] = p;
67:     vis[c] = 1;
68:     for(auto [v,w] : adj[c]){ // split
69:         if(vis[v]) continue;
70:         build(v,c);
71:     }
72:     return c;
73: }
74:
75: // LCA
76: int anc[N][LG], dep[N];
77: void precompute(int u, int p){
78:     anc[u][0] = p;
79:     dep[u] = dep[p]+1;
80:     for(int i=1; i<LG; i++) anc[u][i] = anc[anc[u][i-1]][i-1];
81:     for(auto [v,w] : adj[u]){
82:         if(v==p) continue;
83:         precompute(v,u);
84:     }
85: }
86:

```

```

87: int lca(int u, int v){
88:     if(dep[u] < dep[v])swap(u,v);
89:     for(int i=LG-1; i>=0; i--)if(dep[anc[u][i]] >= dep[v])u = anc[u][i];
90:     if(u==v)return u;
91:     for(int i=LG-1; i>=0; i--){
92:         if(anc[u][i] != anc[v][i]){
93:             u = anc[u][i];
94:             v = anc[v][i];
95:         }
96:     }
97:     return anc[u][0];
98: }
99:
100: int dis(int u, int v){
101:     int w = lca(u,v);
102:     return dep[u]+dep[v]-2*dep[w];
103: }
104:
105: // Centroid decomposition
106: vector<bool>vis2(N,0);
107:
108: void add(int u, int p,int rt){
109:     C[rt].push(dis(u,fa[rt]));
110:     for(auto [v,w] : adj[u]){
111:         if(vis2[v] || v == p)continue;
112:         add(v,u,rt);
113:     }
114: }
115:
116:
117: void solve(int u){
118:     vis2[u] = 1;
119:     B[u].push(0);
120:     add(u,fa[u],u);
121:     for(auto v : G[u]){
122:         solve(v);
123:         B[u].push(C[v].top());
124:     }
125:     pusha(u);
126: }
127:
128:
129: // Update
130: vector<bool>light(N,0);
131:
132: void on(int x){
133:     dela(x);
134:     B[x].del(0);
135:     pusha(x);
136:     int v = x;
137:     while(fa[v]){
138:         dela(fa[v]);
139:         if(C[v].size())B[fa[v]].del(C[v].top());
140:         C[v].del(dis(x,fa[v]));
141:         if(C[v].size())B[fa[v]].push(C[v].top());
142:         pusha(fa[v]);
143:         v = fa[v];
144:     }
145: }
146:
147: void off(int x){
148:     dela(x);
149:     B[x].push(0);
150:     pusha(x);
151:     int v = x;
152:     while(fa[v]){
153:         dela(fa[v]);
154:         if(C[v].size())B[fa[v]].del(C[v].top());
155:         C[v].push(dis(x,fa[v]));
156:         B[fa[v]].push(C[v].top());
157:         pusha(fa[v]);
158:         v = fa[v];
159:     }
160: }
161:
162: int main(){
163:     ios::sync_with_stdio(0);
164:     cin.tie(0);
165:     int n;
166:     cin >> n;
167:
168:     for(int i=1; i<n; i++){
169:         int u, v;
170:         cin >> u >> v;
171:         adj[u].push_back({v,1});
172:         adj[v].push_back({u,1});

```

```
173:     }
174:     precompute(1,0);
175:     int rt = build(1,0);
176:     solve(rt);
177:
178:     int m;
179:     cin >> m;
180:     int tot = n;
181:     for(int i=1; i<=m; i++){
182:         char c;
183:         int x;
184:         cin >> c;
185:         if(c=='C'){
186:             cin >> x;
187:             if(light[x]){
188:                 tot++;
189:                 off(x);
190:             }else{
191:                 tot--;
192:                 on(x);
193:             }
194:             light[x] = !light[x];
195:         }else{
196:             if(tot==0)cout << "-1\n";
197:             else if(tot==1)cout << "0\n";
198:             else cout << A.top() << "\n";
199:         }
200:     }
201: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 2e5+5;
5: vector<int>adj[N];
6:
7: void circuit(int n){
8:     if(n==0) return ;
9:
10:    vector<int>s, ans;
11:    s.push_back(0);
12:
13:    while(!s.empty()){
14:        int u = s.back();
15:        if(!adj[u].empty()){ // need to continue visit
16:            int v = adj[u].back();
17:            adj[u].pop_back();
18:            s.push_back(v);
19:        }else{
20:            ans.push_back(u);
21:            s.pop_back();
22:        }
23:    }
24:
25:    // reverse
26:    reverse(ans.begin(), ans.end());
27: }
```

```

1: #include <bits/stdc++.h>
2:
3: template<class T>
4: struct MaxFlow {
5:     struct _Edge {
6:         int to;
7:         T cap;
8:         _Edge(int to, T cap) : to(to), cap(cap) {}
9:     };
10:    int n;
11:    std::vector<_Edge> e;
12:    std::vector<std::vector<int>>> g;
13:    std::vector<int> cur, h;
14:    MaxFlow() {}
15:    MaxFlow(int n) {
16:        init(n);
17:    }
18:    void init(int n) {
19:        this->n = n;
20:        e.clear();
21:        g.assign(n, {});
22:        cur.resize(n);
23:        h.resize(n);
24:    }
25:    bool bfs(int s, int t) {
26:        h.assign(n, -1);
27:        std::queue<int> que;
28:        h[s] = 0;
29:        que.push(s);
30:        while (!que.empty()) {
31:            const int u = que.front();
32:            que.pop();
33:            for (int i : g[u]) {
34:                auto [v, c] = e[i];
35:                if (c > 0 && h[v] == -1) {
36:                    h[v] = h[u] + 1;
37:                    if (v == t) {
38:                        return true;
39:                    }
40:                    que.push(v);
41:                }
42:            }
43:        }
44:        return false;
45:    }
46:    T dfs(int u, int t, T f) {
47:        if (u == t) {
48:            return f;
49:        }
50:        auto r = f;
51:        for (int &i = cur[u]; i < int(g[u].size()); ++i) {
52:            const int j = g[u][i];
53:            auto [v, c] = e[j];
54:            if (c > 0 && h[v] == h[u] + 1) {
55:                auto a = dfs(v, t, std::min(r, c));
56:                e[j].cap -= a;
57:                e[j ^ 1].cap += a;
58:                r -= a;
59:                if (r == 0) {
60:                    return f;
61:                }
62:            }
63:        }
64:        return f - r;
65:    }
66:    void addEdge(int u, int v, T c) {
67:        g[u].push_back(e.size());
68:        e.emplace_back(v, c);
69:        g[v].push_back(e.size());
70:        e.emplace_back(u, 0);
71:    }
72:    T flow(int s, int t) {
73:        T ans = 0;
74:        while (bfs(s, t)) {
75:            cur.assign(n, 0);
76:            ans += dfs(s, t, std::numeric_limits<T>::max());
77:        }
78:        return ans;
79:    }
80:    std::vector<bool> minCut() {
81:        std::vector<bool> c(n);
82:        for (int i = 0; i < n; i++) {
83:            c[i] = (h[i] != -1);
84:        }
85:        return c;
86:    }

```

