



# Notebook - Maratona de Programação

Cabo HDMI, VGA, USB

## Contents

|          |                      |          |
|----------|----------------------|----------|
| <b>1</b> | <b>Graph</b>         | <b>2</b> |
| 1.1      | BFS . . . . .        | 2        |
| 1.2      | DFS . . . . .        | 2        |
| <b>2</b> | <b>String</b>        | <b>2</b> |
| 2.1      | Z function . . . . . | 2        |

# 1 Graph

## 1.1 BFS

Complexity:  $O(N + M)$ , N: Vertices, M: Arestas  
Uma bfs simples

```
1 // TITLE: BFS
2 // COMPLEXITY:  $O(N + M)$ , N: Vertices, M: Arestas
3 // DESCRIPTION: Uma bfs simples
4
5 vector<int> adj[MAX];
6 bool visited[MAX];
7
8 void bfs(int start)
9 {
10     queue<int> q;
11     q.push(start);
12
13     while(not q.empty()){
14         auto a = q.top();
15         q.pop();
16         if (visited[a]) continue;
17         visited[a] = true;
18         for (auto b: adj[a]){
19             q.push(b);
20         }
21     }
22 }
23
24 void solve(){
25 // Alguma coisa
26 }
```

## 1.2 DFS

Complexity:  $O(N + M)$ , N: Vertices, M: Arestas  
Uma dfs simples

```
1 // TITLE: DFS
2 // COMPLEXITY:  $O(N + M)$ , N: Vertices, M: Arestas
3 // DESCRIPTION: Uma dfs simples
4
5 vector<int> adj[MAX];
6 void dfs(int a)
7 {
8     if (visited[a]) return;
9     visited[a] = true;
10
11     for (auto b: adj[a]) {
12         dfs(b);
13     }
14 }
15
16 void solve(){
17 // Alguma coisa
18 }
```

# 2 String

## 2.1 Z function

Complexity: Z function complexity  
z function

```
1 // TITLE: Z function
2 // COMPLEXITY: Z function complexity
3 // DESCRIPTION: z function
4
5 void z_function(string& s)
6 {
7     return;
8 }
```