

1 Bitwise Trick

1.1 tricks

```

1 bool isOdd(int x) {
2     return x & 1;
3 }
4
5 int isolateRightmostSetBit(int x) {
6     return x & -x; // 取得最右邊的1
7 }
8
9 bool isPowerOfTwo(int x) {
10    return x > 0 && (x & (x - 1)) == 0;
11 }
12
13 int countSetBits(int x) {
14     int count = 0;
15     while (x) {
16         x &= (x - 1); // 清除最右邊的 1
17         count++;
18     }
19     return count;
20 }
21
22 int turnOffRightmostSetBit(int x) {
23     return x & (x - 1); // 將最右邊的1設為0
24 }
25
26 int setBit(int x, int pos) {
27     return x | (1 << pos); // 設該位元為1
28 }
29
30 int clearBit(int x, int pos) {
31     return x & ~(1 << pos); // 設該位元為0
32 }
33
34 int toggleBit(int x, int pos) {
35     return x ^ (1 << pos); // 切換該位元
36 }
37
38 int getBit(int x, int pos) {
39     return (x >> pos) & 1; // 取得該位元值
40 }
41
42 int modPowerOfTwo(int x, int mod) {
43     return x & (mod - 1); // x % mod, mod 必須
44     // 是 2 的幕次
45 }
46
47 // a + b = (a ^ b) + 2 * (a & b)
48 int add(int a, int b) {
49     while (b != 0) {
50         int carry = a & b; // 計算進位
51         a = a ^ b; // 無進位相加
52         b = carry << 1; // 將進位左移一位
53     }
54     return a;
55 }
56
57 // a - b = (a ^ b) - 2 * ((~a) & b)
58 int subtract(int a, int b) {

```

```

57     while (b != 0) {
58         int borrow = (~a) & b; // 計算借位
59         a = a ^ b; // 無借位相減
60         b = borrow << 1; // 將借位左移一位
61     }
62     return a;
63 }
64 // a * b = sum(a << i) for each set bit i in b
65 int multiply(int a, int b) {
66     int result = 0;
67     while (b > 0) {
68         if (b & 1) { // 如果 b 的最低位是 1
69             result = add(result, a); // 使用
70             // 上面的加法函數
71         }
72         a <<= 1; // 將 a 左移一位 (相當於乘以 2)
73         b >>= 1; // 將 b 右移一位 (相當於除以 2)
74     }
75     return result;
76 }
77 // a / b = sum(1 << i) for each set bit i in a
78 int divide(int a, int b) {
79     if (b == 0) throw std::invalid_argument("Division by zero");
80     int result = 0;
81     int power = 1;
82     int tempB = b;
83     while (tempB <= a && tempB > 0) { // 防止溢位
84         tempB <<= 1;
85         power <<= 1;
86     }
87     while (power > 1) {
88         tempB >>= 1;
89         power >>= 1;
90         if (a >= tempB) {
91             a = subtract(a, tempB); // 使用
92             // 上面的減法函數
93             result = add(result, power); // 使用上面的加法函數
94         }
95     }
96     return result;
97 }
98
99 int findUnique(const std::vector<int>& nums) {
100    {
101        int unique = 0;
102        for (int num : nums) {
103            unique ^= num; // XOR 相同的數字會抵消
104        }
105        return unique;
106    }
107 }
108
109 int graycode(int n) { // n 轉 Gray code

```

2 DP

2.1 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     return dp[pos][tight][cnt] = res;
66 }
67
68 int main() {
69     ios::sync_with_stdio(false);
70     cin.tie(nullptr);
71
72     cin >> num >> k;
73
74     memset(dp, -1, sizeof(dp));
75
76     cout << count(0, true, k) << '\n';
77
78     return 0;
79 }
80

```

2.2 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

```

3 D&C

3.1 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 - 1 + 1);
4 while (1 <= 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 (1 <= 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 }

```

4 Data Structure

4.1 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];

```

```

14 cc -= 1;
15 }
16 void init(ll n) {
17     djs.clear(); djs.resize(n + 1);
18     for (ll i = 1; i <= n; ++i) djs[i] = i;
19     sz.clear(); sz.resize(n + 1, 1);
20     cc = n;
21 }

```

4.2 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
40 void push(ll l, ll r, ll n) {
41     ll mid = l + (r - 1)/2;
42
43     if (nds[n].isSet) {
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

```

