UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

Algoritmos em Grafo
Busca em Largura

Prof. Dr. Luis Augusto Martins Pereira

Busca em largura

- **Problema**: dado um grafo **G(V, E)** (com ou sem ciclo, dirigido ou não) e um vértice fonte **s** em **V**, encontre o número de arestas entre **s** e qualquer outro vértice **t** em **V** se existir um caminho entre **s** e **t**.

- Algoritmo da Busca em Largura ou BFS (Breadth First Search):
  - **Entrada: G(V, E)** e um vértice **s** em **V;**
  - O algoritmo percorre sistematicamente as arestas de **G**, descobrindo todos os vértices atingíveis a partir de **s**;
  - **Saída:** a distância em número de arestas entre **s** e qualquer outro vértice atingível em **G** a partir de **s**.

Controle do processo da busca

- **Fila** (Fist in, First out): guardar os nós visitados
- **Atributos dos nós:**
  - Cor :
    - BRANCO: não visitado;
    - CINZA: visitado;
    - PRETO: visitado e fora da fila. Se um vértice é preto, os seus vizinhos são necessariamente cinza.
  - Distância (dist): distância, em arestas, do nó fonte **s** até um nó **t**;
  - Predecessor (pred): vértice antecessor ao vértice **t** no caminho de **s** a **t**.

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO
4.     v.pred = NIL
5.     v.dist = INF
6.  Endfor

7.  s.cor = CINZA
8.  s.dist = 0
9.  s.pred = NIL
10. Q.insere(s)

12. While |Q| != vazio:
13.       u = Q.remove()
14.       For each v in Adj[u]:
15.           If v.cor == BRANCO:
16.               v.cor = CINZA
17.               v.dist = u.dist + 1
18.               v.pred = u
19.               Q.insere(v)
20.           Endif
21.       Endfor
22.       u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA
8.  s.dist = 0
9.  s.pred = NIL
10. Q.insere(s)

12. While |Q| != vazio:
13.       u = Q.remove()
14.       For each v in Adj[u]:
15.           If v.cor == BRANCO:
16.               v.cor = CINZA
17.               v.dist = u.dist + 1
18.               v.pred = u
19.               Q.insere(v)
20.           Endif
21.       Endfor
22.       u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA // Pinte o nó "s" de CINZA (primeiro nó visitado)
8.  s.dist = 0    // Distância de "s" para "s" é zero
9.  s.pred = NIL  // Predecessor de "s" é NIL
10. Q.insere(s)   // Insira "s" na fila

12. While |Q| != vazio:
13.      u = Q.remove()
14.      For each v in Adj[u]:
15.          If v.cor == BRANCO:
16.              v.cor = CINZA
17.              v.dist = u.dist + 1
18.              v.pred = u
19.              Q.insere(v)
20.          Endif
21.      Endfor
22.      u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA // Pinte o nó "s" de CINZA (primeiro nó visitado)
8.  s.dist = 0    // Distância de "s" para "s" é zero
9.  s.pred = NIL  // Predecessor de "s" é NIL
10. Q.insere(s)   // Insira "s" na fila

12. While |Q| != vazio: // Enquanto a fila não estiver vazia
13.      u = Q.remove() // Remova um nó do início da fila
14.      For each v in Adj[u]:
15.          If v.cor == BRANCO:
16.              v.cor = CINZA
17.              v.dist = u.dist + 1
18.              v.pred = u
19.              Q.insere(v)
20.          Endif
21.      Endfor
22.      u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA // Pinte o nó "s" de CINZA (primeiro nó visitado)
8.  s.dist = 0    // Distância de "s" para "s" é zero
9.  s.pred = NIL  // Predecessor de "s" é NIL
10. Q.insere(s)   // Insira "s" na fila

12. While |Q| != vazio: // Enquanto a fila não estiver vazia
13.      u = Q.remove() // Remova um nó do início da fila
14.      For each v in Adj[u]: // Todos os nós "v" adjacentes ao nó "u"
15.          If v.cor == BRANCO:
16.              v.cor = CINZA
17.              v.dist = u.dist + 1
18.              v.pred = u
19.              Q.insere(v)
20.          Endif
21.      Endfor
22.      u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA // Pinte o nó "s" de CINZA (primeiro nó visitado)
8.  s.dist = 0    // Distância de "s" para "s" é zero
9.  s.pred = NIL  // Predecessor de "s" é NIL
10. Q.insere(s)   // Insira "s" na fila

12. While |Q| != vazio: // Enquanto a fila não estiver vazia
13.      u = Q.remove() // Remova um nó do início da fila
14.      For each v in Adj[u]: // Todos os nós "v" adjacentes ao nó "u"
15.          If v.cor == BRANCO: // Verifique se "v" não foi visitado
16.              v.cor = CINZA
17.              v.dist = u.dist + 1
18.              v.pred = u
19.              Q.insere(v)
20.          Endif
21.      Endfor
22.      u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

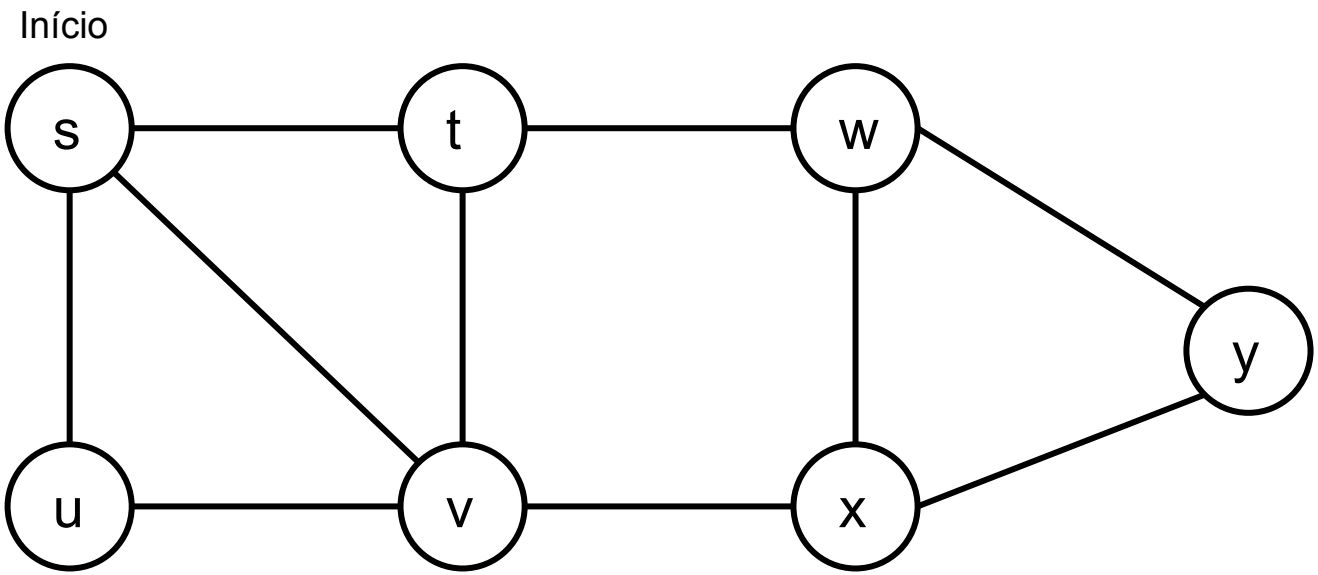
```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA // Pinte o nó "s" de CINZA (primeiro nó visitado)
8.  s.dist = 0    // Distância de "s" para "s" é zero
9.  s.pred = NIL  // Predecessor de "s" é NIL
10. Q.insere(s)   // Insira "s" na fila

12. While |Q| != vazio: // Enquanto a fila não estiver vazia
13.      u = Q.remove() // Remova um nó do início da fila
14.      For each v in Adj[u]: // Todos os nós "v" adjacentes ao nó "u"
15.          If v.cor == BRANCO: // Verifique se "v" não foi visitado
16.              v.cor = CINZA   // Pinte v de cinza (visitado)
17.              v.dist = u.dist + 1 // A distância de "v" é a distância até u + 1
18.              v.pred = u // Atualize predecessor
19.              Q.insere(v) // Insira "v"
20.          Endif
21.      Endfor
22.      u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Pseudocódigo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO // Pinte todos os nós de BRANCO
4.     v.pred = NIL   // Predecessor de todos os "v" para NIL
5.     v.dist = INF   // Dist iniciais iguais a infinito
6.  Endfor

7.  s.cor = CINZA // Pinte o nó "s" de CINZA (primeiro nó visitado)
8.  s.dist = 0    // Distância de "s" para "s" é zero
9.  s.pred = NIL  // Predecessor de "s" é NIL
10. Q.insere(s)   // Insira "s" na fila

12. While |Q| != vazio: // Enquanto a fila não estiver vazia
13.      u = Q.remove() // Remova um nó do início da fila
14.      For each v in Adj[u]: // Todos os nós "v" adjacentes ao nó "u"
15.          If v.cor == BRANCO: // Verifique se "v" não foi visitado
16.              v.cor = CINZA   // Pinte v de cinza (visitado)
17.              v.dist = u.dist + 1 // A distância de "v" é a distância até u + 1
18.              v.pred = u // Atualize predecessor
19.              Q.insere(v) // Insira "v"
20.          Endif
21.      Endfor
22.      u.cor = PRETO // Pinte "u" de preto (fora da fila)
23.  Endwhile
24. EndFunction
```

