Notebook - Maratona de Programação

Prisioneiras de WA e WAstros

Contents

1 Grafos

1.1 Binary Lifting

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

using namespace std;
// O problema deu 109 o k entao log2 10e9 e 30;
const int MAXLOG2 = 30;
const int MAXN = 2e5+10;
vector<vector<int>> pre(MAXLOG2+1, vector<int>(MAXN));

 // Pre processa os pais em expoentes de 2, [0] significa 2e0
void pre_processamento(vector<int> sucessores, int n){
  for(int i=1;i<=n; i++){
    pre[0][i] = sucessores[i];
  }

  for(int p=1; p<=MAXLOG2; p++){
    for(int i=1; i<= n; i++){
      if(pre[p-1][i] == -1){
        pre[p][i] = -1;
      }
      else{
        pre[p][i] = pre[p-1][pre[p-1][i]];
      }
      }

  }
}
 //kesimo pai do no
int kth_succ(int n, int k){
  int aux = n;
  for(int i=0; i<=MAXLOG2; i++){
    if(k & (1 << i)){
      aux = pre[i][aux];
      if(aux == -1)
        return -1;
    }
  }
  return aux;
}

int main(){
  int n, q; cin >> n >> q;
  vector<int> sucessores(n+1);
  sucessores[1] = -1;
  sucessores[0] = -1;
  for(int i=2;i<= n; i++){
   int x; cin >> x;
   sucessores[i] = x;
  }

  pre_processamento(sucessores, n);

  while(q--){
    int a, b;
    cin >> a >> b;
    cout << kth_succ(a, b) << endl;
  }
}
```

2 Strings

2.1 Kmp

```
string p;
int neighbor[N];
int walk(int u, char c) { // leader after inputting ‘c‘
    while (u != -1 && (u+1 >= (int)p.size() || p[u + 1] != c)) // leader doesn
    't match
        u = neighbor[u];
    return p[u + 1] == c ? u+1 : u;
}
void build() {
    neighbor[0] = -1; // -1 is the leftmost state
    for (int i = 1; i < (int)p.size(); i++)
        neighbor[i] = walk(neighbor[i-1], p[i]);
}
```

2.2 Lcs

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

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

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

        currRow = 1 - currRow;
    }

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

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

2.3 Trie

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

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

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

2.4 Suffix Array

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

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

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

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

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

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

2.5 Hash

```
// String Hash template
// constructor(s) - O(|s|)
// query(l, r) - returns the hash of the range [l,r] from left to right - O(1)
// query_inv(l, r) from right to left - O(1)

struct Hash {
    const ll P = 31;
    int n; string s;
    vector<ll> h, hi, p;
    Hash() {}
    Hash(string s): s(s), n(s.size()), h(n), hi(n), p(n) {
        for (int i=0;i<n;i++) p[i] = (i ? P*p[i-1]:1) % MOD;
        for (int i=0;i<n;i++)
            h[i] = (s[i] + (i ? h[i-1]:0) * P) % MOD;
        for (int i=n-1;i>=0;i--)
            hi[i] = (s[i] + (i+1<n ? hi[i+1]:0) * P) % MOD;
    }
    int query(int l, int r) {
        ll hash = (h[r] - (l ? h[l-1]*p[r-l+1]%MOD : 0));
        return hash < 0 ? hash + MOD : hash;
    }
    int query_inv(int l, int r) {
        ll hash = (hi[l] - (r+1 < n ? hi[r+1]*p[r-l+1] % MOD : 0));
        return hash < 0 ? hash + MOD : hash;
    }
};
```

3 DP

3.1 Sumset

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

using namespace std;
int MAXN = 1e5+1;
int MAXM = 1e3+1;
vector<vector<int>> dp(MAXM, vector<int> (MAXN, -1));
vector<int> xs;
int response = 0;

int solve(int n, int valor){
```

```
  if(n == -1 and valor == 0 ){
    response = 1;
     return 1;
  }
 if(n == -1 or valor < 0){
    return 0;
  }

 auto &rep = dp[n][valor];
 if(rep != -1){
   return rep;
  }

 rep = solve(n-1, valor) || solve(n-1, valor-xs[n]);
return rep;

}

int main(){
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int V, N;
  cin >> V >> N;
  xs = vector<int>(N);
  for(auto &i: xs)
    cin >> i;
  solve(N-1, V);
  // se for possivel formar o valor v com o array
  cout << (response ? "S" : "N") << endl;

}
```

4 Geometria

4.1 Area Polygon

```
double polygonArea(const std::vector<std::pair<double, double>>& vertices) {
    int n = vertices.size();
    double area = 0.0;

    for (int i = 0; i < n; i++) {
        int j = (i + 1) % n;
        area += vertices[i].first * vertices[j].second;
        area -= vertices[i].second * vertices[j].first;
    }

    return std::fabs(area) / 2.0;
}
```

4.2 Intersect Polygon

```
bool intersect(vector<point> A, vector<point> B) // Ordered ccw
{
    for(auto a: A)
        if(inside(B, a))
            return true;
    for(auto b: B)
        if(inside(A, b))
            return true;

    if(inside(B, center(A)))
        return true;

    return false;
}
```

4.3 Convex Hull

```
vp convex_hull(vp P)
{
    sort(P.begin(), P.end());
    vp L, U;
    for(auto p: P){
        while(L.size()>=2 and ccw(L.end()[-2], L.back(), p)!=1)
            L.pop_back();
        L.push_back(p);
    }
    reverse(P.begin(), P.end());
    for(auto p: P){
        while(U.size()>=2 and ccw(U.end()[-2], U.back(), p)!=1)
            U.pop_back();
        U.push_back(p);
    }
    L.pop_back();
    L.insert(L.end(), U.begin(), U.end()-1);
    return L;
}
```

