Competitive programming Notebook 🎈

Meia noite eu te conto

Contents

1 General

1.1 Random

```
random_device dev;
mt19937 rng(dev());

uniform_int_distribution<mt19937::result_type> dist
    (1, 6); // distribution in range [1, 6]

int val = dist(rng);
```

1.2 Split

```
vector<string> split(string s, char key=' ') {
    vector<string> ans;
    string aux = "";

    for (int i = 0; i < (int)s.size(); i++) {
        if (s[i] == key) {
            if (aux.size() > 0) {
                ans.push_back(aux);
                aux = "";
            }
        } else {
            aux += s[i];
        }
    }

    if ((int)aux.size() > 0) {
        ans.push_back(aux);
    }

    return ans;
}
```

1.3 Base Converter

```
const string digits = "0123456789
    ABCDEFGHIJKLMNOPQRSTUVWXYZ";

ll tobase10(string number, int base) {
    map<char, int> val;
    for (int i = 0; i < digits.size(); i++) {
        val[digits[i]] = i;
    }

    ll ans = 0, pot = 1;

    for (int i = number.size() - 1; i >= 0; i--) {
        ans += val[number[i]] * pot;
        pot *= base;
    }

    return ans;
}

string frombase10(ll number, int base) {
    if (number == 0) return "0";

    string ans = "";

    while (number > 0) {
        ans += digits[number % base];
        number /= base;
    }

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

    return ans;
}

// verifica se um nÃžmero estÃą na base especificada
bool verify_base(string num, int base) {
    map<char, int> val;
    for (int i = 0; i < digits.size(); i++) {
        val[digits[i]] = i;
    }

    for (auto digit : num) {
        if (val[digit] >= base) {
            return false;
        }
    }

    return true;
}
```

1.4 Template

```
// MEIA NOITE EU TE CONTO
#include <bits/stdc++.h>

using namespace std;

#define _ ios_base::sync_with_stdio(0);cin.tie(0);

typedef long long ll;

const int INF = 0x3f3f3f3f;
const ll LINF = 0x3f3f3f3f3f3f3f3fll;

int main() { _
    return 0;
}
```

2 Math

2.1 2sat

```
// 2SAT
//
// verifica se existe e encontra soluÃğÃčo
// para fÃşrmulas booleanas da forma
// (a or b) and (!a or c) and (...)
//
// indexado em 0
// n(a) = 2*x e n(~a) = 2*x+1
// a = 2 ; n(a) = 4 ; n(~a) = 5 ; n(a)^1 = 5 ; n(~a)
    ^1 = 4
//
// https://cses.fi/problemset/task/1684/
// https://codeforces.com/gym/104120/problem/E
// (add_eq, add_true, add_false e at_most_one nÃčo
    foram testadas)
//
// O(n + m)

struct sat {
    int n, tot;
    vector<vector<int>> adj, adjt; // grafo original,
     grafo transposto
    vector<int> vis, comp, ans;
    stack<int> topo; // ordem topolÃşgica

    sat() {}
    sat(int n_) : n(n_), tot(n), adj(2*n), adjt(2*n)
    {}

    void dfs(int x) {
        vis[x] = true;

        for (auto e : adj[x]) {
            if (!vis[e]) dfs(e);
        }

        topo.push(x);
    }

    void dfst(int x, int& id) {
        vis[x] = true;
        comp[x] = id;

        for (auto e : adjt[x]) {
            if (!vis[e]) dfst(e, id);
        }
    }

    void add_impl(int a, int b) { // a -> b = (!a or b)
        a = (a >= 0 ? 2*a : -2*a-1);
        b = (b >= 0 ? 2*b : -2*b-1);

        adj[a].push_back(b);
        adj[b^1].push_back(a^1);

        adjt[b].push_back(a);
        adjt[a^1].push_back(b^1);
    }

    void add_or(int a, int b) { // a or b
        add_impl(~a, b);
    }

    void add_nor(int a, int b) { // a nor b = !(a or b)
        add_or(~a, b), add_or(a, ~b), add_or(~a, ~b);
    }

    void add_and(int a, int b) { // a and b
        add_or(a, b), add_or(~a, b), add_or(a, ~b);
    }

    void add_nand(int a, int b) { // a nand b = !(a and b)
        add_or(~a, ~b);
    }

    void add_xor(int a, int b) { // a xor b = (a != b)
        add_or(a, b), add_or(~a, ~b);
    }

    void add_xnor(int a, int b) { // a xnor b = !(a xor b) = (a = b)
        add_xor(~a, b);
    }

    void add_true(int a) { // a = T
        add_or(a, ~a);
    }

    void add_false(int a) { // a = F
        add_and(a, ~a);
    }

    // magia - brunomaletta
    void add_true_old(int a) { // a = T (n sei se funciona)
        add_impl(~a, a);
    }

    void at_most_one(vector<int> v) { // no max um verdadeiro
        adj.resize(2*(tot+v.size()));
        for (int i = 0; i < v.size(); i++) {
            add_impl(tot+i, ~v[i]);
            if (i) {
                add_impl(tot+i, tot+i-1);
                add_impl(v[i], tot+i-1);
            }
        }
        tot += v.size();
    }

    pair<bool, vector<int>> solve() {
        ans.assign(n, -1);
        comp.assign(2*tot, -1);
        vis.assign(2*tot, 0);
        int id = 1;

        for (int i = 0; i < 2*tot; i++) if (!vis[i]) dfs(i);

        vis.assign(2*tot, 0);
        while (topo.size()) {
            auto x = topo.top();
            topo.pop();

            if (!vis[x]) {
                dfst(x, id);
                id++;
            }
        }

        for (int i = 0; i < tot; i++) {
            if (comp[2*i] == comp[2*i+1]) return {false, {}};
            ans[i] = (comp[2*i] > comp[2*i+1]);
        }

        return {true, ans};
    }
};
```