Simulação da Busca em Largura

Todos os nós são brancos

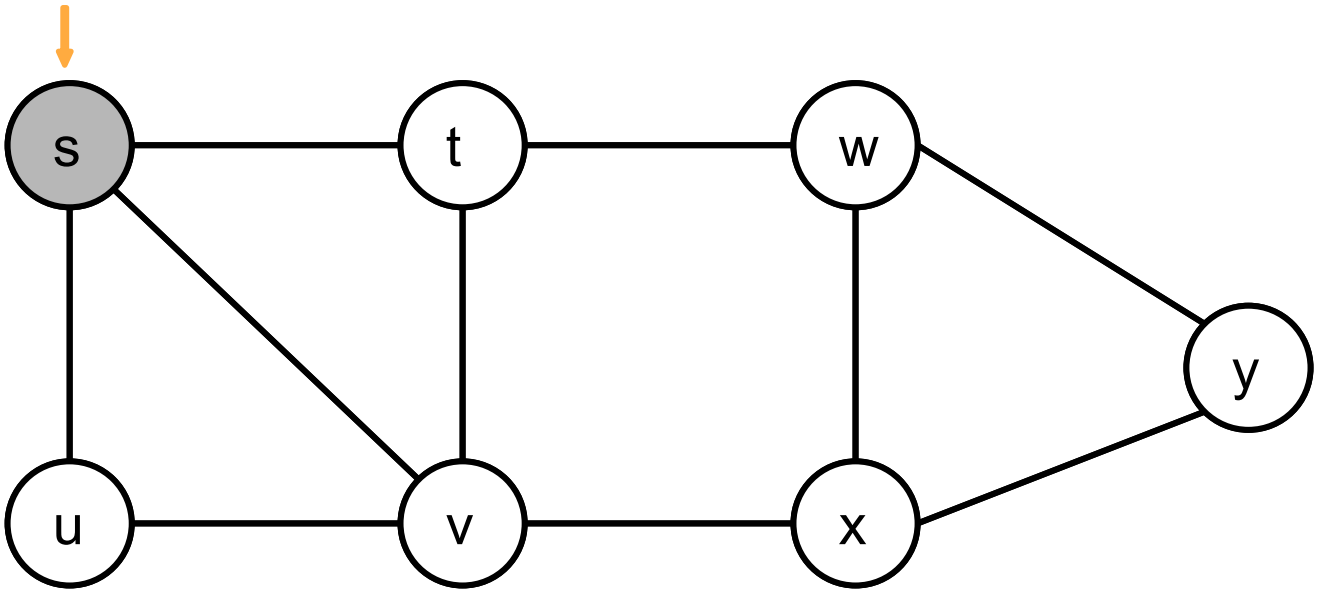



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	INF	INF	INF	INF	INF	INF	INF
pred	NIL	NIL	NIL	NIL	NIL	NIL	NIL

Fila vazia!

Simulação da Busca em Largura

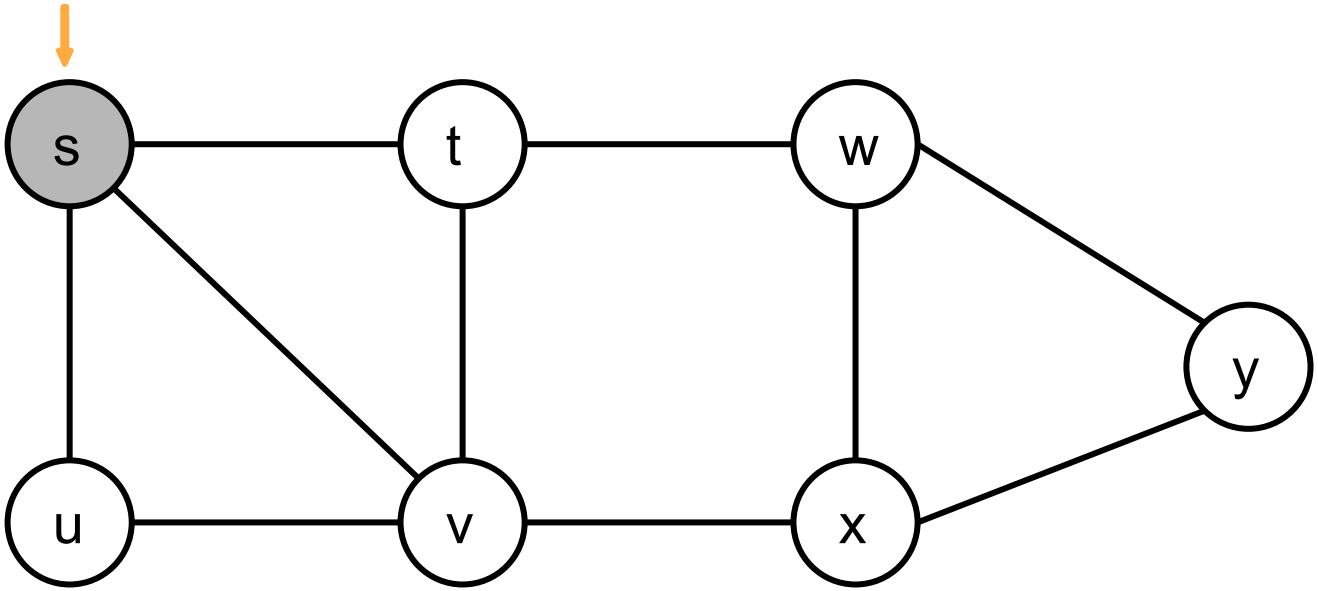



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	INF	INF	INF	INF	INF	INF
pred	NIL	NIL	NIL	NIL	NIL	NIL	NIL

	1
Fila	s

Simulação da Busca em Largura

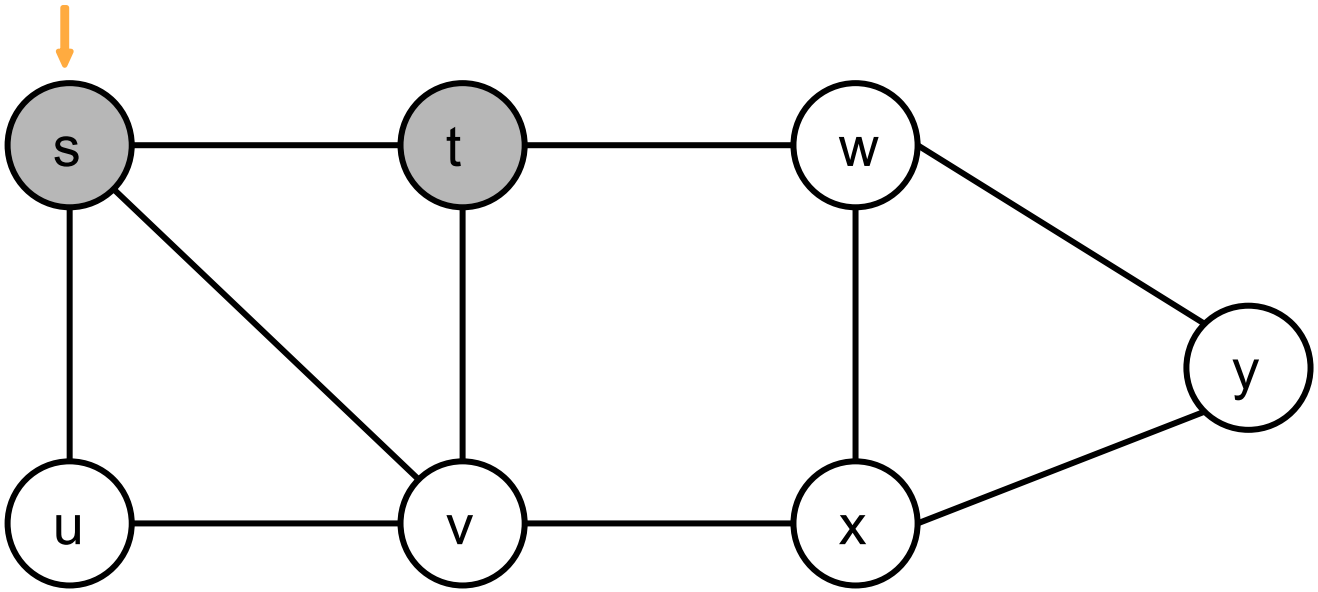



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	INF	INF	INF	INF	INF	INF
pred	NIL	NIL	NIL	NIL	NIL	NIL	NIL

Fila vazia!

Simulação da Busca em Largura

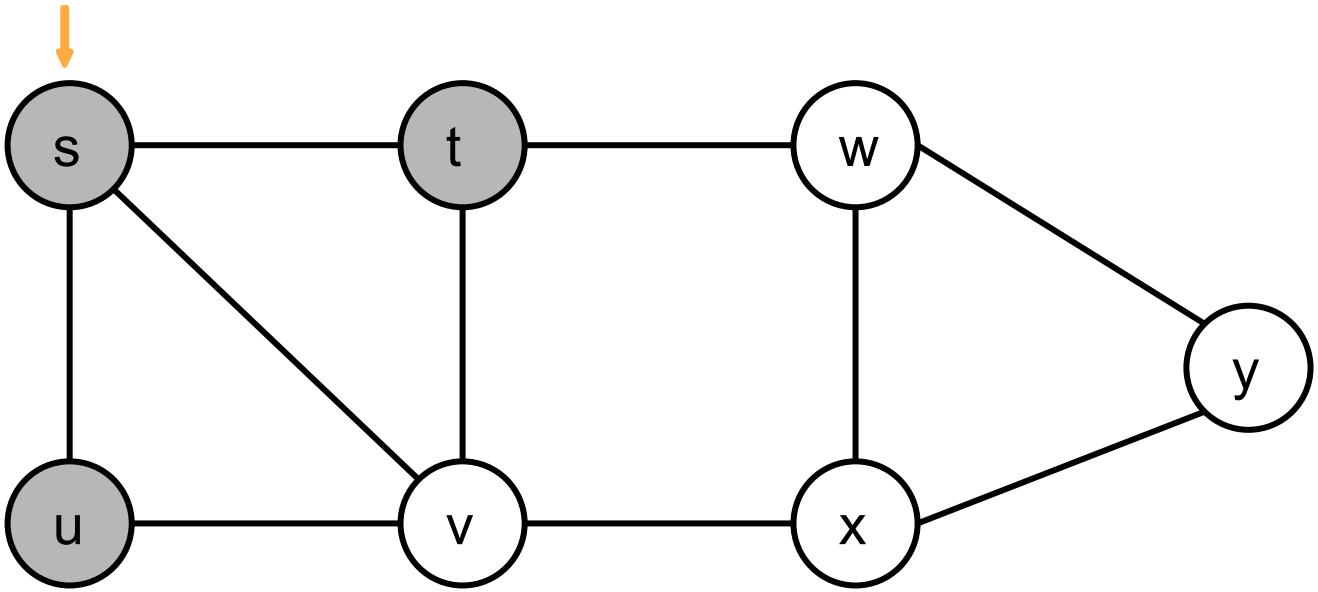



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	INF	INF	INF	INF	INF
pred	NIL	s	NIL	NIL	NIL	NIL	NIL

