1 DP

1.1 Digit DP

```
string num;

ll dp[the max len of num + 1][2][2][...];
memset(dp, -1, sizeof(dp));

ll count(ll pos, bool tight, bool
     leadingZero, ...) {
    if (dp[pos][tight][leadingZero][...] !=
         -1) {
        return dp[pos][tight][leadingZero
            ][...];
    }

    if (base-case) {  // e.g. pos == num.
        size() or other base case condition
      // do something
    }

    ll res = 0;
    ll up = (tight ? num[pos] - '0' : 9);
    for (ll d = 0; d <= up; ++d) {
        res += count(
            pos + 1,
            tight and (d == num[pos] - '0'),
            leadingZero and (d == 0),
            ...
        );
    }

    return dp[pos][tight][leadingZero][...]
         = res;
}

count(0, true, true, ...)  // the answer

// ============================
// ============================

// AtCoder ABC154E
// Find the number of integers between 1 and
     N (inclusive)
// that contains exactly K non-zero digits
     when written in base ten.
// 1 <= N <= 10^100, 1 <= K <= 3

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

ll k;
string num;

ll dp[102][2][4];

ll count(ll pos, bool tight, ll cnt) {
    if (dp[pos][tight][cnt] != -1) {
        return dp[pos][tight][cnt];
    }

    if (cnt == 0) return dp[pos][tight][cnt]
         = 1;
    else if (pos == num.size()) return dp[
        pos][tight][cnt] = 0;

    ll res = 0;
    ll up = (tight ? num[pos] - '0' : 9);
    for (ll d = 0; d <= up; ++d) {
        res += count(
            pos + 1,
            tight and (d == num[pos] - '0'),
            cnt + (d == 0 ? 0 : -1)
        );
    }

    return dp[pos][tight][cnt] = res;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> num >> k;

    memset(dp, -1, sizeof(dp));

    cout << count(0, true, k) << '\n';

    return 0;
}
```

1.2 Interval DP

```
ll n;  // the length of the array

vector<ll> a(n + 1);

vector<vector<ll>> dp(n + 1, vector<ll>(n +
     1));

for (ll i = 1; i <= n; ++i) {
  dp[i][i] = "initial value";  // length 1
       interval [i, i]
}
for (ll len = 2; len <= n; ++len) {  // loop
     through all length from 2 to n
  for (ll l = 1; l <= n - len + 1; ++l) {
    ll r = l + len - 1;

    dp[l][r] = ...;  // interval [l, r]
  }
}

dp[1][n]  // the answer (interval [1, n])
```

1.3 Knapsack

```
// 0/1 knapsack

const ll INF = 9e18;

ll n, wmx; cin >> n >> wmx;
vector<ll> w(n + 1); for (ll i = 1; i <= n;
     ++i) cin >> w[i];
vector<ll> v(n + 1); for (ll i = 1; i <= n;
     ++i) cin >> v[i];

vector<ll> dp(wmx + 1, -INF); dp[0] = 0;
for (ll i = 1; i <= n; ++i) {
  for (ll j = wmx; j >= w[i]; --j) {
    dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
  }
}

ll mx = dp[0];
for (ll j = 1; j <= wmx; ++j) {
  mx = max(mx, dp[j]);
}
cout << mx << '\n';

// ------------------------------

// Unbounded
// just change the inner for loop to the
     following line
// for (ll j = w[i]; j <= wmx; ++j) ...
```

1.4 Subset DP

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

1.5 LIS

```
ll LISLength(const vector<ll>& arr) {
    vector<ll> d;
    for (ll x : arr) {
        auto it = lower_bound(d.begin(), d.
             end(), x);
        if (it == d.end()) d.push_back(x);
        else *it = x;
    }
    return d.size();
}

// find LIS sequence
for (ll i = 0; i < num.size(); ++i){
    if(lis.empty() || lis.back() < num[i]){
        lis.push_back(num[i]);
        dp[i] = lis.size();
    }
    else{
        auto iter = lower_bound(all(lis),num
             [i]);
        dp[i] = iter - lis.begin() + 1;
        *iter = num[i];
    }
}
ll length = lis.size();
for (ll i = num.size() - 1; i>=0; --i){
    if(dp[i] == length){
        ans.push_back(num[i]);
        length -= 1;
    }
}
```

1.6 LCS

```
ll n; cin >> n;
string s; cin >> s; s = " " + s;
ll m; cin >> m;
string t; cin >> t; t = " " + t;

vector<vector<ll>> dp(n + 1, vector<ll>(m +
     1, 0));
for (ll i = 1; i <= n; ++i) {
  for (ll j = 1; j <= m; ++j) {
    if (s[i] == t[j]) {
      dp[i][j] = dp[i - 1][j - 1] + 1;
    }
    else {
      dp[i][j] = max(dp[i - 1][j], dp[i][j -
           1]);
    }
  }
}

cout << dp[n][m] << '\n';  // the length of
     LCS
```

2 D&C

2.1 MergeSort Finds the Number of Inversions

```
void merge(vector<int>& arr, ll l, ll r, ll
     x, ll y) {
  ll left = l;
  vector<ll> tmp; tmp.reserve(y - l + 1);
  while (l <= r and x <= y) {
    if (arr[l] <= arr[x]) {
      tmp.push_back(arr[l++]);
    }
    else {
      tmp.push_back(arr[x++]);
    }
  }
  while (l <= r) {
```

```
    tmp.push_back(arr[l++]);
  }
  while (x <= y) {
    tmp.push_back(arr[x++]);
  }
  for (ll i = left; i <= y; ++i) {
    arr[i] = tmp[i - left];
  }
}
ll mergeSort(vector<ll>& arr, ll l, ll r) {
    if (l == r) return 0;
    ll mid = l + (r - l)/2;
    ll lcnt = mergeSort(arr, l, mid);
    ll rcnt = mergeSort(arr, mid + 1, r);

  // ---------- main logic ----------
    ll cnt = lcnt + rcnt;
    ll a = l, b = mid, c = mid + 1, d = r;
        // c is the current checking position
    while (a <= b) {
        while (c <= d and arr[a] > arr[c]) c += 1;
        cnt += c - (mid + 1);
        a += 1;
    }
    // --------------------------------

    merge(arr, l, mid, mid + 1, r);

    return cnt;
}
```

