Codebook

crazycoder00

1 Templates and Scripts

1.1 template.cpp

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

// Template
// ============================================
    // pbds
    // #include <ext/pb_ds/assoc_container.hpp>
    // #include <ext/pb_ds/tree_policy.hpp>
    // using namespace __gnu_pbds;
    // template<typename T, typename comp = less<T>>
    // using ordered_set =  tree<T, null_type, comp,
     rb_tree_tag, tree_order_statistics_node_update>;

    // Debugging
    #ifdef LOCAL
    #include "debug.h"
    #else
    #define debug(...)
    #define see(x)
    #endif

    typedef long long ll;
    typedef vector<int> VI;
    typedef vector<long long> VLL;
    typedef vector<bool> VB;
    typedef vector<vector<int>> VVI;
    typedef pair<int, int> PI;
    typedef pair<ll, ll> PLL;
    typedef vector<pair<int, int>> VPI;

    #define pb push_back
    #define ff first
    #define ss second
    #define mp make_pair
    #define all(a) a.begin(), a.end()
    #define revall(a) a.rbegin(), a.rend()

    #define loop(i, s, e) for (int i = s; i < e; ++i)
    #define inp(v) for (auto& x : v) cin >> x
    #define outp(v) for (int i = 0, n = v.size(); i < n; ++
     i) cout << v[i] << " \n"[i == n - 1]

    #define nl "\n"
    #define yep cout << "YES\n"
    #define nope cout << "NO\n"

    #define INF (int) 1e9
    #define INFL (ll) 1e18
    // #define MOD 998244353
    #define MOD 1000000007
    #define MAXN 300002
// ============================================

void solve()
{
}

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

    int t = 1;
    cin >> t;
    while(t--) solve();

    #ifdef LOCAL
    cerr << "Execution time: " << 1000.f * clock() /
     CLOCKS_PER_SEC << " ms." << nl;
    #endif

    return 0;
}
```

1.2 debug.h

```cpp
#include <iostream>
using namespace std;

#define see(x) cerr << #x << ": " << x << nl

template<typename... Args>
void debug(Args... args)
{
    ((cerr << " " << args), ...) << "\n";
}
```

1.3 sublime.build

```
{
    "shell_cmd": "g++ -Wall -std=c++20 -O2 -DLOCAL $file -o
      $file_base_name && ./$file_base_name <input.in>
     output.out 2> error.log",
    "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?:? (.*)$",
    "working_dir": "${file_path}",
    "selector": "source.c++, source.c, source.cpp",

    "variants":
    [
        {
            "name": "sanitize",
            "shell_cmd": "g++ -Wall -std=c++20 -fsanitize=
     address -fsanitize=undefined -DLOCAL $file -o
     $file_base_name && ./$file_base_name <input.in> output
     .out 2> error.log"
        }
    ]
}
```

1.4 tasks.json (vs code)

```
{
    "version": "2.0.0",
    "tasks": [
        {
```

```
            "label": "cp",
            "type": "shell",
            "command": "",
            "args": [
                "g++",
                "-Wall",
                "-std=c++20",
                "-O2",
                "-DLOCAL",
                "\"${file}\"",
                "-o",
                "\"${fileDirname}/${fileBasenameNoExtension
    }\"",
                "&&",
                "\"${fileDirname}/${fileBasenameNoExtension
    }\"",
                "<",
                "input.in",
                ">",
                "output.out",
                "2>",
                "error.log"
            ],
            "group": "build",
            "presentation": {
                "reveal": "silent"
            },
            "problemMatcher": {
                "owner": "cpp",
                "fileLocation": ["relative", "${
    workspaceRoot}"],
                "pattern": {
                "regexp": "^(.*):(\\d+):(\\d+):\\s+(warning
    |error):\\s+(.*)$",
                "file": 1,
                "line": 2,
                "column": 3,
                "severity": 4,
                "message": 5
                }
            }
        }
    ]
}
```

1.5 Interactive

