

# Grafos

*BFS 0/1*

**Prof. Edson Alves**

***Faculdade UnB Gama***

## **Características da BFS 0/1**

## Características da BFS 0/1

- ★ Especialização do algoritmo de Dijkstra

## Características da BFS 0/1

- ★ Especialização do algoritmo de Dijkstra
- ★ Aplicável em grafos cujos pesos das arestas são iguais ou a 0 ou a  $x$

## Características da BFS 0/1

- ★ Especialização do algoritmo de Dijkstra
- ★ Aplicável em grafos cujos pesos das arestas são iguais ou a 0 ou a  $x$
- ★ O nome do algoritmo provém do caso  $x = 1$

## Características da BFS 0/1

- ★ Especialização do algoritmo de Dijkstra
- ★ Aplicável em grafos cujos pesos das arestas são iguais ou a 0 ou a  $x$
- ★ O nome do algoritmo provém do caso  $x = 1$
- ★ Complexidade:  $O(V + E)$

**Fila x fila com prioridades**

## Fila x fila com prioridades

★ O algoritmo de Dijkstra usa uma fila com prioridades para identificar o vértice  $u$  mais próximo de  $s$  ainda não processado



## Fila x fila com prioridades

- ★ O algoritmo de Dijkstra usa uma fila com prioridades para identificar o vértice  $u$  mais próximo de  $s$  ainda não processado
- ★ Com a restrição dos pesos  $w_i$  ao conjunto  $\{0, 1\}$ , o relaxamento terá apenas duas opções

## Fila x fila com prioridades

- ★ O algoritmo de Dijkstra usa uma fila com prioridades para identificar o vértice  $u$  mais próximo de  $s$  ainda não processado
- ★ Com a restrição dos pesos  $w_i$  ao conjunto  $\{0, 1\}$ , o relaxamento terá apenas duas opções
- ★ Se  $(u, v)$  tem peso  $w$  e  $\text{dist}(s, v) > \text{dist}(s, u) + w$ , então após o relaxamento ou  $\text{dist}(s, v) = \text{dist}(s, u)$  ou  $\text{dist}(s, v) = \text{dist}(s, u) + 1$

## Fila x fila com prioridades

- ★ O algoritmo de Dijkstra usa uma fila com prioridades para identificar o vértice  $u$  mais próximo de  $s$  ainda não processado
- ★ Com a restrição dos pesos  $w_i$  ao conjunto  $\{0, 1\}$ , o relaxamento terá apenas duas opções
  - ★ Se  $(u, v)$  tem peso  $w$  e  $\text{dist}(s, v) > \text{dist}(s, u) + w$ , então após o relaxamento ou  $\text{dist}(s, v) = \text{dist}(s, u)$  ou  $\text{dist}(s, v) = \text{dist}(s, u) + 1$
- ★ Deste modo, a fila de prioridades pode ser substituída por uma fila simples

# Pseudocódigo

## Pseudocódigo

**Entrada:** um grafo  $G(V, E)$  cujos pesos  $w_i \in \{0, x\}$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

## Pseudocódigo

**Entrada:** um grafo  $G(V, E)$  cujos pesos  $w_i \in \{0, x\}$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

1. Faça  $d[s] = 0$ ,  $d[u] = \infty$  se  $u \neq s$  e seja  $Q = \{s\}$  a fila

# Pseudocódigo

**Entrada:** um grafo  $G(V, E)$  cujos pesos  $w_i \in \{0, x\}$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

1. Faça  $d[s] = 0$ ,  $d[u] = \infty$  se  $u \neq s$  e seja  $Q = \{s\}$  a fila
2. Enquanto  $Q \neq \emptyset$ :
  - (a) Seja  $u$  o primeiro elemento de  $Q$
  - (b) Se  $(u, v)$  torna  $d[v] = d[u]$ , insira  $v$  na primeira posição de  $Q$
  - (c) Se  $(u, v)$  torna  $d[v] = d[u] + 1$ , insira  $v$  na última posição de  $Q$
  - (d) Remova  $u$  de  $Q$

