Codebook

MU_Codebenders

Metropolitan University, Sylhet

Sudipto Dey Himel

Maria Akter Mariam

Md. Tanimur Rahman

Part1 → Number Theory

1. BigMod

```
ll bigMod(ll n, ll k) {
    ll res=1;
    n %= MOD;
    while(k) {
        if(k&1)  res = (res*n)%MOD;
        n = (n*n)%MOD;
        k >>= 1;
    }
    return res;
}
```

2. Sieve

```
ll const M = 1e18;
bitset<M> isPrime;

void sieve(int n) {
    isPrime.set();
    isPrime[0] = isPrime[1] = 0;
    for(int i=2; i*i<=n; i++) {
        if(isPrime[i]) {
            for(int j=i*i; j<=n; j+=i) {
                isPrime[j] = 0;
            }
        }
    }
}
```

3. nCr (1)

```
ll nCr(ll n, ll r) {
    ll p = 1, q = 1;
    if(r > n-r) r = n-r;
    if(r) {
        while(r) {
            p *= n; q *= r;
            ll x = gcd(p, q);
            p /= x; q /= x;

            n--; r--;
        }
    }
    else p = 1;

    return p;
}
```

4. nCr(2)

```
ll nCr(ll n, ll r) {   // using bionomial coefficient
```

```
        if(r > n-r) r = n-r;
        ll ans = 1;
        for(ll i=0; i<r; i++) {
            ans *= (n-i);
            ans /= (i+1);
        }
        return ans;
    }
```

5. Divisors

```
const ll M = 1e7+5;
vector<ll> V(M);

 void divisorCount() {
    for(ll i=1; i<=M; i++) {
        for(ll j=i; j<=M; j+=i) {
            V[j]++;
        }
    }
 }
```

6. Divisors of n

```
vector<int> divisors

void divisors(ll n) {
    for(int i=1; i*i <= n; i++) {
        if(n%i == 0) divisors.push_back(i);
        if(n/i != i) divisors.push_back(n/i);
    }
}
```

7. Number of Common Divisors

```
ll count_common_divisor(ll a, ll b) {
    ll n = __gcd(a, b);
    ll count = 0;
    for(ll i=1; i*i <= n; i++) {
        if(n%i == 0) {
            count += 2;
            if(i*i == n) count--;
        }
    }
    return count;
}
```

8. Prime Fectorization

```
ll primeFact(ll n) {
    set<ll> st;
    for(ll i=2; i*i<=n; i++) {
        if(n%i==0) {
            int count=0;
```

```
                while(n%i==0) {
                    count++;
                    n/=i;
                }
                st.insert(i);
            }
        }
        if(n>1) st.insert(n);

        return st.size();
    }
```

Part2 → Graph

1. DFS

```
void DFS(int st) {
    visited[st] = true;
    for (auto neighbor : graph[st]) {
        if (!visited[neighbor])
            DFS(neighbor);
    }
}
```

2. BFS

```
void BFS(int start) {
    visited[start] = true;
    queue<int> q;
    q.push(start);
    while (!q.empty()) {
        int current = q.front();
        q.pop();
        for (auto neighbor : graph[current]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }
    }
}
```

3. Shortest Path

```
vector<ll> g[M];

void bfs(ll src, vector<ll>&par, vector<ll>&dist) {
    queue<ll> q;
    q.push(src);
    dist[src] = 0;
    while(!q.empty()) {
        ll u = q.front();
        q.pop();
```

```
            for(auto v : g[u]) {
                if(dist[v] == inf) {
                    dist[v] = dist[u] + 1;
                    par[v] = u;
                    q.push(v);
                }
            }
        }
    }


    void solve() {
        ll v, e; cin >> v >> e;

        g->clear();

        for(ll i=0; i<e; i++) {
            ll u, v; cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }

        vector<ll> dist(v+1, inf);
        vector<ll> par(v+1, -1);

        ll src = 1, dest = v;

        bfs(src, par, dist);

        vector<ll> path;
        ll curr = v;

        while(curr != -1) {
            path.push_back(curr);
            curr = par[curr];
        }

        if(dist[v] == inf) {
            cout << "Impossible\n"; return;
        }

        cout << dist[v] + 1 << endl;

        reverse(path.begin(), path.end());

        for(auto i : path) cout << i << " "; cout << endl;
    }
```