3 DS

3.1 Dsu

```
/*
DSU - Disjoint Set Union (or Union Find)

find(x) -> find component that x is on
join(a, b) -> union of a set containing 'a' and set containing b

find / join with path compreension -> O(inv_Ackermann(n)) [O(1)]
find / join without path compreension -> O(logN)

https://judge.yosupo.jp/submission/126864
*/

struct DSU {

    int n = 0, components = 0;
    vector<int> parent;
    vector<int> size;

    DSU(int nn){
        n = nn;
        components = n;
        size.assign(n + 5, 1);
        parent.assign(n + 5, 0);
        iota(parent.begin(), parent.end(), 0);
    }

```

```cpp
    int find(int x){
        if(x == parent[x]) {
            return x;
        }
        //path compression
        return parent[x] = find(parent[x]);
    }

    void join(int a, int b){
        a = find(a);
        b = find(b);
        if(a == b) {
            return;
        }
        if(size[a] < size[b]) {
            swap(a, b);
        }
        parent[b] = a;
        size[a] += size[b];
        components -= 1;
    }

    int sameSet(int a, int b) {
        a = find(a);
        b = find(b);
        return a == b;
    }

};
```

3.2 Ordered Set

```cpp
// Ordered Set
//
// set roubado com mais operacoes
//
// para alterar para multiset
// trocar less para less_equal
//
// ordered_set<int> s
//
// order_of_key(k) // number of items strictly smaller than k -> int
// find_by_order(k) // k-th element in a set (counting from zero) -> iterator
//
// https://cses.fi/problemset/task/2169
//
// O(log N) para insert, erase (com iterator), order_of_key, find_by_order

using namespace __gnu_pbds;
template <typename T>
using ordered_set = tree<T,null_type,less<T>, rb_tree_tag,tree_order_statistics_node_update>;
```

4 DP

4.1 Knapsack

4.2 Edit Distance

```cpp
// Edit Distance / Levenshtein Distance
//
// numero minimo de operacoes
// para transformar
// uma string em outra
//
// tamanho da matriz da dp eh |a| x |b|
// edit_distance(a.size(), b.size(), a, b)
//
// https://cses.fi/problemset/task/1639
//
// O(n^2)

int tb[MAX][MAX];

int edit_distance(int i, int j, string &a, string &b) {
    if (i == 0) return j;
    if (j == 0) return i;

    int &ans = tb[i][j];

    if (ans != -1) return ans;

    ans = min({
        edit_distance(i-1, j, a, b) + 1,
        edit_distance(i, j-1, a, b) + 1,
        edit_distance(i-1, j-1, a, b) + (a[i-1] != b[j-1])
    });

    return ans;
}
```

4.3 Lcs

```cpp
// LCS (Longest Common Subsequence)
//
// maior subsequencia comum entre duas strings
//
// tamanho da matriz da dp eh |a| x |b|
// lcs(a, b) = string da melhor resposta
// dp[a.size()][b.size()] = tamanho da melhor resposta
//
// https://atcoder.jp/contests/dp/tasks/dp_f
//
// O(n^2)

string lcs(string a, string b) {
    int n = a.size();
    int m = b.size();

    int dp[n+1][m+1];
    pair<int, int> p[n+1][m+1];

    memset(dp, 0, sizeof(dp));
    memset(p, -1, sizeof(p));

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (a[i-1] == b[j-1]) {
                dp[i][j] = dp[i-1][j-1] + 1;
                p[i][j] = {i-1, j-1};
            } else {
                if (dp[i-1][j] > dp[i][j-1]) {
                    dp[i][j] = dp[i-1][j];
                    p[i][j] = {i-1, j};
                } else {
                    dp[i][j] = dp[i][j-1];
                    p[i][j] = {i, j-1};
                }
            }
        }
    }

    // recuperar resposta

    string ans = "";
    pair<int, int> curr = {n, m};

    while (curr.first != 0 && curr.second != 0) {
        auto [i, j] = curr;

        if (a[i-1] == b[j-1]) {
            ans += a[i-1];
        }

        curr = p[i][j];
    }

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

    return ans;
}
```

5 Graph

5.1 Dfs

```
// DFS
//
// Percorre todos os vertices
// priorizando profundidade
//
// O(n+m)

vector<vector<int>> g;
vector<bool> vis;

void dfs(int s){
    if(vis[s]) return;
    vis[s] = true;
    for(auto v : g[s]){
        dfs(v);
    }
}
```