```
87:     struct Edge {
88:         int from;
89:         int to;
90:         T cap;
91:         T flow;
92:     };
93:     std::vector<Edge> edges() {
94:         std::vector<Edge> a;
95:         for (int i = 0; i < e.size(); i += 2) {
96:             Edge x;
97:             x.from = e[i + 1].to;
98:             x.to = e[i].to;
99:             x.cap = e[i].cap + e[i + 1].cap;
100:            x.flow = e[i + 1].cap;
101:            a.push_back(x);
102:        }
103:        return a;
104:    }
105: };
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: pair <int, vector<int> > hungarian(const vector<vector<int> > &a){
5:     if (a.empty()) return {0, {}};
6:     int n = (int)a.size() + 1, m = (int)a[0].size() + 1;
7:     vector <int> u(n), v(m), p(m), ans(n - 1);
8:     for (int i = 1; i < n; i++){
9:         p[0] = i;
10:        int j0 = 0;
11:        vector <int> dist(m, INT_MAX), pre(m, -1);
12:        vector <bool> done(m + 1);
13:        do {
14:            done[j0] = true;
15:            int i0 = p[j0], j1, delta = INT_MAX;
16:            for (int j = 1; j < m; j++){
17:                if (!done[j]){
18:                    auto cur = a[i0 - 1][j - 1] - u[i0] - v[j];
19:                    if (cur < dist[j])
20:                        dist[j] = cur, pre[j] = j0;
21:                    if (dist[j] < delta)
22:                        delta = dist[j], j1 = j;
23:                }
24:            }
25:            for (int j = 0; j < m; j++){
26:                if (done[j])
27:                    u[p[j]] += delta, v[j] -= delta;
28:                else
29:                    dist[j] -= delta;
30:            }
31:            j0 = j1;
32:        } while (p[j0]);
33:        while (j0){
34:            int j1 = pre[j0];
35:            p[j0] = p[j1], j0 = j1;
36:        }
37:    }
38:    for (int j = 1; j < m; j++){
39:        if (p[j])
40:            ans[p[j] - 1] = j - 1;
41:    }
42:    return {-v[0], ans};
43: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: typedef pair<ll, ll> pii;
6:
7: int main(){
8:     ll h, x, y, z;
9:     cin >> h >> x >> y >> z;
10:
11:     ll dis[x];
12:     for(int i=0; i<x; i++)dis[i] = 1e18;
13:     dis[0] = 0;
14:
15:     priority_queue<pii, vector<pii>, greater<pii>>pq;
16:     pq.push({0,0});
17:
18:     bool vis[x];
19:     memset(vis, 0, sizeof vis);
20:
21:     while(!pq.empty()){
22:         ll w = pq.top().first;
23:         ll u = pq.top().second;
24:         pq.pop();
25:         if(vis[u])continue;
26:         vis[u] = 1;
27:
28:         // use y
29:         if(dis[(u+y)%x] > dis[u]+y){
30:             dis[(u+y)%x] = dis[u]+y;
31:             pq.push({dis[(u+y)%x], (u+y)%x});
32:         }
33:
34:         // use z
35:         if(dis[(u+z)%x] > dis[u]+z){
36:             dis[(u+z)%x] = dis[u]+z;
37:             pq.push({dis[(u+z)%x], (u+z)%x});
38:         }
39:     }
40:
41:     ll ans = 0;
42:     for(int i=0; i<x; i++)if(h>=dis[i])ans += (h-dis[i])/x+1;
43:     cout << ans;
44: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 2e5+5;
5: vector<bool>ins(N,0);
6: vector<int>adj[N], st(N), low(N), scc(N), s;
7: int tin = 0, k = 0;
8:
9: void dfs(int u){
10:     st[u] = low[u] = ++tin;
11:     s.emplace_back(u); // store descendent
12:     ins[u] = 1;
13:
14:     for(int v : adj[u]){
15:         if(!st[v]){
16:             dfs(v);
17:             low[u] = min(low[u], low[v]);
18:         }else if(ins[v]){ // back / forward edge
19:             low[u] = min(low[u], st[v]);
20:         }
21:     }
22:     if(st[u]==low[u]){
23:         ++k;
24:         for(int x=-1; x!=u; s.pop_back()){
25:             x = s.back();
26:             scc[x] = k;
27:             ins[x] = 0;
28:         }
29:     }
30: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: typedef pair<ll, ll> pii;
6:
7: const int N = 2e5+10;
8: vector<pii>adj[N];
9:
10: struct edge{
11:     int u, v, w;
12: };
13: vector<edge>E(2*N);
14:
15: vector<ll>dij1(int s, int n){
16:     vector<bool>vis(n+1,0);
17:     vector<ll>d(n+1,1e18);
18:     d[s] = 0;
19:     priority_queue<pii, vector<pii>, greater<pii>>pq;
20:     pq.push({d[s],s});
21:     while(!pq.empty()){
22:         int u = pq.top().second;
23:         pq.pop();
24:         if(vis[u])continue;
25:         vis[u] = 1;
26:         for(auto [v,i] : adj[u]){
27:             ll w = E[i].w;
28:             if(d[v] > d[u]+w){
29:                 d[v] = d[u]+w;
30:                 pq.push({d[v],v});
31:             }
32:         }
33:     }
34:     return d;
35: }
36:
37: int main(){
38:     ios::sync_with_stdio(0);
39:     cin.tie(0);
40:     int n, m;
41:     cin >> n >> m;
42:
43:     for(int i=1; i<=m; i++){
44:         cin >> E[i].u >> E[i].v >> E[i].w;
45:         adj[E[i].u].push_back({E[i].v, i});
46:         adj[E[i].v].push_back({E[i].u, i});
47:     }
48:
49:     vector<ll>d1 = dij1(1,n), d2 = dij1(n,n);
50:     vector<int>l(n+1,1e9), r(n+1,0);
51:     vector<bool>used(m+1,0);
52:
53:     // l[i] = first edge not in 1->i
54:     // r[i] = last edge not in i->n
55:     l[1] = 1;
56:     int tot = 0;
57:     vector<int>E2(m+1,0);
58:     for(int u=1; u!=n;){
59:         for(auto [v,i] : adj[u]){
60:             ll w = E[i].w;
61:             if(d2[v]+w == d2[u]){
62:                 E2[tot+1] = i;
63:                 u = v;
64:                 used[i] = 1;
65:                 break;
66:             }
67:         }
68:         r[u] = ++tot;
69:         l[u] = tot+1;
70:     }
71:
72:     vector<int>ord(n);
73:     iota(ord.begin(), ord.end(),1);
74:
75:     sort(ord.begin(), ord.end(), [&](int x, int y){
76:         return d1[x] < d1[y];
77:     });
78:     for(int u : ord){
79:         for(auto [v,i] : adj[u]){
80:             if(used[i])continue;
81:             ll w = E[i].w;
82:             if(d1[v]==d1[u]+w)l[v] = min(l[v],l[u]);
83:         }
84:     }
85:
86:     sort(ord.begin(), ord.end(), [&](int x, int y){

```