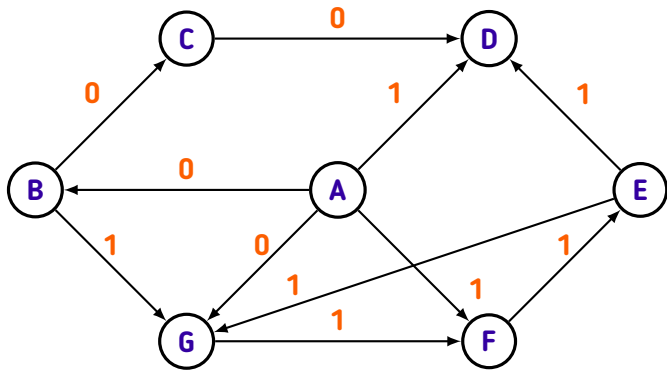
# Pseudocódigo

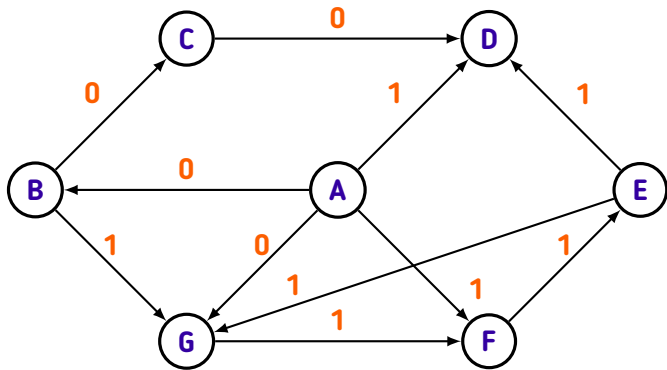
**Entrada:** um grafo  $G(V, E)$  cujos pesos  $w_i \in \{0, x\}$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

1. Faça  $d[s] = 0$ ,  $d[u] = \infty$  se  $u \neq s$  e seja  $Q = \{s\}$  a fila
2. Enquanto  $Q \neq \emptyset$ :
  - (a) Seja  $u$  o primeiro elemento de  $Q$
  - (b) Se  $(u, v)$  torna  $d[v] = d[u]$ , insira  $v$  na primeira posição de  $Q$
  - (c) Se  $(u, v)$  torna  $d[v] = d[u] + 1$ , insira  $v$  na última posição de  $Q$
  - (d) Remova  $u$  de  $Q$
3. Retorne  $d$



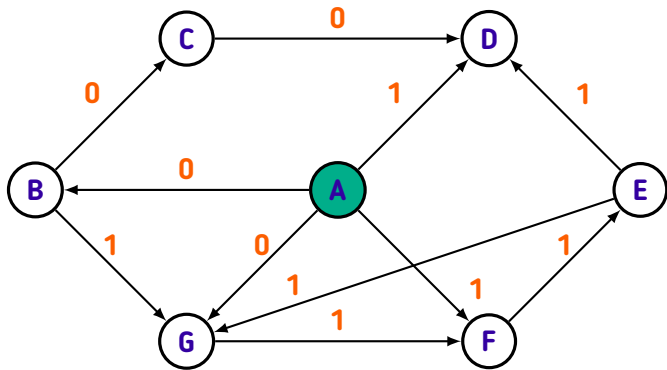




$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

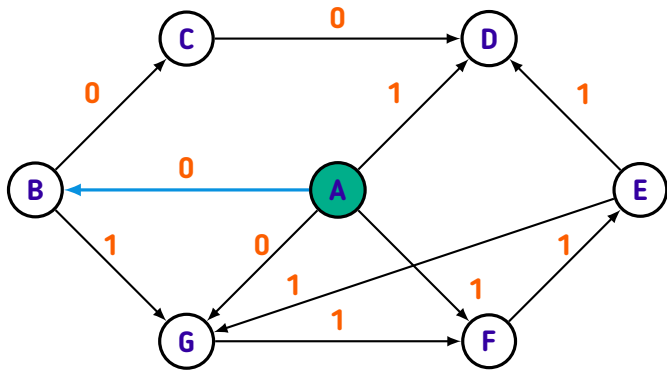
$$Q = \{ \mathbf{A} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