4.4 3d

```
// typedef ll cod;
// bool eq(cod a, cod b){ return (a==b); }

const ld EPS = 1e-6;
#define vp vector<point>
typedef ld cod;
bool eq(cod a, cod b){ return fabs(a - b) <= EPS; }

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

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

```
        return {x/t, y/t, z/t};
    }
    bool operator==(const point &o) const {
        return eq(x, o.x) and eq(y, o.y) and eq(z, o.z);
    }
    cod operator*(const point &o) const { // dot
        return x*o.x + y*o.y + z*o.z;
    }
    point operator^(const point &o) const { // cross
        return point(y*o.z - z*o.y,
                     z*o.x - x*o.z,
                     x*o.y - y*o.x);
    }
};

ld norm(point a) { // Modulo
    return sqrt(a * a);
}
cod norm2(point a) {
    return a * a;
}
bool nulo(point a) {
    return (eq(a.x, 0) and eq(a.y, 0) and eq(a.z, 0));
}
ld proj(point a, point b) { // a sobre b
    return (a*b)/norm(b);
}
ld angle(point a, point b) { // em radianos
    return acos((a*b) / norm(a) / norm(b));
}

cod triple(point a, point b, point c) {
    return (a * (b^c)); // Area do paralelepipedo
}

point normilize(point a) {
    return a/norm(a);
}

struct plane {
    cod a, b, c, d;
    point p1, p2, p3;
    plane(point p1=0, point p2=0, point p3=0): p1(p1), p2(p2), p3(p3) {
        point aux = (p1-p3)^(p2-p3);
        a = aux.x; b = aux.y; c = aux.z;
        d = -a*p1.x - b*p1.y - c*p1.z;
    }
    plane(point p, point normal) {
        normal = normilize(normal);
        a = normal.x; b = normal.y; c = normal.z;
        d = -(p*normal);
    }

    // ax+by+cz+d = 0;
    cod eval(point &p) {
        return a*p.x + b*p.y + c*p.z + d;
    }
};
```

```
cod dist(plane pl, point p) {
    return fabs(pl.a*p.x + pl.b*p.y + pl.c*p.z + pl.d) / sqrt(pl.a*pl.a + pl.b
    *pl.b + pl.c*pl.c);
}

point rotate(point v, point k, ld theta) {
    // Rotaciona o vetor v theta graus em torno do eixo k
    // theta *= PI/180; // graus
    return (
        v*cos(theta)) +
        ((k^v)*sin(theta)) +
        (k*(k*v))*(1-cos(theta)
    );
}

// 3d line inter / mindistance
cod d(point p1, point p2, point p3, point p4) {
    return (p2-p1) * (p4-p3);
}
vector<point> inter3d(point p1, point p2, point p3, point p4) {
    cod mua = ( d(p1, p3, p4, p3) * d(p4, p3, p2, p1) - d(p1, p3, p2, p1) * d(
    p4, p3, p4, p3) )
           / ( d(p2, p1, p2, p1) * d(p4, p3, p4, p3) - d(p4, p3, p2, p1) * d(
    p4, p3, p2, p1) );
    cod mub = ( d(p1, p3, p4, p3) + mua * d(p4, p3, p2, p1) ) / d(p4, p3, p4,
    p3);
    point pa = p1 + (p2-p1) * mua;
    point pb = p3 + (p4-p3) * mub;
    if (pa == pb) return {pa};
    return {};
}
```

4.5 Inside Polygon

```
// Convex O(logn)

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

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

```
    return insideT(p[0], p[r-1], p[r], e);
}


// Any O(n)

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

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

4.6 Sweepline

```
vector<int> max_intersection(const vector<pll>& is)
{
    vector<pll> es;

    for (size_t i = 0; i < is.size(); ++i)
    {
        auto [a, b] = is[i];

        es.emplace_back(a, i + 1);      // Evento de íincio
        es.emplace_back(b, -(i + 1));   // Evento de fim
    }

    sort(es.begin(), es.end());

    set<int> active, max_set;

    for (const auto& [_, i] : es)
    {
        if (i > 0)
            active.emplace(i);
        else
            active.erase(-i);

        if (active.size() >= max_set.size())
            max_set = active;
    }

    return { max_set.begin(), max_set.end() };
}
```

4.7 Polygon Diameter

```
pair<point, point> polygon_diameter(vp p) {
    p = convex_hull(p);
  int n = p.size(), j = n<2 ? 0:1;
  pair<ll, vp> res({0, {p[0], p[0]}});
  for (int i=0;i<j;i++){
    for (;; j = (j+1) % n) {
      res = max(res, {norm2(p[i] - p[j]), {p[i], p[j]}});
      if ((p[(j + 1) % n] - p[j]) ^ (p[i + 1] - p[i]) >= 0)
        break;
    }
  }
  return res.second;
}

double diameter(const vector<point> &p) {
    vector<point> h = convexHull(p);
    int m = h.size();
    if (m == 1)
        return 0;
    if (m == 2)
        return dist(h[0], h[1]);
    int k = 1;
    while (area(h[m - 1], h[0], h[(k + 1) % m]) > area(h[m - 1], h[0], h[k]))
        ++k;
    double res = 0;
    for (int i = 0, j = k; i <= k && j < m; i++) {
        res = max(res, dist(h[i], h[j]));
        while (j < m && area(h[i], h[(i + 1) % m], h[(j + 1) % m]) > area(h[i
    ], h[(i + 1) % m], h[j])) {
            res = max(res, dist(h[i], h[(j + 1) % m]));
            ++j;
        }
    }
    return res;
}
```

4.8 Mindistpair