```
87:         return d2[x] < d2[y];
88:     });
89:     for(int u : ord){
90:         for(auto [v,i] : adj[u]){
91:             if(used[i])continue;
92:             ll w = E[i].w;
93:             if(d2[v] == d2[u]+w)r[v] = max(r[v],r[u]);
94:         }
95:     }
96:
97:     vector<ll>p1[tot+1], p2[tot+1];
98:     for(int i=1; i<=m; i++){
99:         if(used[i])continue;
100:         auto [u,v,w] = E[i];
101:         if(l[u] <= r[v]){
102:             p1[l[u]].push_back(d1[u]+d2[v]+w);
103:             p2[r[v]].push_back(d1[u]+d2[v]+w);
104:         }
105:         if(l[v] <= r[u]){
106:             p1[l[v]].push_back(d1[v]+d2[u]+w);
107:             p2[r[u]].push_back(d1[v]+d2[u]+w);
108:         }
109:     }
110:
111:     multiset<ll>s;
112:     for(int i=1; i<=tot; i++){
113:         for(ll w : p1[i])s.insert(w);
114:         if(*s.begin() == d1[n])used[E2[i]] = 0;
115:         for(ll w : p2[i])s.erase(s.find(w));
116:     }
117:
118:     for(int i=1; i<=m; i++){
119:         if(used[i])cout << "Yes\n";
120:         else cout << "No\n";
121:     }
122:
123:     // if(ans == d1[n])cnt = m;
124:     // cout << ans << " " << cnt;
125: }
126:
127: /*
128: 1. dijkstra shortest path
129: 2. Build shortest path tree
130: 3. Check each shortest path edge when removed have what answer
131: 4. Observe that new path = 1 -> l[u] -> (u,v) -> r[v] -> n
132: 5. (u,v) will be used only if removed edge [l[u],r[v]]
133: 6. use multiset to maintain ans when edge i is removed
134: */
```

```
1: #include <bits/stdc++.h>
2: #define all(a) a.begin(), a.end()
3: #define sz(a) (int)a.size()
4: #define int long long
5: #define Tp template <typename T>
6: #define Ts template <typename T, typename... Ar>
7: using namespace std;
8:
9: template <class T>
10: using vc = vector<T>;
11: template <class T>
12: using vvc = vector<vc<T>>;
13: template <class T>
14: using vvvc = vector<vvc<T>>;
15: template <class T>
16: using vvvvc = vector<vvvc<T>>;
17: typedef pair<int, int> pii;
18:
19: template <typename T>
20: ostream &operator<<(ostream &os, const pair<T, T> &data) { return os << data.first << ' ' <<
data.second; }
21: template <typename T>
22: istream &operator>>(istream &is, pair<T, T> &data) { return is >> data.first >> data.second; }
23: template <typename T>
24: ostream &operator<<(ostream &os, const vector<T> &data)
25: {
26:     for (T x : data)
27:         os << x << ' ';
28:     return os;
29: }
30: template <typename T>
31: istream &operator>>(istream &is, vector<T> &data)
32: {
33:     for (T &x : data)
34:         is >> x;
35:     return is;
36: }
37:
38: template <class T>
39: inline void chmin(T &x, T y)
40: {
41:     if (y < x)
42:         x = y;
43: }
44: template <class T>
45: inline void chmax(T &x, T y)
46: {
47:     if (y > x)
48:         x = y;
49: }
50: namespace Debug
51: {
52:     Tp void _debug(char *f, T t) { cerr << f << '=' << t << endl; }
53:     Ts void _debug(char *f, T x, Ar... y)
54:     {
55:         while (*f != ',')
56:             cerr << *f++;
57:         cerr << '=' << x << ", ";
58:         _debug(f + 1, y...);
59:     }
60: #ifdef LOCAL
61: #define err(...) _debug((char *)__VA_ARGS__, __VA_ARGS__)
62: #else
63: #define err(...) void()
64: #endif
65: }
66: using namespace Debug;
67:
68: mt19937_64 rng(chrono::steady_clock::time_point::clock().now().time_since_epoch().count());
69:
70: const int N = 5e5 + 10;
71: const int INF = 1e18;
72: const int MOD = 998244353;
73: const int LOG = 20;
74:
75: typedef __int128_t ll;
76:
77: int solve()
78: {
79: }
80:
81: int32_t main()
82: {
83:     ios::sync_with_stdio(false);
84:     cin.tie(0);
85: }
```

