

1 DP

1.1 Digit DP

```

1 string num;
2
3 ll dp[the max len of num + 1][2][2][...];
4 memset(dp, -1, sizeof(dp));
5
6 ll count(ll pos, bool tight, bool
    leadingZero, ...) {
7     if (dp[pos][tight][leadingZero][...] !=
        -1) {
8         return dp[pos][tight][leadingZero
            ][...];
9     }
10
11     if (base-case) { // e.g. pos == num.
        size() or other base case condition
12         // do something
13     }
14
15     ll res = 0;
16     ll up = (tight ? num[pos] - '0' : 9);
17     for (ll d = 0; d <= up; ++d) {
18         res += count(
19             pos + 1,
20             tight and (d == num[pos] - '0'),
21             leadingZero and (d == 0),
22             ...
23         );
24     }
25
26     return dp[pos][tight][leadingZero][...]
        = res;
27 }
28
29 count(0, true, true, ...) // the answer
30
31 // =====
32 // =====
33
34 // AtCoder ABC154E
35 // Find the number of integers between 1 and
    N (inclusive)
36 // that contains exactly K non-zero digits
    when written in base ten.
37 // 1 <= N <= 10^100, 1 <= K <= 3
38
39 #include <bits/stdc++.h>
40 using namespace std;
41 using ll = long long;
42
43 ll k;
44 string num;
45
46 ll dp[102][2][4];
47
48 ll count(ll pos, bool tight, ll cnt) {
49     if (dp[pos][tight][cnt] != -1) {
50         return dp[pos][tight][cnt];
51     }
52

```

```

53     if (cnt == 0) return dp[pos][tight][cnt]
        = 1;
54     else if (pos == num.size()) return dp[
        pos][tight][cnt] = 0;
55
56     ll res = 0;
57     ll up = (tight ? num[pos] - '0' : 9);
58     for (ll d = 0; d <= up; ++d) {
59         res += count(
60             pos + 1,
61             tight and (d == num[pos] - '0'),
62             cnt + (d == 0 ? 0 : -1)
63         );
64     }
65
66     return dp[pos][tight][cnt] = res;
67 }
68
69 int main() {
70     ios::sync_with_stdio(false);
71     cin.tie(nullptr);
72
73     cin >> num >> k;
74
75     memset(dp, -1, sizeof(dp));
76
77     cout << count(0, true, k) << '\n';
78
79     return 0;
80 }

```

1.2 Interval DP

```

1 ll n; // the length of the array
2
3 vector<ll> a(n + 1);
4
5 vector<vector<ll>> dp(n + 1, vector<ll>(n +
    1));
6
7 for (ll i = 1; i <= n; ++i) {
8     dp[i][i] = "initial value"; // Length 1
        interval [i, i]
9 }
10 for (ll len = 2; len <= n; ++len) { // Loop
    through all length from 2 to n
11     for (ll l = 1; l <= n - len + 1; ++l) {
12         ll r = l + len - 1;
13
14         dp[l][r] = ...; // interval [l, r]
15     }
16 }
17
18 dp[1][n] // the answer (interval [1, n])

```

1.3 Knapsack

```

1 // 0/1 knapsack
2
3 const ll INF = 9e18;

```

```

4 ll n, wmx; cin >> n >> wmx;
5 vector<ll> w(n + 1); for (ll i = 1; i <= n;
    ++i) cin >> w[i];
6 vector<ll> v(n + 1); for (ll i = 1; i <= n;
    ++i) cin >> v[i];
7
8 vector<ll> dp(wmx + 1, -INF); dp[0] = 0;
9 for (ll i = 1; i <= n; ++i) {
10     for (ll j = wmx; j >= w[i]; --j) {
11         dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
12     }
13 }
14
15 ll mx = dp[0];
16 for (ll j = 1; j <= wmx; ++j) {
17     mx = max(mx, dp[j]);
18 }
19 cout << mx << '\n';
20
21 // -----
22 // Unbounded
23 // just change the inner for loop to the
    following line
24 // for (ll j = w[i]; j <= wmx; ++j) ...

```

1.4 Subset DP

```

1 for (ll s = 1; s < (1LL << n); ++s) {
2     for (ll sbs = s; sbs > 0; sbs = ((sbs - 1)
        & s)) {
3         ll c = s - sbs; // ll c = sbs^s; //
            this one is equivalent
4     }
5 }
6 // O(3^n), n <= 16
7 // sbs 是降序的 s 的子集
8 // c 是升序的 s 的子集

```

1.5 LIS

```

1 ll LISLength(const vector<ll>& arr) {
2     vector<ll> d;
3     for (ll x : arr) {
4         auto it = lower_bound(d.begin(), d.
            end(), x);
5         if (it == d.end()) d.push_back(x);
6         else *it = x;
7     }
8     return d.size();
9 }
10
11 // find LIS sequence
12 for (ll i = 0; i < num.size(); ++i) {
13     if (lis.empty() || lis.back() < num[i]) {
14         lis.push_back(num[i]);
15         dp[i] = lis.size();
16     }

```

```

17     else {
18         auto iter = lower_bound(all(lis), num
            [i]);
19         dp[i] = iter - lis.begin() + 1;
20         *iter = num[i];
21     }
22 }
23 ll length = lis.size();
24 for (ll i = num.size() - 1; i >= 0; --i) {
25     if (dp[i] == length) {
26         ans.push_back(num[i]);
27         length -= 1;
28     }
29 }

```

1.6 LCS

```

1 ll n; cin >> n;
2 string s; cin >> s; s = " " + s;
3 ll m; cin >> m;
4 string t; cin >> t; t = " " + t;
5
6 vector<vector<ll>> dp(n + 1, vector<ll>(m +
    1, 0));
7 for (ll i = 1; i <= n; ++i) {
8     for (ll j = 1; j <= m; ++j) {
9         if (s[i] == t[j]) {
10             dp[i][j] = dp[i - 1][j - 1] + 1;
11         }
12         else {
13             dp[i][j] = max(dp[i - 1][j], dp[i][j -
                1]);
14         }
15     }
16 }
17
18 cout << dp[n][m] << '\n'; // the length of
    LCS

```

2 D&C

2.1 MergeSort Finds the Number of Inversions

```

1 void merge(vector<int>& arr, ll l, ll r, ll
    x, ll y) {
2     ll left = l;
3     vector<ll> tmp; tmp.reserve(y - l + 1);
4     while (l <= r and x <= y) {
5         if (arr[l] <= arr[x]) {
6             tmp.push_back(arr[l++]);
7         }
8         else {
9             tmp.push_back(arr[x++]);
10        }
11    }
12    while (l <= r) {

```

```

13 tmp.push_back(arr[l++]);
14 }
15 while (x <= y) {
16     tmp.push_back(arr[x++]);
17 }
18 for (ll i = left; i <= y; ++i) {
19     arr[i] = tmp[i - left];
20 }
21 }
22 ll mergeSort(vector<ll>& arr, ll l, ll r) {
23     if (l == r) return 0;
24     ll mid = l + (r - 1)/2;
25     ll lcnt = mergeSort(arr, l, mid);
26     ll rcnt = mergeSort(arr, mid + 1, r);
27
28     // ----- main Logic -----
29     ll cnt = lcnt + rcnt;
30     ll a = l, b = mid, c = mid + 1, d = r;
31     // c is the current checking
32     // position
33     while (a <= b) {
34         while (c <= d and arr[a] > arr[c]) c
35             += 1;
36         cnt += c - (mid + 1);
37         a += 1;
38     }
39     // -----
40     merge(arr, l, mid, mid + 1, r);
41     return cnt;
42 }

```

3 Data Structure

3.1 DSU

```

1 ll cc;
2 vector<ll> djs, sz;
3 ll find(ll u) {
4     if (u == djs[u]) return u;
5     return djs[u] = find(djs[u]);
6 }
7 void join(ll u, ll v) {
8     u = find(u);
9     v = find(v);
10    if (u == v) return; // don't forgot
11    // this line
12    if (sz[u] < sz[v]) swap(u, v);
13    djs[v] = u;
14    sz[u] += sz[v];
15    cc -= 1;
16 }
17 void init(ll n) {
18     djs.clear(); djs.resize(n + 1);
19     for (ll i = 1; i <= n; ++i) djs[i] = i;
20     sz.clear(); sz.resize(n + 1, 1);
21     cc = n;
22 }