4. Dijkstra

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

#define mx 100123  // maximum number of nodes
#define infLL LLONG_MAX // define infinity as the largest possible value of
long long

typedef long long ll;
typedef pair<ll, ll> pll;
vector<pll> adj[mx];  // adjacency list with weight
ll dist[mx];  // distance array

void dijkstra(int s, int n) {
    for (int i = 1; i <= n; i++) dist[i] = infLL; // initialize distances
to infinity
    dist[s] = 0; // initialize source distance to 0
    priority_queue<pll, vector<pll>, greater<pll>> pq;  // min-heap
priority queue
    pq.push({0, s});  // push source node with distance 0

    while (!pq.empty()) {
        int u = pq.top().second;  // node u
        ll curD = pq.top().first; // current distance to u
        pq.pop();

        if (dist[u] < curD) continue; // if current distance is not
optimal, continue

        // relax all neighbors
        for (auto p : adj[u]) {
            int v = p.first; // adjacent node v
            ll w = p.second; // weight of the edge from u to v
            if (curD + w < dist[v]) { // relax the edge
                dist[v] = curD + w;
                pq.push({dist[v], v});  // push the updated distance of v
            }
        }
    }
}

int main() {
    int n, m;
    cin >> n >> m;  // input number of nodes and edges
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        cin >> u >> v >> w; // input edges with weights
        adj[u].push_back({v, w});  // add edge from u to v with weight w
        adj[v].push_back({u, w});  // add reverse edge for undirected graph
    }
    dijkstra(1, n);  // run Dijkstra starting from node 1

    // print the shortest distance from node 1 to all nodes
    for (int i = 1; i <= n; i++) {
        if (dist[i] == infLL) cout << "-1 "; // unreachable node
        else cout << dist[i] << " "; // reachable node
    }
    cout << endl;

    return 0;
}
```

Part3 → Utilities

1. Bit Manipulation

```
ll Set(ll num, ll pos) {
    return num | (1LL << pos);
}

ll Clear(ll num, ll pos) {
    return num & ~(1LL << pos);
}

ll Toggle(ll num, ll pos) {
    return num ^ (1LL << pos);
}

bool Check(ll num, ll pos) {
    return (bool)(num & (1LL << pos));
}
```

2. Direction

```
int dx4[] = {0, 0, -1, 1};
int dy4[] = {1, -1, 0, 0};
int dx8[] = {1, 1, 1, 0, 0, -1, -1, -1};
int dy8[] = {1, 0, -1, 1, -1, 1, 0, -1};
int dx_horse[] = {1, 1, -1, -1, 2, 2, -2, -2}
Int dy_horse[] = {2, -2, 2, -2, 1, -1, 1, -1}
```

Part4 → Geometry & Math

Formula:

- Triangle:
  Area:
  Using coordinates: A = 1/2 * |x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2)|
  A = (b+h)/2 // b = base, h = height
  Heron's formula:
  A = sqrt(s*(s-a)*(s-b)*(s-c)) // s = (a+b+c)/2
  Perimeter: (a+b+c)

Volume of a triangular prism: A * H // A = area of triangle, H = height of prism

- Distance between 2 point: d = sqrt((x2-x1)^2 + (y2-y1)^2)

  Mid point: M = ((x1+x2)/2, (y1+y2)/2)

- Cylinder:

  Area: A = 2 * pi * r * (r + h)
  Volume: V = pi * r * r * h

Part5 → Segment Tree

1. Normal

```
vector<ll> arr;
vector<ll> tree;

void  init(ll node, ll begin, ll end) {
    if(begin == end) {
        tree[node] = arr[begin]; return;
    }

    ll mid = (begin + end) >> 1;

    init(node << 1, begin, mid);
    init((node << 1) + 1, mid + 1, end);

    tree[node] = tree[node << 1] + tree[(node << 1) + 1];
}

ll query(ll node, ll begin, ll end, ll left, ll right) {
    if(left > end || right < begin) return 0;

    if(left <= begin && right >= end) return tree[node];

    ll mid = (begin + end) >> 1;
    ll sumL = query(node << 1, begin, mid, left, right);
    ll sumR = query((node << 1) + 1, mid + 1, end, left, right);

    return (sumL + sumR);
}

void update(ll node, ll begin, ll end, ll i, ll value) {
    if(i > end || i < begin) return;
    if(i <= begin && i >= end) {
        tree[node] = value; return;
    }

    ll mid = (begin + end) >> 1;

    update(node << 1, begin, mid, i, value);
    update((node << 1) + 1, mid+1, end, i, value);

    tree[node] = tree[node << 1] + tree[(node << 1) + 1];
}

void solve() {
    ll n, q; cin >> n >> q;

    arr.resize(n+1); tree.resize(4*n);

    for(ll i=1; i<=n; i++) cin >> arr[i];

    init(1, 1, n);

    while(q--) {
        ll type; cin >> type;
        if(type == 1) {
            ll i, v; cin >> i >> v;
            update(1, 1, n, i, v);
        }
        else {
            ll l, r; cin >> l >> r;
            cout << query(1, 1, n, l, r) << endl;
        }
    }
}
```

2. Lazy