3 Data Structure

3.1 DSU

```
ll cc;
vector<ll> djs, sz;
ll find(ll u) {
    if (u == djs[u]) return u;
    return djs[u] = find(djs[u]);
}
void join(ll u, ll v) {
    u = find(u);
    v = find(v);
    if (u == v) return;  // don't forgot this line
    if (sz[u] < sz[v]) swap(u, v);
    djs[v] = u;
    sz[u] += sz[v];
    cc -= 1;
}
void init(ll n) {
    djs.clear(); djs.resize(n + 1);
    for (ll i = 1; i <= n; ++i) djs[i] = i;
    sz.clear(); sz.resize(n + 1, 1);
    cc = n;
}
```

3.2 Segment Tree

```
// CSES Range Updates and Sums
// 1. Increase each value in range [a,b] by x.
// 2. Set each value in range [a,b] to x.
// 3. Calculate the sum of values in range [a,b].

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

#define lson 2*n + 1
#define rson 2*n + 2

class Node {
public:
    ll val;
    ll setVal;
    ll addVal;
    bool isSet;
};

ll sz, q;
vector<ll> a;
vector<Node> nds;

void build(ll l, ll r, ll n = 0) {
    if (l == r) {
        nds[n].val = a[l];
        nds[n].setVal = 0;
        nds[n].addVal = 0;
        nds[n].isSet = false;
        return;
    }
    ll mid = l + (r - l)/2;
    build(l, mid, lson);
    build(mid + 1, r, rson);
    nds[n].val = nds[lson].val + nds[rson].val;
}

void push(ll l, ll r, ll n) {
    ll mid = l + (r - l)/2;

    if (nds[n].isSet) {

        nds[lson].val = nds[n].setVal*(mid - l + 1);
        nds[lson].setVal = nds[n].setVal;
        nds[lson].addVal = 0;
        nds[lson].isSet = true;

        nds[rson].val = nds[n].setVal*(r - (mid + 1) + 1);
        nds[rson].setVal = nds[n].setVal;
        nds[rson].addVal = 0;
        nds[rson].isSet = true;

        nds[n].isSet = false;
    }

    nds[lson].addVal += nds[n].addVal;

    nds[rson].val += nds[n].addVal*(r - (mid + 1) + 1);
    nds[rson].addVal += nds[n].addVal;

    nds[n].addVal = 0;
}

void setVal(ll x, ll y, ll val, ll l, ll r, ll n = 0) {
    if (l == x and r == y) {
        nds[n].val = val*(y - x + 1);
        nds[n].setVal = val;
        nds[n].addVal = 0;
        nds[n].isSet = true;
        return;
    }
    push(l, r, n);
    ll mid = l + (r - l)/2;
    if (y <= mid) {
        setVal(x, y, val, l, mid, lson);
    } else if (x >= mid + 1) {
        setVal(x, y, val, mid + 1, r, rson);
    } else {
        setVal(x, mid, val, l, mid, lson);
        setVal(mid + 1, y, val, mid + 1, r, rson);
    }
    nds[n].val = nds[lson].val + nds[rson].val;
}

void addVal(ll x, ll y, ll val, ll l, ll r, ll n = 0) {
    if (l == x and r == y) {
        nds[n].val += val*(y - x + 1);
        nds[n].addVal += val;
        return;
    }
    push(l, r, n);
    ll mid = l + (r - l)/2;
    if (y <= mid) {
        addVal(x, y, val, l, mid, lson);
    } else if (x >= mid + 1) {
        addVal(x, y, val, mid + 1, r, rson);
    } else {
        addVal(x, mid, val, l, mid, lson);
        addVal(mid + 1, y, val, mid + 1, r, rson);
    }
    nds[n].val = nds[lson].val + nds[rson].val;
}

ll query(ll x, ll y, ll l, ll r, ll n = 0) {
    if (l == x and r == y) return nds[n].val;
    push(l, r, n);
    ll mid = l + (r - l)/2;
    if (y <= mid) {
        return query(x, y, l, mid, lson);
    } else if (x >= mid + 1) {
        return query(x, y, mid + 1, r, rson);
    } else {
        ll a = query(x, mid, l, mid, lson);
        ll b = query(mid + 1, y, mid + 1, r, rson);
        return a + b;
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> sz >> q;

    a.resize(sz + 1);
    for (ll i = 1; i <= sz; ++i) cin >> a[i];

    nds.resize(4*sz);
    build(1, sz);

    while (q--) {
        ll act; cin >> act;
        if (act == 1) {
            ll l, r, val; cin >> l >> r >> val;
            addVal(l, r, val, 1, sz);
        } else if (act == 2) {
            ll l, r, val; cin >> l >> r >> val;
            setVal(l, r, val, 1, sz);
        } else if (act == 3) {
            ll l, r; cin >> l >> r;
            cout << query(l, r, 1, sz) << '\n';
        }
    }

    return 0;
}
```

3.3 BIT

```
inline ll lowbit(ll x) {
    return x & -x;
}

ll n;
vector<ll> a, bit;

void add(ll pos, ll val) {
    for (ll i = pos; i <= n; i += lowbit(i)) {
        bit[i] += val;
    }
}

void init() {
    for (ll i = 1; i <= n; ++i) {
        add(i, a[i]);
    }
}

ll query(ll pos) {  // [1, pos]
```

```
    ll sum = 0;
    for (ll i = pos; i >= 1; i -= lowbit(i))
        {
        sum += bit[i];
    }
    return sum;
}
```

3.4 Sparse Table

```
// if the max size of arr is 200000
vector<ll> arr;

// ------------------------------

// Sparse Table
const ll lgmx = 17;  // floor(log2(200000))
ll rmq[lgmx + 1][200000 + 1];
void init(ll n) {  // O(nlogn)
    for (ll i = 1; i <= n; ++i) rmq[0][i] =
        arr[i];
    for (ll h = 1; h <= lgmx; ++h)
        for (ll i = 1; i + (1 << h) - 1 <= n
            ; ++i)
            rmq[h][i] = min(rmq[h - 1][i],
                rmq[h - 1][i + (1 << (h - 1)
                )]);
}
ll flg(ull x) {return 63 - __builtin_clzll(x
    );}
ll query(ll l, ll r) {  // O(1)
    ll h = flg(r - l + 1);
    return min(rmq[h][l], rmq[h][r - (1 << h
        ) + 1]);
}

// ------------------------------

// initialize the array
ll n; cin >> n;
arr.resize(n + 1);
for (ll i = 1; i <= n; ++i) cin >> arr[i];

// initialize the sparse table
init(n);
```

