

Caderno de Programação Competitiva

Capangas do Ribas

Sumário

1	Algoritmos	2
1.1	DP	2
1.2	Estruturas de Dados	2
1.3	General	2
1.4	Geometry	2
1.5	Grafos	2
	1.5.1 DFS	2
	1.5.2 BFS	2
	1.5.3 Dijkstra	2
1.6	Math	2
	1.6.1 Combinações	2
1.7	Primitives	2
1.8	String	2

1 Algoritmos

1.1 DP

1.2 Estruturas de Dados

1.3 General

1.4 Geometry

1.5 Grafos

1.5.1 DFS

```

1  const int MAX { 200010 };
2
3  bitset<MAX> visited;
4  vector<int> adj[MAX];
5
6  void dfs(int u)
7  {
8      if (visited[u])
9          return;
10
11     // processa/visita u
12
13     visited[u] = true;
14
15     for (auto v : adj[u])
16         dfs(v);
17 }

```

1.5.2 BFS

```

1  vector<int> bfs(int s, int N) {
2      vector<int> dist(N + 1, oo);
3      queue<int> q;
4
5      dist[s] = 0; q.push(s);
6
7      while (not q.empty())
8      {
9          auto u = q.front(); q.pop();
10
11         // visita/processa u
12
13         for (auto v : adj[u]) {
14             if (dist[v] == oo) {
15                 dist[v] = dist[u] + 1; q.push(v);
16             }
17         }
18     }
19
20     return dist;
21 }

```

1.5.3 Dijkstra

```

1  using ii = pair<int, int>;
2  using edge = tuple<int, int, int>;

```

```

3
4  const int MAX { 100010 };
5  vector<ii> adj[MAX];
6
7  vector<int> dijkstra(int s, int N)
8  {
9      const int oo { 1000000010 };
10
11     vector<int> dist(N + 1, oo);
12     dist[s] = 0;
13
14     set<ii> U;
15     U.emplace(0, s);
16
17     while (not U.empty())
18     {
19         auto [d, u] = *U.begin();
20         U.erase(U.begin());
21
22         for (auto [v, w] : adj[u])
23         {
24             if (dist[v] > d + w)
25             {
26                 if (U.count(ii(dist[v], v)))
27                     U.erase(ii(dist[v], v));
28
29                 dist[v] = d + w;
30                 U.emplace(dist[v], v);
31             }
32         }
33     }
34
35     return dist;
36 }

```

1.6 Math

1.6.1 Combinações

```

1  using ll = long long;
2
3  ll binom(int n, int m)
4  {
5      if (m > n)
6          return 0;
7
8      vector<ll> dp(m + 1, 0);
9      dp[0] = 1;
10
11     for (int i = 1; i <= n; ++i)
12         for (int j = m; j > 0; --j)
13             dp[j] = dp[j] + dp[j - 1];
14
15     return dp[m];
16 }

```

1.7 Primitives

1.8 String