```
socat EXEC:"./interactor" EXEC:"./soln"
```

2 Number Theory

2.1 Mod Template

```
ll mod(ll n)
{
    n %= MOD;
    return n < 0 ? MOD + n : n;
}

ll modpow(ll n, ll k)
{
    ll ret = 1;

    while (k)
    {
        if (k & 1) ret = (ret * n) % MOD;
        n = (n * n) % MOD;
        k >>= 1;
    }

    return ret;
}

inline ll inv(ll n)
{
    return modpow(n, MOD - 2);
}

inline ll mul(ll x, ll y)
{
    return ((x % MOD) * (y % MOD)) % MOD;
}

inline ll dvd(ll x, ll y)
{
    return mul(x, inv(y));
}
```

2.2 Binomial Coeffecient

```
ll fact[MAXN];

void calc_fact()
{
    fact[0] = 1;
    for (int i = 1; i < MAXN; ++i)
    {
        fact[i] = (fact[i - 1] * i) % MOD;
    }
}

ll C(int n, int k)
{
    if (k > n) return 0;
    return (1LL * fact[n] *
        inv((1LL * fact[k] * fact[n - k]) % MOD)) % MOD;
}
```

2.3 Sieve

```
VI primes;
bool isprime[MAXN];

void sieve()
{
    isprime[1] = false;
    for (int i = 2; i < MAXN; ++i) isprime[i] = true;

    for (int i = 2; i < MAXN; ++i)
    {
        if (!isprime[i]) continue;
        primes.pb(i);

        if (1LL * i * i >= MAXN) continue;
        for (int j = i * i; j < MAXN; j += i)
            isprime[j] = false;

    }
}
```

2.4 CRT

```
int extended_gcd(ll a, ll b, ll& x, ll& y)
{
    if (!b)
    {
        x = 1, y = 0;
        return a;
    }

    ll g = extended_gcd(b, a % b, x, y);
    swap(x, y);
    y -= x * (a / b);
    return g;
}

ll modinv(ll a, ll m)
{
    ll x, y;
    extended_gcd(a, m, x, y);

    x %= m;
    return x < 0 ? x + m : x;
}

ll crt(vector<pair<ll, ll>>& mods)
{
    ll mod = 1;
    for (auto& p : mods) mod *= p.ss;

    ll ans = 0;
    for (auto [x, p] : mods)
    {
```

```
        ll m = mod / p;
        m = (m * modinv(m, p)) % mod;
        m = (m * x) % mod;

        ans = (ans + m) % mod;
    }

    return ans;
}
```

3 Data Structure

3.1 Fenwick Tree

```
struct fenwick
{
    int n;
    vector<ll> tree;

    fenwick(int s)
    {
        n = s;
        tree.assign(n + 1, 0);
    }

    ll get(int i)
    {
        ll res = 0;
        for(; i > 0; i -= i & -i) res += tree[i];
        return res;
    }

    void add(int i, int x)
    {
        for (; i <= n; i += i & -i) tree[i] += x;
    }

    ll rsq(int l, int r)
    {
        return get(r) - get(l - 1);
    }
};
```

3.2 Segment Tree

```
template<typename T>
struct segtree
{
    #define left(u) 2 * u + 1
    #define right(u) 2 * u + 2

    int n;
    T init_val;
    vector<T> tree;
    function<T (T, T)> f;

    bool is_lazy = false;
    T init_lazy;
    vector<T> lazy;
    function<void (T&, T, int, int)> apply_lazy;
    function<void (T&, T)> merge_lazy;

    segtree() {} // Dummy constructor for global declaration

    segtree(int sz, function<T (T, T)> f, T val = 0)
    {
        n = sz;
        this->f = f;
        init_val = val;

        tree.assign(1 << (__lg(n - 1) + 2), init_val);
    }

    // apply(tree[u], lazy[u], l, r)
    // merge(lazy[left(u)], lazy[u])
    void make_lazy(function<void (T&, T, int, int)> apply,
     function<void (T&, T)> merge, T init = 0)
    {
        is_lazy = true;
        init_lazy = init;
        apply_lazy = apply;
        merge_lazy = merge;

