NTJ
UDESC

Eric Grochowicz, Enzo de Almeida Rodrigues e João Marcos de Oliveira

3 de maio de 2023

Índice

1 Problemas

1.1 Kth digito na string infinita de digitos

```
// Retorna qual o numero e qual o algarismo do Kth digito
// na string infinita dos numeros naturais
   (12345678910111213...)
// Complexidade: O(log_10(k))

pair<ll,ll> kthdig(ll k){
    ll qtd = 1, num_alg = 1, base = 1;
    while(1){
        ll add = (9 * base) * num_alg;
        if(qtd + add < k){
            qtd += add;
        } else break;
        base *= 10, num_alg++;
    }
    ll algarismo = (k - qtd) % num_alg;
    ll numero = (k - qtd) / num_alg + base;
    return {numero, algarismo};
}
```

2 Estruturas

2.1 Fenwick Tree

```
// Processas queries de operacao com inverso
// Build: O(n)
// Query: O(log(n))
// Update: O(log(n))

typedef long long ll;

struct fenwick {
    vector<ll> bit;
    fenwick(int n) { bit.assign(n+1, 0); }
    fenwick(vector<ll>& v) {
        int n = v.size();
        bit.assign(n+1, 0);
        for(int i = 1; i <= n; i++) bit[i] = v[i-1];
        for(int i = 1; i <= n; i++) {
            int j = i + (i & -i);
            if(j <= n) bit[j] += bit[i];
        }
    }
    ll query(int i){
        ll res = 0;
        for(; i; i -= (i & -i))
            res += bit[i];
        return res;
    }
    ll query(int l, int r){
        return query(r) - query(l-1);
    }
    void update(int i, ll d){
        for(; i && i < (int)bit.size(); i += (i & -i))
            bit[i] += d;
    }
};
```

3 Grafos

3.1 Bridges e Edge Biconnected Components

```
// Acha todas as pontes em O(n)
// Tambem constroi a arvore condensada, mantendo
// so as pontes como arestas e o resto comprimindo
// em nodos

const int maxn = 4e5;
int n, m;
bool vis[maxn];
int dp[maxn], dep[maxn];
vector<int> adj[maxn];
vector<ii> bridges;

void dfs_dp(int u, int p = -1, int d = 0){
    dp[u] = 0, dep[u] = d, vis[u] = 1;
    for(auto v : adj[u]) if(v != p) {
        if(vis[v]){
            if(dep[v] < dep[u]) dp[v]--, dp[u]++;
        } else {
            dfs_dp(v, u, d+1);
            dp[u] += dp[v];
        }
    }
    if(dp[u] == 0 && p != -1){ // edge {u, p} eh uma
       ponte
        bridges.emplace_back(u, p);
    }
}

void find_bridges(){
    memset(vis, 0, n+1);
    for(int i = 1; i <= n; i++){
        if(!vis[i]) dfs_dp(i);
    }
}

// EDGE BICONNECTED COMPONENTS (requer todo codigo acima)
int ebcc[maxn], ncc = 1;
vector<int> adjbcc[maxn];

void dfs_ebcc(int u, int p = -1, int cc = 1){
    vis[u] = 1;
    if(dp[u] == 0 && p != -1){
        cc = ++ncc;
    }
    ebcc[u] = cc;
    for(auto v : adj[u]) if(!vis[v]) {
        dfs_ebcc(v, u, cc);
    }
}

void build_ebcc_graph(){
    find_bridges();
    memset(vis, 0, n+1);
    for(int i = 1; i <= n; i++){
        if(!vis[i]) dfs_ebcc(i);
    }
    // Opcao 1 - constroi o grafo condensado passando
       por todas as edges
    for(int u = 1; u <= n; u++){
        for(auto v : adj[u]){
            if(ebcc[u] != ebcc[v]){
                adjbcc[ebcc[u]].emplace_back(ebcc[v]);
            } else {
                // faz algo
            }
        }
    }
    // Opcao 2 - constroi o grafo condensado passando so
       pelas pontes
    for(auto [u,v] : bridges){
        adjbcc[ebcc[u]].emplace_back(ebcc[v]);
        adjbcc[ebcc[v]].emplace_back(ebcc[u]);
    }
}
```

4 Extra

4.1 Config do Vim

```
// .vimrc

set number
set nohls
set ai

set belloff=all

syntax on
filetype plugin indent on

set ts=4
set sw=4
set expandtab
set noshiftround

set showmode
set showcmd

" bracket remap
inoremap {} {}<Left><Return><Up><End><Return>

" bracket translator (trocar pra C cedilha)
nnoremap c :g/{/normal kJx<return>
nnoremap C :g/{/normal $xo{<return>
```

4.2 Gerador aleatorio de inteiros em [l, r]

```
mt19937 rng(chrono::steady_clock::now()
    .time_since_epoch().count());

ll uniform(ll l, ll r){
    uniform_int_distribution<int> uid(l, r);
    return uid(rng);
}
```

4.3 Rand C++

```
mt19937 rng(chrono::steady_clock::now()
    .time_since_epoch().count());
```

4.4 Script de stress test

```
set -e
g++ code.cpp -o code
g++ brute.cpp -o brute
g++ gen.cpp -o gen

for((i = 1; ; ++i)); do
    ./gen > input_file
    ./code < input_file > myAnswer
    ./brute < input_file > correctAnswer
    diff myAnswer correctAnswer > /dev/null || break
    echo "Passed test: "  $i
done

echo "WA on the following test:"
cat input_file
echo "Your answer is:"
cat myAnswer
echo "Correct answer is:"
cat correctAnswer
```

4.5 Template C++

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

using namespace std;

void solve(){

}

signed main(){
    ios_base::sync_with_stdio(0); cin.tie(0);
```

```
    solve();
}
```

4.6 Template de debug simples

```
void _print() { }
template<typename T, typename... U> void _print(T a,
    U... b) {
     if(sizeof... (b)){
         cerr << a << ", ";
         _print(b...);
     } else cerr << a;
}
#define debug(x...) cerr << "[" << #x << "] = [",
    _print(x), cerr << "]" << endl

// #define debug(...)
```