teambrbr002
UFMG

Emanuel Silva, Felipe Mota e Kaio Vieira

14 de outubro de 2023

Índice

1 DP

1.1 Mochila

```
// Resolve mochila, recuperando a resposta
//
// O(n * cap), O(n + cap) de memoria

add int v[MAX], w[MAX]; // valor e peso
582 int dp[2][MAX_CAP];

// DP usando os itens [l, r], com capacidade = cap
0d6 void get_dp(int x, int l, int r, int cap) {
f8f     memset(dp[x], 0, (cap+1)*sizeof(dp[x][0]));
574     for (int i = l; i <= r; i++) for (int j = cap; j >= 0; j--)
3a9         if (j - w[i] >= 0) dp[x][j] = max(dp[x][j], v[i] + dp[x][j
    - w[i]]);
b95 }

5ab void solve(vector<int>& ans, int l, int r, int cap) {
```

```
893     if (l == r) {
9ff         if (w[l] <= cap) ans.push_back(l);
505         return;
13a     }
ee4     int m = (l+r)/2;
283     get_dp(0, l, m, cap), get_dp(1, m+1, r, cap);
056     int left_cap = -1, opt = -INF;
c94     for (int j = 0; j <= cap; j++)
2f2         if (int at = dp[0][j] + dp[1][cap - j]; at > opt)
91d             opt = at, left_cap = j;
da3     solve(ans, l, m, left_cap), solve(ans, m+1, r, cap - left_cap);
d75 }

0d7 vector<int> knapsack(int n, int cap) {
dab     vector<int> ans;
1e0     solve(ans, 0, n-1, cap);
ba7     return ans;
e4d }
```

2 Strings

2.1 Z

```
// z[i] = lcp(s, s[i..n))
//
// Complexidades:
// z - O(|s|)
// match - O(|s| + |p|)

a19 vector<int> get_z(string s) {
163     int n = s.size();
2b1     vector<int> z(n, 0);

fae     int l = 0, r = 0;
6f5     for (int i = 1; i < n; i++) {
0af         if (i <= r) z[i] = min(r - i + 1, z[i - l]);
457         while (i + z[i] < n and s[z[i]] == s[i + z[i]]) z[i]++;
65e         if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
5cd     }

070     return z;
74a }
```

3 Matematica

3.1 Totiente

```
// O(sqrt(n))

a7e int tot(int n){
0f6     int ret = n;

505     for (int i = 2; i*i <= n; i++) if (n % i == 0) {
b0c         while (n % i == 0) n /= i;
125         ret -= ret / i;
34a     }
af4     if (n > 1) ret -= ret / n;

edf     return ret;
fae }
```

4 Grafos

4.1 Virtual Tree

```
// Comprime uma arvore dado um conjunto S de vertices, de forma que
// o conjunto de vertices da arvore comprimida contenha S e seja
// minimal e fechado sobre a operacao de LCA
// Se |S| = k, a arvore comprimida tem menos que 2k vertices
// As arestas de virt possuem a distancia do vertice ate o vizinho
// Retorna a raiz da virtual tree
//
// lca::pos deve ser a ordem de visitacao no dfs
// voce pode usar o LCAcomHLD, por exemplo
//
// O(k log(k))

b36 vector<pair<int, int>> virt[MAX];

d41 #warning lembrar de buildar o LCA antes
c14 int build_virt(vector<int> v) {
b46     auto cmp = [&](int i, int j) { return lca::pos[i] <
    lca::pos[j]; };
074     sort(v.begin(), v.end(), cmp);
e85     for (int i = v.size()-1; i; i--) v.push_back(lca::lca(v[i],
    v[i-1]));
```

```
074     sort(v.begin(), v.end(), cmp);
d76     v.erase(unique(v.begin(), v.end()), v.end());
37c     for (int i = 0; i < v.size(); i++) virt[v[i]].clear();
197     for (int i = 1; i < v.size(); i++) virt[lca::lca(v[i-1],
    v[i])].clear();
ad7     for (int i = 1; i < v.size(); i++) {
51b         int parent = lca::lca(v[i-1], v[i]);
290         int d = lca::dist(parent, v[i]);
d41 #warning soh to colocando aresta descendo
4d0         virt[parent].emplace_back(v[i], d);
fe5     }
832     return v[0];
142 }
```