4 Geometry

4.1 Convex Hull

```
// pts = {p0, p1, ... pn-1}, 0-based
// the points in pts should be distinct
vector<vec> convexHull(const vector<vec> &
    _pts) {
    vector<vec> pts = _pts;
    sort(pts.begin(), pts.end());

    ll n = pts.size();
    vector<vec> hull(1, pts[0]);
    for (ll i = 1; i < n; ++i) {
        while (hull.size() >= 2 &&
               ori(hull[hull.size() - 2],
                   hull.back(), pts[i]) < 0)
                    {
            hull.pop_back();
        }
        hull.push_back(pts[i]);
    }

    ll m = hull.size();
    for (ll i = n - 2; i >= 0; --i) {
        while (hull.size() - m + 1 >= 2 &&
               ori(hull[hull.size() - 2],
                   hull.back(), pts[i]) < 0)
                    {
            hull.pop_back();
        }
        hull.push_back(pts[i]);
    }
    hull.pop_back();

    return hull;
}
```

4.2 Vector

```
class vec {
public:
    ll x, y;
    vec() {}
    vec(ll _x, ll _y) : x(_x), y(_y) {}
    vec operator+(const vec& v) const {
        return vec(this->x + v.x, this->y +
            v.y);
    }
    vec operator-(const vec& v) const {
        return vec(this->x - v.x, this->y -
            v.y);
    }
    ll operator*(const vec& v) const {
        return this->x * v.x + this->y * v.y
            ;
    }
    ll operator^(const vec& v) const {
        return this->x * v.y - this->y * v.x
            ;
    }
    bool operator<(const vec& v) const {
        if (this->x != v.x) return this->x <
            v.x;
        return this->y < v.y;
    }
    vec& operator=(const vec& v) {
        this->x = v.x;
        this->y = v.y;
        return *this;
    }
};

ll sign(ll x) {
    if (x == 0) return 0;
    if (x < 0) return -1;
    return 1;
}

ll ori(const vec& o, const vec& a,const vec&
    b) {
    return sign((a - o) ^ (b - o));
}

bool isCollinear(const vec& a, const vec& b,
    const vec& c) {
    return ori(a, b, c) == 0;
}

bool isOnSeg(const vec& a, const vec& b,
    const vec& p) {
    return isCollinear(a, b, p) && sign((p -
        a) * (p - b)) <= 0;
}
```

4.3 Simple Polygon Area

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

4.4 Line Segment Intersection Test

```
bool isSegInter(const vec& a, const vec& b,
    const vec& c,const vec& d) {
    ll ori1 = ori(a, b, c);
    ll ori2 = ori(a, b, d);
    ll ori3 = ori(c, d, a);
    ll ori4 = ori(c, d, b);
    if (isCollinear(a, b, c) && isCollinear(
        a, b, d)) {
        return isOnSeg(a, b, c) || isOnSeg(a
            , b, d) ||
               isOnSeg(c, d, a) || isOnSeg(c
                   , d, b);
    }
    return ori1 * ori2 <= 0 && ori3 * ori4
        <= 0;
}
```

5 Graph

5.1 Kosaraju

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define pb push_back
#define rep(n) for (ll _ = 1; _ <= n; ++_)

ll V, E;

stack<ll> stk;
vector<bool> vis;
vector<vector<ll>> adj;
void dfs1(ll u) {
    vis[u] = true;
    for (ll v : adj[u]) {
        if (vis[v]) continue;
        dfs1(v);
    }
    stk.push(u);
}

ll sccCnt;
vector<ll> scc;
vector<vector<ll>> radj;
void dfs2(ll u, ll sccIdx) {
    scc[u] = sccIdx;
    for (ll v : radj[u]) {
        if (scc[v] != -1) continue;
        dfs2(v, sccIdx);
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> V >> E;
    vis.resize(V + 1, false);
    adj.resize(V + 1);
    radj.resize(V + 1);
    rep (E) {
        ll x, y; cin >> x >> y;
        adj[x].pb(y);
        radj[y].pb(x);
    }

    for (ll u = 1; u <= V; ++u) {
        if (vis[u]) continue;
        dfs1(u);
    }

    sccCnt = 0;
    scc.resize(V + 1, -1);
    while (not stk.empty()) {
        ll u = stk.top(); stk.pop();
        if (scc[u] != -1) continue;
        dfs2(u, ++sccCnt);
    }

    cout << sccCnt << '\n';
    for (ll i = 1; i <= V; ++i) {
        cout << scc[i] << ' ';
    } cout << '\n';

    return 0;
}
```

5.2 AP

```cpp
void dfs(ll u, ll pa = -1) {
    ll ch = 0;  // 紀錄子節點樹
    low[u] = in[u] = ++t;
    for (ll v : adj[u]) {
        if (v == pa) continue;
        if (in[v] == -1) {
            ch += 1;  // 子節點數加一
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if (pa != -1 and low[v] >= in[u
                ]) "u is AP"  // find AP
        } else if (in[v] < in[u]) {
            low[u] = min(low[u], in[v]);
        }
    }
    if (pa == -1 and ch >= 2) "u is AP" //
        root 額外判斷
}
```

5.3 Dijkstra

```cpp
int main() {
  ios::sync_with_stdio(false);
    cin.tie(nullptr);

  const ll INF = 1e18;

  ll n, m; cin >> n >> m;
  vector<vector<pll>> adj(n + 1);
  rep (m) {
    ll u, v, w; cin >> u >> v >> w;
    adj[u].pb(pll(w, v));
  }

  vector<ll> dis(n + 1, INF);
  priority_queue<pll, vector<pll>, greater<
      pll>> pq;

  dis[1] = 0;
  pq.push(pll(0, 1));
  while (not pq.empty()) {
    auto [uw, u] = pq.top(); pq.pop();

    if (uw > dis[u]) continue;

    for (auto [w, v] : adj[u]) {
      if (dis[u] + w < dis[v]) {
        dis[v] = dis[u] + w;
        pq.push(pll(dis[v], v));
      }
    }
  }
}
```

5.4 Floyed Warshall