```
ll MinDistPair(vp &vet){
    int n = vet.size();
    sort(vet.begin(), vet.end());
    set<point> s;

    ll best_dist = LLINF;
    int j=0;
    for(int i=0;i<n;i++){
        ll d = ceil(sqrt(best_dist));
        while(j<n and vet[i].x-vet[j].x >= d){
            s.erase(point(vet[j].y, vet[j].x));
            j++;
        }

        auto it1 = s.lower_bound({vet[i].y - d, vet[i].x});
        auto it2 = s.upper_bound({vet[i].y + d, vet[i].x});

        for(auto it=it1; it!=it2; it++){
```

```
            ll dx = vet[i].x - it->y;
            ll dy = vet[i].y - it->x;
            if(best_dist > dx*dx + dy*dy){
                best_dist = dx*dx + dy*dy;
                // vet[i] e inv(it)
            }
        }

        s.insert(point(vet[i].y, vet[i].x));
    }
    return best_dist;
}
```

4.9 2d

```
#define vp vector<point>
#define ld long double
const ld EPS = 1e-6;
const ld PI = acos(-1);

typedef ld T;
bool eq(T a, T b){ return abs(a - b) <= EPS; }

struct point{
    T x, y;
    int id;
    point(T x=0, T y=0): x(x), y(y){}

    point operator+(const point &o) const{ return {x + o.x, y + o.y}; }
    point operator-(const point &o) const{ return {x - o.x, y - o.y}; }
    point operator*(T t) const{ return {x * t, y * t}; }
    point operator/(T t) const{ return {x / t, y / t}; }
    T operator*(const point &o) const{ return x * o.x + y * o.y; }
    T operator^(const point &o) const{ return x * o.y - y * o.x; }
    bool operator<(const point &o) const{
        return (eq(x, o.x) ? y < o.y : x < o.x);
    }
    bool operator==(const point &o) const{
        return eq(x, o.x) and eq(y, o.y);
    }
  friend ostream& operator<<(ostream& os, point p) {
    return os << "(" << p.x << "," << p.y << ")"; }
};

int ccw(point a, point b, point e){ // -1=dir; 0=collinear; 1=esq;
    T tmp = (b-a) ^ (e-a); // vector from a to b
    return (tmp > EPS) - (tmp < -EPS);
}

ld norm(point a){ // Modulo
    return sqrt(a * a);
}
T norm2(point a){
    return a * a;
}
bool nulo(point a){
    return (eq(a.x, 0) and eq(a.y, 0));
}
```

```
point rotccw(point p, ld a){
    // a = PI*a/180; // graus
    return point((p.x*cos(a)-p.y*sin(a)), (p.y*cos(a)+p.x*sin(a)));
}
point rot90cw(point a) { return point(a.y, -a.x); };
point rot90ccw(point a) { return point(-a.y, a.x); };

ld proj(point a, point b){ // a sobre b
    return a*b/norm(b);
}
ld angle(point a, point b){ // em radianos
    ld ang = a*b / norm(a) / norm(b);
    return acos(max(min(ang, (ld)1), (ld)-1));
}
ld angle_vec(point v){
    // return 180/PI*atan2(v.x, v.y); // graus
    return atan2(v.x, v.y);
}
ld order_angle(point a, point b){ // from a to b ccw (a in front of b)
    ld aux = angle(a,b)*180/PI;
    return ((a^b)<=0 ? aux:360-aux);
}
bool angle_less(point a1, point b1, point a2, point b2){ // ang(a1,b1) <= ang(
    a2,b2)
    point p1((a1*b1), abs((a1^b1)));
    point p2((a2*b2), abs((a2^b2)));
    return (p1^p2) <= 0;
}

ld area(vp &p){ // (points sorted)
    ld ret = 0;
    for(int i=2;i<(int)p.size();i++)
        ret += (p[i]-p[0])^(p[i-1]-p[0]);
    return abs(ret/2);
}
ld areaT(point &a, point &b, point &c){
    return abs((b-a)^(c-a))/2.0;
}

point center(vp &A){
    point c = point();
    int len = A.size();
    for(int i=0;i<len;i++)
        c=c+A[i];
    return c/len;
}

point forca_mod(point p, ld m){
    ld cm = norm(p);
    if(cm<EPS) return point();
    return point(p.x*m/cm,p.y*m/cm);
}

ld param(point a, point b, point v){
    // v = t*(b-a) + a // return t;
    // assert(line(a, b).inside_seg(v));
    return ((v-a) * (b-a)) / ((b-a) * (b-a));
}
```

```
bool simetric(vp &a){ //ordered
    int n = a.size();
    point c = center(a);
    if(n&1) return false;
    for(int i=0;i<n/2;i++)
        if(ccw(a[i], a[i+n/2], c) != 0)
            return false;
    return true;
}

point mirror(point m1, point m2, point p){
    // mirror point p around segment m1m2
    point seg = m2-m1;
    ld t0 = ((p-m1)*seg) / (seg*seg);
    point ort = m1 + seg*t0;
    point pm = ort-(p-ort);
    return pm;
}


////////////
//  Line  //
////////////

struct line{
    point p1, p2;
    T a, b, c; // ax+by+c = 0;
    // y-y1 = ((y2-y1)/(x2-x1))(x-x1)
    line(point p1=0, point p2=0): p1(p1), p2(p2){
        a = p1.y - p2.y;
        b = p2.x - p1.x;
        c = p1 ^ p2;
    }