```
86:         int tt;
87:         cin >> tt;
88:         for (int i = 1; i <= tt; i++)
89:         {
90:             solve();
91:         }
92: }
93:
94: // 0 2 1 -> 1 1 0 0 0
95: // 2 1 0 -> 0 0 0 0 2
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: #define int long long
5: typedef array<int,2>ar;
6:
7: const int N = 1e5+5;
8: const int LG = 60;
9: const int LG2 = 23;
10:
11: struct linearBasis{
12:     int d[LG];
13:     int num;
14:
15:     linearBasis() {clear();}
16:
17:     void clear(){
18:         memset(d, 0, sizeof d);
19:         num = 0;
20:     }
21:
22:     bool add(int x){
23:         for(int i=LG-1; i>=0; i--){
24:             if((x>>i)&1){
25:                 if(!d[i]){
26:                     d[i] = x;
27:                     num++;
28:                     for(int j=i-1; j>=0; j--)if(d[j]&(1ll<<j))d[i] ^= d[j];
29:                     for(int j=i+1; j<LG; j++)if(d[j]&(1ll<<i))d[j] ^= d[i];
30:                     return 1;
31:                 }
32:                 x ^= d[i];
33:             }
34:         }
35:         return 0;
36:     }
37:
38:     int queryMax(){
39:         int res = 0;
40:         for(int i=LG-1; i>=0; i--)if((res^d[i])>res)res ^= d[i];
41:         return res;
42:     }
43:
44:     int queryMin(){
45:         for(int i=0; i<LG; i++)if(d[i])return d[i];
46:         return 0;
47:     }
48:
49:     vector<int> getIndependent(){
50:         vector<int> res;
51:         for(int i=0; i<LG; i++)if(d[i])res.push_back(d[i]);
52:         return res;
53:     }
54:
55:     int querykth(int k){
56:         vector<int> vec = getIndependent();
57:         int sz = vec.size();
58:         if(k >= (1ll<<sz))return -1;
59:         int res = 0;
60:         for(int i=0; i<sz; i++)if(k&(1ll<<i))res ^= vec[i];
61:         return res;
62:     }
63:
64:     friend linearBasis merge(const linearBasis &a, const linearBasis &b){
65:         linearBasis res = a;
66:         for(int i=0; i<LG; i++)if(b.d[i])res.add(b.d[i]);
67:         return res;
68:     }
69: };
70:
71: int32_t main(){
72:     int n, q;
73:     cin >> n >> q;
74:
75:     vector<int>a(n+1), dep(n+1), lg(n+1,0);
76:     for(int i=1; i<=n; i++)cin >> a[i];
77:     for(int i=2; i<=n; i++)lg[i] = lg[i/2]+1;
78:
79:     vector<vector<int>>g(n+1);
80:     for(int i=1; i<n; i++){
81:         int u, v;
82:         cin >> u >> v;
83:         g[u].push_back(v);
84:         g[v].push_back(u);
85:     }
86:

```

```

87:     vector<vector<int>>>anc(n+1,vector<int>(LG2));
88:     vector<vector<linearBasis>>>sum(n+1,vector<linearBasis>(LG2));
89:
90:     auto dfs = [&](auto&&self, int u, int p) -> void{
91:         dep[u] = dep[p]+1;
92:         anc[u][0] = p;
93:         sum[u][0].add(a[u]);
94:         for(int i=1; i<LG2; i++){
95:             anc[u][i] = anc[anc[u][i-1]][i-1];
96:             sum[u][i] = merge(sum[u][i-1],sum[anc[u][i-1]][i-1]);
97:         }
98:         for(int v : g[u]){
99:             if(v==p)continue;
100:             self(self, v, u);
101:         }
102:     };
103:     dfs(dfs,1,0);
104:
105:     auto lca = [&](int x, int y) -> int{
106:         if(dep[x] < dep[y])swap(x,y);
107:         for(int i=LG2-1; i>=0; i--)if(dep[anc[x][i]] >= dep[y])x = anc[x][i];
108:         if(x==y)return x;
109:         for(int i=LG2-1; i>=0; i--){
110:             if(anc[x][i] != anc[y][i]){
111:                 x = anc[x][i];
112:                 y = anc[y][i];
113:             }
114:         }
115:         return anc[x][0];
116:     };
117:
118:     auto kth = [&](int x, int k) -> int{
119:         for(int i=LG2-1; i>=0; i--)if(k&(1ll<<i))x = anc[x][i];
120:         return x;
121:     };
122:
123:     while(q--){
124:         int x, y;
125:         cin >> x >> y;
126:         int w = lca(x,y);
127:
128:         linearBasis lb;
129:         lb.add(a[w]);
130:         int lenx = lg[dep[x]-dep[w]], leny = lg[dep[y]-dep[w]];
131:         lb = merge(lb,sum[x][lenx]);
132:         lb = merge(lb,sum[kth(x,dep[x]-dep[w]-(1ll<<lenx))][lenx]);
133:         lb = merge(lb,sum[y][leny]);
134:         lb = merge(lb,sum[kth(y,dep[y]-dep[w]-(1ll<<leny))][leny]);
135:
136:         cout << lb.queryMax() << "\n";
137:     }
138: }

```

```
1: typedef long long ll;
2:
3: ll exgcd(ll a, ll b, ll &x, ll &y, ll d = 0){ // a*x + b*y = d = gcd(a,b)
4:     if(b==0)x = 1, y = 0, d = a;
5:     else d = exgcd(b, a%b, y, x), y-= a/b*x;
6:     return d;
7: }
8:
9: ll inv(ll a, ll m, ll x = 0, ll y = 0){
10:     exgcd(a,m,x,y);
11:     return (x%m+m)%m;
12: }
13:
14: ll crt(vector<ll>&m, vector<ll>&a){
15:     int n = m.size();
16:     ll mul = 1;
17:     for(int i=0; i<n; i++)mul *= m[i];
18:
19:     ll ans = 0;
20:     for(int i=0; i<n; i++){
21:         ll t = mul/m[i], c = inv(t,m[i]);
22:         ans = (ans+((a[i]*t)%mul)*c%mul)%mul;
23:     }
24:
25:     return ans;
26: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: #define int long long
5: const int MOD = 998244353;
6: const int N = 1e5+1;
7:
8: vector<int> precompute_phi(int n){
9:     vector<int> phi(n+1);
10:    iota(phi.begin(), phi.end(), 0);
11:    for(int i=2; i<=n; i++){
12:        if(phi[i] == i){
13:            for(int j=i; j<=n; j+=i){
14:                phi[j] -= phi[j]/i;
15:            }
16:        }
17:    }
18:    return phi;
19: }
20:
21:
22:
23: int32_t main(){
24:     int n;
25:     cin >> n;
26:
27:     vector<int> fact[N];
28:     for(int i=1; i<N; i++) for(int j=i; j<N; j+=i) fact[j].push_back(i);
29:
30:     vector<int> a(n+1), phi = precompute_phi(N), g(N,0);
31:     for(int i=1; i<=n; i++) cin >> a[i];
32:
33:     vector<int> po2(n+1,1);
34:     for(int i=1; i<=n; i++) po2[i] = (po2[i-1]*2)%MOD;
35:
36:     int ans = 0;
37:     for(int i=1; i<=n; i++){
38:         ans = (ans*2)%MOD;
39:         for(auto d : fact[a[i]]) ans = (ans+phi[d]*g[d])%MOD;
40:         for(auto d : fact[a[i]]) g[d] = (g[d] + po2[i-1])%MOD;
41:         cout << ans << "\n";
42:     }
43: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 2e5+5;
5: const int LG = 60;
6:
7: struct linearBasis{
8:     int d[LG];
9:     int num;
10:
11:     linearBasis() {clear();}
12:
13:     void clear(){
14:         memset(d, 0, sizeof d);
15:         num = 0;
16:     }
17:
18:     bool add(int x){
19:         for(int i=LG-1; i>=0; i--){
20:             if((x>>i)&&1){
21:                 if(!d[i]){
22:                     d[i] = x;
23:                     num++;
24:                     for(int j=i-1; j>=0; j--)if(d[i]&(1ll<<j))d[i] ^= d[j];
25:                     for(int j=i+1; j<LG; j++)if(d[j]&(1ll<<i))d[j] ^= d[i];
26:                     return 1;
27:                 }
28:                 x ^= d[i];
29:             }
30:         }
31:         return 0;
32:     }
33:
34:     int queryMax(){
35:         int res = 0;
36:         for(int i=LG-1; i>=0; i--)if(res^d[i]>res)res ^= d[i];
37:         return res;
38:     }
39:
40:     int queryMin(){
41:         for(int i=0; i<LG; i++)if(d[i])return d[i];
42:         return 0;
43:     }
44:
45:     vector<int> getIndependent(){
46:         vector<int> res;
47:         for(int i=0; i<LG; i++)if(d[i])res.push_back(d[i]);
48:         return res;
49:     }
50:
51:     int querykth(int k){
52:         vector<int> vec = getIndependent();
53:         int sz = vec.size();
54:         if(k >= (1ll<<sz))return -1;
55:         int res = 0;
56:         for(int i=0; i<sz; i++)if(k&(1ll<<i))res ^= vec[i];
57:         return res;
58:     }
59:
60:     friend linearBasis merge(const linearBasis &a, const linearBasis &b){
61:         linearBasis res = a;
62:         for(int i=0; i<LG; i++)if(b.d[i])res.add(b.d[i]);
63:         return res;
64:     }
65: };
```