5 Primitivas

5.1 Aritmetica Modular

```
// O mod tem q ser primo

429 template<int p> struct mod_int {
c68     ll expo(ll b, ll e) {
c85         ll ret = 1;
c87         while (e) {
cad             if (e % 2) ret = ret * b % p;
9d2             e /= 2, b = b * b % p;
c42         }
edf         return ret;
734     }
1f6     ll inv(ll b) { return expo(b, p-2); }

4d7     using m = mod_int;
d93     int v;
fe0     mod_int() : v(0) {}
e12     mod_int(ll v_) {
019         if (v_ >= p or v_ <= -p) v_ %= p;
bc6         if (v_ < 0) v_ += p;
2e7         v = v_;
7f3     }
74d     m& operator +=(const m& a) {
2fd         v += a.v;
ba5         if (v >= p) v -= p;
357         return *this;
c8b     }
eff     m& operator -=(const m& a) {
8b4         v -= a.v;
cc8         if (v < 0) v += p;
357         return *this;
f8d     }
4c4     m& operator *=(const m& a) {
8a5         v = v * ll(a.v) % p;
357         return *this;
d4c     }
3f9     m& operator /=(const m& a) {
5d6         v = v * inv(a.v) % p;
357         return *this;
62d     }
d65     m operator -(){ return m(-v); }
b3e     m& operator ^=(ll e) {
06d         if (e < 0) {
6e2             v = inv(v);
00c             e = -e;
275         }
94a         v = expo(v, e%(p-1));
357         return *this;
ba3     }
423     bool operator ==(const m& a) { return v == a.v; }
69f     bool operator !=(const m& a) { return v != a.v; }

1c6     friend istream& operator >>(istream& in, m& a) {
d1c         ll val; in >> val;
d48         a = m(val);
091         return in;
870     }
44f     friend ostream& operator <<(ostream& out, m a) {
5a0         return out << a.v;
214     }
399     friend m operator +(m a, m b) { return a += b; }
f9e     friend m operator -(m a, m b) { return a -= b; }
9c1     friend m operator *(m a, m b) { return a *= b; }
51b     friend m operator /(m a, m b) { return a /= b; }
08f     friend m operator ^(m a, ll e) { return a ^= e; }
1af };

055 typedef mod_int<(int)1e9+7> mint;
```

6 Estruturas

6.1 Wavelet Tree

```
// Usa O(sigma + n log(sigma)) de memoria,
// onde sigma = MAXN - MINN
// Depois do build, o v fica ordenado
// count(i, j, x, y) retorna o numero de elementos de
// v[i, j) que pertencem a [x, y]
// kth(i, j, k) retorna o elemento que estaria
// na poscicao k-1 de v[i, j), se ele fosse ordenado
// sum(i, j, x, y) retorna a soma dos elementos de
// v[i, j) que pertencem a [x, y]
// sumk(i, j, k) retorna a soma dos k-esimos menores
// elementos de v[i, j) (sum(i, j, 1) retorna o menor)
//
// Complexidades:
// build - O(n log(sigma))
// count - O(log(sigma))
// kth   - O(log(sigma))
// sum   - O(log(sigma))
// sumk  - O(log(sigma))

597 int n, v[MAX];
578 vector<int> esq[4*(MAXN-MINN)], pref[4*(MAXN-MINN)];

f8d void build(int b = 0, int e = n, int p = 1, int l = MINN, int r =
    MAXN) {
58f     int m = (l+r)/2; esq[p].push_back(0); pref[p].push_back(0);
f2f     for (int i = b; i < e; i++) {
6b9         esq[p].push_back(esq[p].back()+(v[i]<=m));
26f         pref[p].push_back(pref[p].back()+v[i]);
206     }
8ce     if (l == r) return;
3a7     int m2 = stable_partition(v+b, v+e, [=](int i){return i <=
    m;}) - v;
347     build(b, m2, 2*p, l, m), build(m2, e, 2*p+1, m+1, r);
0fb }

540 int count(int i, int j, int x, int y, int p = 1, int l = MINN, int
    r = MAXN) {
2ad     if (y < l or r < x) return 0;
4db     if (x <= l and r <= y) return j-i;
ddc     int m = (l+r)/2, ei = esq[p][i], ej = esq[p][j];
0a5     return count(ei, ej, x, y, 2*p, l, m)+count(i-ei, j-ej, x, y,
    2*p+1, m+1, r);
3cf }

f62 int kth(int i, int j, int k, int p=1, int l = MINN, int r = MAXN) {
3ce     if (l == r) return l;
ddc     int m = (l+r)/2, ei = esq[p][i], ej = esq[p][j];
585     if (k <= ej-ei) return kth(ei, ej, k, 2*p, l, m);
28b     return kth(i-ei, j-ej, k-(ej-ei), 2*p+1, m+1, r);
8b6 }

f2c int sum(int i, int j, int x, int y, int p = 1, int l = MINN, int r
    = MAXN) {
2ad     if (y < l or r < x) return 0;
2a9     if (x <= l and r <= y) return pref[p][j]-pref[p][i];
ddc     int m = (l+r)/2, ei = esq[p][i], ej = esq[p][j];
43b     return sum(ei, ej, x, y, 2*p, l, m) + sum(i-ei, j-ej, x, y,
    2*p+1, m+1, r);
b6d }

b84 int sumk(int i, int j, int k, int p = 1, int l = MINN, int r =
    MAXN) {
8a1     if (l == r) return l*k;
ddc     int m = (l+r)/2, ei = esq[p][i], ej = esq[p][j];
50c     if (k <= ej-ei) return sumk(ei, ej, k, 2*p, l, m);
4c9     return pref[2*p][ej]-pref[2*p][ei]+sumk(i-ei, j-ej, k-(ej-ei),
    2*p+1, m+1, r);
940 }
```