```

3.2 Segment Tree

```

1 // CSES Range Updates and Sums
2 // 1. Increase each value in range [a,b] by
3 // 2. Set each value in range [a,b] to x.
4 // 3. Calculate the sum of values in range [
5 // a,b].
6 #include <bits/stdc++.h>
7 using namespace std;
8 using ll = long long;
9
10 #define lson 2*n + 1
11 #define rson 2*n + 2
12
13 class Node {
14 public:
15     ll val;
16     ll setVal;
17     ll addVal;
18     bool isSet;
19 };
20
21 ll sz, q;
22 vector<ll> a;
23 vector<Node> nds;
24
25 void build(ll l, ll r, ll n = 0) {
26     if (l == r) {
27         nds[n].val = a[l];
28         nds[n].setVal = 0;
29         nds[n].addVal = 0;
30         nds[n].isSet = false;
31         return;
32     }
33     ll mid = l + (r - 1)/2;
34     build(l, mid, lson);
35     build(mid + 1, r, rson);
36     nds[n].val = nds[lson].val + nds[rson].
37         val;
38 }
39 void push(ll l, ll r, ll n) {
40     ll mid = l + (r - 1)/2;
41
42     if (nds[n].isSet) {
43
44         nds[lson].val = nds[n].setVal*(mid -
45             1 + 1);
46         nds[lson].setVal = nds[n].setVal;
47         nds[lson].addVal = 0;
48         nds[lson].isSet = true;
49
50         nds[rson].val = nds[n].setVal*(r - (
51             mid + 1) + 1);
52         nds[rson].setVal = nds[n].setVal;
53         nds[rson].addVal = 0;
54         nds[rson].isSet = true;
55
56         nds[n].isSet = false;
57     }
58
59     nds[lson].val += nds[n].addVal*(mid - 1
60         + 1);

```

```

58 nds[lson].addVal += nds[n].addVal;
59
60 nds[rson].val += nds[n].addVal*(r - (mid
61     + 1) + 1);
62 nds[rson].addVal += nds[n].addVal;
63
64 nds[n].addVal = 0;
65 }
66 void setVal(ll x, ll y, ll val, ll l, ll r,
67     ll n = 0) {
68     if (l == x and r == y) {
69         nds[n].val = val*(y - x + 1);
70         nds[n].setVal = val;
71         nds[n].addVal = 0;
72         nds[n].isSet = true;
73         return;
74     }
75     push(l, r, n);
76     ll mid = l + (r - 1)/2;
77     if (y <= mid) {
78         setVal(x, y, val, l, mid, lson);
79     } else if (x >= mid + 1) {
80         setVal(x, y, val, mid + 1, r, rson);
81     } else {
82         setVal(x, mid, val, l, mid, lson);
83         setVal(mid + 1, y, val, mid + 1, r,
84             rson);
85     }
86     nds[n].val = nds[lson].val + nds[rson].
87         val;
88 }
89 void addVal(ll x, ll y, ll val, ll l, ll r,
90     ll n = 0) {
91     if (l == x and r == y) {
92         nds[n].val += val*(y - x + 1);
93         nds[n].addVal += val;
94         return;
95     }
96     push(l, r, n);
97     ll mid = l + (r - 1)/2;
98     if (y <= mid) {
99         addVal(x, y, val, l, mid, lson);
100     } else if (x >= mid + 1) {
101         addVal(x, y, val, mid + 1, r, rson);
102     } else {
103         addVal(x, mid, val, l, mid, lson);
104         addVal(mid + 1, y, val, mid + 1, r,
105             rson);
106     }
107     nds[n].val = nds[lson].val + nds[rson].
108         val;
109 }
110 ll query(ll x, ll y, ll l, ll r, ll n = 0) {
111     if (l == x and r == y) return nds[n].val
112         ;
113     push(l, r, n);
114     ll mid = l + (r - 1)/2;
115     if (y <= mid) {
116         return query(x, y, l, mid, lson);
117     } else if (x >= mid + 1) {
118         return query(x, y, mid + 1, r, rson)
119         ;
120     } else {

```

```

115     ll a = query(x, mid, l, mid, lson);
116     ll b = query(mid + 1, y, mid + 1, r,
117         rson);
118     return a + b;
119 }
120
121 int main() {
122     ios::sync_with_stdio(false);
123     cin.tie(nullptr);
124
125     cin >> sz >> q;
126
127     a.resize(sz + 1);
128     for (ll i = 1; i <= sz; ++i) cin >> a[i]
129         ;
130
131     nds.resize(4*sz);
132     build(1, sz);
133
134     while (q--) {
135         ll act; cin >> act;
136         if (act == 1) {
137             ll l, r, val; cin >> l >> r >>
138                 val;
139             addVal(l, r, val, 1, sz);
140         } else if (act == 2) {
141             ll l, r, val; cin >> l >> r >>
142                 val;
143             setVal(l, r, val, 1, sz);
144         } else if (act == 3) {
145             ll l, r; cin >> l >> r;
146             cout << query(l, r, 1, sz) << '\n';
147         }
148     }
149     return 0;
150 }

```

3.3 BIT

```

1 inline ll lowbit(ll x) {
2     return x & -x;
3 }
4
5 ll n;
6 vector<ll> a, bit;
7
8 void add(ll pos, ll val) {
9     for (ll i = pos; i <= n; i += lowbit(i))
10         {
11             bit[i] += val;
12         }
13 }
14 void init() {
15     for (ll i = 1; i <= n; ++i) {
16         add(i, a[i]);
17     }
18 }
19
20 ll query(ll pos) { // [1, pos]

```

```

21 ll sum = 0;
22 for (ll i = pos; i >= 1; i -= lowbit(i))
23     {
24         sum += bit[i];
25     }
26 return sum;

```

3.4 Sparse Table

```

1 // if the max size of arr is 200000
2 vector<ll> arr;
3
4 // -----
5
6 // Sparse Table
7 const ll lgmx = 17; // floor(Log2(200000))
8 ll rmq[lgmx + 1][200000 + 1];
9 void init(ll n) { // O(nlogn)
10     for (ll i = 1; i <= n; ++i) rmq[0][i] = arr[i];
11     for (ll h = 1; h <= lgmx; ++h)
12         for (ll i = 1; i <= n; i += (1 << h) - 1 <= n; ++i)
13             rmq[h][i] = min(rmq[h - 1][i],
14                             rmq[h - 1][i + (1 << (h - 1))]);
15 ll flg(ull x) {return 63 - __builtin_clzll(x);}
16 ll query(ll l, ll r) { // O(1)
17     ll h = flg(r - l + 1);
18     return min(rmq[h][l], rmq[h][r - (1 << h) + 1]);
19 }
20
21 // -----
22
23 // initialize the array
24 ll n; cin >> n;
25 arr.resize(n + 1);
26 for (ll i = 1; i <= n; ++i) cin >> arr[i];
27
28 // initialize the sparse table
29 init(n);

```

4 Geometry

4.1 Convex Hull

```

1 // pts = {p0, p1, ... pn-1}, 0-based
2 // the points in pts should be distinct
3 vector<vec> convexHull(const vector<vec> &
4     _pts) {
5     vector<vec> pts = _pts;
6     sort(pts.begin(), pts.end());
7
8     ll n = pts.size();

```

```

8     vector<vec> hull(1, pts[0]);
9     for (ll i = 1; i < n; ++i) {
10         while (hull.size() >= 2 &&
11             ori(hull[hull.size() - 2],
12                 hull.back(), pts[i]) < 0)
13             hull.pop_back();
14         hull.push_back(pts[i]);
15     }
16     ll m = hull.size();
17     for (ll i = n - 2; i >= 0; --i) {
18         while (hull.size() - m + 1 >= 2 &&
19             ori(hull[hull.size() - 2],
20                 hull.back(), pts[i]) < 0)
21             hull.pop_back();
22         hull.push_back(pts[i]);
23     }
24     hull.pop_back();
25     return hull;
26 }