```

1: template <int P>
2: class mod_int {
3:     using Z = mod_int;
4:
5: private:
6:     static int mo(int x) { return x < 0 ? x + P : x; }
7:
8: public:
9:     int x;
10:    int val() const { return x; }
11:    mod_int() : x(0) {}
12:    template <class T>
13:    mod_int(const T &x_) : x(x_ >= 0 && x_ < P ? static_cast<int>(x_) : mo(static_cast<int>(x_ % P))) {}
14:    bool operator==(const Z &rhs) const { return x == rhs.x; }
15:    bool operator!=(const Z &rhs) const { return x != rhs.x; }
16:    Z operator-() const { return Z(x ? P - x : 0); }
17:    Z pow(long long k) const {
18:        Z res = 1, t = *this;
19:        while (k) {
20:            if (k & 1) res *= t;
21:            if (k >= 1) t *= t;
22:        }
23:        return res;
24:    }
25:    Z &operator++() {
26:        x < P - 1 ? ++x : x = 0;
27:        return *this;
28:    }
29:    Z &operator--() {
30:        x ? --x : x = P - 1;
31:        return *this;
32:    }
33:    Z operator++(int) {
34:        Z ret = x;
35:        x < P - 1 ? ++x : x = 0;
36:        return ret;
37:    }
38:    Z operator--(int) {
39:        Z ret = x;
40:        x ? --x : x = P - 1;
41:        return ret;
42:    }
43:    Z inv() const { return pow(P - 2); }
44:    Z &operator+=(const Z &rhs) {
45:        (x += rhs.x) >= P && (x -= P);
46:        return *this;
47:    }
48:    Z &operator-=(const Z &rhs) {
49:        (x -= rhs.x) < 0 && (x += P);
50:        return *this;
51:    }
52:    Z &operator*=(const Z &rhs) {
53:        x = 1ULL * x * rhs.x % P;
54:        return *this;
55:    }
56:    Z &operator/=(const Z &rhs) { return *this *= rhs.inv(); }
57:    #define setO(T, o) \
58:        friend T operator o(const Z &lhs, const Z &rhs) {\ \
59:            Z res = lhs; \
60:            return res o## = rhs; \
61:        }
62:    setO(Z, +) setO(Z, -) setO(Z, *) setO(Z, /)
63:    #undef setO
64: };
65: const int P = 998244353;
66: using Z = mod_int<P>;
67:

```

```
1: #include <bits/stdc++.h>
2: #define int long long
3: using namespace std;
4:
5: const int MOD = 998244353; // bruh
6:
7: int pow(int x, int y){
8:     int res = 1;
9:     x = x%MOD;
10:    while(y > 0){
11:        if(y&1) res = (res*x)%MOD;
12:        y = y >> 1;
13:        x = (x*x)%MOD;
14:    }
15:    return res;
16: }
17:
18: int inv(int x){
19:     return pow(x, MOD-2);
20: }
21:
22: const int N = 1e7+1;
23: vector<int> fact{1}, invfact{1};
24:
25: int C(int n, int r){
26:     if(n < r) return 0;
27:     for(int i = fact.size(); i <= n; i++){
28:         fact.push_back(fact.back() * i % MOD);
29:         invfact.push_back(inv(fact.back()));
30:     }
31:     return ((fact[n]*invfact[r])%MOD*invfact[n-r])%MOD;
32: }
33:
34: void precompute(){
35:     fact.resize(N);
36:     invfact.resize(N);
37:     fact[0] = invfact[0] = 1;
38:     for(int i = 1; i<N; i++) fact[i] = (fact[i-1]*i)%MOD;
39:     invfact[N-1] = inv(fact[N-1]);
40:     for(int i=N-2; i>=1; i--) invfact[i] = invfact[i+1]*(i+1)%MOD;
41:
42: }
```