	1
Fila	t

Simulação da Busca em Largura

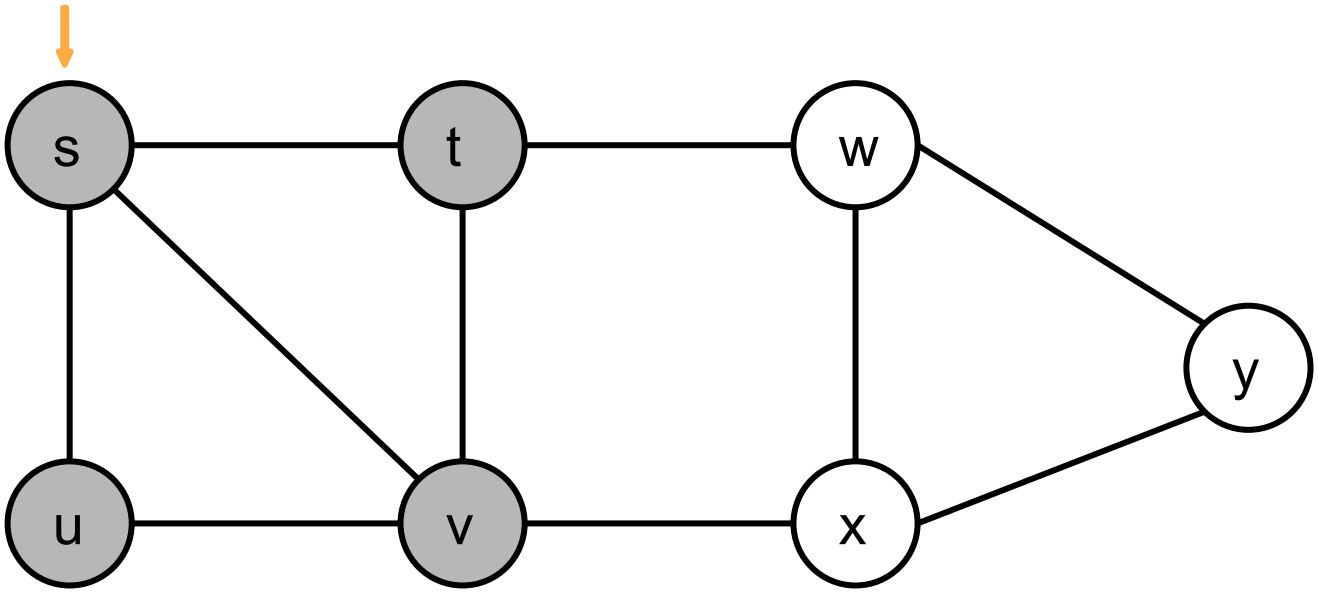



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.     Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	INF	INF	INF	INF
pred	NIL	s	s	NIL	NIL	NIL	NIL

	1	2
Fila	t	u

Simulação da Busca em Largura

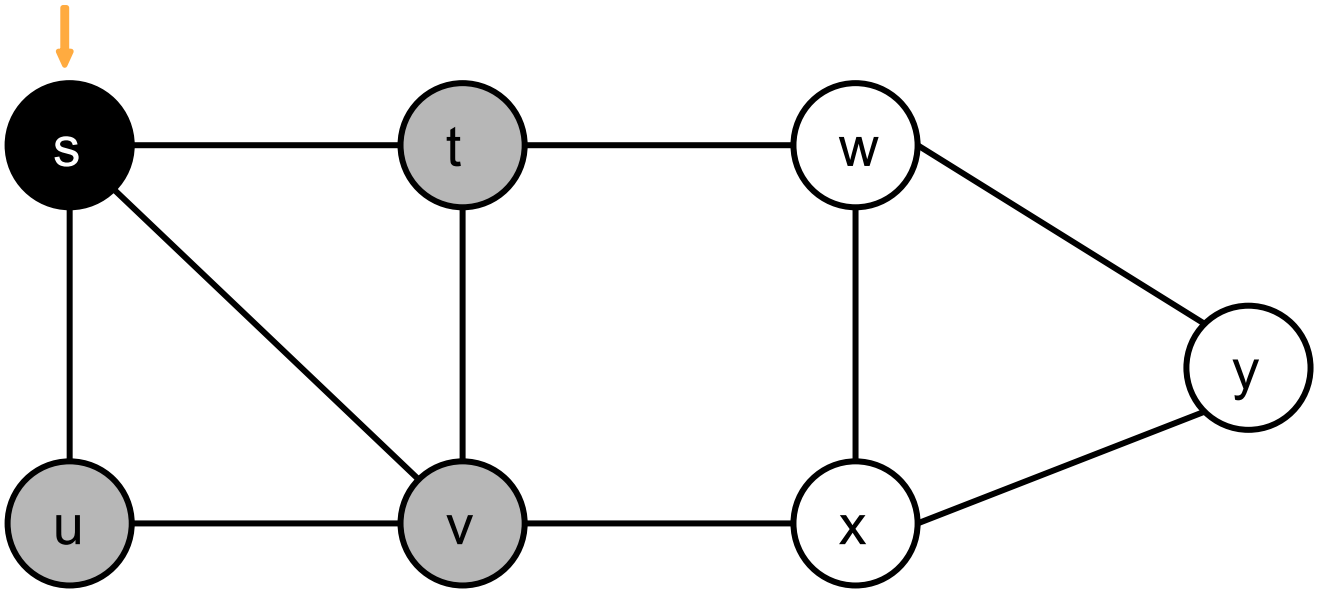



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.     Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	INF	INF	INF
pred	NIL	s	s	s	NIL	NIL	NIL

	1	2	3
Fila	t	u	v

Simulação da Busca em Largura

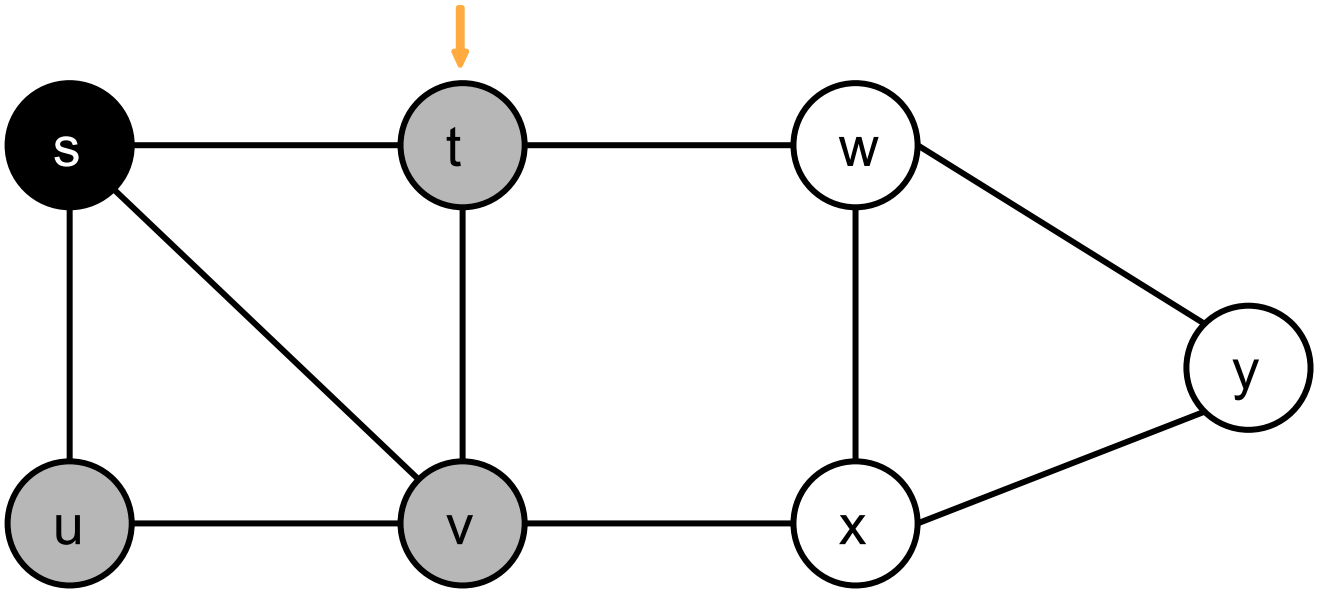



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	INF	INF	INF
pred	NIL	s	s	s	NIL	NIL	NIL