    T eval(point p){
        return a*p.x+b*p.y+c;
    }
    bool inside(point p){
        return eq(eval(p), 0);
    }
    point normal(){
        return point(a, b);
    }

    bool inside_seg(point p){
        return (
            ((p1-p) ^ (p2-p)) == 0 and
            ((p1-p) * (p2-p)) <= 0
        );
    }

};

// be careful with precision error
vp inter_line(line l1, line l2){
    ld det = l1.a*l2.b - l1.b*l2.a;
    if(det==0) return {};
    ld x = (l1.b*l2.c - l1.c*l2.b)/det;
    ld y = (l1.c*l2.a - l1.a*l2.c)/det;
    return {point(x, y)};
}

// segments not collinear
vp inter_seg(line l1, line l2){
    vp ans = inter_line(l1, l2);
    if(ans.empty() or !l1.inside_seg(ans[0]) or !l2.inside_seg(ans[0]))
        return {};
    return ans;
}
bool seg_has_inter(line l1, line l2){
    return ccw(l1.p1, l1.p2, l2.p1) * ccw(l1.p1, l1.p2, l2.p2) < 0 and
           ccw(l2.p1, l2.p2, l1.p1) * ccw(l2.p1, l2.p2, l1.p2) < 0;
}

ld dist_seg(point p, point a, point b){ // point - seg
    if((p-a)*(b-a) < EPS) return norm(p-a);
    if((p-b)*(a-b) < EPS) return norm(p-b);
    return abs((p-a)^(b-a)) / norm(b-a);
}

ld dist_line(point p, line l){ // point - line
    return abs(l.eval(p))/sqrt(l.a*l.a + l.b*l.b);
}

line bisector(point a, point b){
    point d = (a+b)/2;
    return line(d, d + rot90ccw(a-b));
}

line perpendicular(line l, point p){ // passes through p
    return line(l.b, -l.a, -l.b*p.x + l.a*p.y);
}


////////////
// Circle //
////////////

struct circle{
    point c; T r;
    circle() : c(0, 0), r(0){}
    circle(const point o) : c(o), r(0){}
    circle(const point a, const point b){
        c = (a+b)/2;
        r = norm(a-c);
    }
    circle(const point a, const point b, const point cc){
        assert(ccw(a, b, cc) != 0);
        c = inter_line(bisector(a, b), bisector(b, cc))[0];
        r = norm(a-c);
    }
    bool inside(const point &a) const{
        return norm(a - c) <= r + EPS;
    }
};
```

```
pair<point, point> tangent_points(circle cr, point p) {
    ld d1 = norm(p-cr.c), theta = asin(cr.r/d1);
    point p1 = rotccw(cr.c-p, -theta);
    point p2 = rotccw(cr.c-p, theta);
    assert(d1 >= cr.r);
    p1 = p1 * (sqrt(d1*d1-cr.r*cr.r) / d1) + p;
    p2 = p2 * (sqrt(d1*d1-cr.r*cr.r) / d1) + p;
    return {p1, p2};
}


circle incircle(point p1, point p2, point p3){
    ld m1 = norm(p2-p3);
    ld m2 = norm(p1-p3);
    ld m3 = norm(p1-p2);
    point c = (p1*m1 + p2*m2 + p3*m3)*(1/(m1+m2+m3));
    ld s = 0.5*(m1+m2+m3);
    ld r = sqrt(s*(s-m1)*(s-m2)*(s-m3)) / s;
    return circle(c, r);
}

circle circumcircle(point a, point b, point c) {
    circle ans;
    point u = point((b-a).y, -(b-a).x);
    point v = point((c-a).y, -(c-a).x);
    point n = (c-b)*0.5;
    ld t = (u^n)/(v^u);
    ans.c = ((a+c)*0.5) + (v*t);
    ans.r = norm(ans.c-a);
    return ans;
}

vp inter_circle_line(circle C, line L){
    point ab = L.p2 - L.p1, p = L.p1 + ab * ((C.c-L.p1)*(ab) / (ab*ab));
    ld s = (L.p2-L.p1)^(C.c-L.p1), h2 = C.r*C.r - s*s / (ab*ab);
    if (h2 < -EPS) return {};
    if (eq(h2, 0)) return {p};
    point h = (ab/norm(ab)) * sqrt(h2);
    return {p - h, p + h};
}

vp inter_circle(circle c1, circle c2){
    if (c1.c == c2.c) { assert(c1.r != c2.r); return {}; }
    point vec = c2.c - c1.c;
    ld d2 = vec * vec, sum = c1.r + c2.r, dif = c1.r - c2.r;
    ld p = (d2 + c1.r * c1.r - c2.r * c2.r) / (2 * d2);
    ld h2 = c1.r * c1.r - p * p * d2;
    if (sum * sum < d2 or dif * dif > d2) return {};
    point mid = c1.c + vec * p, per = point(-vec.y, vec.x) * sqrt(fmax(0, h2)
    / d2);
    if (eq(per.x, 0) and eq(per.y, 0)) return {mid};
    return {mid + per, mid - per};
}