```

4.2 Vector

```

1 class vec {
2 public:
3     ll x, y;
4     vec() {}
5     vec(ll _x, ll _y) : x(_x), y(_y) {}
6     vec operator+(const vec& v) const {
7         return vec(this->x + v.x, this->y + v.y);
8     }
9     vec operator-(const vec& v) const {
10        return vec(this->x - v.x, this->y - v.y);
11    }
12    ll operator*(const vec& v) const {
13        return this->x * v.x + this->y * v.y;
14    }
15    ll operator^(const vec& v) const {
16        return this->x * v.y - this->y * v.x;
17    }
18    bool operator<(const vec& v) const {
19        if (this->x != v.x) return this->x < v.x;
20        return this->y < v.y;
21    }
22    vec& operator=(const vec& v) {
23        this->x = v.x;
24        this->y = v.y;
25        return *this;
26    }
27 };
28
29 ll sign(ll x) {
30     if (x == 0) return 0;

```

```

31     if (x < 0) return -1;
32     return 1;
33 }
34
35 ll ori(const vec& o, const vec& a, const vec&
36     b) {
37     return sign((a - o) ^ (b - o));
38 }
39
40 bool isCollinear(const vec& a, const vec& b,
41     const vec& c) {
42     return ori(a, b, c) == 0;
43 }
44
45 bool isOnSeg(const vec& a, const vec& b,
46     const vec& p) {
47     return isCollinear(a, b, p) && sign((p - a) * (p - b)) <= 0;
48 }

```

4.3 Simple Polygon Area

```

1 // pts = {p0, p1, ... pn-1, p0}, 0-based
2 // for simple polygon area
3 ll shoelace2(const vector<vec> &pts) {
4     ll n = pts.size() - 2, res = 0;
5     for (ll i = 0; i <= n; ++i) {
6         res += pts[i].x * pts[i + 1].y;
7         res -= pts[i].y * pts[i + 1].x;
8     }
9     return abs(res);
10 }

```

4.4 Line Segment Intersection Test

```

1 bool isSegInter(const vec& a, const vec& b,
2     const vec& c, const vec& d) {
3     ll ori1 = ori(a, b, c);
4     ll ori2 = ori(a, b, d);
5     ll ori3 = ori(c, d, a);
6     ll ori4 = ori(c, d, b);
7     if (isCollinear(a, b, c) && isCollinear(a, b, d)) {
8         return isOnSeg(a, b, c) || isOnSeg(a, b, d) ||
9             isOnSeg(c, d, a) || isOnSeg(c, d, b);
10    }
11    return ori1 * ori2 <= 0 && ori3 * ori4 <= 0;

```

5 Graph

5.1 Kosaraju

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 #define pb push_back
5 #define rep(n) for (ll _ = 1; _ <= n; ++_)
6
7 ll V, E;
8
9 stack<ll> stk;
10 vector<bool> vis;
11 vector<vector<ll>> adj;
12 void dfs1(ll u) {
13     vis[u] = true;
14     for (ll v : adj[u]) {
15         if (vis[v]) continue;
16         dfs1(v);
17     }
18     stk.push(u);
19 }
20
21 ll sccCnt;
22 vector<ll> scc;
23 vector<vector<ll>> radj;
24 void dfs2(ll u, ll sccIdx) {
25     scc[u] = sccIdx;
26     for (ll v : radj[u]) {
27         if (scc[v] != -1) continue;
28         dfs2(v, sccIdx);
29     }
30 }
31
32 int main() {
33     ios::sync_with_stdio(false);
34     cin.tie(nullptr);
35
36     cin >> V >> E;
37     vis.resize(V + 1, false);
38     adj.resize(V + 1);
39     radj.resize(V + 1);
40     rep (E) {
41         ll x, y; cin >> x >> y;
42         adj[x].pb(y);
43         radj[y].pb(x);
44     }
45
46     for (ll u = 1; u <= V; ++u) {
47         if (vis[u]) continue;
48         dfs1(u);
49     }
50
51     sccCnt = 0;
52     scc.resize(V + 1, -1);
53     while (not stk.empty()) {
54         ll u = stk.top(); stk.pop();
55         if (scc[u] != -1) continue;
56         dfs2(u, ++sccCnt);
57     }
58
59     cout << sccCnt << '\n';
60     for (ll i = 1; i <= V; ++i) {
61         cout << scc[i] << ' ';
62     } cout << '\n';
63
64     return 0;
65 }

```

5.2 AP

```

1 void dfs(ll u, ll pa = -1) {
2     ll ch = 0; // 紀錄子節點樹
3     low[u] = in[u] = ++t;
4     for (ll v : adj[u]) {
5         if (v == pa) continue;
6         if (in[v] == -1) {
7             ch += 1; // 子節點數加一
8             dfs(v, u);
9             low[u] = min(low[u], low[v]);
10            if (pa != -1 and low[v] >= in[u]
11                ) "u is AP" // find AP
12            } else if (in[v] < in[u]) {
13                low[u] = min(low[u], in[v]);
14            }
15        }
16        if (pa == -1 and ch >= 2) "u is AP" //
            root 額外判斷

```

5.3 Dijkstra

```

1 int main() {
2     ios::sync_with_stdio(false);
3     cin.tie(nullptr);
4
5     const ll INF = 1e18;
6
7     ll n, m; cin >> n >> m;
8     vector<vector<pll>> adj(n + 1);
9     rep (m) {
10         ll u, v, w; cin >> u >> v >> w;
11         adj[u].pb(pll(w, v));
12     }
13
14     vector<ll> dis(n + 1, INF);
15     priority_queue<pll, vector<pll>, greater<
16         pll>> pq;
17
18     dis[1] = 0;
19     pq.push(pll(0, 1));
20     while (not pq.empty()) {
21         auto [uw, u] = pq.top(); pq.pop();
22
23         if (uw > dis[u]) continue;
24
25         for (auto [w, v] : adj[u]) {
26             if (dis[u] + w < dis[v]) {
27                 dis[v] = dis[u] + w;
28                 pq.push(pll(dis[v], v));
29             }
30         }
31     }

```

5.4 Floyd Warshall

```

1 int main() {
2     ios::sync_with_stdio(false);
3     cin.tie(nullptr);
4
5     cin >> n >> m;
6     mtx.clear(); mtx.resize(n + 1, vector<ll>(
7         n + 1, INF));
8     for (ll i = 1; i <= n; ++i) mtx[i][i] = 0;
9     rep (m) {
10         ll x, y, w; cin >> x >> y >> w;
11         mtx[x][y] = min(mtx[x][y], w);
12         mtx[y][x] = min(mtx[y][x], w);
13     } // "min" is used to prevent the
        multiple edges
14
15     for (ll k = 1; k <= n; ++k) {
16         for (ll i = 1; i <= n; ++i) {
17             for (ll j = 1; j <= n; ++j) {
18                 mtx[i][j] = min(mtx[i][j],
19                     min(mtx[i][k], mtx[k][j])
20                     + mtx[k][j]);
21             }
22         }
23     }

```

5.5 MST Prim

```

1 int main() {
2     ios::sync_with_stdio(false);
3     cin.tie(nullptr);
4
5     constexpr ll INF = 1e18;
6
7     ll n, m; cin >> n >> m;
8     vector<vector<pll>> adj(n + 1);
9     rep (m) {
10         ll u, v, w; cin >> u >> v >> w;
11         adj[u].pb(pll(w, v));
12         adj[v].pb(pll(w, u));
13     }
14
15     priority_queue<pll, vector<pll>, greater<
16         pll>> pq;
17     vector<bool> vis(n + 1, false);
18     vector<ll> curw(n + 1, INF);
19
20     pq.push(pll(0, 1));
21     curw[1] = 0;
22
23     ll mstCost = 0, mstEdgesCnt = 0;
24     while (mstEdgesCnt < n - 1) {
25         if (pq.empty()) {
26             // the graph is disconnected (
27                 MST D.N.E.)
28             return 0;
29         }
30         auto [uw, u] = pq.top(); pq.pop();
31
32         if (uw > curw[u]) continue;
33
34         vis[u] = true;
35         mstCost += uw;

```

```

35         mstEdgesCnt += (u == 1 ? 0 : 1);
36
37         for (const auto& [w, v] : adj[u]) {
38             if (not vis[v] and w < curw[v]) {
39                 pq.push(pll(w, v));
40                 curw[v] = w;
41             }
42         }
43     }
44     return 0;
45 }

```

5.6 MST Kruskal