        lazy.assign(tree.size(), init_lazy);
    }

    void build(vector<T>& a) { _build(0, 0, n - 1, a); }
    void update(int i, T x) { _update(0, 0, n - 1, i, x); }
    void update(int l, int r, T x) { _update_range(0, 0, n
     - 1, l, r, x); }
    T get(int l, int r) { return _get(0, 0, n - 1, l, r); }

    void _build(int u, int l, int r, vector<T>& a)
    {
        if (l == r)
        {
            tree[u] = a[l];
            return;
        }

        int mid = (l + r) / 2;
        _build(left(u), l, mid, a);
        _build(right(u), mid + 1, r, a);
        tree[u] = f(tree[left(u)], tree[right(u)]);
    }

    void _update(int u, int l, int r, int i, T x)
    {
        if (l == r)
        {
            tree[u] = x;
            return;
        }

        int mid = (l + r) / 2;
        if (i <= mid) _update(left(u), l, mid, i, x);
        else _update(right(u), mid + 1, r, i, x);
        tree[u] = f(tree[left(u)], tree[right(u)]);
    }

    void propagate(int u, int l, int r)
    {
        if (lazy[u] == init_lazy || l == r) return;

        int m = (l + r) / 2;
        apply_lazy(tree[left(u)], lazy[u], l, m);
        merge_lazy(lazy[left(u)], lazy[u]);
        apply_lazy(tree[right(u)], lazy[u], m + 1, r);
        merge_lazy(lazy[right(u)], lazy[u]);

        lazy[u] = init_lazy;
    }

    void _update_range(int u, int tl, int tr, int l, int r,
      T x)
    {
        if (l > tr || r < tl) return;
        if (tl >= l && tr <= r)
        {
            apply_lazy(tree[u], x, tl, tr);
            merge_lazy(lazy[u], x);
            return;
        }

        propagate(u, tl, tr);

        int mid = (tl + tr) / 2;
        _update_range(left(u), tl, mid, l, r, x);
        _update_range(right(u), mid + 1, tr, l, r, x);

        tree[u] = f(tree[left(u)], tree[right(u)]);
    }

    T _get(int u, int tl, int tr, int l, int r)
    {
        if (l > tr || r < tl) return init_val;

        if (l <= tl && r >= tr) return tree[u];

        if (is_lazy) propagate(u, tl, tr);

        int mid = (tl + tr) / 2;
        return f(_get(left(u), tl, mid, l, r),
                 _get(right(u), mid + 1, tr, l, r));
    }

    #undef left
    #undef right
```

```
};
```

3.3 DSU

```
struct dsu
{
    vector<int> parent, size;

    dsu(int n)
    {
        parent.resize(n + 1);
        size.resize(n + 1);

        for (int i = 1; i <= n; ++i)
        {
            parent[i] = i;
            size[i] = 1;
        }
    }

    int find(int u)
    {
        if (parent[u] == u) return u;
        return parent[u] = find(parent[u]);
    }

    void merge(int u, int v)
    {
        u = find(u);
        v = find(v);

        if (u != v)
        {
            if (size[u] < size[v]) swap(u, v);
            parent[v] = u;
            size[u] += size[v];
        }
    }
};
```

4 Graph

4.1 Dijkstra

```
struct Node
{
    int u;
    ll dist;

    bool operator< (const Node& v) const
    {
        return dist > v.dist;
    }
};

void dijkstra(int start, vector<ll>& dis, vector<VPI>& adj)
{
    priority_queue<Node> q;
    dis[start] = 0;
    q.push({start, 0});

    while (!q.empty())
    {
        Node node = q.top();
        int u = node.u;
        q.pop();

        if (dis[u] < node.dist) continue;

        for (auto e : adj[u])
        {
            if (node.dist + e.ss < dis[e.ff])
            {
                dis[e.ff] = node.dist + e.ss;
                q.push({e.ff, dis[e.ff]});
            }
        }
    }
}
```

4.2 LCA