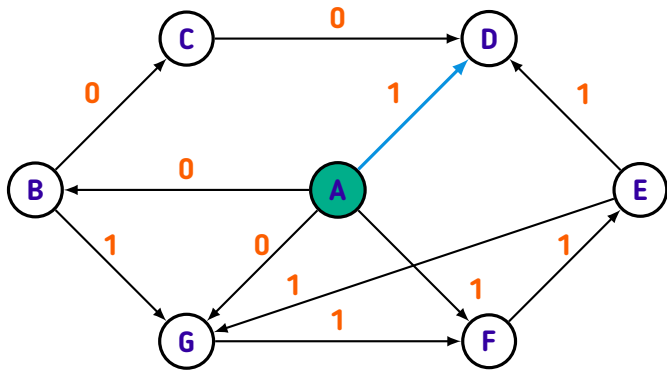
$$Q = \{ \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

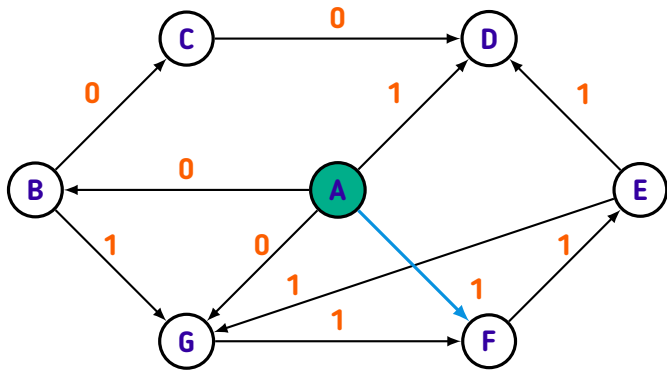
$$Q = \{ \mathbf{B} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	1	$\infty$	$\infty$	$\infty$

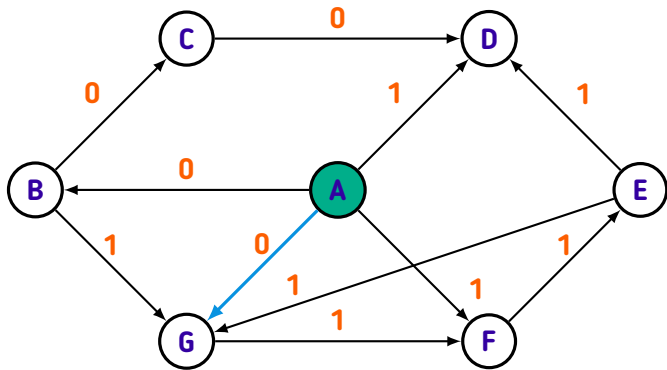
$$Q = \{ \mathbf{B}, \mathbf{D} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	1	$\infty$	1	$\infty$

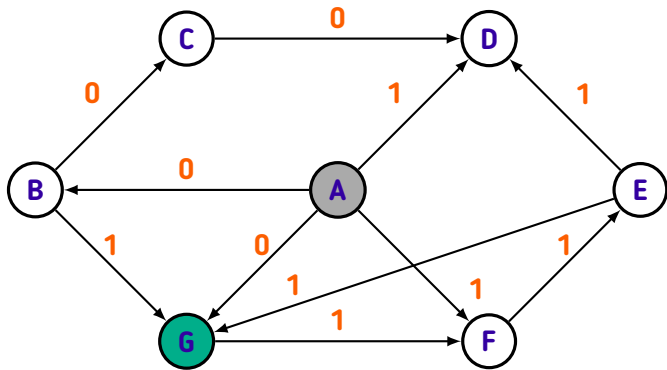
$$Q = \{ \mathbf{B}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	1	$\infty$	1	0

