Competitive Programming Notebook

Programadores Roblox

Contents

1 DP

1.1 Lis

1.2 Lcs

1.3 Knapsack

```cpp
// dp[i][j] => i-esimo item com j-carga sobrando na mochila
// O(N * W)

for(int j = 0; j < MAXN; j++) {
    dp[0][j] = 0;
}
for(int i = 1; i <= N; i++) {
    for(int j = 0; j <= W; j++) {
        if(items[i].first > j) {
            dp[i][j] = dp[i-1][j];
        }
        else {
            dp[i][j] = max(dp[i-1][j], dp[i-1][j-items[i].first] + items[i].second);
        }
    }
}
```

2 String

3 Geometry

4 Graph

4.1 Dijkstra

```cpp
// SSP com pesos positivos.
// O((V + E) log V).

vector<int> dijkstra(int S) {
    vector<bool> vis(MAXN, 0);
    vector<ll> dist(MAXN, LLONG_MAX);
    dist[S] = 0;
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({0, S});
    while(pq.size()) {
        ll v = pq.top().second;
        pq.pop();
        if(vis[v]) continue;
        vis[v] = 1;
        for(auto &[peso, vizinho] : adj[v]) {
            if(dist[vizinho] > dist[v] + peso) {
                dist[vizinho] = dist[v] + peso;
                pq.push({dist[vizinho], vizinho});
            }
        }
    }
    return dist;
}
```

5 Math

5.1 Exgcd

```cpp
// O retorno da funÃğÃčo Ãľ {n, m, g}
// e significa que gcd(a, b) = g e
// n e m sÃčo inteiros tais que an + bm = g
array<ll, 3> exgcd(int a, int b) {
    if(b == 0) return {1, 0, a};
    auto [m, n, g] = exgcd(b, a % b);
    return {n, m - a / b * n, g};
}
```

5.2 Fexp

```cpp
// a^e mod m
// O(log n)

ll fexp(ll a, ll e, ll m) {
    a %= m;
    ll ans = 1;
    while (e > 0){
        if (e & 1) ans = ansa % m;
        a = aa % m;
        e /= 2;
    }
    return ans%m;
}
```

5.3 Equacao Diofantina

```cpp
// resolve equacao ax + by = c
// retorno {existe sol., x, y, g}
array<ll, 4> find_any_solution(ll a, ll b, ll c) {
    auto[x, y, g] = exgcd(a, b);
    if (c % g) return {false, 0, 0, 0};
    x *= c / g;
    y *= c / g;
    return {true, x, y, g};
}
```

6 DS

7 Primitives

8 General

8.1 Bitwise

```cpp
int check_kth_bit(int x, int k) {
  return (x >> k) & 1;
}

void print_on_bits(int x) {
  for (int k = 0; k < 32; k++) {
    if (check_kth_bit(x, k)) {
      cout << k << ' ';
    }
  }
  cout << '\n';
}

int count_on_bits(int x) {
  int ans = 0;
  for (int k = 0; k < 32; k++) {
    if (check_kth_bit(x, k)) {
      ans++;
    }
  }
  return ans;
}

bool is_even(int x) {
  return ((x & 1) == 0);
}

int set_kth_bit(int x, int k) {
  return x | (1 << k);
}

int unset_kth_bit(int x, int k) {
  return x & (~(1 << k));
}

int toggle_kth_bit(int x, int k) {
  return x ^ (1 << k);
}

bool check_power_of_2(int x) {
  return count_on_bits(x) == 1;
}
```