// minimum circle cover O(n) amortizado
circle min_circle_cover(vp v){
    random_shuffle(v.begin(), v.end());
    circle ans;
    int n = v.size();
    for(int i=0;i<n;i++) if(!ans.inside(v[i])){
        ans = circle(v[i]);
        for(int j=0;j<i;j++) if(!ans.inside(v[j])){
            ans = circle(v[i], v[j]);
            for(int k=0;k<j;k++) if(!ans.inside(v[k])){
                ans = circle(v[i], v[j], v[k]);
            }
        }
    }
    return ans;
}
```

5 Matematica

5.1 Primes

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

using namespace std;

const int MAX { 10000001 };

bool is_prime(int n)
{
    if (n < 2)
        return false;

    for (int i = 2; i < n; ++i)
        if (n % i == 0)
            return false;

    return true;
}

bool is_prime2(int n)
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    if (n % 2 == 0)
        return false;

    for (int i = 3; i < n; i += 2)
        if (n % i == 0)
            return false;

    return true;
}

bool is_prime3(int n)
{
    if (n < 2)
```

```
        return false;

    if (n == 2)
        return true;

    if (n % 2 == 0)
        return false;

    for (int i = 3; i * i <= n; i += 2)
        if (n % i == 0)
            return false;

    return true;
}

vector<int> primes(int N)
{
    vector<int> ps;

    for (int i = 2; i <= N; ++i)
        if (is_prime3(i))
            ps.push_back(i);

    return ps;
}

vector<int> primes2(int N) {
    vector<int> ps;
    bitset<MAX> sieve;              // MAX deve ser maior do que N
    sieve.set();                    // Todos ãso "potencialmente" primos
    sieve[1] = false;               // 1 ãno é primo

    for (int i = 2; i <= N; ++i) {
        if (sieve[i]) {             // i é primo
            ps.push_back(i);

            for (int j = 2 * i; j <= N; j += i)
                sieve[j] = false;
        }
    }

    return ps;
}

vector<int> primes3(int N)
{
    bitset<MAX> sieve;                  // MAX deve ser maior do que N
    vector<int> ps { 2 };               // Os pares ãso tratados à parte
    sieve.set();                        // Todos ãso "potencialmente" primos

    for (int i = 3; i <= N; i += 2) {   // Apenas ímpares ãso verificados agora
        if (sieve[i]) {                 // i é primo
            ps.push_back(i);

            for (int j = 2 * i; j <= N; j += i)
                sieve[j] = false;
        }
    }
    return ps;
}

vector<long long> primes4(long long N)
{
    bitset<MAX> sieve;                  // MAX deve ser maior do que N
    vector<long long> ps { 2 };         // Os pares ãso tratados à parte
    sieve.set();                        // Todos ãso "potencialmente" primos

    for (long long i = 3; i <= N; i += 2) {   // Apenas ímpares ãso verificados
     agora
        if (sieve[i]) {                       // i é primo
            ps.push_back(i);

            for (long long j = i * i; j <= N; j += 2*i)   // úMltiplos ímpares
    >= i*i
                sieve[j] = false;
        }
    }

    return ps;
}

vector<long long> primes5(long long N)
{
    bitset<MAX> sieve;              // MAX deve ser maior do que N
    vector<long long> ps { 2, 3 };  // Pares e úmltiplos de 3 ãso tratados à
    parte
    sieve.set();                    // Todos ãso "potencialmente" primos

    // O incremento alterna entre saltos de 2 ou 4, evitando os úmltiplos de 3
    for (long long i = 5, step = 2; i <= N; i += step, step = 6 - step) {
        if (sieve[i]) {                                   // i é primo
            ps.push_back(i);

            for (long long j = i * i; j <= N; j += 2*i)   // úMltiplos ímpares
    >= i*i
                sieve[j] = false;
        }
    }
    return ps;
}

int main()
{
    cout << "==== Testes de primalidade:\n\n";

    auto p = 999983;
    auto start = chrono::system_clock::now();
    auto ok = is_prime(p);
    auto end = chrono::system_clock::now();
    chrono::duration<double> t = end - start;

    cout.precision(15);
    cout << fixed;
    cout << "is_prime(" << p << ")  = " << ok << " (" << t.count() << " ms)\n"
    ;
```

```
start = chrono::system_clock::now();
ok = is_prime2(p);
end = chrono::system_clock::now();
t = end - start;

cout << "is_prime2(" << p << ") = " << ok << " (" << t.count() << " ms)\n"
;

start = chrono::system_clock::now();
ok = is_prime3(p);
end = chrono::system_clock::now();
t = end - start;

cout << "is_prime3(" << p << ") = " << ok << " (" << t.count() << " ms)\n"
;

cout << "\n\n==== çãGerao de primos éat N:\n\n";
auto N = 10000000;

start = chrono::system_clock::now();
auto ps = primes(N);
end = chrono::system_clock::now();
t = end - start;

cout << "primes(" << N << ")  = " << ps.size() << " (" << t.count() << "
ms)\n";

start = chrono::system_clock::now();
ps = primes2(N);
end = chrono::system_clock::now();
t = end - start;

cout << "primes2(" << N << ") = " << ps.size() << " (" << t.count() << "
ms)\n";

start = chrono::system_clock::now();
ps = primes3(N);
end = chrono::system_clock::now();
t = end - start;

cout << "primes3(" << N << ") = " << ps.size() << " (" << t.count() << "
ms)\n";

long long M = N;
start = chrono::system_clock::now();
auto qs = primes4(M);
end = chrono::system_clock::now();
t = end - start;

cout << "primes4(" << N << ") = " << qs.size() << " (" << t.count() << "
ms)\n";

start = chrono::system_clock::now();
qs = primes5(M);
end = chrono::system_clock::now();
t = end - start;

    cout << "primes5(" << N << ") = " << qs.size() << " (" << t.count() << "
    ms)\n";

    return 0;
}
```