```cpp
int main() {
  ios::sync_with_stdio(false);
    cin.tie(nullptr);

  cin >> n >> m;
  mtx.clear(); mtx.resize(n + 1, vector<ll>(
      n + 1, INF));
  for (ll i = 1; i <= n; ++i) mtx[i][i] = 0;
  rep (m) {
    ll x, y, w; cin >> x >> y >> w;
    mtx[x][y] = min(mtx[x][y], w);
    mtx[y][x] = min(mtx[y][x], w);
  }  // "min" is used to prevent the
     multiple edges

  for (ll k = 1; k <= n; ++k) {
    for (ll i = 1; i <= n; ++i) {
      for (ll j = 1; j <= n; ++j) {
        mtx[i][j] = min(mtx[i][j], mtx[i][k]
            + mtx[k][j]);
      }
    }
  }
}
```

5.5 MST Prim

```cpp
int main() {
  ios::sync_with_stdio(false);
    cin.tie(nullptr);

    constexpr ll INF = 1e18;

    ll n, m; cin >> n >> m;
    vector<vector<pll>> adj(n + 1);
    rep (m) {
        ll u, v, w; cin >> u >> v >> w;
        adj[u].pb(pll(w, v));
        adj[v].pb(pll(w, u));
    }

    priority_queue<pll, vector<pll>, greater
        <pll>> pq;
    vector<bool> vis(n + 1, false);
    vector<ll> curw(n + 1, INF);

    pq.push(pll(0, 1));
    curw[1] = 0;

    ll mstCost = 0, mstEdgesCnt = 0;
    while (mstEdgesCnt < n - 1) {
        if (pq.empty()) {
            // the graph is disconnected (
                MST D.N.E.)
            return 0;
        }

        auto [uw, u] = pq.top(); pq.pop();

        if (uw > curw[u]) continue;

        vis[u] = true;
        mstCost += uw;
        mstEdgesCnt += (u == 1 ? 0 : 1);

        for (const auto& [w, v] : adj[u]) {
            if (not vis[v] and w < curw[v])
                {
                pq.push(pll(w, v));
                curw[v] = w;
            }
        }
    }

  return 0;
}
```

5.6 MST Kruskal

```cpp
vector<ll> djs, sz;
ll find(ll u) {
    if (u == djs[u]) return u;
    return djs[u] = find(djs[u]);
}
void join(ll u, ll v) {
    u = find(u);
    v = find(v);
    if (u == v) return;
    if (sz[u] < sz[v]) swap(u, v);
    djs[v] = u;
    sz[u] += sz[v];
}
void init(ll n) {
    djs.clear(); djs.resize(n + 1);
    for (ll i = 1; i <= n; ++i) djs[i] = i;
    sz.clear(); sz.resize(n + 1, 1);
}

class Edge{
public:
    ll u, v, w;
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

  // nodes are 1-indexed
  // edges are 0-indexed

    ll n, m; cin >> n >> m;
    vector<Edge> edges(m);
    for (auto& [u, v, w] : edges) {
        cin >> u >> v >> w;
    }
    sort(all(edges), [](const Edge& e1,
         const Edge e2) -> bool {
        return e1.w < e2.w;
    });
    init(n);

    ll mstCost = 0, mstEdgesCnt = 0;
    for (const auto& [u, v, w] : edges) {
        if (find(u) == find(v)) continue;
        join(u, v);
        mstCost += w;
        mstEdgesCnt += 1;
        if (mstEdgesCnt == n - 1) break;
    }

    // if (mstEdgesCnt < n - 1) // the graph
        is disconnected (MST D.N.E.)

    return 0;
}
```

5.7 Topo-sort DFS

```cpp
ll v;
vector<ll> topo;
vector<bool> vis;
vector<vector<ll>> adj(v + 1);
void dfs(ll node) {
  vis[node] = true;
  for (ll neighbor : adj[node]) {
    if (vis[neighbor]) continue;
    dfs(neighbor);
  }
  topo.pb(node);
}

reverse(topo.begin(), topo.end());
```

5.8 Topo-sort Kahn

```cpp
int main() {
  ios::sync_with_stdio(false);
    cin.tie(nullptr);

  ll n, m; cin >> n >> m;
  vector<ll> indeg(n + 1, 0);
  vector<vector<ll>> adj(n + 1);
  rep (m) {
    ll u, v; cin >> u >> v;
    adj[u].pb(v);
    indeg[v] += 1;
  }

  vector<ll> res; res.reserve(n);
  queue<ll> q;  // you can use any pool data
      structure here
  for (ll u = 1; u <= n; ++u) {
    if (indeg[u] == 0) {
      q.push(u);
      res.pb(u);
    }
  }
  while (not q.empty()) {
    ll u = q.front(); q.pop();
    for (ll v : adj[u]) {
      indeg[v] -= 1;
      if (indeg[v] == 0) {
        q.push(v);
        res.pb(v);
      }
    }
```