```

1 vector<ll> djs, sz;
2 ll find(ll u) {
3     if (u == djs[u]) return u;
4     return djs[u] = find(djs[u]);
5 }
6 void join(ll u, ll v) {
7     u = find(u);
8     v = find(v);
9     if (u == v) return;
10    if (sz[u] < sz[v]) swap(u, v);
11    djs[v] = u;
12    sz[u] += sz[v];
13 }
14 void init(ll n) {
15     djs.clear(); djs.resize(n + 1);
16     for (ll i = 1; i <= n; ++i) djs[i] = i;
17     sz.clear(); sz.resize(n + 1, 1);
18 }
19
20 class Edge {
21 public:
22     ll u, v, w;
23 };
24
25 int main() {
26     ios::sync_with_stdio(false);
27     cin.tie(nullptr);
28
29     // nodes are 1-indexed
30     // edges are 0-indexed
31
32     ll n, m; cin >> n >> m;
33     vector<Edge> edges(m);
34     for (auto& [u, v, w] : edges) {
35         cin >> u >> v >> w;
36     }
37     sort(all(edges), [](const Edge& e1,
38         const Edge& e2) -> bool {
39         return e1.w < e2.w;
40     });
41     init(n);
42
43     ll mstCost = 0, mstEdgesCnt = 0;
44     for (const auto& [u, v, w] : edges) {
45         if (find(u) == find(v)) continue;
46         join(u, v);
47         mstCost += w;

```

```

47         mstEdgesCnt += 1;
48         if (mstEdgesCnt == n - 1) break;
49     }
50
51     // if (mstEdgesCnt < n - 1) // the graph
52     // is disconnected (MST D.N.E.)
53     return 0;
54 }

```

5.7 Topo-sort DFS

```

1 ll v;
2 vector<ll> topo;
3 vector<bool> vis;
4 vector<vector<ll>> adj(v + 1);
5 void dfs(ll node) {
6     vis[node] = true;
7     for (ll neighbor : adj[node]) {
8         if (vis[neighbor]) continue;
9         dfs(neighbor);
10    }
11    topo.pb(node);
12 }
13
14 reverse(topo.begin(), topo.end());

```

5.8 Topo-sort Kahn

```

1 int main() {
2     ios::sync_with_stdio(false);
3     cin.tie(nullptr);
4
5     ll n, m; cin >> n >> m;
6     vector<ll> indeg(n + 1, 0);
7     vector<vector<ll>> adj(n + 1);
8     rep (m) {
9         ll u, v; cin >> u >> v;
10        adj[u].pb(v);
11        indeg[v] += 1;
12    }
13
14    vector<ll> res; res.reserve(n);
15    queue<ll> q; // you can use any pool data
16    // structure here
17    for (ll u = 1; u <= n; ++u) {
18        if (indeg[u] == 0) {
19            q.push(u);
20            res.pb(u);
21        }
22    }
23    while (not q.empty()) {
24        ll u = q.front(); q.pop();
25        for (ll v : adj[u]) {
26            indeg[v] -= 1;
27            if (indeg[v] == 0) {
28                q.push(v);
29                res.pb(v);
30            }
31        }
32    }

```

```

31 }
32
33 // if (res.size() < n) // there exists a
    directed cycle
34
35 return 0;
36 }

```

5.9 Bridge

```

1 void dfs(ll u, ll pa = -1) {
2     low[u] = in[u] = ++t;
3     for (ll v : adj[u]) {
4         if (v == pa) continue;
5         if (in[v] == -1) {
6             dfs(v, u);
7             low[u] = min(low[u], low[v]);
8             if (low[v] > in[u]) "edge (u, v)
                is bridge" // find bridge
9         } else if (in[v] < in[u]) {
10             low[u] = min(low[u], in[v]);
11         }
12     }
13 }

```

5.10 Bellman Ford Detects Negative Cycle

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 using pll = pair<ll, ll>;
5 #define ff first
6 #define ss second
7 #define pb push_back
8 #define rep(n) for (ll _ = 1; _ <= n; ++_)
9
10 int main() {
11     ios::sync_with_stdio(false);
12     cin.tie(nullptr);
13
14     const ll INF = 5e12 + 1000;
15
16     ll n, m; cin >> n >> m;
17     vector<ll> dis(n + 1, INF);
18     vector<vector<pll>> adj(n + 1);
19     rep(m) {
20         ll u, v, w; cin >> u >> v >> w;
21         adj[u].pb(pll(w, v));
22     }
23     for (ll i = 1; i <= n; ++i) {
24         adj[0].pb(pll(0, i));
25     }
26
27     dis[0] = 0;
28     rep(n - 1) {
29         bool updated = false;
30         for (ll u = 0; u <= n; ++u) {
31             for (const auto& [w, v] : adj[u]) {

```

```

32         if (dis[u] < INF and dis[u] + w <
            dis[v]) {
33             dis[v] = dis[u] + w;
34             updated = true;
35         }
36     }
37     if (not updated) break;
38 }
39
40 bool hasNegativeCycle = false;
41 for (ll u = 0; not hasNegativeCycle and u
    <= n; ++u) {
42     for (const auto& [w, v] : adj[u]) {
43         if (dis[u] < INF and dis[u] + w < dis[
            v]) {
44             hasNegativeCycle = true;
45             break;
46         }
47     }
48 }
49 return 0;
50 }

```

6 Math

6.1 Big Integer Addition and Multiplication

```

1 vector<int> strToVec(string str, int sz) {
2     vector<int> r(sz, 0);
3     int strLength = str.length();
4     for (int i = strLength - 1, idx = 0; i
        >= 0; --i, ++idx) {
5         r[idx] = str[i] - '0';
6     }
7     return r;
8 }
9 // for example:
10 // strToVec("677", 4) -> 7 7 6 0
11 // strToVec("8829", 4) -> 9 2 8 8
12
13 // addition
14 string add(string x, string y) {
15     ll n = max(x.length(), y.length());
16     vector<ll> xdigit = strToVec(x, n + 1);
17     vector<ll> ydigit = strToVec(y, n + 1);
18     vector<ll> result(n + 1, 0);
19     ll carry = 0;
20     for (ll i = 0; i < n + 1; ++i) {
21         result[i] = xdigit[i] + ydigit[i] +
            carry;
22         if (result[i] >= 10) {
23             result[i] %= 10;
24             carry = 1;
25         }
26         else carry = 0;
27     }
28     ll start;

```

```

29     for (ll i = n; i >= 0; --i) {
30         if (result[i] != 0) {
31             start = i;
32             break;
33         }
34     }
35     string r = "";
36     for (ll i = start; i >= 0; --i) {
37         r += result[i] + '0';
38     }
39     return r;
40 }
41
42 // multiplication
43 string product(string x, string y) {
44     ll xlength = x.length();
45     ll ylength = y.length();
46     ll n = max(xlength, ylength);
47     vector<ll> xdigit = strToVec(x, xlength);
48     vector<ll> ydigit = strToVec(y, ylength);
49     vector<ll> result(2*n, 0);
50     for (ll i = 0; i < xlength; ++i) {
51         for (ll j = 0; j < ylength; ++j) {
52             result[i + j] += xdigit[i]*
                ydigit[j];
53             if (result[i + j] >= 10) {
54                 result[i + j + 1] += result[
                    i + j]/10;
55                 result[i + j] %= 10;
56             }
57         }
58     }
59     ll start;
60     for (ll i = 2*n - 1; i >= 0; --i) {
61         if (result[i] != 0) {
62             start = i;
63             break;
64         }
65     }
66     string r = "";
67     for (ll i = start; i >= 0; --i) {
68         r += result[i] + '0';
69     }
70     return r;
71 }

```

6.2 Modular Inverse

```

1 // extend Euclidean
2 pll extgcd(ll a, ll b) {
3     if (b == 0) return pll(1, 0);
4     pll p = extgcd(m, a%b);
5     ll x = p.ff, y = p.ss;
6     return pll(y, x - y*(a/b));
7 }
8 extgcd(a, MOD).ff // the modular inverse
9 (extgcd(a, MOD).ff%MOD + MOD)%MOD // if you
    want to ensure that it is non-negative
10
11 // fast exponentiation (make sure that MOD
    is a prime number)