5.2 Fatorization

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

using namespace std;

map<long long, long long> factorization(long long n) {
    map<long long, long long> fs;

    for (long long d = 2, k = 0; d * d <= n; ++d, k = 0) {
        while (n % d == 0) {
            n /= d;
            ++k;
        }

        if (k) fs[d] = k;
    }

    if (n > 1) fs[n] = 1;

    return fs;
}

map<long long, long long> factorization(long long n, vector<long long>& primes
    )
{
    map<long long, long long> fs;

    for (auto p : primes)
    {
        if (p * p > n)
            break;

        long long k = 0;

        while (n % p == 0) {
            n /= p;
            ++k;
        }

        if (k)
            fs[p] = k;
    }

    if (n > 1)
        fs[n] = 1;

    return fs;
}

int main()
{
```

```
    long long n;
    cin >> n;

    auto fs = factorization(n);
    bool first = true;

    cout << n << " = ";
    for (auto [p, k] : fs)
    {
        if (not first)
            cout << " x ";

        cout << p << "^" << k;
        first = false;
    }

    cout << endl;

    return 0;
}
```

5.3 Polinomial-degree

```
int evaluate(const polynomial& p, int x)
{
    int y = 0, N = degree(p);

    for (int i = N; i >= 0; --i)
    {
        y *= x;
        y += p[i];
    }

    return y;
}
```

5.4 Phandfp

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

using namespace std;

using ll = long long;

ll fp(ll a, ll b){

  if ( not b)
    return 1;

  ll pr = fp(a, b/2);

  return ~b & 1 ? pr * pr : pr * pr * a;

}
```

```
ll ph(ll x){

  if ( x == 1)
    return 1;

  map<int, int> m;

  for ( int i = 2; i * i <= x; i++)
    while ( x % i == 0){
      x/=i;
      m[i]++;
  }

  if (x and x != 1)
    m[x]++;


  ll res = 1;
  for ( auto [primo, potencia ] : m)
    res = (primo - 1) fp(primo, potencia - 1);

  return res;

}


int main(){
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cout << ph(400) << endl;



}
```

5.5 Mdc

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

using namespace std;

long long gcd(long long a, long long b)
{
    return b ? gcd(b, a % b) : a;
}

long long ext_gcd(long long a, long long b, long long& x, long long& y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }

    long long x1, y1;
```

```
    long long d = ext_gcd(b, a % b, x1, y1);

    x = y1;
    y = x1 - y1*(a/b);

    return d;
}

int main()
{
    long long a, b;
    cin >> a >> b;

    cout << "(" << a << ", " << b << ") = " << gcd(a, b) << '\n';

    long long x, y;
    auto d = ext_gcd(a, b, x, y);

    cout << d << " = (" << a << ")(" << x << ") + (" << b << ")(" << y << ")\n
    ";

    return 0;
}
```

5.6 Mod

```
long long add(long long a, long long b, long long m)
{
    auto r = (a + b) % m;

    return r < 0 ? r + m : r;
}

long long mul(long long a, long long b, long long m)
{
    auto r = (a * b) % m;

    return r < 0 ? r + m : r;
}

long long fast_exp_mod(long long a, long long n, long long m) {
    long long res = 1, base = a;

    while (n) {
        if (n & 1)
            res = mul(res, base, m);

        base = mul(base, base);
        n >= 1;
    }

    return res;
}

long long inv(long long a, long long p) {
    return fast_exp_mod(a, p - 2, p);
}
```

```
// É assumido que (a, m) = 1
long long inverse(long long a, long long m)
{
    return fast_exp_mod(a, phi(m) - 1, m);
}

int mod(int a, int m)
{
    return ((a % m) + m) % m;
}
```

5.7 Primos

```
//(N ** fi de p) % p == 1 sempre
// sistema reduzido de íresduo é os diferentes restos que deixam (7 vai ter t
    =6) - pega todos os restos
// únmeros coprimos - únmero que mdc entre eles é 1
// coprimos de 6 = 1,4,5


// TEOREMA DE FERMAT
// a^p é congruente a a(mod p) - a é inteiro e p é primo
//

// TEOREMA DE EULER
// a^fi de m é congruente a 1 mod m
// ós de primo o fi é -1
// fatora em primo e sabe que é -1
// fi de qulquer valor é = fi de primo 1 * fi de primo 2

// Fatoracao em primos
#define ll long long

ll phi(){

}

ll fatp(int x){
    map<int, int> m;

    for(int i = 2; i * i < x; i++){
       while(x%i == 0){
        x/=i;
        m[i]++;
       }



    }


}

// verificar se é primo
bool is_p(int n){
    if(n < 2)
        return false;
```

```
    if(n == 2)
        return true;

    if(n%2 == 0)
        return false;

    for(int i = 3; i * i <= n; i+=2){
        if(n%i == 0)
            return false;
    }
    return true;
}

// crivo
vector<long, long> primes(ll N){
    bitset<MAX> sieve;
    vector<long long> ps{2};
    sieve.set();

    for(ll i = 3; i<=N; i+=2){
        if(sieve[i]){
            ps.push_back(i);
            for(ll j = i * i; j<=N; j+=2*i){
                sieve[j] = false;
            }
        }
    }
    return ps;
}
```

5.8 Funcoes Multiplicativas