	1	2	3
Fila	t	u	v

Simulação da Busca em Largura

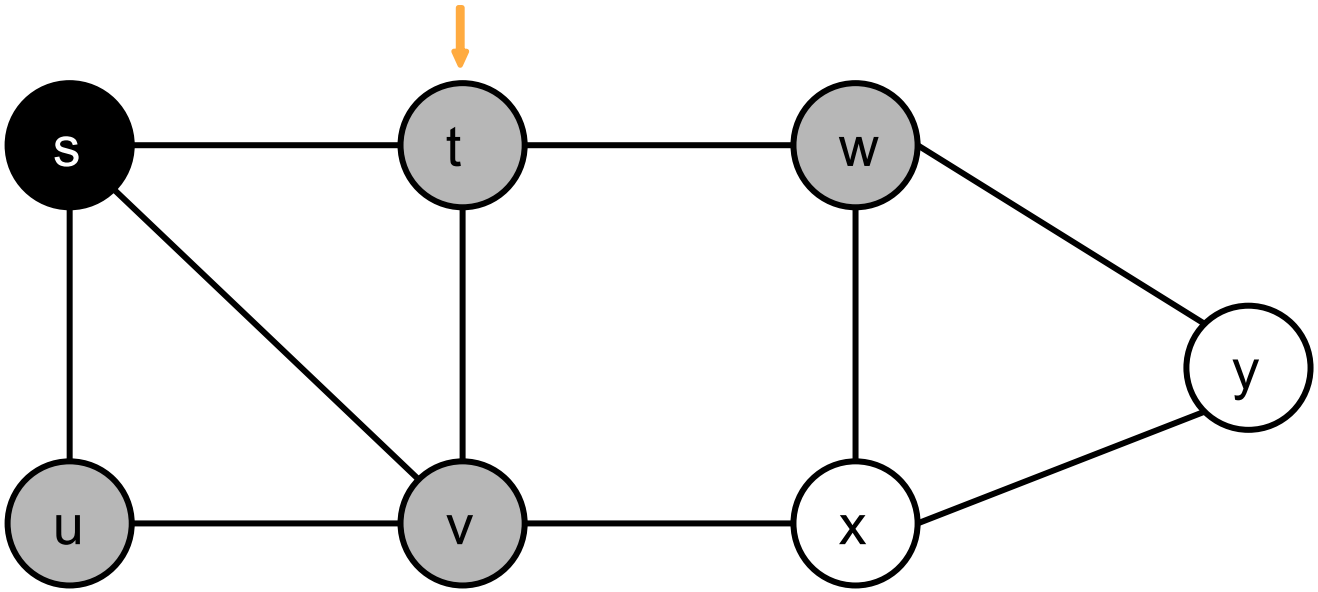



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	INF	INF	INF
pred	NIL	s	s	s	NIL	NIL	NIL

	1	2
Fila	u	v

Simulação da Busca em Largura

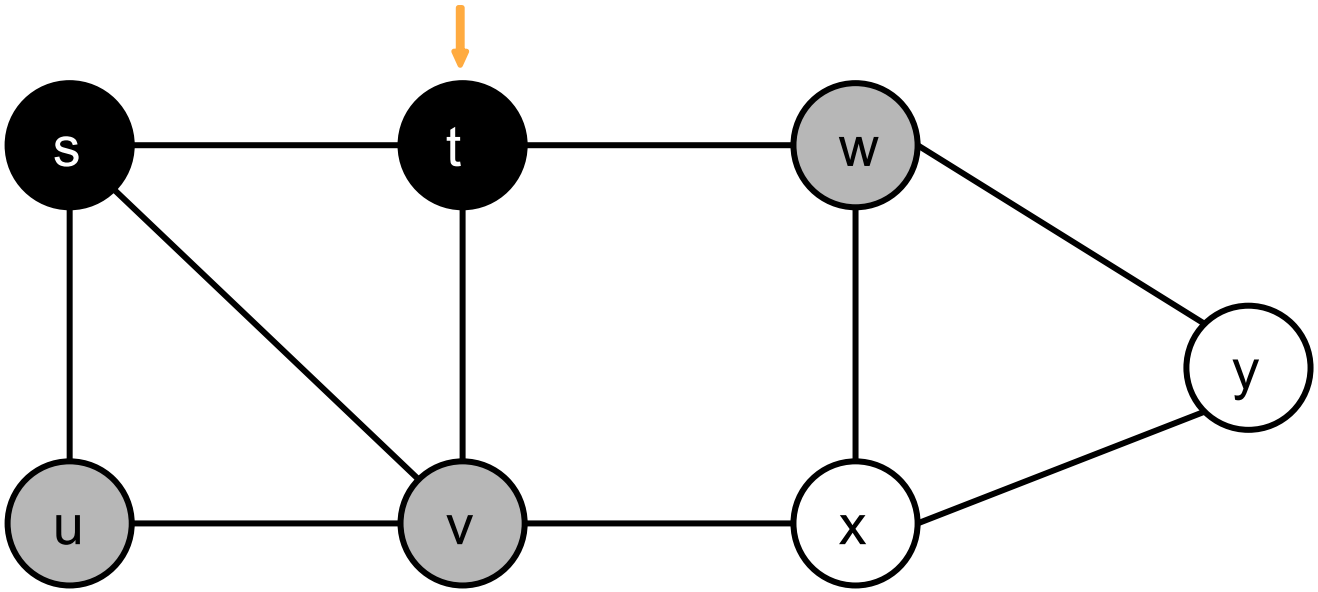



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	INF	2	INF
pred	NIL	s	s	s	NIL	t	NIL

Fila	1	2	3
	u	v	w

Simulação da Busca em Largura



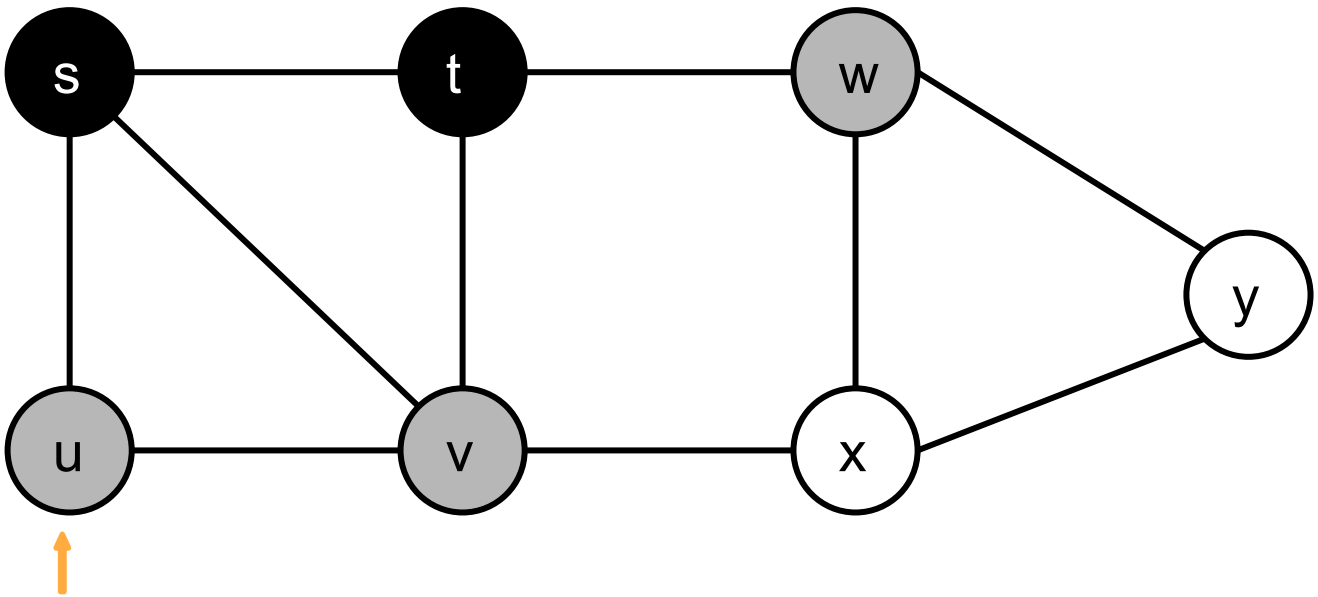

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	INF	2	INF
pred	NIL	s	s	s	NIL	t	NIL