$$Q = \{ \mathbf{G}, \mathbf{B}, \mathbf{D}, \mathbf{F} \}$$

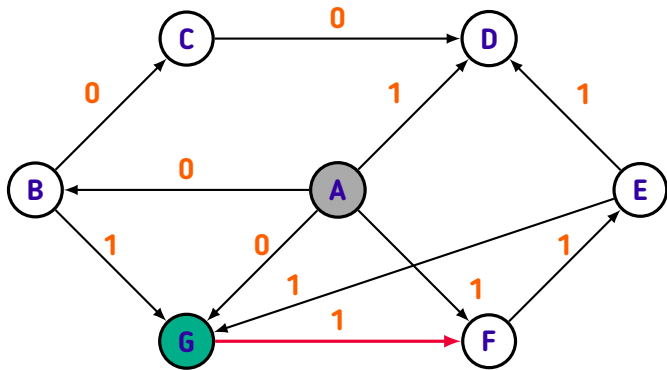


$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	1	$\infty$	1	0

$$Q = \{ \mathbf{B}, \mathbf{D}, \mathbf{F} \}$$

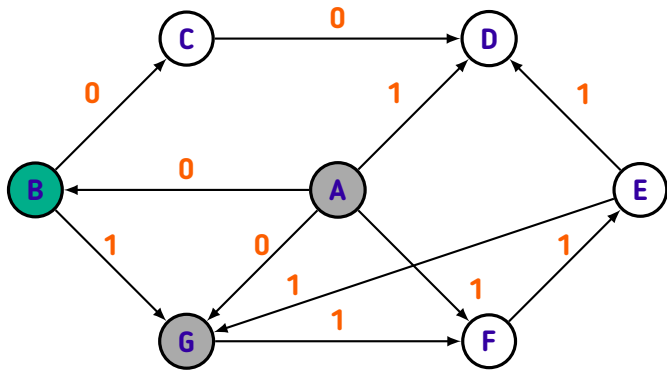




$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	1	$\infty$	1	0

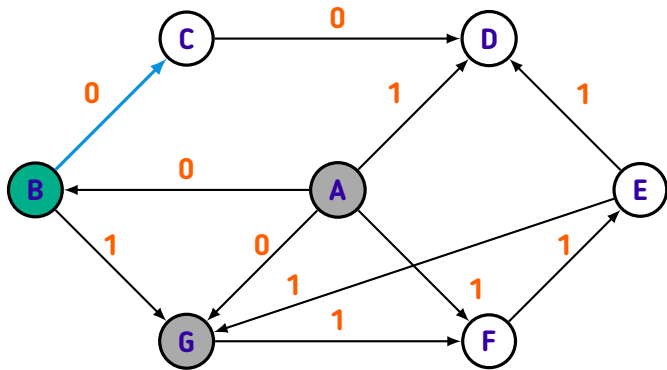
$$Q = \{ \mathbf{B}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	$\infty$	1	$\infty$	1	0

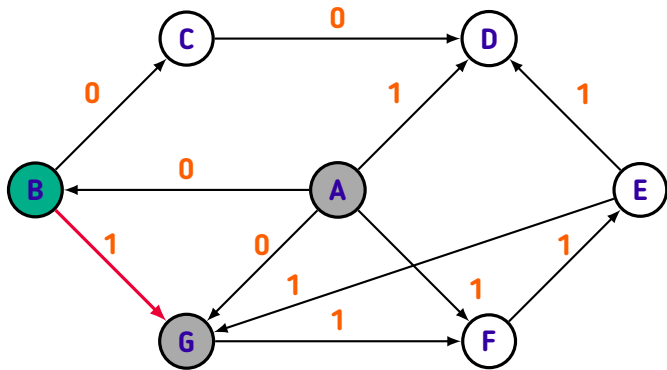
$$Q = \{ \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	1	$\infty$	1	0

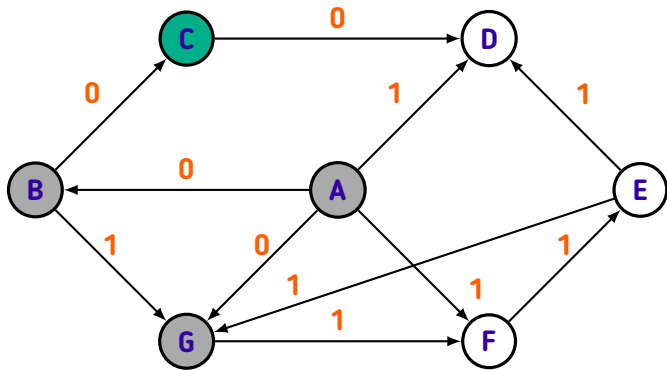
$$Q = \{ \mathbf{C}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	1	$\infty$	1	0