```

49     nds[rson].val = nds[n].setVal*(r - (
50         mid + 1) + 1);
51     nds[rson].setVal = nds[n].setVal;
52     nds[rson].addVal = 0;
53     nds[rson].isSet = true;
54
55     nds[n].isSet = false;
56 }
57
58 nds[lson].val += nds[n].addVal*(mid - 1
59     + 1);
60 nds[lson].addVal += nds[n].addVal;
61
62 nds[rson].val += nds[n].addVal*(r - (mid
63     + 1) + 1);
64 nds[rson].addVal += nds[n].addVal;
65
66 nds[n].addVal = 0;
67 }
68
69 void setVal(ll x, ll y, ll val, ll l, ll r,
70     ll n = 0) {
71     if (l == x and r == y) {
72         nds[n].val = val*(y - x + 1);
73         nds[n].setVal = val;
74         nds[n].addVal = 0;
75         nds[n].isSet = true;
76         return;
77     }
78     push(l, r, n);
79     ll mid = l + (r - 1)/2;
80     if (y <= mid) {
81         setVal(x, y, val, l, mid, lson);
82     } else if (x >= mid + 1) {
83         setVal(x, y, val, mid + 1, r, rson);
84     } else {
85         setVal(x, mid, val, l, mid, lson);
86         setVal(mid + 1, y, val, mid + 1, r,
87             rson);
88     }
89     nds[n].val = nds[lson].val + nds[rson].
90         val;
91
92 void addVal(ll x, ll y, ll val, ll l, ll r,
93     ll n = 0) {
94     if (l == x and r == y) {
95         nds[n].val += val*(y - x + 1);
96         nds[n].addVal += val;
97         return;
98     }
99     push(l, r, n);
100    ll mid = l + (r - 1)/2;
101    if (y <= mid) {
102        addVal(x, y, val, l, mid, lson);
103    } else if (x >= mid + 1) {
104        addVal(x, y, val, mid + 1, r, rson);
105    } else {
106        addVal(x, mid, val, l, mid, lson);
107        addVal(mid + 1, y, val, mid + 1, r,
108            rson);
109    }
110    nds[n].val = nds[lson].val + nds[rson].
111        val;

```

```

112 ll query(ll x, ll y, ll l, ll r, ll n = 0) {
113     if (l == x and r == y) return nds[n].val;
114     ;
115     push(l, r, n);
116     ll mid = l + (r - 1)/2;
117     if (y <= mid) {
118         return query(x, y, l, mid, lson);
119     } else if (x >= mid + 1) {
120         return query(x, y, mid + 1, r, rson);
121     } else {
122         ll a = query(x, mid, l, mid, lson);
123         ll b = query(mid + 1, y, mid + 1, r,
124             rson);
125         return a + b;
126     }
127 }
128
129 int main() {
130     ios::sync_with_stdio(false);
131     cin.tie(nullptr);
132
133     cin >> sz >> q;
134
135     a.resize(sz + 1);
136     for (ll i = 1; i <= sz; ++i) cin >> a[i]
137         ];
138
139     nds.resize(4*sz);
140     build(1, sz);
141
142     while (q--) {
143         ll act; cin >> act;
144         if (act == 1) {
145             ll l, r, val; cin >> l >> r >>
146                 val;
147             addVal(l, r, val, 1, sz);
148         } else if (act == 2) {
149             ll l, r, val; cin >> l >> r >>
150                 val;
151             setVal(l, r, val, 1, sz);
152         } else if (act == 3) {
153             ll l, r; cin >> l >> r;
154             cout << query(l, r, 1, sz) << '\n';
155         }
156     }
157     return 0;
158 }

```

4.3 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         ;

```

```

10     bit[i] += val;
11 }
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;
27 }