```
struct LCA
{
    int step = 0;
    VI vis_start, vis_end;
    VVI st;

    int k;
    VVI& tree;

    LCA(VVI& tree) : tree(tree)
    {
        init(tree.size() - 1);
    }

    void build_st(int u, int p)
    {
        st[u][0] = p;
        for (int i = 1; i <= k; ++i)
            st[u][i] = st[st[u][i - 1]][i - 1];
    }

    void dfs(int u, int p)
    {
        vis_start[u] = step++;
        build_st(u, p);

        for (auto v : tree[u])
        {
            if (v != p) dfs(v, u);
        }

        vis_end[u] = step++;
    }

    bool is_ancestor(int a, int b)
    {
        return vis_start[a] < vis_start[b] && vis_end[a] >
     vis_end[b];
    }

    int get(int a, int b)
    {
        if (a == b) return a;
        if (is_ancestor(a, b)) return a;
        if (is_ancestor(b, a)) return b;

        for (int i = k; i >= 0; --i)
        {
            if (!is_ancestor(st[a][i], b)) a = st[a][i];
        }

        return st[a][0];
    }

    void init(int n)
    {
        vis_start.assign(n + 1, 0);
        vis_end.assign(n + 1, 0);

        k = __lg(2 * n - 1);
        st.assign(n + 1, VI(k + 1));
        dfs(1, 1);
    }

    // Extra
    int get_ancestor(int u, int n)
    {
        for (int i = 0; n > 0; ++i)
        {
            if (n & (1 << i))
            {
                u = st[u][i];
                n -= (1 << i);
            }
        }

        return u;
    }
};
```

4.3 SCC

```
VVI scc;
VI stk;
```

```
int disc[MAXN], low[MAXN];
bool instk[MAXN];
int cur_time;

void dfs_scc(int u, VVI& g)
{
    disc[u] = low[u] = ++cur_time;
    instk[u] = true;
    stk.pb(u);

    for (int v : g[u])
    {
        if (!disc[v])
        {
            dfs_scc(v, g);
            low[u] = min(low[u], low[v]);
        }
        else if (instk[v])
            low[u] = min(low[u], disc[v]);
    }

    if (low[u] < disc[u]) return;

    scc.pb(VI());
    bool rem = true;
    while (rem)
    {
        int v = stk.back();
        stk.pop_back();
        instk[v] = false;
        scc.back().pb(v);
        rem = v != u;
    }
}

void tarjan(VVI& g)
{
    cur_time = 0;
    int n = g.size();

    for (int i = 0; i < n; ++i) disc[i] = 0;
    for (int u = 1; u < n; ++u) if (!disc[u]) dfs_scc(u, g)
     ;
}
```

4.4 HLD

```
VI heavy, chain, label, par, sz, dep;
int timer;

void dfsz(int u, int p, VVI& tree)
{
    sz[u] = 1;
    par[u] = p;
    dep[u] = dep[p] + 1;
    int mx = -1;
    for (int v : tree[u])
    {
        if (v == p) continue;

        dfsz(v, u, tree);
        sz[u] += sz[v];
        if (sz[v] > mx) mx = sz[v], heavy[u] = v;
    }
}

void dfshld(int u, int p, VVI& tree, int head)
{
    label[u] = timer++;
    chain[u] = head;

    if (heavy[u] != -1) dfshld(heavy[u], u, tree, head);

    for (int v : tree[u])
    {
        if (v == p || v == heavy[u]) continue;

        dfshld(v, u, tree, v);
    }
}

void inithld(int n, VVI& tree)
{
    heavy = VI(n + 1, -1);
    chain = VI(n + 1);
    label = VI(n + 1);
    sz = VI(n + 1);
    par = VI(n + 1);
    dep = VI(n + 1);

    timer = 1;
    dfsz(1, 0, tree);
    dfshld(1, 0, tree, 1);
}

int qryup(int u, int v)
{
    if (dep[u] < dep[v]) swap(u, v);

    int head = chain[u];
    int ret = 0; // init value
    while (dep[head] > dep[v])
    {
        // ret = ...(label[head], label[u]);
        u = par[head];
        head = chain[u];
    }