```
#define ll long long

ll number_of_divisors(int n, const vector<int>& primes){
    auto fs = factorization(n, primes);
    ll res = 1;

    for(auto [p, k] : fs)
        res*=(k+1);

    return res;
}

ll sum_of_divisors(int n, const vector<int>& primes){
    auto fs = factorization(n, primes);
    ll res = 1;

    for(auto [p, k] : fs){
        ll pk = p;

        while(k--){
            pk *= p;
        }

        res *= (pk-1)/(p-1);
    }
    return res;
}
```

```
int phi(int n, const vector<int>& primes){
    if(n==1)
        return 1;

    auto fs = factorization(n, primes);
    auto res = n;

    for( auto [p, k] : fs){
        res /= p;
        res *= (p-1);
    }
    return res;
}
```

5.9 Polinomy-add

```
polynomial operator+(const polynomial& p, const polynomial& q)
{
    int N = degree(p), M = degree(q);
    polynomial r(max(N, M) + 1, 0);

    for (int i = 0; i <= N; ++i)
        r[i] += p[i];

    for (int i = 0; i <= M; ++i)
        r[i] += q[i];

    while (not r.empty() and r.back() == 0)
        r.pop_back();

    if (r.empty())
        r.push_back(0);

    return r;
}
```

5.10 Fatorial

```
map<int, int> factorial_factorization(int n, const vector<int>& primes)
{
    map<int, int> fs;

    for (const auto& p : primes)
    {
        if (p > n)
            break;

        fs[p] = E(n, p);
    }

    return fs;
}
```

5.11 Permutacoes

```cpp
#include <bits/stdc++.h>
#include <vector>
#define ll long long

template<typename T>
ll permutations(const vector<T>& A){
    map<T, int>hist;

    for(auto a: A)
        ++hist[a];

    ll res = factorial(A.size());

    for(auto [a, ni]: hist)
        res/= factorial(ni);

    return res;
}

int main(){
    vector<int> A {5, 3, 4, 1, 2};

    sort(A.begin(), A.end());

    do{
        for(size_t i = 0; i<A.size(); ++i){
            cout << A[i] << (i+1 == A.size() ? '\n' : ' ');
        }
    } while (next_permutations(A.begin(), A.end()));
    return 0;
}
```

5.12 Fast Exp

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

using namespace std;

long long fast_exp(long long a, int n)
{
    if (n == 1)
        return a;

    auto x = fast_exp(a, n / 2);

    return x * x * (n % 2 ? a : 1);
}

long long fast_exp_it(long long a, int n)
{
    long long res = 1, base = a;

    while (n)
    {
        if (n & 1)
            res *= base;

        base *= base;
        n >>= 1;
    }

    return res;
}

int main()
{
    long long a;
    int n;

    cin >> a >> n;

    cout << a << "^" << n << " = " << fast_exp(a, n) << '\n';

    return 0;
}
```

5.13 Arranjos

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

#define ll long long;

ll A(ll n, ll p){
    if(n < p)
        return 0;

    ll res = 1;

    for(ll i = n; i > p; --i){
        res*=i;
    }

    return res;
}

//long long ós aguenta 10!
//maior N! ou A^B

ll dp(int k, int a, int b){
    if(a < 0 || b < 0)
        return 0;

    if(k == 0)
        return 1;

    if(st[k][a][b] != -1)
        return st[k][a][b];

    auto res = dp(k-1, a-1, b) + dp(k-1, a, b-1);

    st[k][a][b] = res;
    return res;
}
```

6 EstruturaDados

6.1 Union Find

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

using namespace std;

#define ff first;
#define ss second;
#define ii pair<int, int>
#define vi vector<int>
#define ll long long
#define ld long double
#define ios ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);

vector<int> si(100001);
vector<int> dad(100001);
int can = 1;

int find_set(int v){
  if(v == dad[v])
    return v;
  return dad[v] = find_set(dad[v]);
}
void make_set(int v){
  dad[v] = v;
  si[v] =1;
}

void union_sets(int a, int b){
  a = find_set(a);
  b = find_set(b);
  if(a != b) {
    if(si[a] < si[b])
      swap(a, b);
    dad[b] = a;
    si[a]+=si[b];
  }
}

int main(){
  ios;
  int n, m;
  cin >> n >> m;
  int aux = m;
  vector<vector<int>> xs(n+1);
  for(int i=1; i<n; i++){
   dad[i] = i;
  }
  set<int> ns;
  while(m--){
    int A, B; cin >> A>> B;
    if(xs[A].size() == 2 or xs[B].size() == 2)
      can = 0;
    else if(!ns.empty() and ns.count(A) and ns.count(B) and find_set(A) ==
    find_set(B))
      can = 0;
    else{
      ns.insert(A);
      ns.insert(B);
      xs[A].push_back(B);
      xs[B].push_back(A);
      union_sets(A, B);
    }

  }
  cout << (can ? "Yes" : "No") << endl;

  }
```

6.2 Seg Tree

```
template<typename T>
class SegTree {

  T e;

  std::function<T(T a, T b)> op;

  std::vector<T> ps;

  size_t p;

public:

  SegTree(size_t n, T e, std::function<T(T a, T b)> op): e(e), op(op), ps(4*n,
     e){

    p = 1;
    while (p < n)
      p <<= 1;

  }


  void update(size_t i, T value) {

    i += p;

    ps[i] = value;

    i>>=1;

    while (i){
      ps[i] = op(ps[i*2], ps[i*2 + 1]);
      i >>= 1;

    }

  }


  T query(size_t a, size_t b) {
```

```
    a+= p;
    b+= p;

    T la = e, ra = e;

    while (a <= b) {

      if ( a & 1) la = op(ps[a++], la);
      if (~b & 1) ra = op(ra, ps[b--]);

      a>>=1;
      b>>=1;
    }


    return op(la, ra);

  }



};


int main(){
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, q;
  cin >> n >> q;
  // Pra soma std::plus<int>()
  // pra xor std::xor<int>()
  SegTree<int> segtree(n, (int) 2e9, [](int a, int b) { return std::min(a, b)
    ;});
  for(int i=1; i<=n; i++){
    ll x;
    cin >> x;
   segtree.update(i, x);
  }
  while(q--){
    ll x, a, b;
    cin >> x >> a >> b;
    if(x == 2)
      cout << segtree.query(a, b) << endl;
    else
      segtree.update(a, b);
  }
}
```