```
  }

  // if (res.size() < n) // there exists a directed cycle

  return 0;
}
```

5.9 Bridge

```
void dfs(ll u, ll pa = -1) {
    low[u] = in[u] = ++t;
    for (ll v : adj[u]) {
        if (v == pa) continue;
        if (in[v] == -1) {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > in[u]) "edge (u, v) is bridge"  // find bridge
        } else if (in[v] < in[u]) {
            low[u] = min(low[u], in[v]);
        }
    }
}
```

5.10 Bellman Ford Detects Negative Cycle

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using pll = pair<ll, ll>;
#define ff first
#define ss second
#define pb push_back
#define rep(n) for (ll _ = 1; _ <= n; ++_)

int main() {
  ios::sync_with_stdio(false);
    cin.tie(nullptr);

  const ll INF = 5e12 + 1000;

  ll n, m; cin >> n >> m;
  vector<ll> dis(n + 1, INF);
  vector<vector<pll>> adj(n + 1);
  rep (m) {
    ll u, v, w; cin >> u >> v >> w;
    adj[u].pb(pll(w, v));
  }
  for (ll i = 1; i <= n; ++i) {
    adj[0].pb(pll(0, i));
  }

  dis[0] = 0;
  rep (n - 1) {
    bool updated = false;
    for (ll u = 0; u <= n; ++u) {
      for (const auto& [w, v] : adj[u]) {
        if (dis[u] < INF and dis[u] + w < dis[v]) {
          dis[v] = dis[u] + w;
          updated = true;
        }
      }
    }
    if (not updated) break;
  }

  bool hasNegativeCycle = false;
  for (ll u = 0; not hasNegativeCycle and u <= n; ++u) {
    for (const auto& [w, v] : adj[u]) {
      if (dis[u] < INF and dis[u] + w < dis[v]) {
        hasNegativeCycle = true;
        break;
      }
    }
  }

  return 0;
}
```

6 Math

6.1 Big Integer Addition and Multiplication

```
vector<int> strToVec(string str, int sz) {
    vector<int> r(sz, 0);
    int strLength = str.length();
    for (int i = strLength - 1, idx = 0; i >= 0; --i, ++idx) {
        r[idx] = str[i] - '0';
    }
    return r;
}
// for example:
// strToVec("677", 4) -> 7 7 6 0
// strToVec("8829", 4) -> 9 2 8 8

// addition
string add(string x, string y) {
    ll n = max(x.length(), y.length());
    vector<ll> xdigit = strToVec(x, n + 1);
    vector<ll> ydigit = strToVec(y, n + 1);
    vector<ll> result(n + 1, 0);
    ll carry = 0;
    for (ll i = 0; i < n + 1; ++i) {
        result[i] = xdigit[i] + ydigit[i] + carry;
        if (result[i] >= 10) {
            result[i] %= 10;
            carry = 1;
        }
        else carry = 0;
    }
    ll start;
    for (ll i = n; i >= 0; --i) {
        if (result[i] != 0) {
            start = i;
            break;
        }
    }
    string r = "";
    for (ll i = start; i >= 0; --i) {
        r += result[i] + '0';
    }
    return r;
}

// multiplication
string product(string x, string y) {
    ll xlength = x.length();
    ll ylength = y.length();
    ll n = max(xlength, ylength);
    vector<ll> xdigit = strToVec(x, xlength);
    vector<ll> ydigit = strToVec(y, ylength);
    vector<ll> result(2*n, 0);
    for (ll i = 0; i < xlength; ++i) {
        for (ll j = 0; j < ylength; ++j) {
            result[i + j] += xdigit[i]*ydigit[j];
            if (result[i + j] >= 10) {
                result[i + j + 1] += result[i + j]/10;
                result[i + j] %= 10;
            }
        }
    }
    ll start;
    for (ll i = 2*n - 1; i >= 0; --i) {
        if (result[i] != 0) {
            start = i;
            break;
        }
    }
    string r = "";
    for (ll i = start; i >= 0; --i) {
        r += result[i] + '0';
    }
    return r;
}
```

6.2 Modular Inverse

```
// extend Euclidean
pll extgcd(ll a, ll b) {
    if (b == 0) return pll(1, 0);
    pll p = extgcd(m, a%b);
    ll x = p.ff, y = p.ss;
    return pll(y, x - y*(a/b));
}
extgcd(a, MOD).ff  // the modular inverse
(extgcd(a, MOD).ff%MOD + MOD)%MOD  // if you want to ensure that it is non-negative

// fast exponentiation (make sure that MOD is a prime number)
fastPow(a, MOD - 2)  // the modular inverse
```

6.3 Fast Exponentiation

```
ll fastPow(ll b, ll e) {
  b = b%MOD;
  ll res = 1;
  while (e > 0) {
    if (e%2 == 1) res = (res*b)%MOD;
    b = (b*b)%MOD;
    e /= 2;
  }
  return res;
}
```

6.4 Matrix

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

struct Matrix {
    int n, m;                   // n 行, m 列
    vector<vector<long long>> a;
    static const long long MOD = 1e9+7; // 如果題目需要取模，可以改這裡

    Matrix(int n, int m, bool ident = false)
        : n(n), m(m) {
        a.assign(n, vector<long long>(m, 0));
        if(ident) { // 單位矩陣
            for(int i=0; i<min(n,m); i++) a[i][i] = 1;
        }
    }

    // 輸出矩陣
    void print() const {
        for(int i=0; i<n; i++) {
            for(int j=0; j<m; j++) cout << a[i][j] << " ";
            cout << "\n";
        }
    }

    // 矩陣乘法
    Matrix operator*(const Matrix& o) const {
        assert(m == o.n);
        Matrix res(n, o.m);
        for(int i=0; i<n; i++) {
            for(int k=0; k<m; k++) if(a[i][k]) {
                for(int j=0; j<o.m; j++) {
                    res.a[i][j] = (res.a[i][j] + a[i][k] * o.a[k][j]) % MOD;
                }
            }
```

```
        }
        return res;
    }

    // 矩陣加法
    Matrix operator+(const Matrix& o) const {
        assert(n == o.n && m == o.m);
        Matrix res(n, m);
        for(int i=0; i<n; i++) {
            for(int j=0; j<m; j++) {
                res.a[i][j] = (a[i][j] + o.a[i][j]) % MOD;
            }
        }
        return res;
    }

    // 矩陣快速冪 (n×n 方陣才能做)
    Matrix pow(long long exp) const {
        assert(n == m);
        Matrix res(n, n, true), base = *this;
        while(exp > 0) {
            if(exp & 1) res = res * base;
            base = base * base;
            exp >>= 1;
        }
        return res;
    }
};
```

6.5 Prime Sieve

```
// Sieve of Eratosthenes
ll MAXN;
vector<ll> spf(MAXN + 1, -1);
spf[0] = -2; spf[1] = -2;
for (ll i = 2; i*i <= MAXN; ++i) {
    if (spf[i] == -1) {
        for (ll j = i*i; j <= MAXN; j += i) {
            if (spf[j] == -1) {
                spf[j] = i;
            }
        }
    }
}
```

6.6 Combination