```

```

12 fastPow(a, MOD - 2) // the modular inverse

```

6.3 Fast Exponentiation

```

1 ll fastPow(ll b, ll e) {
2     b = b%MOD;
3     ll res = 1;
4     while (e > 0) {
5         if (e%2 == 1) res = (res*b)%MOD;
6         b = (b*b)%MOD;
7         e /= 2;
8     }
9     return res;
10 }

```

6.4 Matrix

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Matrix {
5     int n, m; // n 行, m 列
6     vector<vector<long long>> a;
7     static const long long MOD = 1e9+7; //
        如果題目需要取模，可以改這裡
8
9     Matrix(int n, int m, bool ident = false)
        : n(n), m(m) {
10         a.assign(n, vector<long long>(m, 0));
11         if(ident) { // 單位矩陣
12             for(int i=0; i<m(n,m); i++) a[
                i][i] = 1;
13         }
14     }
15
16     // 輸出矩陣
17     void print() const {
18         for(int i=0; i<n; i++) {
19             for(int j=0; j<m; j++) cout << a
                [i][j] << " ";
20             cout << "\n";
21         }
22     }
23
24     // 矩陣乘法
25     Matrix operator*(const Matrix& o) const
        {
26         assert(m == o.n);
27         Matrix res(n, o.m);
28         for(int i=0; i<n; i++) {
29             for(int k=0; k<m; k++) if(a[i][k]
                ) {
30                 for(int j=0; j<o.m; j++) {
31                     res.a[i][j] = (res.a[i][
                        j] + a[i][k] * o.a[k][
                            j]) % MOD;
32                 }
33             }
34         }
35     }

```

```

34     }
35     return res;
36 }
37
38 // 矩陣加法
39 Matrix operator+(const Matrix& o) const
40 {
41     assert(n == o.n && m == o.m);
42     Matrix res(n, m);
43     for(int i=0; i<n; i++) {
44         for(int j=0; j<m; j++) {
45             res.a[i][j] = (a[i][j] + o.a
46                 [i][j]) % MOD;
47         }
48     }
49     return res;
50 }
51
52 // 矩陣快速冪 (n×n 方陣才能做)
53 Matrix pow(long long exp) const {
54     assert(n == m);
55     Matrix res(n, n, true), base = *this
56     ;
57     while(exp > 0) {
58         if(exp & 1) res = res * base;
59         base = base * base;
60         exp >>= 1;
61     }
62     return res;
63 }

```

6.5 Prime Sieve

```

1 // Sieve of Eratosthenes
2 ll MAXN;
3 vector<ll> spf(MAXN + 1, -1);
4 spf[0] = -2; spf[1] = -2;
5 for (ll i = 2; i*i <= MAXN; ++i) {
6     if (spf[i] == -1) {
7         for (ll j = i*i; j <= MAXN; j += i)
8             if (spf[j] == -1) {
9                 spf[j] = i;
10            }
11        }
12    }
13 }

```

6.6 Combination

```

1 ll MAXN;
2
3 vector<ll> fact; // factorial
4 vector<ll> invs; // modular inverse
5 void init() {
6     fact.resize(MAXN + 1, 1);
7     invs.resize(MAXN + 1, 1);
8     for (ll i = 2; i <= MAXN; ++i) {

```

```

9         fact[i] = (fact[i - 1]*i)%MOD;
10        invs[i] = fastPow(fact[i], MOD - 2);
11    }
12 }
13
14 inline ll C(ll n, ll k) {
15     return fact[n]*invs[k]%MOD*invs[n - k]%MOD
16 }

```

7 Others

7.1 GCC Builtin Functions

```

1 // count the number of 1 bit in x
2 __builtin_popcount(unsigned int x)
3 __builtin_popcountll(unsigned long long x)
4
5 // count leading zero of x
6 __builtin_clz(unsigned int x)
7 __builtin_clzll(unsigned long long x)
8
9 // count trailing zero of x
10 __builtin_ctz(unsigned int x)
11 __builtin_ctzll(unsigned long long x)

```

7.2 mt19937 Usage

```

1 #include <chrono>
2 #include <random>
3 using namespace std;
4 using ll = long long;
5
6 // mt19937 is a random number generator
7 // chrono::steady_clock::now().
8 // time_since_epoch().count() is used as a
9 // seed
10 mt19937 rng(chrono::steady_clock::now().
11     time_since_epoch().count());
12
13 // dist describes a range of random numbers
14 uniform_int_distribution<ll> dist(lb, ub);
15
16 dist(rng) // use rng to generate a random
17     integer in [lb, ub]

```

7.3 Custom Hash

```

1 #include <chrono>
2
3 // =====
4 // =====
5
6 class custom_hash {
7 private:
8     static uint64_t splitmix64(uint64_t x) {

```

```

9     x += 0x9e3779b97f4a7c15;
10    x = (x ^ (x >> 30)) * 0
11        xbf58476d1ce4e5b9;
12    x = (x ^ (x >> 27)) * 0
13        x94d049bb133111eb;
14    return x ^ (x >> 31);
15 }
16
17 public:
18     size_t operator()(uint64_t x) const {
19         static const uint64_t FIXED_RANDOM =
20             std::chrono::steady_clock::now
21             ().time_since_epoch().count();
22         return splitmix64(x + FIXED_RANDOM);
23     }
24 };
25
26 // for example:
27 // unordered_map<LL, LL, custom_hash> mp;
28
29 // =====
30 // =====
31
32 class custom_hash {
33 private:
34     static const uint64_t FIXED_RANDOM;
35     static uint64_t splitmix64(uint64_t x) {
36         x += 0x9e3779b97f4a7c15;
37         x = (x ^ (x >> 30)) * 0
38             xbf58476d1ce4e5b9;
39         x = (x ^ (x >> 27)) * 0
40             x94d049bb133111eb;
41         return x ^ (x >> 31);
42     }
43 }
44
45 public:
46     size_t operator()(uint64_t x) const {
47         return splitmix64(x + FIXED_RANDOM);
48     }
49
50     size_t operator()(pair<uint64_t,
51         uint64_t> p) const {
52         uint64_t h1 = splitmix64(p.first +
53             FIXED_RANDOM);
54         uint64_t h2 = splitmix64(p.second +
55             FIXED_RANDOM + 1);
56         return h1 ^ (h2*47);
57     }
58 };
59
60 const uint64_t custom_hash::FIXED_RANDOM =
61     std::chrono::steady_clock::now().
62     time_since_epoch().count();
63
64 // for example:
65 // unordered_map<pLL, LL, custom_hash> mp;

```

8 String

8.1 String Hashing

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 using pll = pair<ll, ll>;
5 #define pb push_back

```

```

7 class strHash {
8 private:
9     const ll m1 = 1e9 + 7, m2 = 1e9 + 9, p =
10         239017;
11     ll n;
12     string s;
13     vector<ll> h1, h2, p1, p2;
14 public:
15     strHash(const string& _s) {
16         n = _s.size();
17         s = _s;
18
19         h1.resize(n); h1[0] = s[0];
20         h2.resize(n); h2[0] = s[0];
21         for (ll i = 1; i < n; ++i) {
22             h1[i] = (h1[i - 1]*p%m1 + s[i])%
23                 m1;
24             h2[i] = (h2[i - 1]*p%m2 + s[i])%
25                 m2;
26         }
27
28         p1.resize(n); p1[0] = 1;
29         p2.resize(n); p2[0] = 1;
30         for (ll i = 1; i < n; ++i) {
31             p1[i] = p1[i - 1]*p%m1;
32             p2[i] = p2[i - 1]*p%m2;
33         }
34
35         pll hash(ll l, ll r) const { // [l, r]
36             if (l == 0) return pll(h1[r], h2[r]);
37             ;
38             return pll(
39                 ((h1[r] - h1[l - 1]*p1[r - l +
40                     1]%m1)%m1 + m1)%m1,
41                 ((h2[r] - h2[l - 1]*p2[r - l +
42                     1]%m2)%m2 + m2)%m2
43             );
44         }
45     };
46 };
47
48 int main() {
49     ios::sync_with_stdio(false);
50     cin.tie(nullptr);
51
52     string s1, s2; cin >> s1 >> s2;
53     ll n = s1.size(), m = s2.size();
54
55     if (n < m) {
56         cout << 0 << '\n';
57         return 0;
58     }
59
60     ll cnt = 0;
61     strHash sh1(s1), sh2(s2);
62     pll tarHash = sh2.hash(0, m - 1);
63     for (ll i = 0; i < n - m + 1; ++i) {
64         if (sh1.hash(i, i + m - 1) ==
65             tarHash) cnt += 1;
66     }
67     cout << cnt << '\n';
68     return 0;
69 }