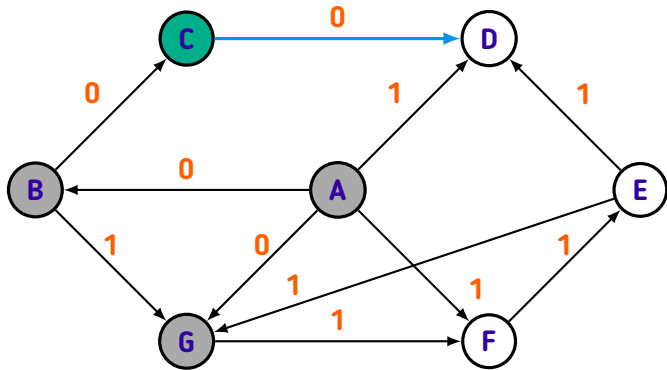
$$Q = \{ \mathbf{C}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	1	$\infty$	1	0

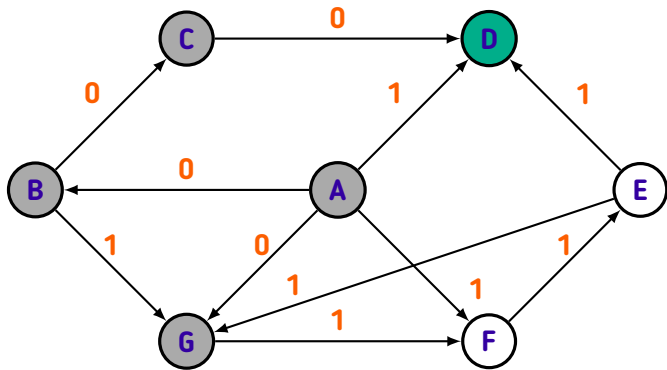
$$Q = \{ \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	$\infty$	1	0

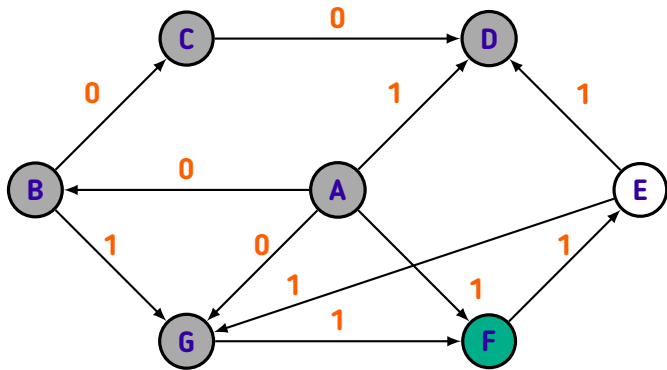
$$Q = \{ \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	$\infty$	1	0