```
ll MAXN;

vector<ll> fact;  // factorial
vector<ll> invs;  // modular inverse
void init() {
    fact.resize(MAXN + 1, 1);
    invs.resize(MAXN + 1, 1);
    for (ll i = 2; i <= MAXN; ++i) {
        fact[i] = (fact[i - 1]*i)%MOD;
        invs[i] = fastPow(fact[i], MOD - 2);
    }
}

inline ll C(ll n, ll k) {
  return fact[n]*invs[k]%MOD*invs[n - k]%MOD
}
```

7 Others

7.1 GCC Builtin Functions

```
// count the number of 1 bit in x
__builtin_popcount(unsigned int x)
__builtin_popcountll(unsigned long long x)

// count leading zero of x
__builtin_clz(unsigned int x)
__builtin_clzll(unsigned long long x)

// count trailing zero of x
__builtin_ctz(unsigned int x)
__builtin_ctzll(unsigned long long x)
```

7.2 mt19937 Usage

```
#include <chrono>
#include <random>
using namespace std;
using ll = long long;

// mt19937 is a random number generator
// chrono::steady_clock::now().time_since_epoch().count() is used as a seed
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

// dist describes a range of random numbers
uniform_int_distribution<ll> dist(lb, ub);

dist(rng) // use rng to generate a random integer in [lb, ub]
```

7.3 Custom Hash

```
#include <chrono>

// ============================
// ============================

class custom_hash {
private:
    static uint64_t splitmix64(uint64_t x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }
public:
    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = std::chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};
// for example:
// unordered_map<ll, ll, custom_hash> mp;

// ============================
// ============================

class custom_hash {
private:
    static const uint64_t FIXED_RANDOM;
    static uint64_t splitmix64(uint64_t x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }
public:
    size_t operator()(uint64_t x) const {
        return splitmix64(x + FIXED_RANDOM);
    }
    size_t operator()(pair<uint64_t, uint64_t> p) const {
        uint64_t h1 = splitmix64(p.first + FIXED_RANDOM);
        uint64_t h2 = splitmix64(p.second + FIXED_RANDOM + 1);
        return h1 ^ (h2*47);
    }
};
const uint64_t custom_hash::FIXED_RANDOM = std::chrono::steady_clock::now().time_since_epoch().count();
// for example:
// unordered_map<pll, ll, custom_hash> mp;
```

8 String

8.1 String Hashing

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using pll = pair<ll, ll>;
#define pb push_back

class strHash {
private:
    const ll m1 = 1e9 + 7, m2 = 1e9 + 9, p = 239017;
    ll n;
    string s;
    vector<ll> h1, h2, p1, p2;
public:
    strHash(const string& _s) {
        n = _s.size();
        s = _s;

        h1.resize(n); h1[0] = s[0];
        h2.resize(n); h2[0] = s[0];
        for (ll i = 1; i < n; ++i) {
            h1[i] = (h1[i - 1]*p%m1 + s[i])%m1;
            h2[i] = (h2[i - 1]*p%m2 + s[i])%m2;
        }

        p1.resize(n); p1[0] = 1;
        p2.resize(n); p2[0] = 1;
        for (ll i = 1; i < n; ++i) {
            p1[i] = p1[i - 1]*p%m1;
            p2[i] = p2[i - 1]*p%m2;
        }
    }
    pll hash(ll l, ll r) const {  // [l, r]
        if (l == 0) return pll(h1[r], h2[r]);
        return pll(
            ((h1[r] - h1[l - 1]*p1[r - l + 1]%m1)%m1 + m1)%m1,
            ((h2[r] - h2[l - 1]*p2[r - l + 1]%m2)%m2 + m2)%m2
        );
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s1, s2; cin >> s1 >> s2;
    ll n = s1.size(), m = s2.size();

    if (n < m) {
        cout << 0 << '\n';
        return 0;
    }

    ll cnt = 0;
    strHash sh1(s1), sh2(s2);
    pll tarHash = sh2.hash(0, m - 1);
    for (ll i = 0; i < n - m + 1; ++i) {
        if (sh1.hash(i, i + m - 1) == tarHash) cnt += 1;
    }
    cout << cnt << '\n';

    return 0;
}
```

8.2 KMP

```
string a; // 文本串
string b; // 模板串 (將被匹配的字串)
int kmp_next[N]; // next數組

void getNext(int m = b.size()){  // 初始化
  int j = 0;
  kmp_next[0] = 0;
  for(int i = 1; i < m; ++i){
    while(j > 0 && b[i] != b[j]) j =
        kmp_next[j-1];
    if(b[i] == b[j]) ++j;
    kmp_next[i] = j;
  }
}

int kmp(int n = a.size(), int m = b.size()){
      // 使用KMP尋找匹配位置
  int i, j = 0;
  int p = -1;
  getNext(m);
  for(i = 0; i < n; ++i){
    while(j > 0 && b[j] != a[i]) j =
        kmp_next[j-1];
    if(b[j] == a[i]) ++j;
    if(j == m){
      p = i - m + 1;
      break;
    }
  }
  return p;
}

int kmp(int n = a.size(), int m = b.size()){
      // 使用KMP計算匹配次數
  int i, j = 0, res = 0;
  getNext(m);
  for(i = 0; i < n; ++i){
    while(j > 0 && b[j] != a[i]) j =
        kmp_next[j-1];
    if(b[j] == a[i]) ++j;
    if(j == m) ++res;
  }
  return res;
}
```

8.3 LPS

```
//最長迴文子字串
#define T(x) ((x) % 2 ? s[(x) / 2] : '.')

string s;
int n;

int ex(int l, int r){
    int i = 0;
    while(l - i >= 0 && r + i < n && T(l - i
        ) == T(r + i)) i++;
    return i;
}

int main(){
    cin >> s;
    n = 2 * s.size() + 1;

    int mx = 0;
    int center = 0;
    vector<int> r(n);
    int ans = 1;
    r[0] = 1;
    for(int i = 1; i < n; i++){
        int ii = center - (i - center);
        int len = mx - i + 1;
        if(i > mx){
            r[i] = ex(i, i);
            center = i;
            mx = i + r[i] - 1;
        }
        else if(r[ii] == len){
            r[i] = len + ex(i - len, i + len
                );
            center = i;
            mx = i + r[i] - 1;
        }
        else{
            r[i] = min(r[ii], len);
        }
        ans = max(ans, r[i]);
    }

    cout << ans - 1 << "\n";
    return 0;
}
```