```


8.2 KMP

```

1 string a; // 文本串
2 string b; // 模板串 (將被匹配的字串)
3 int kmp_next[N]; // next數組
4
5 void getNext(int m = b.size()){ // 初始化
6     int j = 0;
7     kmp_next[0] = 0;
8     for(int i = 1; i < m; ++i){
9         while(j > 0 && b[i] != b[j]) j =
10             kmp_next[j-1];
11         if(b[i] == b[j]) ++j;
12         kmp_next[i] = j;
13     }
14
15     int kmp(int n = a.size(), int m = b.size()){
16         // 使用KMP尋找匹配位置
17         int i, j = 0;
18         int p = -1;
19         getNext(m);
20         for(i = 0; i < n; ++i){
21             while(j > 0 && b[j] != a[i]) j =
22                 kmp_next[j-1];
23             if(b[j] == a[i]) ++j;
24             if(j == m){
25                 p = i - m + 1;
26                 break;
27             }
28         }
29         return p;
30     }
31
32     int kmp(int n = a.size(), int m = b.size()){
33         // 使用KMP計算匹配次數
34         int i, j = 0, res = 0;
35         getNext(m);
36         for(i = 0; i < n; ++i){
37             while(j > 0 && b[j] != a[i]) j =
38                 kmp_next[j-1];
39             if(b[j] == a[i]) ++j;
40             if(j == m) ++res;
41         }
42         return res;
43     }
44 }

```

8.3 LPS

```

1 //最長迴文子字串
2 #define T(x) ((x) % 2 ? s[(x) / 2] : '.')
3
4 string s;
5 int n;
6
7 int ex(int l, int r){
8     int i = 0;
9     while(l - i >= 0 && r + i < n && T(l - i)
10         == T(r + i)) i++;
11     return i;
12 }

```

```

12 int main(){
13     int main(){
14         cin >> s;
15         n = 2 * s.size() + 1;
16
17         int mx = 0;
18         int center = 0;
19         vector<int> r(n);
20         int ans = 1;
21         r[0] = 1;
22         for(int i = 1; i < n; i++){
23             int ii = center - (i - center);
24             int len = mx - i + 1;
25             if(i > mx){
26                 r[i] = ex(i, i);
27                 center = i;
28                 mx = i + r[i] - 1;
29             }
30             else if(r[ii] == len){
31                 r[i] = len + ex(i - len, i + len);
32             }
33             center = i;
34             mx = i + r[i] - 1;
35         }
36         else{
37             r[i] = min(r[ii], len);
38         }
39         ans = max(ans, r[i]);
40     }
41
42     cout << ans - 1 << "\n";
43     return 0;
44 }

```

8.4 Trie

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Trie {
5     struct Node {
6         bool endofWord;
7         vector<int> children;
8         Node() : endofWord(false), children
9             (26, -1) {}
10     };
11
12     vector<Node> trie;
13
14 public:
15     Trie() {
16         trie.emplace_back();
17     }
18
19     void insert(const string& word) {
20         int cur = 0;
21         for (char c : word) {
22             int idx = c - 'a';
23             if (trie[cur].children[idx] ==
24                 -1) {
25                 trie[cur].children[idx] =
26                     trie.size();
27                 trie.emplace_back();
28             }
29             cur = trie[cur].children[idx];
30         }
31         trie[cur].endofWord = true;
32     }
33
34     bool search(const string& word) {
35         int cur = 0;
36         for (char c : word) {
37             int idx = c - 'a';
38             if (trie[cur].children[idx] ==
39                 -1) return false;
40             cur = trie[cur].children[idx];
41         }
42         return trie[cur].endofWord;
43     }
44
45     bool startsWith(const string& prefix) {
46         int cur = 0;
47         for (char c : prefix) {
48             int idx = c - 'a';
49             if (trie[cur].children[idx] ==
50                 -1) return false;
51             cur = trie[cur].children[idx];
52         }
53         return true;
54     }
55
56     void deleteWord(const string& word) {
57         int cur = 0;
58         for (char c : word) {
59             int idx = c - 'a';
60             if (trie[cur].children[idx] ==
61                 -1) return;
62             cur = trie[cur].children[idx];
63         }
64         trie[cur].endofWord = false;
65     }
66
67     void print(int node, string prefix) {
68         const {
69             if (trie[node].endofWord) {
70                 cout << prefix << "\n";
71             }
72             for (int i = 0; i < 26; i++) {
73                 if (trie[node].children[i]
74                     != -1) {
75                     print(trie[node].children[i],
76                         prefix + char('a' + i));
77                 }
78             }
79         }
80     }
81
82     void print() const { print(0, ""); }
83 };
84
85 int main() {
86     Trie trie;
87
88     trie.insert("geek");
89     trie.insert("geeks");
90     trie.insert("code");
91     trie.insert("coder");
92     trie.insert("coding");
93
94     cout << "Trie contents:\n";
95     trie.print();
96
97     cout << "\nSearch results:\n";
98     cout << "geek: " << trie.search("geek")
99         << "\n";
100     cout << "geeks: " << trie.search("geeks")
101         << "\n";
102     cout << "code: " << trie.search("code")
103         << "\n";
104     cout << "coder: " << trie.search("coder")
105         << "\n";
106     cout << "coding: " << trie.search("coding")
107         << "\n";
108     cout << "codex: " << trie.search("codex")
109         << "\n";
110
111     trie.deleteWord("coding");
112     trie.deleteWord("geek");
113
114     cout << "\nTrie contents after deletions
115         :\n";
116     trie.print();
117
118     cout << "\nSearch results after
119         deletions:\n";
120     cout << "coding: " << trie.search("coding")
121         << "\n";
122     cout << "geek: " << trie.search("geek")
123         << "\n";
124
125     return 0;
126 }

```

```

25     }
26     cur = trie[cur].children[idx];
27 }
28 trie[cur].endofWord = true;
29 }
30
31 bool search(const string& word) {
32     int cur = 0;
33     for (char c : word) {
34         int idx = c - 'a';
35         if (trie[cur].children[idx] ==
36             -1) return false;
37         cur = trie[cur].children[idx];
38     }
39     return trie[cur].endofWord;
40 }
41
42 bool startsWith(const string& prefix) {
43     int cur = 0;
44     for (char c : prefix) {
45         int idx = c - 'a';
46         if (trie[cur].children[idx] ==
47             -1) return false;
48         cur = trie[cur].children[idx];
49     }
50     return true;
51 }
52
53 void deleteWord(const string& word) {
54     int cur = 0;
55     for (char c : word) {
56         int idx = c - 'a';
57         if (trie[cur].children[idx] ==
58             -1) return;
59         cur = trie[cur].children[idx];
60     }
61     trie[cur].endofWord = false;
62 }
63
64 void print(int node, string prefix) {
65     const {
66         if (trie[node].endofWord) {
67             cout << prefix << "\n";
68         }
69         for (int i = 0; i < 26; i++) {
70             if (trie[node].children[i]
71                 != -1) {
72                 print(trie[node].children[i],
73                     prefix + char('a' + i));
74             }
75         }
76     }
77 }
78
79 void print() const { print(0, ""); }
80 };
81
82 int main() {
83     Trie trie;
84
85     trie.insert("geek");
86     trie.insert("geeks");
87     trie.insert("code");
88     trie.insert("coder");
89     trie.insert("coding");
90
91     cout << "Trie contents:\n";
92     trie.print();
93
94     cout << "\nSearch results:\n";
95     cout << "geek: " << trie.search("geek")
96         << "\n";
97     cout << "geeks: " << trie.search("geeks")
98         << "\n";
99     cout << "code: " << trie.search("code")
100         << "\n";
101     cout << "coder: " << trie.search("coder")
102         << "\n";
103     cout << "coding: " << trie.search("coding")
104         << "\n";
105     cout << "codex: " << trie.search("codex")
106         << "\n";
107
108     trie.deleteWord("coding");
109     trie.deleteWord("geek");
110
111     cout << "\nTrie contents after deletions
112         :\n";
113     trie.print();
114
115     cout << "\nSearch results after
116         deletions:\n";
117     cout << "coding: " << trie.search("coding")
118         << "\n";
119     cout << "geek: " << trie.search("geek")
120         << "\n";
121
122     return 0;
123 }

```

8.5 Z-value