	1	2	3
Fila	u	v	w

Simulação da Busca em Largura

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.     Endwhile
24.     EndFunction
```

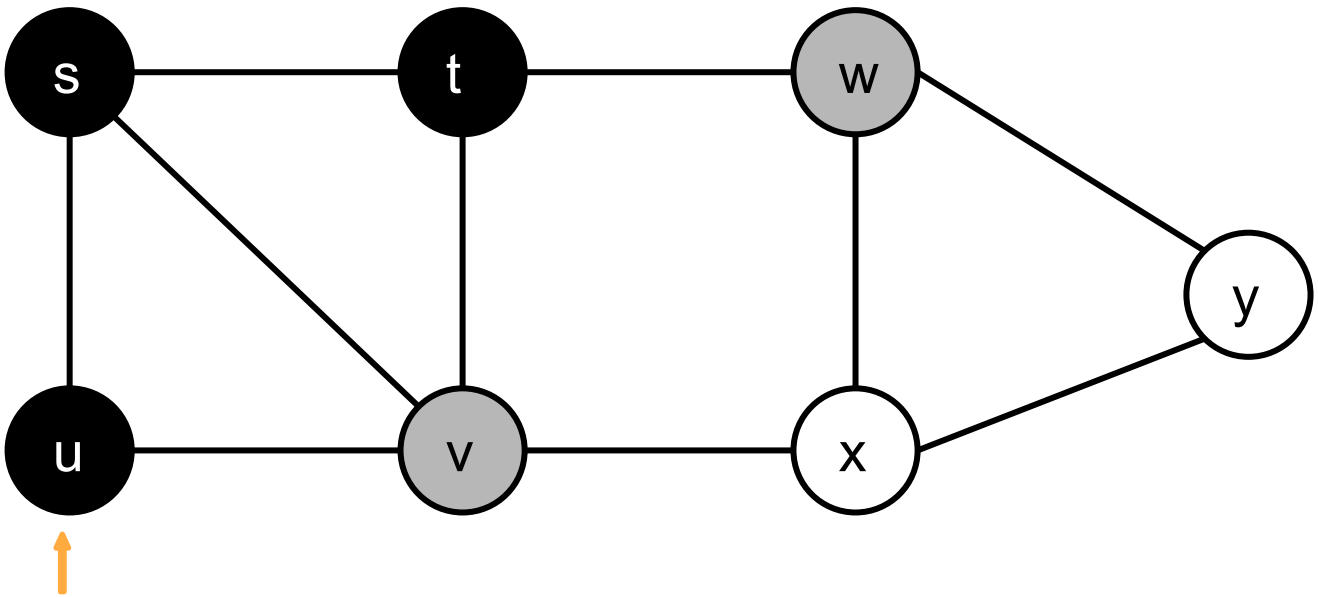


	s	t	u	v	x	w	y
dist	0	1	1	1	INF	2	INF
pred	NIL	s	s	s	NIL	t	NIL

	1	2
Fila	v	w

Simulação da Busca em Largura

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

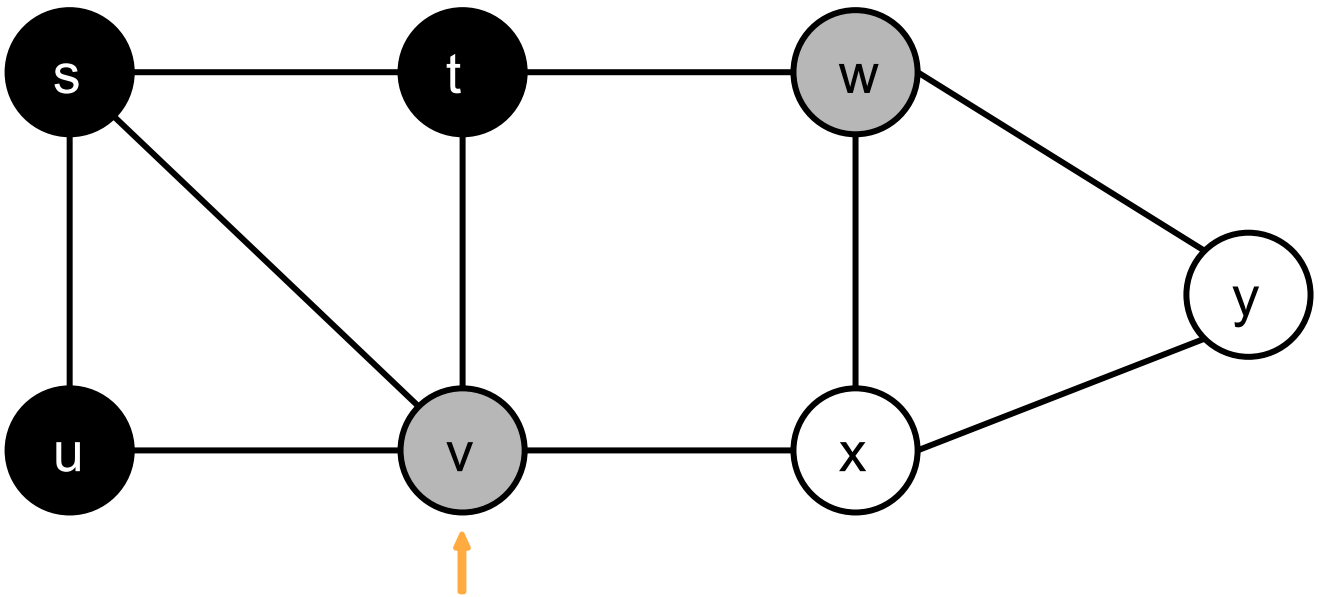


	s	t	u	v	x	w	y
dist	0	1	1	1	INF	2	INF
pred	NIL	s	s	s	NIL	t	NIL

	1	2
Fila	v	w

Simulação da Busca em Largura

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

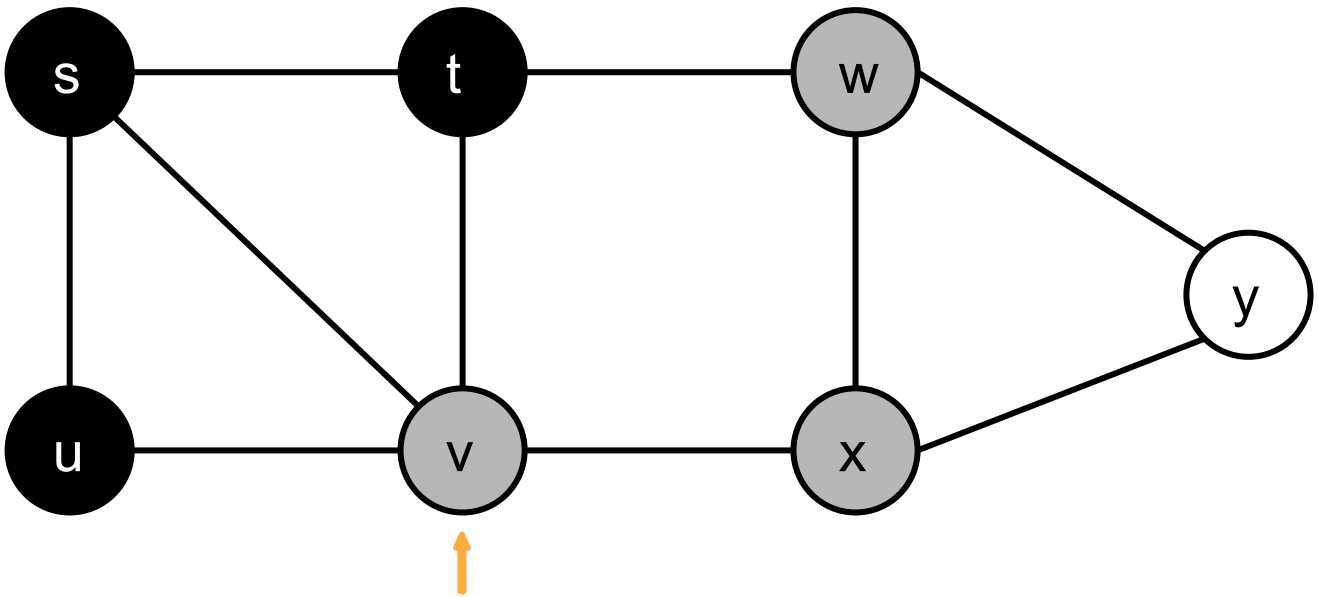


	s	t	u	v	x	w	y
dist	0	1	1	1	INF	2	INF
pred	NIL	s	s	s	NIL	t	NIL

	1
Fila	w

Simulação da Busca em Largura

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.     Endwhile
24.     EndFunction
```