8.4 Trie

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

class Trie {
    struct Node {
        bool endofWord;
        vector<int> children;
        Node() : endofWord(false), children
            (26, -1) {}
    };

    vector<Node> trie;

public:
    Trie() {
        trie.emplace_back();
    }

    void insert(const string& word) {
        int cur = 0;
        for (char c : word) {
            int idx = c - 'a';
            if (trie[cur].children[idx] ==
                -1) {
                trie[cur].children[idx] =
                    trie.size();
                trie.emplace_back();
            }
            cur = trie[cur].children[idx];
        }
        trie[cur].endofWord = true;
    }

    bool search(const string& word) {
        int cur = 0;
        for (char c : word) {
            int idx = c - 'a';
            if (trie[cur].children[idx] ==
                -1) return false;
            cur = trie[cur].children[idx];
        }
        return trie[cur].endofWord;
    }

    bool startsWith(const string& prefix) {
        int cur = 0;
        for (char c : prefix) {
            int idx = c - 'a';
            if (trie[cur].children[idx] ==
                -1) return false;
            cur = trie[cur].children[idx];
        }
        return true;
    }

    void deleteWord(const string& word) {
        int cur = 0;
        for (char c : word) {
            int idx = c - 'a';
            if (trie[cur].children[idx] ==
                -1) return;
            cur = trie[cur].children[idx];
        }
        trie[cur].endofWord = false;
    }

    void print(int node, string prefix)
        const {
        if (trie[node].endofWord) {
            cout << prefix << "\n";
        }
        for (int i = 0; i < 26; i++) {
            int nxt = trie[node].children[i
                ];
            if (nxt != -1) {
                print(nxt, prefix + char('a'
                    + i));
            }
        }
    }

    void print() const { print(0, ""); }
};

int main() {
    Trie trie;

    trie.insert("geek");
    trie.insert("geeks");
    trie.insert("code");
    trie.insert("coder");
    trie.insert("coding");

    cout << "Trie contents:\n";
    trie.print();

    cout << "\nSearch results:\n";
    cout << "geek: " << trie.search("geek")
        << "\n";
    cout << "geeks: " << trie.search("geeks"
        ) << "\n";
    cout << "code: " << trie.search("code")
        << "\n";
    cout << "coder: " << trie.search("coder"
        ) << "\n";
    cout << "coding: " << trie.search("
        coding") << "\n";
    cout << "codex: " << trie.search("codex"
        ) << "\n";

    cout << "\nPrefix results:\n";
    cout << "ge: " << trie.startsWith("ge")
        << "\n";
    cout << "cod: " << trie.startsWith("cod"
        ) << "\n";
    cout << "coz: " << trie.startsWith("coz"
        ) << "\n";

    trie.deleteWord("coding");
    trie.deleteWord("geek");

    cout << "\nTrie contents after deletions
        :\n";
    trie.print();

    cout << "\nSearch results after
        deletions:\n";
    cout << "coding: " << trie.search("
        coding") << "\n";
    cout << "geek: " << trie.search("geek")
        << "\n";

    return 0;
}
```

8.5 Z-value

```
// CSES: String Matching
// Given a string and a pattern, your task
     is to count
// the number of positions where the pattern
     occurs in the string.

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

vector<ll> z_value(const string& s) {
    ll n = s.size();
    vector<ll> z(n);
    ll l = 0, r = 0;
    for (ll i = 1; i < n; ++i) {
        if (i <= r) z[i] = min(z[i - l], r -
            i + 1);
        while (i + z[i] < n && s[z[i]] == s[
            i + z[i]]) z[i] += 1;
```

```
        if (i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
    return z;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s1, s2; cin >> s1 >> s2;
    ll n = s1.size(), m = s2.size();

    ll cnt = 0;
    string s = s2 + "$" + s1;
    vector<ll> z = z_value(s);
    for (ll i = m; i < s.size(); ++i) {
        if (z[i] == m) cnt += 1;
    }
    cout << cnt << '\n';

    return 0;
}

// ==============================
// ==============================

// CSES: Finding Borders
// A border of a string is a prefix that is also a suffix of
// the string but not the whole string. For example,
// the borders of abcababcab are ab and abcab.
// Your task is to find all border lengths of a given string.

#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define pb push_back

vector<ll> z_value(const string& s) {
    ll n = s.size();
    vector<ll> z(n);
    ll l = 0, r = 0;
    for (ll i = 1; i < n; ++i) {
        if (i <= r) z[i] = min(z[i - l], r - i + 1);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) z[i] += 1;
        if (i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
    return z;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s; cin >> s;
    ll n = s.size();

    vector<ll> z = z_value(s), res;
    for (ll i = 1; i < n; ++i) {
        if (i + z[i] == n) res.pb(z[i]);
    }
    sort(res.begin(), res.end());
    for (ll x : res) {
        cout << x << ' ';
    } cout << '\n';

    return 0;
}

// ==============================
// ==============================

// CSES: Finding Periods
// A period of a string is a prefix that can be used to generate
// the whole string by repeating the prefix. The last repetition
// may be partial. For example, the periods of abcabca are abc, abcabc and abcabca.
// Your task is to find all period lengths of a string.

#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define pb push_back

vector<ll> z_value(const string& s) {
    ll n = s.size();
    vector<ll> z(n);
    ll l = 0, r = 0;
    for (ll i = 1; i < n; ++i) {
        if (i <= r) z[i] = min(z[i - l], r - i + 1);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) z[i] += 1;
        if (i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
    return z;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s; cin >> s;
    ll n = s.size();

    vector<ll> z = z_value(s), res;
    for (ll i = 1; i < n; ++i) {
        if (i + z[i] == n) res.pb(i);
    }
    sort(res.begin(), res.end());
    for (ll x : res) {
        cout << x << ' ';
    } cout << n << '\n';

    return 0;
}
```

8.6 Manacher

```
// CSES: Longest Palindrome
// Given a string, your task is to determine the longest
// palindromic substring of the string. For example,
// the longest palindrome in aybabtu is bab.