```

4.4 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(n log n)
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 + (1 << h) - 1 <= n; ++i)
13         rmq[h][i] = min(rmq[h - 1][i],
14                         rmq[h - 1][i + (1 << (h - 1))]);
15 }
16 ll flg(ull x) { return 63 - __builtin_clzll(x); }
17 ll query(ll l, ll r) { // O(1)
18     ll h = flg(r - l + 1);
19     return min(rmq[h][l], rmq[h][r - (1 << h) + 1]);
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);

```

5 Geometry

5.1 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();
9     vector<vec> hull(1, pts[0]);
10    for (ll i = 1; i < n; ++i) {
11        while (hull.size() >= 2 &&
12            ori(hull[hull.size() - 2],
13                hull.back(), pts[i]) < 0)
14        {
15            hull.pop_back();
16        }
17        hull.push_back(pts[i]);
18    }
19
20    ll m = hull.size();
21    for (ll i = n - 2; i >= 0; --i) {
22        while (hull.size() - m + 1 >= 2 &&
23            ori(hull[hull.size() - 2],
24                hull.back(), pts[i]) < 0)
25        {
26            hull.pop_back();
27        }
28        hull.push_back(pts[i]);
29    }
30    hull.pop_back();
31    return hull;
32 }

```

5.2 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 +
8             v.y);
9     }
10    vec operator-(const vec& v) const {
11        return vec(this->x - v.x, this->y -
12            v.y);
13    }
14
15    ll operator*(const vec& v) const {
16        return this->x * v.x + this->y * v.y;
17    }
18
19    ll operator^(const vec& v) const {
20        return this->x * v.y - this->y * v.x;
21    }
22 }

```

```

17 }
18 bool operator<(const vec& v) const {
19     if (this->x != v.x) return this->x <
20         v.x;
21     return this->y < v.y;
22 }
23 vec& operator=(const vec& v) {
24     this->x = v.x;
25     this->y = v.y;
26     return *this;
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 -
48         a) * (p - b)) <= 0;
49 }

```

5.3 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(
8         a, b, d)) {
9         return isOnSeg(a, b, c) || isOnSeg(a,
10             b, d) ||
11             isOnSeg(c, d, a) || isOnSeg(c,
12                 d, b);
13     }
14     return ori1 * ori2 <= 0 && ori3 * ori4
15         <= 0;
16 }

```

6 Graph

6.1 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 }

```

6.2 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

```

6.3 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<
        pll>> pq;
16
17     dis[1] = 0;
18     pq.push(pll(0, 1));
19     while (not pq.empty()) {
20         auto [uw, u] = pq.top(); pq.pop();
21
22         if (uw > dis[u]) continue;
23
24         for (auto [w, v] : adj[u]) {
25             if (dis[u] + w < dis[v]) {
26                 dis[v] = dis[u] + w;
27                 pq.push(pll(dis[v], v));
28             }
29         }
30     }
31 }

```

6.4 MST Prim

```

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         adj[v].pb(pll(w, u));
13     }
14
15     priority_queue<pll, vector<pll>, greater<
        pll>> pq;
16     vector<bool> vis(n + 1, false);
17     vector<ll> curw(n + 1, INF);
18
19     pq.push(pll(0, 1));
20     curw[1] = 0;
21
22     ll mstCost = 0, mstEdgesCnt = 0;
23     while (mstEdgesCnt < n - 1) {
24         if (pq.empty()) {
25             // the graph is disconnected (
                MST D.N.E.)
26             return 0;
27         }
28         auto [uw, u] = pq.top(); pq.pop();
29
30         if (uw > curw[u]) continue;
31
32         vis[u] = true;
33         mstCost += uw;
34         mstEdgesCnt += (u == 1 ? 0 : 1);
35
36         for (const auto& [w, v] : adj[u]) {
37             if (not vis[v] and w < curw[v]) {
38                 pq.push(pll(w, v));
39                 curw[v] = w;
40             }
41         }
42     }
43     return 0;
44 }

```

6.5 MST Kruskal

```

1 vector<ll> djs, sz;
2 ll find(ll u) {...}
3 void join(ll u, ll v) {...}
4 void init(ll n) {...}
5
6 class Edge{
7 public:
8     ll u, v, w;
9 };
10
11 int main() {

```

```

12     ios::sync_with_stdio(false);
13     cin.tie(nullptr);
14
15     // nodes are 1-indexed
16     // edges are 0-indexed
17
18     ll n, m; cin >> n >> m;
19     vector<Edge> edges(m);
20     for (auto& [u, v, w] : edges) {
21         cin >> u >> v >> w;
22     }
23     sort(all(edges), [](const Edge& e1,
24         const Edge e2) -> bool {
25         return e1.w < e2.w;
26     });
27     init(n);
28
29     ll mstCost = 0, mstEdgesCnt = 0;
30     for (const auto& [u, v, w] : edges) {
31         if (find(u) == find(v)) continue;
32         join(u, v);
33         mstCost += w;
34         mstEdgesCnt += 1;
35         if (mstEdgesCnt == n - 1) break;
36     }
37     // if (mstEdgesCnt < n - 1) // the graph
        is disconnected (MST D.N.E.)
38
39     return 0;
40 }

```

6.6 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 }