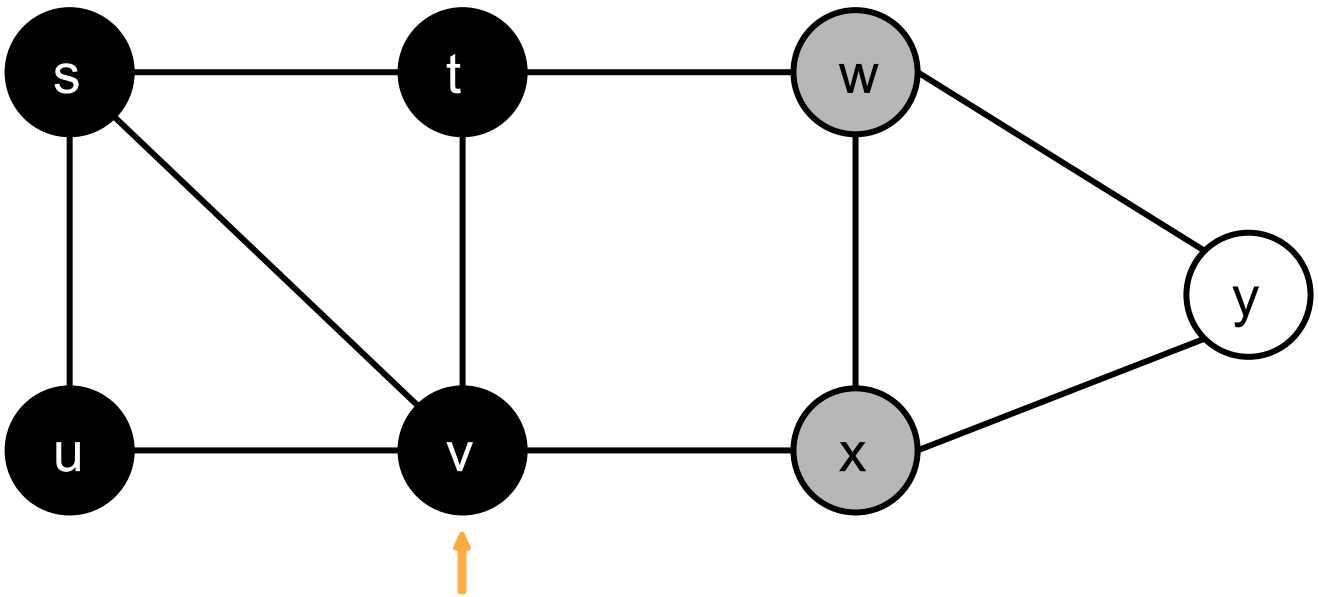


	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	INF
pred	NIL	s	s	s	v	t	NIL

	1	2
Fila	w	x

Simulação da Busca em Largura

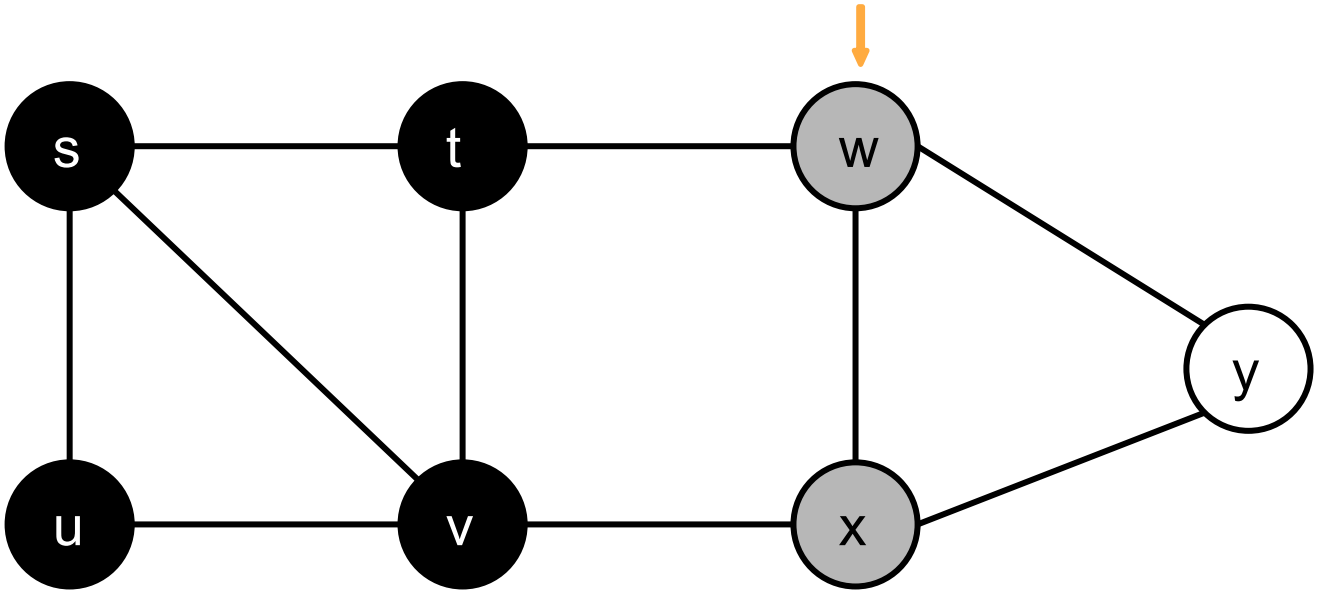
```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```



	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	INF
pred	NIL	s	s	s	v	t	NIL

	1	2
Fila	w	x

Simulação da Busca em Largura

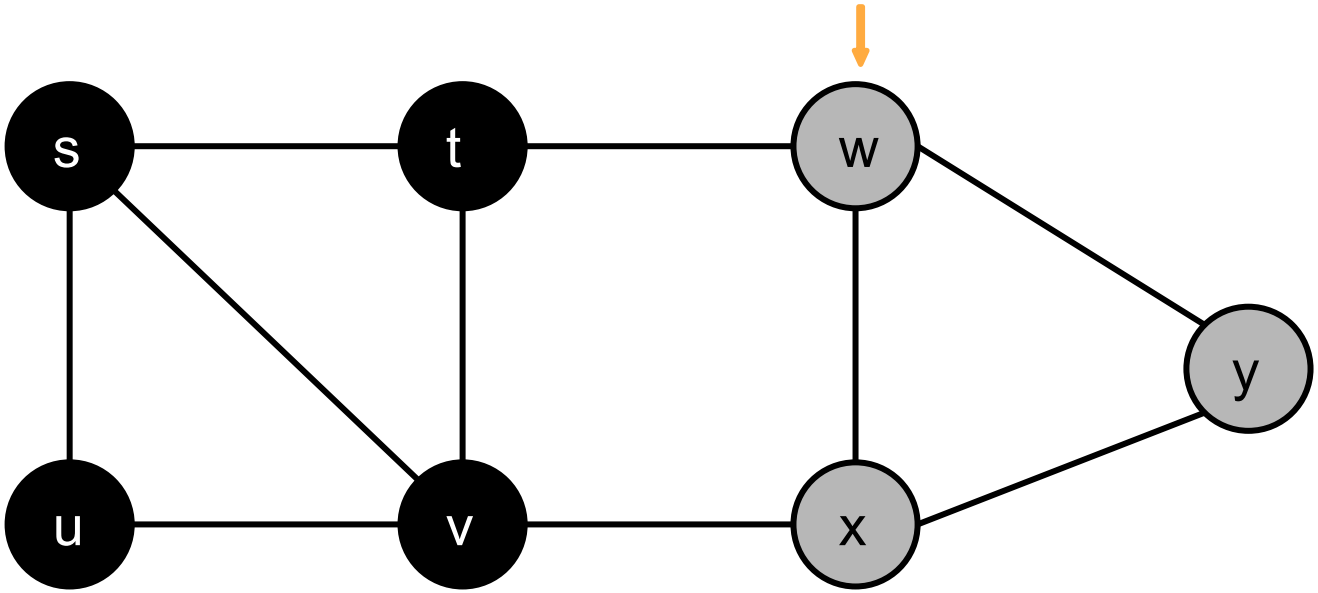



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	INF
pred	NIL	s	s	s	v	t	NIL

	1
Fila	x

Simulação da Busca em Largura

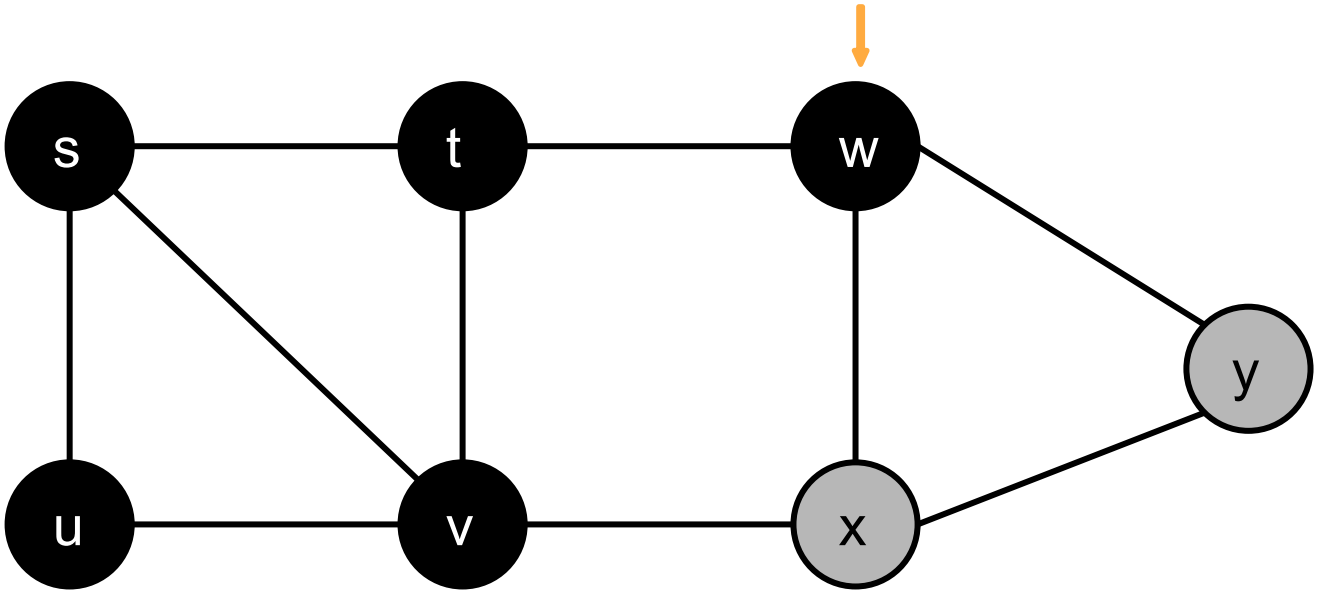



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.     Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

	1	2
Fila	x	y

Simulação da Busca em Largura



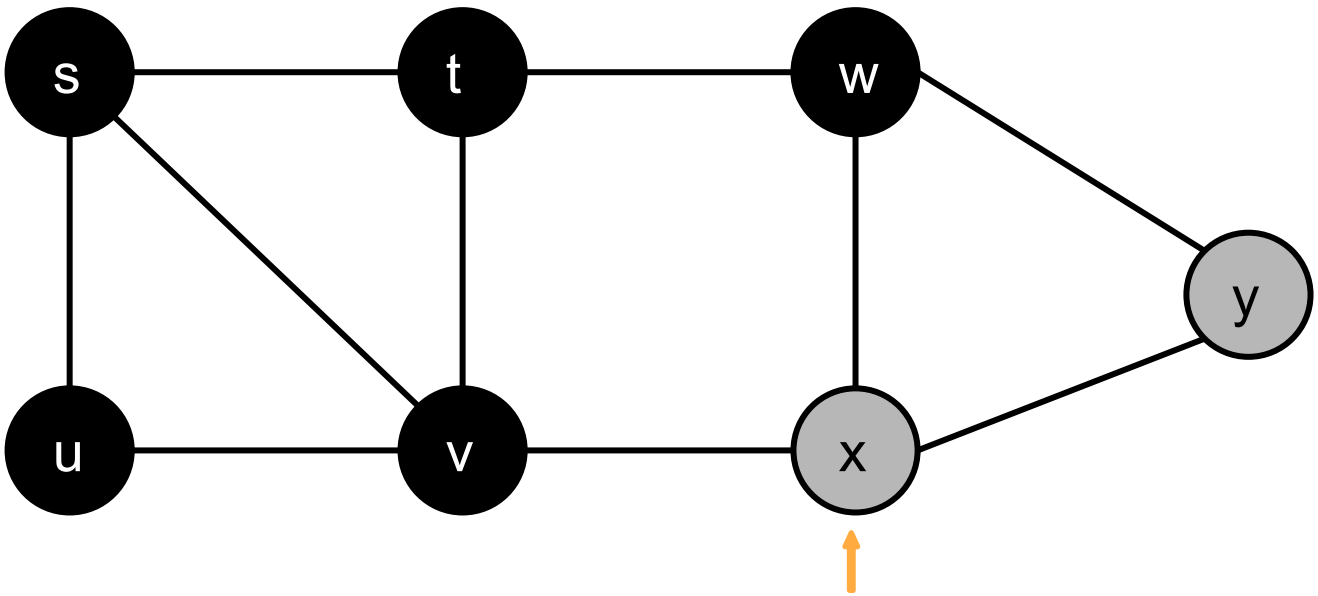

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

	1	2
Fila	x	y

Simulação da Busca em Largura

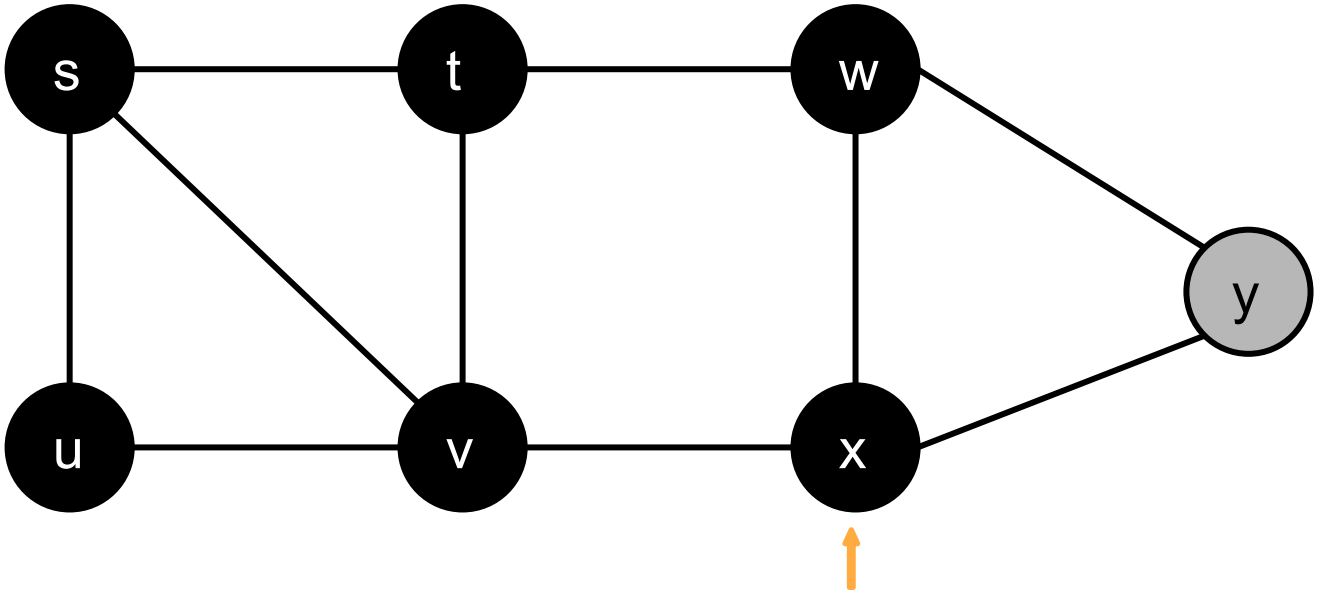
```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.      Endwhile
24.     EndFunction
```



	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

	1
Fila	y

Simulação da Busca em Largura

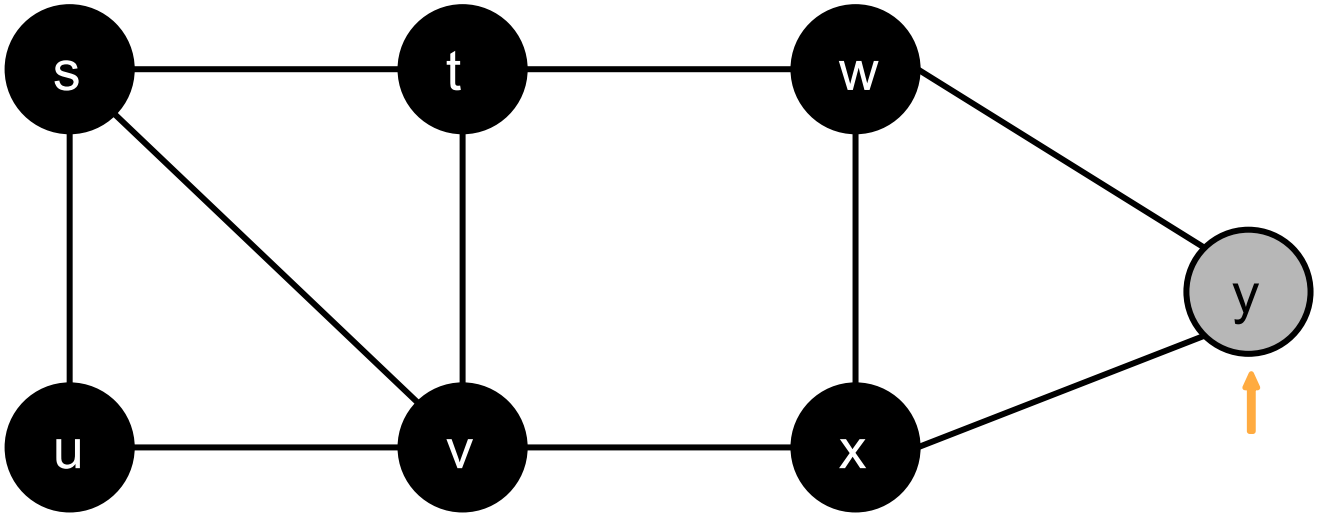



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

	1
Fila	y

Simulação da Busca em Largura

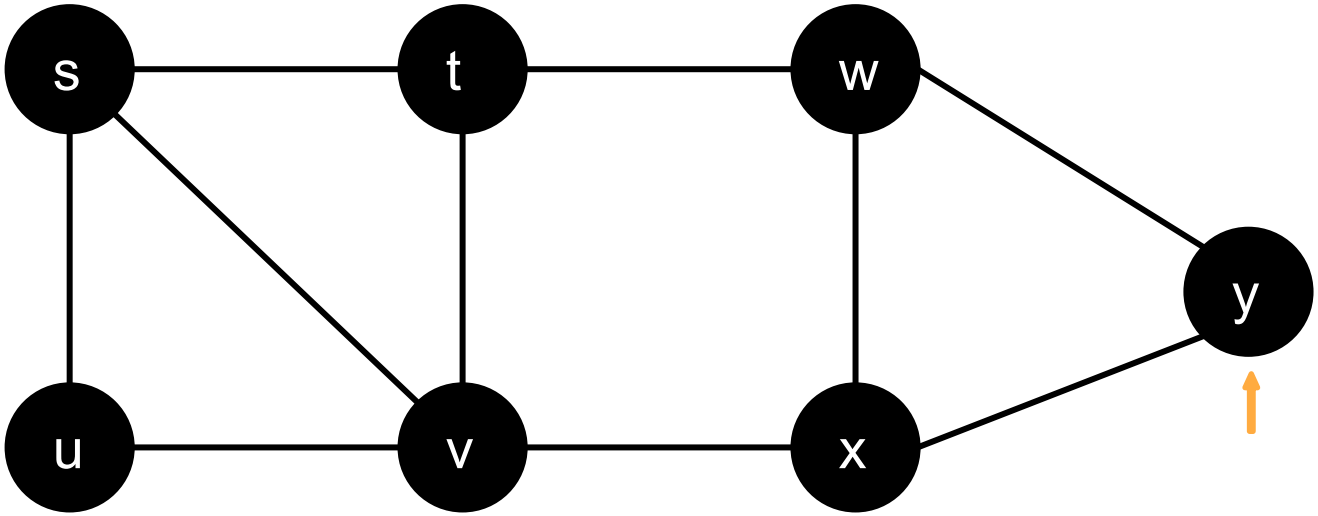