```

1 // CSES: String Matching
2 // Given a string and a pattern, your task
3 // is to count
4 // the number of positions where the pattern
5 // occurs in the string.
6
7 #include <bits/stdc++.h>
8 using namespace std;
9 using ll = long long;
10
11 vector<ll> z_value(const string& s) {
12     ll n = s.size();
13     vector<ll> z(n);
14     ll l = 0, r = 0;
15     for (ll i = 1; i < n; ++i) {
16         if (i <= r) z[i] = min(z[i - l], r -
17             i + 1);
18         while (i + z[i] < n && s[z[i]] == s[
19             i + z[i]]) z[i] += 1;
20     }
21 }

```

```

16     if (i + z[i] - 1 > r) {
17         l = i;
18         r = i + z[i] - 1;
19     }
20 }
21 return z;
22 }
23
24 int main() {
25     ios::sync_with_stdio(false);
26     cin.tie(nullptr);
27
28     string s1, s2; cin >> s1 >> s2;
29     ll n = s1.size(), m = s2.size();
30
31     ll cnt = 0;
32     string s = s2 + "$" + s1;
33     vector<ll> z = z_value(s);
34     for (ll i = m; i < s.size(); ++i) {
35         if (z[i] == m) cnt += 1;
36     }
37     cout << cnt << '\n';
38
39     return 0;
40 }
41
42 // =====
43 // =====
44
45 // CSES: Finding Borders
46 // A border of a string is a prefix that is
47 // also a suffix of
48 // the string but not the whole string. For
49 // example,
50 // the borders of abcababcbab are ab and
51 // abcab.
52 // Your task is to find all border lengths
53 // of a given string.
54
55 #include <bits/stdc++.h>
56 using namespace std;
57 using ll = long long;
58 #define pb push_back
59
60 vector<ll> z_value(const string& s) {
61     ll n = s.size();
62     vector<ll> z(n);
63     ll l = 0, r = 0;
64     for (ll i = 1; i < n; ++i) {
65         if (i <= r) z[i] = min(z[i - l], r -
66             i + 1);
67         while (i + z[i] < n && s[z[i]] == s[
68             i + z[i]]) z[i] += 1;
69         if (i + z[i] - 1 > r) {
70             l = i;
71             r = i + z[i] - 1;
72         }
73     }
74     return z;
75 }
76
77 int main() {
78     ios::sync_with_stdio(false);
79     cin.tie(nullptr);
80
81     string s; cin >> s;
82
83     ll n = s.size();
84
85     vector<ll> z = z_value(s);
86     for (ll i = 1; i < n; ++i) {
87         if (i + z[i] == n) res.pb(i);
88     }
89     sort(res.begin(), res.end());
90     for (ll x : res) {
91         cout << x << ' ';
92     }
93     cout << '\n';
94
95     return 0;
96 }

```

```

76     ll n = s.size();
77
78     vector<ll> z = z_value(s);
79     for (ll i = 1; i < n; ++i) {
80         if (i + z[i] == n) res.pb(z[i]);
81     }
82     sort(res.begin(), res.end());
83     for (ll x : res) {
84         cout << x << ' ';
85     }
86     cout << '\n';
87
88     return 0;
89 }
90 // =====
91 // =====
92
93 // CSES: Finding Periods
94 // A period of a string is a prefix that can
95 // be used to generate
96 // the whole string by repeating the prefix.
97 // The last repetition
98 // may be partial. For example, the periods
99 // of abcbabca are abc, abcbab and abcbabca.
100 // Your task is to find all period lengths
101 // of a string.
102
103 #include <bits/stdc++.h>
104 using namespace std;
105 using ll = long long;
106 #define pb push_back
107
108 vector<ll> z_value(const string& s) {
109     ll n = s.size();
110     vector<ll> z(n);
111     ll l = 0, r = 0;
112     for (ll i = 1; i < n; ++i) {
113         if (i <= r) z[i] = min(z[i - l], r -
114             i + 1);
115         while (i + z[i] < n && s[z[i]] == s[
116             i + z[i]]) z[i] += 1;
117         if (i + z[i] - 1 > r) {
118             l = i;
119             r = i + z[i] - 1;
120         }
121     }
122     return z;
123 }
124
125 int main() {
126     ios::sync_with_stdio(false);
127     cin.tie(nullptr);
128
129     string s; cin >> s;
130
131     ll n = s.size();
132
133     vector<ll> z = z_value(s);
134     for (ll i = 1; i < n; ++i) {
135         if (i + z[i] == n) res.pb(i);
136     }
137     sort(res.begin(), res.end());
138     for (ll x : res) {
139         cout << x << ' ';
140     }
141     cout << '\n';
142
143     return 0;
144 }

```

136 }

8.6 Manacher

```

1 // CSES: Longest Palindrome
2 // Given a string, your task is to determine
3 // the longest
4 // palindromic substring of the string. For
5 // example,
6 // the longest palindrome in aybabetu is bab.
7
8 #include <bits/stdc++.h>
9 using namespace std;
10 using ll = long long;
11
12 int main() {
13     ios::sync_with_stdio(false);
14     cin.tie(nullptr);
15
16     string t, s = "^#"; cin >> t;
17     ll n = t.size(), m = 2*n + 3;
18     for (ll i = 0; i < n; ++i) {
19         s += t[i];
20         s += (i == n - 1 ? "$" : "#");
21     }
22
23     ll c = 1;
24     vector<ll> p(m); p[1] = 0;
25     for (ll i = 2; i <= m - 3; ++i) {
26         if (i < c + p[c]) p[i] = min(p[c -
27             (i - c)], (c + p[c]) - i);
28         while (i + p[i] + 1 < m &&
29             i - p[i] - 1 >= 0 &&
30             s[i + p[i] + 1] == s[i - p[i] - 1])
31             p[i] += 1;
32         if (i + p[i] > c + p[c]) c = i;
33     }
34
35     ll j = 2;
36     for (ll i = 3; i <= m - 3; ++i) {
37         if (p[i] > p[j]) j = i;
38     }
39
40     for (ll i = j - p[j] + 1; i <= j + p[j]
41         - 1; i += 2) {
42         cout << s[i];
43     }
44     cout << '\n';
45
46     return 0;
47 }
48
49 // =====
50 // =====
51
52 // CSES: All Palindromes
53 // Given a string, calculate for each
54 // position the length
55 // of the longest palindrome that ends at
56 // that position.
57
58 #include <bits/stdc++.h>
59 using namespace std;
60 using ll = long long;

```

```

53 ll n, m;
54 string ori_s, s;
55 vector<ll> p, rt, dp;
56
57 int main() {
58     ios::sync_with_stdio(false);
59     cin.tie(nullptr);
60
61     cin >> ori_s;
62     n = ori_s.size();
63     m = 2*n + 3;
64     s = "^#";
65     for (ll i = 0; i < n; ++i) {
66         s += ori_s[i];
67         s += (i == n - 1 ? "$" : "#");
68     }
69
70     ll c = 0;
71     p.resize(m);
72     for (ll i = 2; i <= m - 3; ++i) {
73         if (i < c + p[c]) p[i] = min(p[c -
74             (i - c)], (c + p[c]) - i);
75         while (i - p[i] - 1 >= 0 &&
76             i + p[i] + 1 < m &&
77             s[i - p[i] - 1] == s[i + p[i]
78                 + 1]) p[i] += 1;
79         if (i + p[i] > c + p[c]) c = i;
80     }
81
82     rt.resize(n, 1);
83     for (ll i = 2; i <= m - 3; ++i) {
84         ll y = ((i + p[i] - 1) - 2)/2;
85         if (s[i] == '#') {
86             ll x = ((i + 1) - 2)/2;
87             rt[y] = max(rt[y], (y - x + 1)
88                 *2);
89         } else {
90             ll x = (i - 2)/2;
91             rt[y] = max(rt[y], (y - x)*2 +
92                 1);
93         }
94     }
95
96     dp.resize(n); dp[n - 1] = rt[n - 1];
97     for (ll i = n - 2; i >= 0; --i) {
98         dp[i] = max(rt[i], dp[i + 1] - 2);
99     }
100
101     for (ll x : dp) {
102         cout << x << ' ';
103     }
104     cout << '\n';
105
106     return 0;
107 }

```

9 Tree

9.1 Binary Lifting