```

1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: #define int long long
5:
6: const int N = 2e5+5;
7: const int LG = 60;
8:
9: struct linearBasis{
10:     int d[LG], pos[LG], dc[LG];
11:
12:     linearBasis() {clear();}
13:
14:     void clear(){
15:         memset(d, 0, sizeof d);
16:         memset(dc, 0, sizeof dc);
17:         memset(pos, 0, sizeof pos);
18:     }
19:
20:     bool add(int x, int p){ // edit the add
21:         for(int i=LG-1; i>=0; i--){
22:             if((x>>i)&1){
23:                 if(!d[i]){
24:                     d[i] = x;
25:                     pos[i] = p;
26:                     return 1;
27:                 }
28:                 if(pos[i] < p){
29:                     swap(pos[i], p);
30:                     swap(d[i], x);
31:                 }
32:                 x ^= d[i];
33:             }
34:         }
35:         return 0;
36:     }
37:
38:     void rebuild(){
39:         memcpy(dc,d,sizeof d);
40:         for(int i=LG-1; i>=0; i--){
41:             if(!dc[i])continue;
42:             for(int j=i-1; j>=0; j--){
43:                 if(dc[i] & (1ll<<j))dc[i] ^= dc[j];
44:             }
45:         }
46:     }
47:
48:     int querykth(int k){
49:         rebuild();
50:         vector<int>vec;
51:         for(int i=0; i<LG; i++)if(dc[i])vec.push_back(dc[i]);
52:         int sz = vec.size();
53:         if(k >= (1ll<<sz))return -1;
54:         int res = 0;
55:         for(int i=0; i<sz; i++)if(k&(1ll<<i))res ^= vec[i];
56:         return res;
57:     }
58: };
59:
60:
61: struct prefixLinearBasis{
62:     vector<linearBasis> lbs;
63:     prefixLinearBasis() {lbs.emplace_back();}
64:
65:     void add(int x){
66:         linearBasis lb = lbs.back();
67:         lb.add(x,lbs.size());
68:         lbs.emplace_back(lb);
69:     }
70:
71:     int queryMax(int l, int r){
72:         int res = 0;
73:         for(int i=LG-1; i>=0; i--){
74:             if(lbs[r].pos[i]<l)continue;
75:             if((res^lbs[r].d[i])>res)res ^= lbs[r].d[i];
76:         }
77:         return res;
78:     }
79:
80:     int queryMin(int l, int r){
81:         for(int i=0; i<LG; i++){
82:             if(lbs[r].pos[i]<l)continue;
83:             if(lbs[r].d[i])return lbs[r].d[i];
84:         }
85:         return 0;
86:     }

```

```
87: }plb;
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: int main(){
5:     while(1){
6:         system("data.exe > in.txt");
7:         system("main.exe < in.txt > main.txt");
8:         double begin = clock();
9:         system("test.exe < in.txt > test.txt");
10:        double end = clock();
11:        if(system("fc main.txt test.txt")){
12:            cout << "WA!";
13:            break;
14:        }
15:        if(end-begin > 1000){
16:            cout << "TLE!";
17:            break;
18:        }
19:    }
20: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: int _mergeSort(int arr[], int temp[], int left, int right);
5: int merge(int arr[], int temp[], int left, int mid, int right);
6: int mergeSort(int arr[], int array_size){
7:     int temp[array_size];
8:     return _mergeSort(arr, temp, 0, array_size - 1);
9: }
10:
11: int _mergeSort(int arr[], int temp[], int left, int right) {
12:     int mid, inv_count = 0;
13:     if (right > left) {
14:         mid = (right + left) / 2;
15:         inv_count += _mergeSort(arr, temp, left, mid);
16:         inv_count += _mergeSort(arr, temp, mid + 1, right);
17:         inv_count += merge(arr, temp, left, mid + 1, right);
18:     }
19:     return inv_count;
20: }
21:
22: int merge(int arr[], int temp[], int left, int mid, int right) {
23:     int i, j, k;
24:     int inv_count = 0;
25:
26:     i = left;
27:     j = mid;
28:     k = left;
29:     while ((i <= mid - 1) && (j <= right)) {
30:         if (arr[i] <= arr[j]) {
31:             temp[k++] = arr[i++];
32:         }
33:         else {
34:             temp[k++] = arr[j++];
35:             inv_count = inv_count + (mid - i);
36:         }
37:     }
38:     while (i <= mid - 1) temp[k++] = arr[i++];
39:     while (j <= right) temp[k++] = arr[j++];
40:     for (i = left; i <= right; i++) arr[i] = temp[i];
41:     return inv_count;
42: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef long long ll;
5: const ll MOD = 998244353;
6:
7: class Mint {
8: public:
9:     ll value;
10:
11:     Mint(ll v = 0) : value((v % MOD + MOD) % MOD) {}
12:
13:     Mint operator+(const Mint &other) const {
14:         return Mint(value + other.value);
15:     }
16:
17:     Mint operator-(const Mint &other) const {
18:         return Mint(value - other.value);
19:     }
20:
21:     Mint operator*(const Mint &other) const {
22:         return Mint(value * other.value);
23:     }
24:
25:     Mint operator/(const Mint &other) const {
26:         return *this * other.inverse();
27:     }
28:
29:     Mint inverse() const {
30:         ll a = value, b = MOD, u = 1, v = 0;
31:         while (b) {
32:             ll q = a / b;
33:             a -= q * b; swap(a, b);
34:             u -= q * v; swap(u, v);
35:         }
36:         return Mint(u);
37:     }
38:
39:     Mint pow(ll exp) const {
40:         Mint result = 1, base = *this;
41:         while (exp) {
42:             if (exp % 2) result = result * base;
43:             base = base * base;
44:             exp /= 2;
45:         }
46:         return result;
47:     }
48:
49:     friend ostream &operator<<(ostream &os, const Mint &m) {
50:         return os << m.value;
51:     }
52: };
53:
54: Mint f(ll x, ll y, ll m) {
55:     Mint res = 0;
56:     if (x == 0 || y == 0) return 0;
57:
58:     { // odd row
59:         ll r = (x + 1) / 2, c = (y + 1) / 2;
60:         res = res + Mint(c) * m * r * (r - 1);
61:         res = res + Mint(r) * c * c;
62:     }
63:
64:     { // even row
65:         ll r = x / 2, c = y / 2;
66:         res = res + Mint(c) * m * r * r;
67:         res = res + Mint(r) * c * (c + 1);
68:     }
69:
70:     return res;
71: }
72:
73: int main() {
74:     ll n, m;
75:     cin >> n >> m;
76:
77:     int q;
78:     cin >> q;
79:
80:     while (q--) {
81:         ll a, b, c, d;
82:         cin >> a >> b >> c >> d;
83:         cout << (f(b, d, m) - f(b, c - 1, m) - f(a - 1, d, m) + f(a - 1, c - 1, m)).value << "\n";
84:     }
85: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: typedef pair<int,int> pii;
5:
6: const int N = 2e5+5;
7: const int M = 1e6+5;
8: const int S = 505;
9:
10: int f[S][S];
11: int a[N], b[N], y[N], c[N], d[N], z[N], l[N], r[N], ans[N];
12: vector<pii>g[M];
13: vector<int>qry[M];
14:
15: int par[S*S];
16:
17: int find(int v){
18:     return (par[v] == v) ? v : par[v] = find(par[v]);
19: }
20:
21: void merge(int x, int y){
22:     int fx = find(x), fy = find(y);
23:     if(fx != fy)par[fx] = fy;
24: }
25:
26: int main(){
27:     int h, w;
28:     cin >> h >> w;
29:
30:     for(int i=1; i<=h; i++)for(int j=1; j<=w; j++)cin >> f[i][j];
31:
32:     int q;
33:     cin >> q;
34:     for(int i=1; i<=q; i++)cin >> a[i] >> b[i] >> y[i] >> c[i] >> d[i] >> z[i];
35:
36:     for(int i=1; i<=h; i++){
37:         for(int j=1; j<=w; j++){
38:             if(i+1<=h)g[min(f[i][j],f[i+1][j])].push_back({i*w+j,(i+1)*w+j});
39:             if(j+1<=w)g[min(f[i][j],f[i][j+1])].push_back({i*w+j,i*w+(j+1)});
40:         }
41:     }
42:
43:     const int lim = 1e6;
44:     for(int i=1; i<=q; i++)l[i] = 1, r[i] = lim;
45:
46:     while(1){
47:         bool fg = 1;
48:         for(int i=0; i<=lim; i++)qry[i].clear();
49:         for(int i=1; i<=q; i++){
50:             if(l[i] > r[i])continue;
51:             int m = (l[i]+r[i])/2;
52:             qry[m].push_back(i);
53:             fg = 0;
54:         }
55:         if(fg)break;
56:         for(int i=0; i<S*S; i++)par[i] = i;
57:         for(int i=lim; i>=0; i--){
58:             for(auto [u,v] : g[i])merge(u,v);
59:             for(auto id : qry[i]){
60:                 int u = a[id]*w+b[id], v = c[id]*w+d[id];
61:                 if(find(u) == find(v)){
62:                     ans[id] = i;
63:                     l[id] = i+1;
64:                 }else r[id] = i-1;
65:             }
66:         }
67:     }
68:
69:     for(int i=1; i<=q; i++)cout << y[i]+z[i]-2*min({y[i],z[i],ans[i]}) << "\n";
70: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: const int N = 2e5+5;
5: const int MOD = 998244353;
6:
7: struct AhoCorasick{
8:     struct node{
9:         int son[26];
10:        int fail;
11:        int deg;
12:        int idx;
13:        int ans = 0;
14:        int lst = 0;
15:
16:        void init(){
17:            memset(son, 0, sizeof son);
18:            ans = fail = idx = 0;
19:        }
20:    }tr[N];
21:
22:    int ans[N];
23:    int tot, pidx;
24:
25:    void init(){
26:        tot = pidx = 0;
27:        tr[0].init();
28:    }
29:
30:    void insert(string s, int& idx, int id){
31:        int u = 0;
32:        for(char c : s){
33:            int &son = tr[u].son[c-'a'];
34:            if(!son)son = ++tot, tr[son].init();
35:            u = son;
36:        }
37:        if(!tr[u].idx)tr[u].idx = ++pidx;
38:        tr[u].lst |= 1 << id;
39:        idx = tr[u].idx;
40:    }
41:
42:    void build(){
43:        queue<int>q;
44:        for(int i=0; i<26; i++)if(tr[0].son[i])q.push(tr[0].son[i]);
45:        while(q.size()){
46:            int u = q.front();
47:            q.pop();
48:            for(int i=0; i<26; i++){
49:                if(tr[u].son[i]){
50:                    tr[tr[u].son[i]].fail = tr[tr[u].fail].son[i];
51:                    tr[tr[tr[u].fail].son[i]].deg++;
52:                    q.push(tr[u].son[i]);
53:                }else{
54:                    tr[u].son[i] = tr[tr[u].fail].son[i];
55:                }
56:            }
57:        }
58:    }
59:
60:    void query(string t){
61:        int u = 0;
62:        for(char c : t){
63:            u = tr[u].son[c-'a'];
64:            tr[u].ans++;
65:        }
66:    }
67:
68:    void topu(){
69:        queue<int>q;
70:        for(int i=0; i<=tot; i++)if(!tr[i].deg)q.push(i);
71:        while(q.size()){
72:            int u = q.front();
73:            q.pop();
74:            ans[tr[u].idx] = tr[u].ans;
75:            int v = tr[u].fail;
76:            tr[v].ans += tr[u].ans;
77:            if(--tr[v].deg)q.push(v);
78:        }
79:    }
80: }AC;
81:
82: int dp[2][1<<8][N];
83:
84: int main(){
85:     int n, l;
86:     cin >> n >> l;
```