6.3 Ordered Set

```
// C++ program to demonstrate the
// ordered set in GNU C++
#include <iostream>
using namespace std;

// Header files, namespaces,
// macros as defined above
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

#define ordered_set tree<int, null_type,less<int>, rb_tree_tag,
    tree_order_statistics_node_update>

// To implement in multiset
// template<class T> using ordered_multiset = tree<T, null_type, less_equal<T
    >, rb_tree_tag, tree_order_statistics_node_update>;

//costum cmpare
//
//template<class T>
//struct custom_compare {
//    bool operator()(const T& a, const T& b) const {
//        if (a == b) return true; // Keep duplicates
//        return a > b;
//    }
//};
//
//template<class T> using ordered_multiset = tree<T, null_type, custom_compare
    <T>, rb_tree_tag, tree_order_statistics_node_update>;

int main()
{
    // Ordered set declared with name o_set
    ordered_set o_set;

    // insert function to insert in
    // ordered set same as SET STL
    o_set.insert(5);

    // Finding the second smallest element
    // in the set using * because
    //  find_by_order returns an iterator
    cout << *(o_set.find_by_order(1))
         << endl;

    // Finding the number of elements
    // strictly less than k=4
    cout << o_set.order_of_key(4)
         << endl;

    // Finding the count of elements less
    // than or equal to 4 i.e. strictly less
    // than 5 if integers are present
    cout << o_set.order_of_key(5)
         << endl;

    // removing in a multiset
    //  auto it = ss.find_by_order(ss.order_of_key(2)); // Find iterator to
    the element 2
   // if (it != ss.end()) {
   //     ss.erase(it); // Erase the found element O(log n)
                      //
    // Deleting 2 from the set if it exists
    if (o_set.find(2) != o_set.end())
        o_set.erase(o_set.find(2));
```

```
    // Now after deleting 2 from the set
    // Finding the second smallest element in the set
    cout << *(o_set.find_by_order(1))
         << endl;

    // Finding the number of
    // elements strictly less than k=4
    cout << o_set.order_of_key(4)
         << endl;

    return 0;
}
```

6.4 Venice-set

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

struct VeniceSet {
    multiset<int> St;
    int water_level = 0;

    void add(int x) { St.insert(x + water_level); }

    void remove(int x)
    {
        auto it = St.find(x + water_level);
        if (it != St.end()) {
            St.erase(it);
        }
        else {
            cout << "Element " << x
                 << " not found for removal." << endl;
        }
    }

    void updateAll(int x) { water_level += x; }

    int size() { return St.size(); }
};

int main()
{
    VeniceSet vs;

    // Add elements to the VeniceSet
    vs.add(10);
    vs.add(20);
    vs.add(30);

    // Print size of the set
    cout << "Size of the set: " << vs.size() << endl;

    // Decrease all by 5
    vs.updateAll(5);

    // Remove an element
    // This removes 5 (present height) + 5 (water level) = 10
    vs.remove(5);

    // Attempt to remove an element that does not exist
    vs.remove(40);

    // Print size of the set
    cout << "Size of the set: " << vs.size() << endl;

    return 0;
}
```

6.5 Merge Sort Tree

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

using namespace std;

#define ff first;
#define ss second;
#define ii pair<int, int>
#define vi vector<int>
#define ll long long
#define ld long double
#define ios ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
const int MAXN = (1e5*3)+1;
vector<int> t[4*MAXN];

void build(int a[], int v, int tl, int tr) {
    if (tl == tr) {
        t[v] = vector<int>(1, a[tl]);
    } else {
        int tm = (tl + tr) / 2;
        build(a, v*2, tl, tm);
        build(a, v*2+1, tm+1, tr);
        merge(t[v*2].begin(), t[v*2].end(), t[v*2+1].begin(), t[v*2+1].end(),
              back_inserter(t[v]));
    }
}


int query(int v, int tl, int tr, int l, int r, int x) {
    if (l > r)
        return 0;
    if (l == tl && r == tr) {
      // QTD DE NUMEROS MAIORES OU IGUAIS A X
        vector<int>::iterator pos = lower_bound(t[v].begin(), t[v].end(), x);
        if (pos != t[v].end()){
            return t[v].end()-pos;
        }
        return 0;
    }
    int tm = (tl + tr) / 2;
    //se quiser so o lower bound do intervalo, tem que ser o min das querys
    return (query(v*2, tl, tm, l, min(r, tm), x) + query(v*2+1, tm+1, tr, max(
    l, tm+1), r, x));
}
```

```
int main(){
    ios;
    int N, M;
    cin >> N >> M;
    int adj[N];
    for(int i=0; i<N; i++){
        cin >> adj[i];
    }

    build(adj, 1, 0, N-1);
    while(M--){
        int x, y, z;
        cin >> x >> y >> z;
        int res = 0;
        if(adj[x-1] >= y){
            res = 0;
        }
        else{
            res = query(1, 0, N-1, x-1, x+z-1, y);
        }
        cout << res << endl;

    }
    return 0;

}
```