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string t, s = "^#"; cin >> t;
    ll n = t.size(), m = 2*n + 3;
    for (ll i = 0; i < n; ++i) {
        s += t[i];
        s += (i == n - 1 ? "#$" : "#");
    }

    ll c = 1;
    vector<ll> p(m); p[1] = 0;
    for (ll i = 2; i <= m - 3; ++i) {
        if (i < c + p[c]) p[i] = min(p[c - (i - c)], (c + p[c]) - i);
        while (i + p[i] + 1 < m && s[i - p[i] - 1] == s[i + p[i] + 1]) p[i] += 1;
        if (i + p[i] > c + p[c]) c = i;
    }

    ll j = 2;
    for (ll i = 3; i <= m - 3; ++i) {
        if (p[i] > p[j]) j = i;
    }

    for (ll i = j - p[j] + 1; i <= j + p[j] - 1; i += 2) {
        cout << s[i];
    } cout << '\n';

    return 0;
}

// ==============================
// ==============================

// CSES: All Palindromes
// Given a string, calculate for each position the length
// of the longest palindrome that ends at that position.

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

ll n, m;
string ori_s, s;
vector<ll> p, rt, dp;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> ori_s;
    n = ori_s.size();
    m = 2*n + 3;
    s = "^#";
    for (ll i = 0; i < n; ++i) {
        s += ori_s[i];
        s += (i == n - 1 ? "#$" : "#");
    }

    ll c = 0;
    p.resize(m);
    for (ll i = 2; i <= m - 3; ++i) {
        if (i < c + p[c]) p[i] = min(p[c - (i - c)], (c + p[c]) - i);
        while (i - p[i] - 1 >= 0 && s[i - p[i] - 1] == s[i + p[i] + 1]) p[i] += 1;
        if (i + p[i] > c + p[c]) c = i;
    }

    rt.resize(n, 1);
    for (ll i = 2; i <= m - 3; ++i) {
        ll y = ((i + p[i] - 1) - 2)/2;
        if (s[i] == '#') {
            ll x = ((i + 1) - 2)/2;
            rt[y] = max(rt[y], (y - x + 1)*2);
        } else {
            ll x = (i - 2)/2;
            rt[y] = max(rt[y], (y - x)*2 + 1);
        }
    }

    dp.resize(n); dp[n - 1] = rt[n - 1];
    for (ll i = n - 2; i >= 0; --i) {
        dp[i] = max(rt[i], dp[i + 1] - 2);
    }

    for (ll x : dp) {
        cout << x << ' ';
    } cout << '\n';

    return 0;
}
```

9 Tree

9.1 Binary Lifting

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

```cpp
using namespace std;
using ll = long long;

inline ll flg(ll x) {
    return 63 - __builtin_clzll(x);
}

inline bool isOnBit(ll x, ll i) {
    return ((1LL << i) & x) > 0;
}

ll n, q, lgn;
vector<vector<ll>> blf;

void init() {
    blf[0][1] = -1;
    for (ll u = 2; u <= n; ++u) cin >> blf[0][u];

    for (ll h = 1; h <= lgn; ++h) {
        for (ll u = 1; u <= n; ++u) {
            ll nt = blf[h - 1][u];
            blf[h][u] = nt == -1 ? -1 : blf[h - 1][nt];
        }
    }
}
ll query(ll u, ll step) {
    ll cur = u;
    for (ll i = 30; i >= 0; --i) {
        if (isOnBit(step, i)) {
            cur = blf[i][cur];
            if (cur == -1) return -1;
        }
    }
    return cur;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n >> q;
    lgn = flg(n);
    blf.resize(lgn + 1, vector<ll>(n + 1));
    init();

    while (q--) {
        ll u, step; cin >> u >> step;
        cout << query(u, step) << '\n';
    }

    return 0;
}
```

9.2 LCA

```cpp
// Use binary lifting

#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define pb push_back

inline ll flg(ll x) {
    return 63 - __builtin_clzll(x);
}

inline bool isOnBit(ll x, ll i) {
    return ((1LL << i) & x) > 0;
}

ll n, q, lgn;
vector<ll> d;
vector<vector<ll>> blf, adj;

void init() {
    blf[0][1] = -1;
    for (ll u = 2; u <= n; ++u) {
        ll v; cin >> v;
        blf[0][u] = v;
        adj[v].pb(u);
    }

    for (ll h = 1; h <= lgn; ++h) {
        for (ll u = 1; u <= n; ++u) {
            ll nt = blf[h - 1][u];
            blf[h][u] = nt == -1 ? -1 : blf[h - 1][nt];
        }
    }
}
ll query(ll u, ll step) {
    ll cur = u;
    for (ll i = 30; i >= 0; --i) {
        if (isOnBit(step, i)) {
            cur = blf[i][cur];
            if (cur == -1) return -1;
        }
    }
    return cur;
}

void dfs(ll u, ll dn) {
    d[u] = dn;
    for (ll v : adj[u]) {
        dfs(v, dn + 1);
    }
}

ll lca(ll u, ll v) {
    if (d[u] > d[v]) swap(u, v);
    if (d[u] < d[v]) v = query(v, d[v] - d[u]);
    if (u == v) return u;
    for (ll h = lgn; h >= 0; --h) {
        ll ntu = blf[h][u];
        ll ntv = blf[h][v];
        if (ntu == -1 or ntv == -1 or ntu == ntv) continue;
        u = ntu;
        v = ntv;
    }
    return blf[0][u];
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n >> q;
    lgn = flg(n);
    d.resize(n + 1);
    blf.resize(lgn + 1, vector<ll>(n + 1));
    adj.resize(n + 1);
    init();
    dfs(1, 0);

    while (q--) {
        ll u, v; cin >> u >> v;
        cout << lca(u, v) << '\n';
    }

    return 0;
}
```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

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

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將 block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> diff = end - begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << " byte" << endl;
}
```