```
87:
88:     vector<int>idx(n);
89:     AC.init();
90:     for(int i=0; i<n; i++){
91:         string s;
92:         cin >> s;
93:         AC.insert(s,idx[i], i);
94:         AC.ans[idx[i]] = 0;
95:     }
96:     AC.build();
97:
98:     dp[1][0][0] = 1;
99:     for(int i = 0; i < 1; i++){
100:         for(int bit = 0; bit < (1<<n); bit++){
101:             for(int v = 0; v < AC.tot; v++){
102:                 for(int k = 0; k < 26; k++){
103:                     int to = AC.tr[v].son[k];
104:                     int nbit = bit | AC.tr[to].lst;
105:                     dp[i&1][nbit][to] = (dp[i&1^1][bit][v] + dp[i&1][nbit][to]) % MOD;
106:                 }
107:             }
108:         }
109:
110:         for(int bit = 0; bit < (1<<n); bit++){
111:             for(int v = 0; v <= AC.tot; v++){
112:                 dp[i&1^1][bit][v] = 0;
113:             }
114:         }
115:     }
116:
117:     int ans = 0;
118:     for(int i = 0; i <= AC.tot; i++) ans = (ans + dp[l&1^1][(1<<n)-1][i]);
119:     cout << ans << '\n';
120: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: vector<int> prefix_func(string s){
5:     int n = s.length();
6:     vector<int> pi(n);
7:     for(int i=1, m=0; i<n; i++){
8:         while(m > 0 && s[i] != s[m]) m = pi[m-1];
9:         if(s[i] == s[m]) m++;
10:         pi[i] = m;
11:     }
12:     return pi;
13: }
14:
15: vector<int> kmp(string s, string t){
16:     vector<int> pi = prefix_func(t);
17:     vector<int> res;
18:     int n = s.length(), n2 = t.length();
19:     for(int i=0, m=0; i<n; i++){
20:         while(m > 0 && s[i] != t[m]) m = pi[m-1];
21:         if(s[i] == t[m]) m++;
22:         if(m==n2){
23:             res.push_back(i-n2+1);
24:             m = pi[m-1];
25:         }
26:     }
27:     return res;
28: }
```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: vector<int> manacher_odd(string s) {
5:     int n = s.size();
6:     s = "$" + s + "^";
7:     vector<int> p(n + 2);
8:     int l = 1, r = 1;
9:     for(int i = 1; i <= n; i++) {
10:         p[i] = max(0, min(r - i, p[l + (r - i)]));
11:         while(s[i - p[i]] == s[i + p[i]]) {
12:             p[i]++;
13:         }
14:         if(i + p[i] > r) {
15:             l = i - p[i], r = i + p[i];
16:         }
17:     }
18:     return vector<int>(begin(p) + 1, end(p) - 1);
19: }
20:
21: vector<int> manacher(string s) {
22:     string t;
23:     for(auto c: s) {
24:         t += string("#") + c;
25:     }
26:     auto res = manacher_odd(t + "#");
27:     return vector<int>(begin(res) + 1, end(res) - 1);
28: }
29:
30:
31: int main(){
32:     string s;
33:     cin >> s;
34:     auto v = manacher(s);
35:     for(auto& x : s) x--;
36:
37:     int q;
38:     cin >> q;
39:     while(q--){
40:         int l, r;
41:         cin >> l >> r;
42:         l--, r--;
43:
44:         if(v[l + r] >= (r - l + 1)) cout << "Palindrome\n";
45:         else cout << "No\n";
46:     }
47: }
```

```

1: #include <bits/stdc++.h>
2: #define all(a) (a).begin(), (a).end()
3: #define sz(a) (long long) (a).size()
4: using namespace std;
5:
6: struct SuffixArray{
7:     vector<int> sa, lcp, rank;
8:     SuffixArray(string& s, int lim = 256){
9:         int n = sz(s) + 1, k = 0, a, b;
10:        vector<int> x(all(s) + 1), y(n), ws(max(n, lim));
11:        rank = vector<int>(n);
12:        sa = lcp = y, iota(all(sa), 0);
13:        for (int j = 0, p = 0; p < n; j = max(1LL, j * 2LL), lim = p) {
14:            p = j, iota(all(y), n - j);
15:            for(int i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
16:            fill(all(ws), 0);
17:            for(int i = 0; i < n; i++) ws[x[i]]++;
18:            for(int i = 1; i < lim; i++) ws[i] += ws[i - 1];
19:            for(int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
20:            swap(x, y), p = 1, x[sa[0]] = 0;
21:            for(int i = 1; i < n; i++) a = sa[i - 1], b = sa[i], x[b]
22:                = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1 : p++;
23:            for(int i = 1; i < n; i++) rank[sa[i]] = i;
24:            for(int i = 0, j; i < n - 1; lcp[rank[i++]] = k)
25:                for (k && k--, j = sa[rank[i] - 1];
26:                    i + k < s.size() && j + k < s.size() && s[i + k] == s[j + k]; k++);
27:        }
28:    }
29: };
30: // height == lcp
31: // Note:
32: // height[i] = longest common prefix of suffix(sa[i-1]) suffix(sa[i])
33: // longest common prefix(sa[i], sa[j]) = min(height[i+1]... height[j])
34:
35: // Problem 1
36: // find the longest repeated (occurs at least twice) substring (can overlap)
37: // max value of (height)
38:
39: // Problem 2
40: // find the longest repeated substring, cannot overlap
41: // binary search on answer k
42: // Divide height into blocks [0...i1-1] [i1...i2-1] ...
43: // height[i1] < k
44: // if(max(sa[0...i]) - min(sa[0...i])) for any block i >= k
45: // => longest common prefix at least k and separate at least k apart
46:
47: // Problem 3
48: // Find longest substring that repeated at least k times
49: // same approach -> divide
50: // Then find if exist block with at least k strings
51:
52: // Problem 4
53: // ***æ`217â\200\213â-\227ä,²é\203æ\230-æ\237\220â\200\213â¼\214ç¶'ç\232\204â\211\215ç¶'***
54: // every suffix(sa[k]) contribute n - sa[k] no. of prefix
55: // but height[k] of them are repeated
56: // no. of distinct substring = n (n+1) / 2 - sum of height
57:
58: // Problem 5
59: // Longest palindrome substring
60: // find longest repeated substring of s + '#' + rev(s)
61:
62: // Problem 6
63: // consecutive repeated string (e.g. abababa) ab repeated R = 3 times, R is max
64: // for each k from 1 to n, see if n%k == 0 and RMQ(height[rank[1]]...height[rank[k+1]]) == n - k
65:
66: // Problem 7
67: // Find the substring with most consecutive repeated string R = sqrt(ababababeipi => S = abababab is
the answer
68: // Consider repeated string of length L
69: // if S consist of at least 2 L, then it must contain 2 of the R[0], R[L], R[2*L] ...
70: // so just consider r[L*i] and r[L*(i+1)] can extend to where, let the length be K, ans = max(K/L +
1)
71:
72: // Problem 8
73: // longest common substring of A+B
74: // find Problem 1 of (A + '#' + B), but the suffix are originally in different string