```

6.7 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 <
33                     dis[v]) {
34                     dis[v] = dis[u] + w;
35                     updated = true;
36                 }
37             }
38             if (not updated) break;
39         }
40
41         bool hasNegativeCycle = false;
42         for (ll u = 0; not hasNegativeCycle and u
43             <= n; ++u) {
44             for (const auto& [w, v] : adj[u]) {
45                 if (dis[u] < INF and dis[u] + w < dis[
46                     v]) {
47                     hasNegativeCycle = true;
48                     break;
49                 }
50             }
51             return 0;
52         }

```

7 Math

7.1 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
5         >= 0; --i, ++idx) {
6         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] +
22             carry;
23         if (result[i] >= 10) {
24             result[i] %= 10;
25             carry = 1;
26         }
27         else carry = 0;
28     }
29     ll start;
30     for (ll i = n; i >= 0; --i) {
31         if (result[i] != 0) {
32             start = i;
33             break;
34         }
35     }
36     string r = "";
37     for (ll i = start; i >= 0; --i) {
38         r += result[i] + '0';
39     }
40     return r;
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] *
53                 ydigit[j];
54             if (result[i + j] >= 10) {
55                 result[i + j + 1] += result[
56                     i + j] / 10;
57                 result[i + j] %= 10;
58             }
59         }
60     }
61     ll start;
62     for (ll i = 2*n - 1; i >= 0; --i) {
63         if (result[i] != 0) {
64             start = i;
65             break;
66         }
67     }
68     string r = "";

```

```

67     for (ll i = start; i >= 0; --i) {
68         r += result[i] + '0';
69     }
70     return r;
71 }

```

7.2 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
10 want to ensure that it is non-negative
11 // fast exponentiation (make sure that MOD
12 is a prime number)
13 fastPow(a, MOD - 2) // the modular inverse

```

7.3 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; //
8     // 如果題目需要取模 · 可以改這裡
9
10     Matrix(int n, int m, bool ident = false)
11         : n(n), m(m) {
12         a.assign(n, vector<long long>(m, 0));
13     }
14     if(ident) { // 單位矩陣
15         for(int i=0; i<min(n,m); i++) a[
16             i][i] = 1;
17     }
18 }
19
20 // 輸出矩陣
21 void print() const {
22     for(int i=0; i<n; i++) {
23         for(int j=0; j<m; j++) cout << a
24             [i][j] << " ";
25         cout << "\n";
26     }
27 }
28
29 // 矩陣乘法
30 Matrix operator*(const Matrix& o) const
31 {
32     assert(m == o.n);
33     Matrix res(n, o.m);
34     for(int i=0; i<n; i++) {

```

```

29     for(int k=0; k<m; k++) if(a[i][k
30         ]) {
31         for(int j=0; j<o.m; j++) {
32             res.a[i][j] = (res.a[i][
33                 j] + a[i][k] * o.a[k
34                     ][j]) % MOD;
35         }
36     }
37     return res;
38 }
39
40 // 矩陣加法
41 Matrix operator+(const Matrix& o) const
42 {
43     assert(n == o.n && m == o.m);
44     Matrix res(n, m);
45     for(int i=0; i<n; i++) {
46         for(int j=0; j<m; j++) {
47             res.a[i][j] = (a[i][j] + o.a
48                 [i][j]) % MOD;
49         }
50     }
51     return res;
52 }
53
54 // 矩陣快速幂 (n*n 方陣才能做)
55 Matrix pow(long long exp) const {
56     assert(n == m);
57     Matrix res(n, n, true), base = *this
58     ;
59     while(exp > 0) {
60         if(exp & 1) res = res * base;
61         base = base * base;
62         exp >>= 1;
63     }
64     return res;
65 }
66 }

```

8 Others

8.1 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)

```

9 String

9.1 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
6
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 int main() {
48     ios::sync_with_stdio(false);
49     cin.tie(nullptr);
50
51     string s1, s2; cin >> s1 >> s2;
52     ll n = s1.size(), m = s2.size();
53
54     if (n < m) {
55         cout << 0 << '\n';
56         return 0;
57     }
58
59     ll cnt = 0;

