Caderno de Programação Competitiva

Capangas do Ribas

Sumário

1 Algoritmos

1.1 DP

1.2 Estruturas de Dados

1.3 General

1.4 Geometry

1.5 Grafos

1.5.1 DFS

```cpp
const int MAX { 200010 };

bitset<MAX> visited;
vector<int> adj[MAX];

void dfs(int u)
{
    if (visited[u])
        return;

    // processa/visita u

    visited[u] = true;

    for (auto v : adj[u])
        dfs(v);
}
```

1.5.2 BFS

```cpp
vector<int> bfs(int s, int N) {
    vector<int> dist(N + 1, oo);
    queue<int> q;

    dist[s] = 0; q.push(s);

    while (not q.empty())
    {
        auto u = q.front(); q.pop();

        // visita/processa u

        for (auto v : adj[u]) {
            if (dist[v] == oo) {
                dist[v] = dist[u] + 1; q.push(v);
            }
        }
    }

    return dist;
}
```

1.5.3 Dijkstra

```cpp
using ii = pair<int, int>;
using edge = tuple<int, int, int>;

const int MAX { 100010 };
vector<ii> adj[MAX];

vector<int> dijkstra(int s, int N)
{
    const int oo { 1000000010 };

    vector<int> dist(N + 1, oo);
    dist[s] = 0;

    set<ii> U;
    U.emplace(0, s);

    while (not U.empty())
    {
        auto [d, u] = *U.begin();
        U.erase(U.begin());

        for (auto [v, w] : adj[u])
        {
            if (dist[v] > d + w)
            {
                if (U.count(ii(dist[v], v)))
                    U.erase(ii(dist[v], v));

                dist[v] = d + w;
                U.emplace(dist[v], v);
            }
        }
    }

    return dist;
}
```

1.6 Math

1.7 Primitives

1.8 String