```

```
1: #include <bits/stdc++.h>
2: using namespace std;
3:
4: struct TrieNode{
5:     vector<int> child;
6:     bool isWord = false;
7:     TrieNode() : child(26,-1){}
8: };
9:
10: vector<TrieNode> trie(1);
11:
12: void insert(string S) {
13:     int cur = 0;
14:     for(int i=0; i<S.length(); i++){
15:         if(trie[cur].child[S[i]-'a'] == -1){
16:             trie[cur].child[S[i]-'a'] = trie.size();
17:             trie.emplace_back();
18:         }
19:         cur = trie[cur].child[S[i]-'a'];
20:     }
21:     trie[cur].isWord = 1;
22: }
23:
24: bool lookup(string s){
25:     int cur = 0;
26:     for(int i=0; i<s.length(); i++){
27:         if(trie[cur].child[s[i]-'a'] == -1) return 0;
28:         cur = trie[cur].child[s[i]-'a'];
29:     }
30:     return trie[cur].isWord;
31: }
```

```
1: #include<bits/stdc++.h>
2: using namespace std;
3: vector<int>Zfunc(string& str){
4:     int n=str.size();
5:     vector<int>z(n);
6:     int l=0,r=0;
7:     for(int i=1;i<n;i++){
8:         if(i<=r){
9:             z[i]=min(r-i+1,z[i-1]);
10:        }
11:        while(i+z[i]<n&&str[z[i]]==str[i+z[i]]){
12:            z[i]++;
13:        }
14:        if(i+z[i]-1>r){
15:            l=i;
16:            r=i+z[i]-1;
17:        }
18:    }
19:    return z;
20: }
21:
22: // lcp of string s and string s[i...n]
```