```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

Fila vazia!

Simulação da Busca em Largura



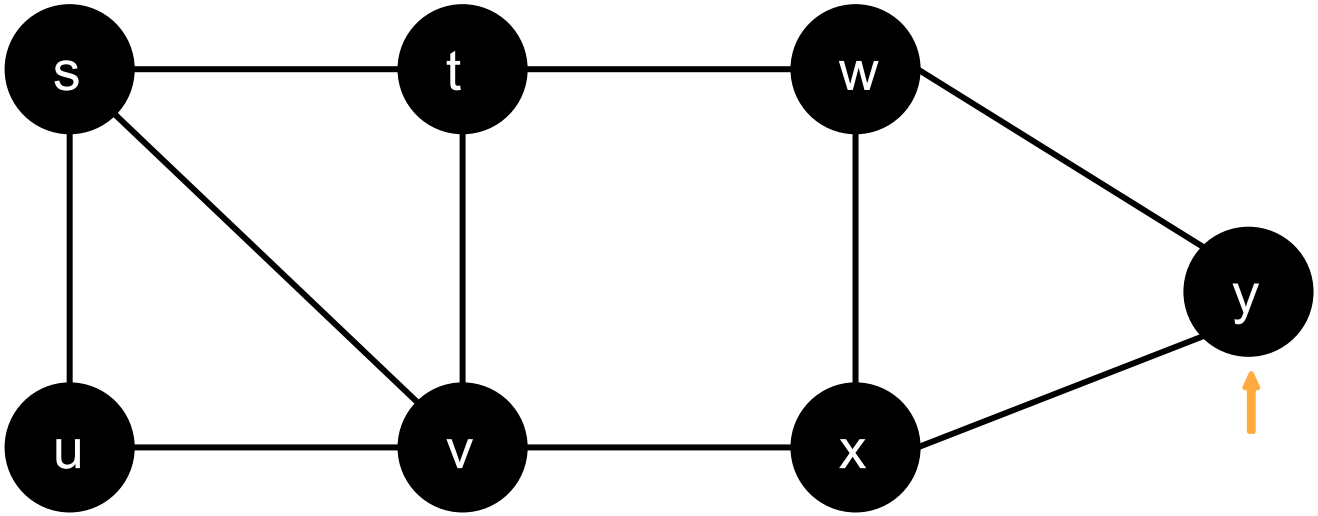

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

Fila vazia!