7 Problemas

7.1 Sweep Direction

```
// Passa por todas as ordenacoes dos pontos definitas por "direcoes"
// Assume que nao existem pontos coincidentes
//
// O(n^2 log n)

4b8 void sweep_direction(vector<pt> v) {
3d2     int n = v.size();
163     sort(v.begin(), v.end(), [](pt a, pt b) {
3a5         if (a.x != b.x) return a.x < b.x;
572         return a.y > b.y;
79a     });
b89     vector<int> at(n);
516     iota(at.begin(), at.end(), 0);
```

```
b79     vector<pair<int, int>> swapp;
25e     for (int i = 0; i < n; i++) for (int j = i+1; j < n; j++)
95f         swapp.push_back({i, j}), swapp.push_back({j, i});

269     sort(swapp.begin(), swapp.end(), [&](auto a, auto b) {
134         pt A = rotate90(v[a.first] - v[a.second]);
247         pt B = rotate90(v[b.first] - v[b.second]);
615         if (quad(A) == quad(B) and !sarea2(pt(0, 0), A, B)) return
    a < b;
224         return compare_angle(A, B);
5e7     });
4e6     for (auto par : swapp) {
e24         assert(abs(at[par.first] - at[par.second]) == 1);
a96         int l = min(at[par.first], at[par.second]),
0d3             r = n-1 - max(at[par.first], at[par.second]);
            // l e r sao quantos caras tem de cada lado do par de
               pontos
            // (cada par eh visitado duas vezes)
9cf         swap(v[at[par.first]], v[at[par.second]]);
1c0         swap(at[par.first], at[par.second]);
241     }
6bb }
```

8 Extra

8.1 makefile

```
CXX = g++
CXXFLAGS = -fsanitize=address,undefined -fno-omit-frame-pointer -g
    -Wall -Wshadow -std=c++17 -Wno-unused-result -Wno-sign-compare
    -Wno-char-subscripts #-fuse-ld=gold
```

8.2 stress.sh

```
P=a
make ${P} ${P}2 gen || exit 1
for ((i = 1; ; i++)) do
    ./gen $i > in
    ./${P} < in > out
    ./${P}2 < in > out2
    if (! cmp -s out out2) then
        echo "--> entrada:"
        cat in
        echo "--> saida1:"
        cat out
        echo "--> saida2:"
        cat out2
        break;
    fi
    echo $i
done
```

8.3 pragma.cpp

```
// Otimizacoes agressivas, pode deixar mais rapido ou mais devagar
#pragma GCC optimize("Ofast")
// Auto explicativo
#pragma GCC optimize("unroll-loops")
// Vetorizacao
#pragma GCC target("avx2")
// Para operacoes com bits
#pragma GCC target("bmi,bmi2,popcnt,lzcnt")
```

8.4 template.cpp

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

using namespace std;

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

#define int             long long int
#define double          long double
#define endl            "\n"
#define print_v(a)      for(auto x : a) cout << x << " "; cout << endl
#define f(i,s,e)        for(int i=s;i<e;i++)
#define rf(i,e,s)       for(int i=e-1;i>=s;i--)

#define dbg(x) cout << #x << " = " << x << endl;

int a, b;

void solve() {

}

int32_t main() { _

    clock_t z = clock();

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


    cerr << fixed << "Run Time : " << ((double)(clock() - z) /
       CLOCKS_PER_SEC) << endl;
    return 0;
}
```

8.5 rand.cpp

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

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