```

```

54 strHash sh1(s1), sh2(s2);
55 pll tarHash = sh2.hash(0, m - 1);
56 for (ll i = 0; i < n - m + 1; ++i) {
57     if (sh1.hash(i, i + m - 1) ==
58         tarHash) cnt += 1;
59 }
60 cout << cnt << '\n';
61 return 0;
62 }

```

9.2 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     }

```

9.3 LPS

```

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

```

9.4 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     vector<Node> trie;
12
13 public:
14     Trie() {

```

```

15         trie.emplace_back();
16     }
17
18     void insert(const string& word) {
19         int cur = 0;
20         for (char c : word) {
21             int idx = c - 'a';
22             if (trie[cur].children[idx] ==
23                 -1) {
24                 trie[cur].children[idx] =
25                     trie.size();
26                 trie.emplace_back();
27             }
28             cur = trie[cur].children[idx];
29         }
30         trie[cur].endofWord = true;
31     }
32
33     bool search(const string& word) {
34         int cur = 0;
35         for (char c : word) {
36             int idx = c - 'a';
37             if (trie[cur].children[idx] ==
38                 -1) return false;
39             cur = trie[cur].children[idx];
40         }
41         return trie[cur].endofWord;
42     }
43
44     bool startsWith(const string& prefix) {
45         int cur = 0;
46         for (char c : prefix) {
47             int idx = c - 'a';
48             if (trie[cur].children[idx] ==
49                 -1) return false;
50             cur = trie[cur].children[idx];
51         }
52         return true;
53     }
54
55     void deleteWord(const string& word) {
56         int cur = 0;
57         for (char c : word) {
58             int idx = c - 'a';
59             if (trie[cur].children[idx] ==
60                 -1) return;
61             cur = trie[cur].children[idx];
62         }
63         trie[cur].endofWord = false;
64     }
65
66     void print(int node, string prefix)
67     const {
68         if (trie[node].endofWord) {
69             cout << prefix << "\n";
70         }
71         for (int i = 0; i < 26; i++) {
72             int nxt = trie[node].children[i];
73             if (nxt != -1) {
74                 print(nxt, prefix + char('a'
75                     + i));
76             }
77         }
78     }
79 }

```

```

73 void print() const { print(0, ""); }
74 };
75
76 int main() {
77     Trie trie;
78
79     trie.insert("geek");
80     trie.insert("geeks");
81     trie.insert("code");
82     trie.insert("coder");
83     trie.insert("coding");
84
85     cout << "Trie contents:\n";
86     trie.print();
87
88     cout << "\nSearch results:\n";
89     cout << "geek: " << trie.search("geek")
90         << "\n";
91     cout << "geeks: " << trie.search("geeks")
92         << "\n";
93     cout << "code: " << trie.search("code")
94         << "\n";
95     cout << "coder: " << trie.search("coder")
96         << "\n";
97     cout << "coding: " << trie.search("coding")
98         << "\n";
99     cout << "codex: " << trie.search("codex")
100         << "\n";
101
102     cout << "\nPrefix results:\n";
103     cout << "ge: " << trie.startsWith("ge")
104         << "\n";
105     cout << "cod: " << trie.startsWith("cod")
106         << "\n";
107     cout << "coz: " << trie.startsWith("coz")
108         << "\n";
109
110     trie.deleteWord("coding");
111     trie.deleteWord("geek");
112
113     cout << "\nTrie contents after deletions
114         :\n";
115     trie.print();
116
117     cout << "\nSearch results after
118         deletions:\n";
119     cout << "coding: " << trie.search("coding")
120         << "\n";
121     cout << "geek: " << trie.search("geek")
122         << "\n";
123
124     return 0;
125 }