    // ret = ...(label[v], label[u]);
    return ret;
}
```

4.5 Centroid Decomposition

```
struct CD
{
    vector<set<int>> tree;
    vector<int> par, sz;

    CD(vector<vector<int>>& tree)
    {
        for (auto& v : tree)
        {
            this->tree.emplace_back(v.begin(), v.end());
        }

        par.resize(tree.size());
        sz.resize(tree.size());
        build(1, 0);
    }

    void build(int u, int p)
    {
        dfs(u, p);
        int centroid = dfs(u, p, sz[u]);
        par[centroid] = p;

        vector<int> adj(tree[centroid].begin(), tree[
     centroid].end());
        for (int v : adj)
        {
            tree[v].erase(centroid);
            tree[centroid].erase(v);

            build(v, centroid);
        }
    }

    void dfs(int u, int p)
    {
        sz[u] = 1;

        for (int v : tree[u])
        {
            if (v == p) continue;

            dfs(v, u);
            sz[u] += sz[v];
        }
    }

    int dfs(int u, int p, int n)
    {
        for (int v : tree[u])
        {
            if (v == p) continue;
```

```
            if (sz[v] > n / 2) return dfs(v, u, n);
        }

        return u;
    }

    int operator[](int i)
    {
        return par[i];
    }
};
```

5 String

5.1 Trie

```
struct trienode
{
    bool endmark;
    vector<int> child;

    trienode(int sz)
    {
        endmark = false;
        child.resize(sz, 0);
    }
};

struct trie
{
    int sz;
    char fst;
    vector<trienode> nodes;

    trie(int alpha_sz, char alpha_start)
    {
        sz = alpha_sz;
        fst = alpha_start;

        // root at idx 0
        nodes.pb(trienode(sz));
    }

    void insert(string& s)
    {
        int cur = 0;
        for (char c : s)
        {
            if (!nodes[cur].child[c - fst])
            {
                nodes[cur].child[c - fst] = nodes.size();
                nodes.pb(trienode(sz));
            }
            cur = nodes[cur].child[c - fst];
        }

        nodes[cur].endmark = true;
    }

    bool search(string& s)
    {
        int cur = 0;
        for (char c : s)
        {
            if (!nodes[cur].child[c - fst])
                return false;

            cur = nodes[cur].child[c - fst];
        }

        return nodes[cur].endmark;
    }

    bool erase(string& s)
    {
        int cur = 0;
        for (char c : s)
        {
            if (!nodes[cur].child[c - fst])
                return false;

            cur = nodes[cur].child[c - fst];
        }

        if (!nodes[cur].endmark) return false;

        nodes[cur].endmark = false;
        return true;
    }
};
```

5.2 Z Algorithm

```
VI z(string s)
{
    int n = s.size();
    VI z(n);

    int l = 0, r = 0;
    for (int i = 0; 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]])
            l = i, r = i + z[i]++;
    }

    return z;
}
```

6 Miscellaneous

6.1 Generationg Permutations

```
VI a;
//...

sort(all(a));
do
{
    // Process permutation...
} while (next_permutation(all(a)));
```

6.2 Iterating over Submasks

```
int n = 17;

for (int mask = 0; mask < (1 << n); ++mask)
{
    for (int sub = mask; sub; sub = (sub - 1) & mask)
    {
        // Process submask...
    }
}
```

6.3 pbds

```
// ordered set
ordered_set<int> st;
// ordered multiset
ordered_set<int, less_equal<int>> multi;

// insert element
st.insert(x);
// get index
st.order_of_key(x);
// access by index
st.find_by_order(idx);
```