$$Q = \{\mathbf{F}\}$$

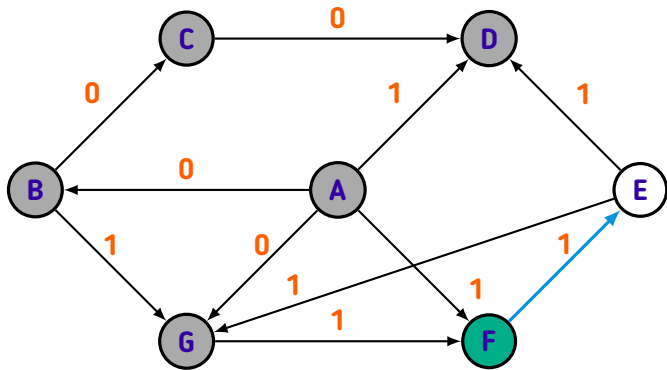


$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	$\infty$	1	0

$Q = \{ \}$

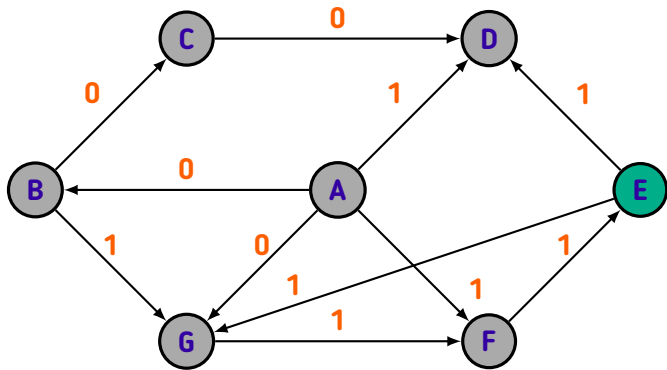




$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	2	1	0

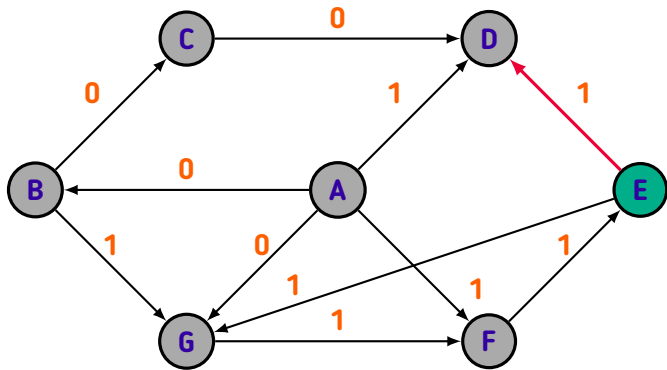
$$Q = \{ \mathbf{E} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	2	1	0

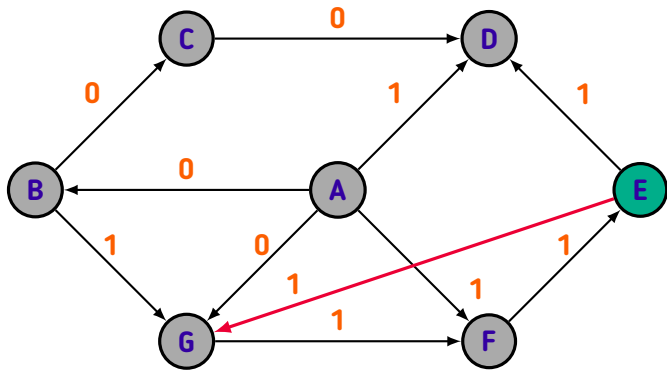
$Q = \{ \}$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	2	1	0

$Q = \{ \}$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F	G
0	0	0	0	2	1	0

$Q = \{ \}$

```
vector<int> bfs_01(int s, int N) {  
    vector<int> dist(N + 1, oo);  
    dist[s] = 0;  
  
    deque<int> q;  
    q.emplace_back(s);  
  
    while (not q.empty()) {  
        auto u = q.front();  
        q.pop_front();  
  
        for (auto [v, w] : adj[u])  
            if (dist[v] > dist[u] + w) {  
                dist[v] = dist[u] + w;  
                w == 0 ? q.emplace_front(v) : q.emplace_back(v);  
            }  
    }  
  
    return dist;  
}
```

## Problemas sugeridos

1. [AtCoder Beginner Contest 176 – Problem D: Wizard in Maze](#)
2. [Codeforces Round #516 \(Div. 1\) – Problem B: Labyrinth](#)
3. [OJ 11573 – Ocean Currents](#)
4. [SPOJ KATHTHI – KATHTHI](#)

## Referências

1. Codeforces, *0-1 BFS [Tutorial]*. himanshujaju's blog, acesso em 19/07/2021.
2. CP-Algorithms, *0-1 BFS*. Acesso em 19/07/2021.
3. HALIM, Felix; HALIM, Steve. *Competitive Programming 3*, 2010.
4. LAAKSONEN, Antti. *Competitive Programmer's Handbook*, 2018.
5. SKIENA, Steven; REVILLA, Miguel. *Programming Challenges*, 2003.