Simulação da Busca em Largura

Terminou!



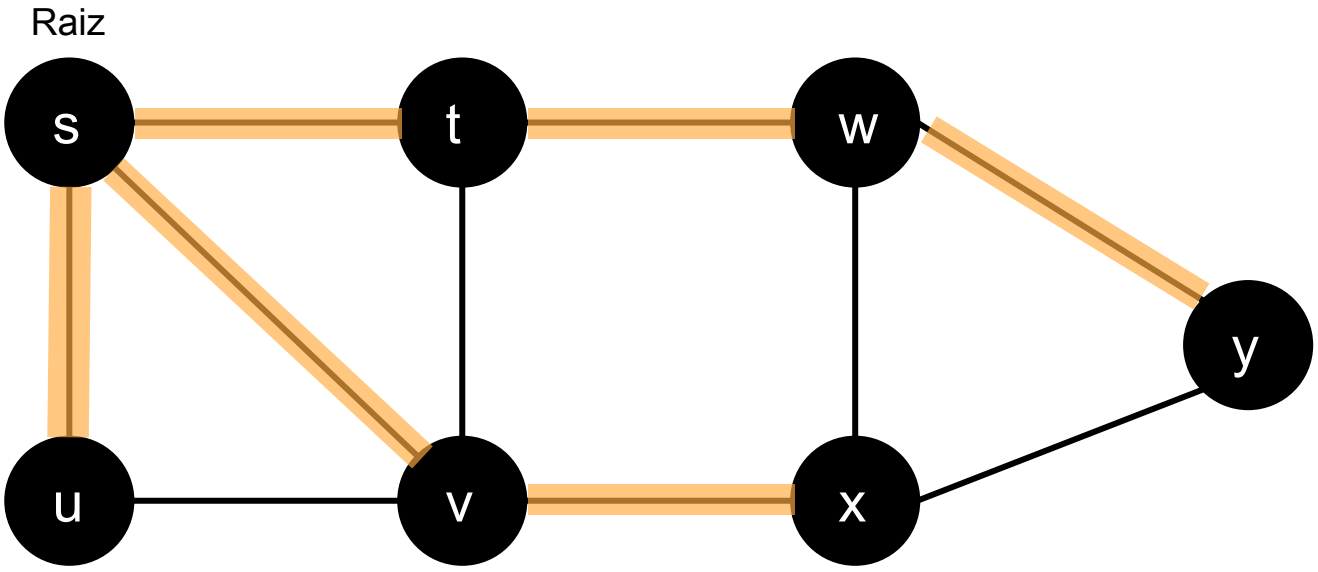

```
1.      Function BFS(V, Adj, s)
2.       For each v in V:
3.          v.cor = BRANCO
4.          v.pred = NIL
5.          v.dist = INF
6.       Endfor

7.       s.cor = CINZA
8.       s.dist = 0
9.       s.pred = NIL
10.     Q.insere(s)

12.     While |Q| != vazio:
13.          u = Q.remove()
14.          For each v in Adj[u]:
15.               If v.cor == BRANCO:
16.                    v.cor = CINZA
17.                    v.dist = u.dist + 1
18.                    v.pred = u
19.                    Q.insere(v)
20.               Endif
21.          Endfor
22.          u.cor = PRETO
23.      Endwhile
24.     EndFunction
```

	s	t	u	v	x	w	y
dist	0	1	1	1	2	2	3
pred	NIL	s	s	s	v	t	w

Fila vazia!

Árvore de Busca em Largura

- **BFS** produz uma árvore de busca em largura com raiz em **s** e que contem todos os vértices acessíveis, determinando o menor caminho (em número de arestas) de **s** a **t** (vértice acessível).



Análise de tempo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.  For each v in V:
3.     v.cor = BRANCO
4.     v.pred = NIL
5.     v.dist = INF
6.  Endfor

7.  s.cor = CINZA
8.  s.dist = 0
9.  s.pred = NIL
10. Q.insere(s)

12. While |Q| != vazio:
13.       u = Q.remove()
14.       For each v in Adj[u]:
15.           If v.cor == BRANCO:
16.               v.cor = CINZA
17.               v.dist = u.dist + 1
18.               v.pred = u
19.               Q.insere(v)
20.           Endif
21.       Endfor
22.       u.cor = PRETO
23.  Endwhile
24. EndFunction
```

Análise de tempo da Busca Largura

```
1. Function BFS(V, Adj, s)
2.   For each v in V:
3.      v.cor = BRANCO
4.      v.pred = NIL
5.      v.dist = INF
6.   Endfor

7.   s.cor = CINZA
8.   s.dist = 0
9.   s.pred = NIL
10.  Q.insere(s)

12.  While |Q| != vazio:
13.        u = Q.remove()
14.        For each v in Adj[u]:
15.            If v.cor == BRANCO:
16.                v.cor = CINZA
17.                v.dist = u.dist + 1
18.                v.pred = u
19.                Q.insere(v)
20.            Endif
21.        Endfor
22.        u.cor = PRETO
23.   Endwhile
24.  EndFunction
```

O(|V|)

Análise de tempo da Busca Largura

```
1.  Function BFS(V, Adj, s)
2.    For each v in V:
3.       v.cor = BRANCO
4.       v.pred = NIL
5.       v.dist = INF
6.    Endfor

7.    s.cor = CINZA
8.    s.dist = 0
9.    s.pred = NIL
10.   Q.insere(s)

12.   While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.    Endwhile
24.  EndFunction
```

O(|V|)

O(1)

Análise de tempo da Busca Largura

```
1.  Function BFS(V, Adj, s)
2.    For each v in V:
3.       v.cor = BRANCO
4.       v.pred = NIL
5.       v.dist = INF
6.    Endfor

7.    s.cor = CINZA
8.    s.dist = 0
9.    s.pred = NIL
10.   Q.insere(s)

12.   While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.    Endwhile
24.   EndFunction
```

O(|V|)

Análise de tempo da Busca Largura

```
1.  Function BFS(V, Adj, s)
2.   For each v in V:
3.      v.cor = BRANCO
4.      v.pred = NIL
5.      v.dist = INF
6.   Endfor

7.   s.cor = CINZA
8.   s.dist = 0
9.   s.pred = NIL
10.  Q.insere(s)

12.  While |Q| != vazio:
13.       u = Q.remove()
14.       For each v in Adj[u]:
15.            If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.            Endif
21.       Endfor
22.       u.cor = PRETO
23.   Endwhile
24.  EndFunction
```

O(|V|)

Análise de tempo da Busca Largura

```
1.  Function BFS(V, Adj, s)
2.    For each v in V:
3.       v.cor = BRANCO
4.       v.pred = NIL
5.       v.dist = INF
6.    Endfor

7.    s.cor = CINZA
8.    s.dist = 0
9.    s.pred = NIL
10.   Q.insere(s)

12.   While |Q| != vazio:
13.         u = Q.remove()
14.         For each v in Adj[u]:
15.             If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.             Endif
21.         Endfor
22.         u.cor = PRETO
23.    Endwhile
24.  EndFunction
```

O(|V|)

Análise de tempo da Busca Largura

```
1.  Function BFS(V, Adj, s)
2.   For each v in V:
3.      v.cor = BRANCO
4.      v.pred = NIL
5.      v.dist = INF
6.   Endfor

7.   s.cor = CINZA
8.   s.dist = 0
9.   s.pred = NIL
10.  Q.insere(s)

12.  While |Q| != vazio:
13.       u = Q.remove()
14.       For each v in Adj[u]:
15.            If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.            Endif
21.       Endfor
22.       u.cor = PRETO
23.   Endwhile
24.  EndFunction
```

O(|V|)

O(|E|)

Análise de tempo da Busca Largura

```
1.  Function BFS(V, Adj, s)
2.   For each v in V:
3.      v.cor = BRANCO
4.      v.pred = NIL
5.      v.dist = INF
6.   Endfor

7.   s.cor = CINZA
8.   s.dist = 0
9.   s.pred = NIL
10.  Q.insere(s)

12.  While |Q| != vazio:
13.       u = Q.remove()
14.       For each v in Adj[u]:
15.            If v.cor == BRANCO:
16.                 v.cor = CINZA
17.                 v.dist = u.dist + 1
18.                 v.pred = u
19.                 Q.insere(v)
20.            Endif
21.       Endfor
22.       u.cor = PRETO
23.   Endwhile
24.  EndFunction
```

$O(|V|+|E|)$

Exercício

- Simulação da Busca em Largura

Aplicações práticas da Busca em Largura

- GeeksfoGeeks

Bibliografia

- CORMEN, Thomas H et al. Algoritmos: teoria e prática. Rio de Janeiro: Campus, 2002. 916 p.
- ISBN 978-85-352-0926-6. tradução de ""Introduction to algorithms"" 2.ed.

- ZIVIANI, N. Projeto de algoritmos: com implementações em PASCAL e C. 2 ed. rev. e ampl. São
- Paulo: Thomson, 2004. 552 p. ISBN 978-85-221-0390-4. (Também disponível em e-book)