```

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

```



```

2 using namespace std;
3 using ll = long long;
4
5 inline ll flg(ll x) {
6     return 63 - __builtin_clzll(x);
7 }
8
9 inline bool isOnBit(ll x, ll i) {
10     return ((1LL << i) & x) > 0;
11 }
12
13 ll n, q, lgn;
14 vector<vector<ll>> blf;
15
16 void init() {
17     blf[0][1] = -1;
18     for (ll u = 2; u <= n; ++u) cin >> blf
19         [0][u];
20
21     for (ll h = 1; h <= lgn; ++h) {
22         for (ll u = 1; u <= n; ++u) {
23             ll nt = blf[h - 1][u];
24             blf[h][u] = nt == -1 ? -1 : blf[
25                 h - 1][nt];
26         }
27     }
28
29 ll query(ll u, ll step) {
30     ll cur = u;
31     for (ll i = 30; i >= 0; --i) {
32         if (isOnBit(step, i)) {
33             cur = blf[i][cur];
34             if (cur == -1) return -1;
35         }
36     }
37     return cur;
38 }
39
40 int main() {
41     ios::sync_with_stdio(false);
42     cin.tie(nullptr);
43
44     cin >> n >> q;
45     lgn = flg(n);
46     blf.resize(lgn + 1, vector<ll>(n + 1));
47     init();
48
49 while (q--) {
50     ll u, step; cin >> u >> step;
51     cout << query(u, step) << '\n';
52 }
53
54 return 0;
55 }

```

9.2 LCA

```

1 // Use binary lifting
2
3 #include <bits/stdc++.h>
4 using namespace std;
5 using ll = long long;
6 #define pb push_back

```

```

7
8 inline ll flg(ll x) {
9     return 63 - __builtin_clzll(x);
10 }
11
12 inline bool isOnBit(ll x, ll i) {
13     return ((1LL << i) & x) > 0;
14 }
15
16 ll n, q, lgn;
17 vector<ll> d;
18 vector<vector<ll>> blf, adj;
19
20 void init() {
21     blf[0][1] = -1;
22     for (ll u = 2; u <= n; ++u) {
23         ll v; cin >> v;
24         blf[0][u] = v;
25         adj[v].pb(u);
26     }
27
28     for (ll h = 1; h <= lgn; ++h) {
29         for (ll u = 1; u <= n; ++u) {
30             ll nt = blf[h - 1][u];
31             blf[h][u] = nt == -1 ? -1 : blf[
32                 h - 1][nt];
33         }
34     }
35
36 ll query(ll u, ll step) {
37     ll cur = u;
38     for (ll i = 30; i >= 0; --i) {
39         if (isOnBit(step, i)) {
40             cur = blf[i][cur];
41             if (cur == -1) return -1;
42         }
43     }
44     return cur;
45 }
46
47 void dfs(ll u, ll dn) {
48     d[u] = dn;
49     for (ll v : adj[u]) {
50         dfs(v, dn + 1);
51     }
52 }
53
54 ll lca(ll u, ll v) {
55     if (d[u] > d[v]) swap(u, v);
56     if (d[u] < d[v]) v = query(v, d[v] - d[u]);
57
58     if (u == v) return u;
59     for (ll h = lgn; h >= 0; --h) {
60         ll ntu = blf[h][u];
61         ll ntv = blf[h][v];
62         if (ntu == -1 or ntv == -1 or ntu ==
63             ntv) continue;
64         u = ntu;
65         v = ntv;
66     }
67     return blf[0][u];
68 }
69
70 int main() {
71     ios::sync_with_stdio(false);
72     cin.tie(nullptr);
73
74     cin >> n >> q;
75     lgn = flg(n);
76     d.resize(n + 1);
77     adj.resize(lgn + 1, vector<ll>(n + 1));
78     init();
79
80 while (q--) {
81     ll u, v; cin >> u >> v;
82     cout << lca(u, v) << '\n';
83 }
84
85 return 0;
86 }

```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

1 DP	1	1.4 Subset DP	1	5 Graph	3	6.5 Prime Sieve	6
1.1 Digit DP	1	1.5 LIS	1	5.1 Kosaraju	3	6.6 Combination	6
1.2 Interval DP	1	1.6 LCS	1	5.2 AP	4		
1.3 Knapsack	1			5.3 Dijkstra	4	7 Others	6
		2 D&C	1	5.4 Floyed Warshall	4	7.1 GCC Builtin Functions	6
		2.1 MergeSort Finds the Number of Inversions	1	5.5 MST Prim	4	7.2 mt19937 Usage	6
				5.6 MST Kruskal	4	7.3 Custom Hash	6
		3 Data Structure	2	5.7 Topo-sort DFS	4		
		3.1 DSU	2	5.8 Topo-sort Kahn	4	8 String	6
		3.2 Segment Tree	2	5.9 Bridge	5	8.1 String Hashing	6
		3.3 BIT	2	5.10 Bellman Ford Detects Nega- tive Cycle	5	8.2 KMP	7
		3.4 Sparse Table	3			8.3 LPS	7
				6 Math	5	8.4 Trie	7
		4 Geometry	3	6.1 Big Integer Addition and Multiplication	5	8.5 Z-value	7
		4.1 Convex Hull	3	6.2 Modular Inverse	5	8.6 Manacher	8
		4.2 Vector	3	6.3 Fast Exponentiation	5		
		4.3 Simple Polygon Area	3	6.4 Matrix	5	9 Tree	8
		4.4 Line Segment Intersection Test	3			9.1 Binary Lifting	8
						9.2 LCA	9

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 namespace system_test {
5
6 const size_t KB = 1024;
7 const size_t MB = KB * 1024;
8 const size_t GB = MB * 1024;

```

```

9 size_t block_size, bound;
10 void stack_size_dfs(size_t depth = 1) {
11     if (depth >= bound)
12         return;
13     int8_t ptr[block_size]; // 若無法編譯將
14                             // block_size 改成常數
15     memset(ptr, 'a', block_size);
16     cout << depth << endl;
17     stack_size_dfs(depth + 1);
18 }
19
20 void stack_size_and_runtime_error(size_t
21     block_size, size_t bound = 1024) {
22     system_test::block_size = block_size;
23     system_test::bound = bound;
24     stack_size_dfs();
25 }
26
27 double speed(int iter_num) {
28     const int block_size = 1024;
29     volatile int A[block_size];
30     auto begin = chrono::high_resolution_clock
31         ::now();
32     while (iter_num--)
33         for (int j = 0; j < block_size; ++j)
34             A[j] += j;
35     auto end = chrono::high_resolution_clock::
36         now();

```

```

37 chrono::duration<double> diff = end -
38     begin;
39 return diff.count();
40 }
41
42 void runtime_error_1() {
43     // Segmentation fault
44     int *ptr = nullptr;
45     *(ptr + 7122) = 7122;
46 }
47
48 void runtime_error_2() {
49     // Segmentation fault
50     int *ptr = (int *)memset;
51     *ptr = 7122;
52 }
53
54 void runtime_error_3() {
55     // munmap_chunk(): invalid pointer
56     int *ptr = (int *)memset;
57     delete ptr;
58 }
59
60 void runtime_error_4() {
61     // free(): invalid pointer
62     int *ptr = new int[7122];
63     ptr += 1;
64     delete[] ptr;
65 }

```

```

62
63 void runtime_error_5() {
64     // maybe illegal instruction
65     int a = 7122, b = 0;
66     cout << (a / b) << endl;
67 }
68
69 void runtime_error_6() {
70     // floating point exception
71     volatile int a = 7122, b = 0;
72     cout << (a / b) << endl;
73 }
74
75 void runtime_error_7() {
76     // call to abort.
77     assert(false);
78 }
79
80 } // namespace system_test
81
82 #include <sys/resource.h>
83 void print_stack_limit() { // only work in
84     Linux
85     struct rlimit l;
86     getrlimit(RLIMIT_STACK, &l);
87     cout << "stack_size = " << l.rlim_cur << "
88         byte" << endl;
89 }

```