```

9.5 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>

```



```

6 using namespace std;
7 using ll = long long;
8
9 vector<ll> z_value(const string& s) {
10     ll n = s.size();
11     vector<ll> z(n);
12     ll l = 0, r = 0;
13     for (ll i = 1; i < n; ++i) {
14         if (i <= r) z[i] = min(z[i - l], r -
15             i + 1);
16         while (i + z[i] < n && s[z[i]] == s[
17             i + z[i]]) z[i] += 1;
18         if (i + z[i] - 1 > r) {
19             l = i;
20             r = i + z[i] - 1;
21         }
22     }
23     return z;
24 }
25
26 int main() {
27     ios::sync_with_stdio(false);
28     cin.tie(nullptr);
29
30     string s1, s2; cin >> s1 >> s2;
31     ll n = s1.size(), m = s2.size();
32
33     ll cnt = 0;
34     string s = s2 + "$" + s1;
35     vector<ll> z = z_value(s);
36     for (ll i = m; i < s.size(); ++i) {
37         if (z[i] == m) cnt += 1;
38     }
39     cout << cnt << '\n';
40     return 0;
41 }
42 // =====
43 // =====
44 // CSES: Finding Borders
45 // A border of a string is a prefix that is
46 // also a suffix of
47 // the string but not the whole string. For
48 // example,
49 // the borders of abcababcbab are ab and
50 // abcab.
51 // Your task is to find all border lengths
52 // of a given string.
53
54 #include <bits/stdc++.h>
55 using namespace std;
56 using ll = long long;
57 #define pb push_back
58
59 vector<ll> z_value(const string& s) {
60     ll n = s.size();
61     vector<ll> z(n);
62     ll l = 0, r = 0;
63     for (ll i = 1; i < n; ++i) {
64         if (i <= r) z[i] = min(z[i - l], r -
65             i + 1);
66         while (i + z[i] < n && s[z[i]] == s[
67             i + z[i]]) z[i] += 1;
68         if (i + z[i] - 1 > r) {
69             l = i;
70             r = i + z[i] - 1;
71         }
72     }
73     return z;
74 }
75
76 int main() {
77     ios::sync_with_stdio(false);
78     cin.tie(nullptr);
79
80     string s; cin >> s;
81
82     ll l = 0, r = 0;
83     for (ll i = 1; i < n; ++i) {
84         if (i <= r) z[i] = min(z[i - l], r -
85             i + 1);
86         while (i + z[i] < n && s[z[i]] == s[
87             i + z[i]]) z[i] += 1;
88         if (i + z[i] - 1 > r) {
89             l = i;
90             r = i + z[i] - 1;
91         }
92     }
93     return z;
94 }

```

```

64         l = i;
65         r = i + z[i] - 1;
66     }
67     return z;
68 }
69
70 int main() {
71     ios::sync_with_stdio(false);
72     cin.tie(nullptr);
73
74     string s; cin >> s;
75     ll n = s.size();
76
77     vector<ll> z = z_value(s);
78     for (ll i = 1; i < n; ++i) {
79         if (i + z[i] == n) res.pb(z[i]);
80     }
81     sort(res.begin(), res.end());
82     for (ll x : res) {
83         cout << x << ' ';
84     } cout << '\n';
85     return 0;
86 }
87
88 // =====
89 // =====
90 // CSES: Finding Periods
91 // A period of a string is a prefix that can
92 // be used to generate
93 // the whole string by repeating the prefix.
94 // The last repetition
95 // may be partial. For example, the periods
96 // of abcbabca are abc, abcbab and abcbabca.
97 // Your task is to find all period lengths
98 // of a string.
99
100 #include <bits/stdc++.h>
101 using namespace std;
102 using ll = long long;
103 #define pb push_back
104
105 vector<ll> z_value(const string& s) {
106     ll n = s.size();
107     vector<ll> z(n);
108     ll l = 0, r = 0;
109     for (ll i = 1; i < n; ++i) {
110         if (i <= r) z[i] = min(z[i - l], r -
111             i + 1);
112         while (i + z[i] < n && s[z[i]] == s[
113             i + z[i]]) z[i] += 1;
114         if (i + z[i] - 1 > r) {
115             l = i;
116             r = i + z[i] - 1;
117         }
118     }
119     return z;
120 }
121
122 int main() {
123     ios::sync_with_stdio(false);
124     cin.tie(nullptr);
125
126     string s; cin >> s;
127
128     ll l = 0, r = 0;
129     for (ll i = 1; i < n; ++i) {
130         if (i <= r) z[i] = min(z[i - l], r -
131             i + 1);
132         while (i + z[i] < n && s[z[i]] == s[
133             i + z[i]]) z[i] += 1;
134         if (i + z[i] - 1 > r) {
135             l = i;
136             r = i + z[i] - 1;
137         }
138     }
139     return z;
140 }

```