```
vector<ll> arr;
vector<ll> tree;
vector<ll> lazy;

void shift(ll node, ll b, ll e) {
    if(lazy[node]) {
        tree[node] += ((e - b + 1) * lazy[node]);
        if(b != e) {
            lazy[node << 1] += lazy[node];
            lazy[(node << 1) + 1] += lazy[node];
        }
        lazy[node] = 0;
    }
}

void init(ll node, ll b, ll e) {
    if(b == e) {
        tree[node] = arr[b];
        return;
    }

    ll mid = (b + e) >> 1;
    init(node << 1, b, mid); init((node << 1) + 1, mid + 1, e);

    tree[node] = tree[node << 1] + tree[(node << 1) + 1];
}

void update(ll node, ll b, ll e, ll i, ll j, ll value) {
    shift(node, b, e);

    if(e < i || b > j) return;
    if(b >= i && e <= j) {
        tree[node] += ((e - b + 1) * value);
        if(b != e) {
            lazy[node << 1] += value;
            lazy[(node << 1) + 1] += value;
        }
        return;
    }
    ll mid = (b + e) >> 1;
    update(node << 1, b, mid, i, j, value);
    update((node << 1) + 1, mid + 1, e, i, j, value);

    tree[node] = tree[node << 1] + tree[(node << 1) + 1];
}

ll query(ll node, ll b, ll e, ll l, ll r) {
    shift(node, b, e);

    if (e < l || b > r) return 0;
    if (b >= l && e <= r) return tree[node];

    ll mid = (b + e) >> 1;
    ll q1 = query(node << 1, b, mid, l, r);
    ll q2 = query((node << 1) + 1, mid + 1, e, l, r);

    return q1 + q2;
}

void solve() {
    ll n, q; cin >> n >> q;

    arr.resize(n+1); tree.resize(n*4); lazy.resize(n*4, 0);

    for(ll i=1; i<=n; i++) cin >> arr[i];

    init(1, 1, n);
```

```
        while(q--) {
            ll t; cin >> t;
            if(t == 0) {
                ll i, j, v; cin >> i >> j >> v;
                update(1, 1, n, i, j, v);
            }
            else {
                ll l, r; cin >> l >> r;
                cout << query(1, 1, n, l, r) << endl;
            }
        }
    }
```

Part6 → DP

1. LIS

```
void LIS() {
    int n; cin >> n;
    vector<int> V(n);
    for(auto &x : V) cin >> x;
    vector<int> lis;
    for(auto i : V) {
        auto it = lower_bound(lis.begin(), lis.end(), i);
        if(it != lis.end()) {
            *it = i;
        }
        else {
            lis.push_back(i);
        }
    }
    cout << lis.size() << endl;
}
```

2. LCS

```
const int Max = 1000 + 5;
int dp[Max][Max];
bool vist[Max][Max];
string s, t;
int n, m;
int lcs(int i, int j)
{
    if (i >= n or j >= m)
    {
        return 0;
    }
    int &ret = dp[i][j];
    bool &vis = vist[i][j];
    if (vis)
    {
        return ret;
    }
    vis = 1;
    int res = 0;
    if (s[i] == t[j])
    {
        res = 1 + lcs(i + 1, j + 1);
    }
    else
    {
        res = max(lcs(i + 1, j), lcs(i, j + 1));
    }
    return ret = res;
}
string ans;
void solution(int i, int j)
{
    if (i >= n or j >= m)
    {
        return;
    }
    if (s[i] == t[j])
    {
        ans += s[i];
        solution(i + 1, j + 1);
    }
    else
    {
        if (lcs(i + 1, j) > lcs(i, j + 1))
        {
            solution(i + 1, j);
        }
        else
        {
            solution(i, j + 1);
        }
    }
}
int main()
{
    int T;
    cin >> T;
    for (int tc = 1; tc <= T; tc++)
    {
        cin >> s >> t;
        n = s.size();
        m = t.size();
        memset(vist, 0, sizeof vist);
        int maxlen = lcs(0, 0);
        ans = "";
        solution(0, 0);
        cout << ans << '\n';
    }
}
```

Extra:

1. **Subset:**

```
void solve() {
    ll n; cin >> n;
    vector<ll> V(n);
    for(auto &x : V) cin >> x;
        bool f = false;
    for(ll mask=0; mask<(1<<n); mask++) {
        vector<ll> temp;
        for(ll i=0; i<n; i++) {
            if(mask & (1<<i)) {
                temp.pb(V[i]);
            }
        }
        for (auto &x : temp) {
            cout << x << " ";
        }
        cout << endl;
    }
}
```

2. **Two pointer:**

```
void solve() {
    ll n, k; cin >> n >> k;
    vector<ll> V(n);
    set<ll> st;
    for(auto &x : V) cin >> x;
    vector<pair<ll, ll>> pr;

    ll j = 0, i = 0, sum = 0, ans = 0;
    while(j < n) {
        sum += V[j];
        if(j-i+1 == k) {
            pr.pb({sum, i});
        }
        if(j-i+1 > k) {
            while(j-i+1 > k) {
                sum -= V[i];
                i++;
            }
            ans = max(sum, ans);
        }
        j++;
    }
    cout << max(ans, sum) < endl;
}
```

3. **Subarray sum:**

```
int main() {
  int n; cin >> n;
  int a[n];
  for (int i = 0; i < n; i++) {
     cin >> a[i];
  }
  long long max_subarray_sum = -1e18;
  long long sum = -1e18;
  for (int i = 0; i < n; i++) {
     sum=max((long long)a[i],a[i]+sum); // max subarray sum ending at index i
     max_subarray_sum = max(max_subarray_sum, sum);
  }
  cout << max_subarray_sum << '\n';
  return 0;
}
```