```

124     ll n = s.size();
125
126     vector<ll> z = z_value(s);
127     for (ll i = 1; i < n; ++i) {
128         if (i + z[i] == n) res.pb(i);
129     }
130     sort(res.begin(), res.end());
131     for (ll x : res) {
132         cout << x << ' ';
133     } cout << '\n';
134     return 0;
135 }
136
137 // =====
138 // =====
139 // CSES: Longest Palindrome
140 // Given a string, your task is to determine
141 // the longest
142 // palindromic substring of the string. For
143 // example,
144 // the longest palindrome in aybabbu is bab.
145
146 #include <bits/stdc++.h>
147 using namespace std;
148 using ll = long long;
149
150 int main() {
151     ios::sync_with_stdio(false);
152     cin.tie(nullptr);
153
154     string t, s = "^#"; cin >> t;
155     ll n = t.size(), m = 2*n + 3;
156     for (ll i = 0; i < n; ++i) {
157         s += t[i];
158         s += (i == n - 1 ? "$" : "#");
159     }
160
161     ll c = 1;
162     vector<ll> p(m); p[1] = 0;
163     for (ll i = 2; i <= m - 3; ++i) {
164         if (i < c + p[c]) p[i] = min(p[c -
165             (i - c)], (c + p[c]) - i);
166         while (i + p[i] + 1 < m &&
167             i - p[i] - 1 >= 0 &&
168             s[i - p[i] - 1] == s[i + p[i]
169                 + 1]) p[i] += 1;
170         if (i + p[i] > c + p[c]) c = i;
171     }
172
173     ll j = 2;
174     for (ll i = 3; i <= m - 3; ++i) {
175         if (p[i] > p[j]) j = i;
176     }
177
178     for (ll i = j - p[j] + 1; i <= j + p[j]
179         - 1; i += 2) {
180         cout << s[i];
181     } cout << '\n';
182     return 0;
183 }

```

9.6 Manacher

```

43 // =====
44 // =====
45 // CSES: ALL Palindromes
46 // Given a string, calculate for each
47 // position the length
48 // of the longest palindrome that ends at
49 // that position.
50
51 #include <bits/stdc++.h>
52 using namespace std;
53 using ll = long long;
54
55 ll n, m;
56 string ori_s, s;
57 vector<ll> p, rt, dp;
58
59 int main() {
60     ios::sync_with_stdio(false);
61     cin.tie(nullptr);
62
63     cin >> ori_s;
64     n = ori_s.size();
65     m = 2*n + 3;
66     s = "^#";
67     for (ll i = 0; i < n; ++i) {
68         s += ori_s[i];
69         s += (i == n - 1 ? "$" : "#");
70     }
71
72     ll c = 0;
73     p.resize(m);
74     for (ll i = 2; i <= m - 3; ++i) {
75         if (i < c + p[c]) p[i] = min(p[c -
76             (i - c)], (c + p[c]) - i);
77         while (i - p[i] - 1 >= 0 &&
78             i + p[i] + 1 < m &&
79             s[i - p[i] - 1] == s[i + p[i]
80                 + 1]) p[i] += 1;
81         if (i + p[i] > c + p[c]) c = i;
82     }
83
84     rt.resize(n, 1);
85     for (ll i = 2; i <= m - 3; ++i) {
86         ll y = ((i + p[i] - 1) - 2)/2;
87         if (s[i] == '#') {
88             ll x = ((i + 1) - 2)/2;
89             rt[y] = max(rt[y], (y - x + 1)
90                 *2);
91         } else {
92             ll x = (i - 2)/2;
93             rt[y] = max(rt[y], (y - x)*2 +
94                 1);
95         }
96     }
97
98     dp.resize(n); dp[n - 1] = rt[n - 1];
99     for (ll i = n - 2; i >= 0; --i) {
100         dp[i] = max(rt[i], dp[i + 1] - 2);
101     }
102
103     for (ll x : dp) {
104         cout << x << ' ';
105     } cout << '\n';
106     return 0;
107 }

```

10 Tree

10.1 Binary Lifting

```

103| }

10 Tree

10.1 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[0][u];
19
20     for (ll h = 1; h <= lgn; ++h) {
21         for (ll u = 1; u <= n; ++u) {
22             ll nt = blf[h - 1][u];
23             blf[h][u] = nt == -1 ? -1 : blf[h - 1][nt];
24         }
25     }
26 }
27 ll query(ll u, ll step) {
28     ll cur = u;
29     for (ll i = 30; i >= 0; --i) {
30         if (isOnBit(step, i)) {
31             cur = blf[i][cur];
32             if (cur == -1) return -1;
33         }
34     }
35     return cur;
36 }
37
38 int main() {
39     ios::sync_with_stdio(false);
40     cin.tie(nullptr);
41
42     cin >> n >> q;
43     lgn = flg(n);
44     blf.resize(lgn + 1, vector<ll>(n + 1));
45     init();
46
47     while (q--) {
48         ll u, step; cin >> u >> step;
49         cout << query(u, step) << '\n';
50     }
51
52     return 0;
53 }

```

10.2 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[h - 1][nt];
32         }
33     }
34 }
35 ll query(ll u, ll step) {
36     ll cur = u;
37     for (ll i = 30; i >= 0; --i) {
38         if (isOnBit(step, i)) {
39             cur = blf[i][cur];
40             if (cur == -1) return -1;
41         }
42     }
43     return cur;
44 }
45
46 void dfs(ll u, ll dn) {
47     d[u] = dn;
48     for (ll v : adj[u]) {
49         dfs(v, dn + 1);
50     }
51 }
52
53 ll lca(ll u, ll v) {
54     if (d[u] > d[v]) swap(u, v);
55     if (d[u] < d[v]) v = query(v, d[v] - d[u]);
56     if (u == v) return u;
57     for (ll h = lgn; h >= 0; --h) {
58         ll ntu = blf[h][u];
59         ll ntv = blf[h][v];
60         if (ntu == -1 or ntv == -1 or ntu ==

```

```

61         u = ntu;
62         v = ntv;
63     }
64     return blf[0][u];
65 }
66
67 int main() {
68     ios::sync_with_stdio(false);
69     cin.tie(nullptr);
70
71     cin >> n >> q;
72     lgn = flg(n);
73     d.resize(n + 1);
74     blf.resize(lgn + 1, vector<ll>(n + 1));
75     adj.resize(n + 1);
76     init();
77     dfs(1, 0);
78
79     while (q--) {
80         ll u, v; cin >> u >> v;
81         cout << lca(u, v) << '\n';
82     }
83
84     return 0;
85 }

```


ACM ICPC

Team Reference -

c0mpile_error

Contents

1	Bitwise Trick	1	2.1	Digit DP	1	5.3	Line Segment Intersection Test	3	7.3	Matrix	5
	1.1 tricks	1	2.2	Subset DP	1	6	Graph	3	8	Others	5
2	DP	1	3	D&C	1	6.1	Kosaraju	3	8.1	GCC Builtin Functions	5
			3.1	MergeSort Finds the Number of Inversions	1	6.2	AP	4	9	String	5
			4	Data Structure	2	6.3	Dijkstra	4	9.1	String Hashing	5
			4.1	DSU	2	6.4	MST Prim	4	9.2	KMP	6
			4.2	Segment Tree	2	6.5	MST Kruskal	4	9.3	LPS	6
			4.3	BIT	2	6.6	Bridge	4	9.4	Trie	6
			4.4	Sparse Table	3	6.7	Bellman Ford Detects Negative Cycle	4	9.5	Z-value	6
			5	Geometry	3	7	Math	4	9.6	Manacher	7
			5.1	Convex Hull	3	7.1	Big Integer Addition and Multiplication	4	10	Tree	8
			5.2	Vector	3	7.2	Modular Inverse	5	10.1	Binary Lifting	8
									10.2	LCA	8

ACM ICPC Judge Test - c0mpile_error

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
10    size_t block_size, bound;
11    void stack_size_dfs(size_t depth = 1) {

```

```

12        if (depth >= bound)
13            return;
14        int8_t ptr[block_size]; // 若無法編譯將
15            block_size 改成常數
16        memset(ptr, 'a', block_size);
17        cout << depth << endl;
18        stack_size_dfs(depth + 1);
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    double speed(int iter_num) {
27        const int block_size = 1024;
28        volatile int A[block_size];
29        auto begin = chrono::high_resolution_clock
30            ::now();
31        while (iter_num--)
32            for (int j = 0; j < block_size; ++j)
33                A[j] += j;
34        auto end = chrono::high_resolution_clock::
35            now();
36        chrono::duration<double> diff = end -
37            begin;

```

```

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

```

```

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

```