

# Grafos: busca em largura

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados II (AE23CP)  
Engenharia de Computação  
Departamento Acadêmico de Informática (Dainf)  
Universidade Tecnológica Federal do Paraná (UTFPR)  
Campus Pato Branco

- Busca em Largura
- Implementação da Busca em Largura
- Análise de Complexidade da Busca em Largura
- Caminho Mais Curto

- Grafos são estruturas mais complexas em comparação com listas, vetores e árvores binárias
- Grande parte dos problemas representados por grafos necessitam de métodos eficientes para a exploração dessa estrutura de dados
- A complexidade de algoritmos para grafos ( $G$ ) é dada em termos de  $V$  e/ou  $E$ 
  - $V$ : conjunto de vértices
  - $E$ : conjunto de arestas

- Informações relevantes sobre a estrutura do grafo podem ser extraídas
  - Podem ser úteis para projetar algoritmos eficientes para determinados problemas
- Busca em grafos: processo de seguir sistematicamente pelas arestas a fim de visitar os vértices do grafo
  - Busca em profundidade
  - Busca em largura

## Busca em Largura

- Examina, sistematicamente, todos os vértices alcançáveis pelo vértice de origem  $s$  (também denominado fonte)
  - Um vértice  $v$  é alcançável a partir de um vértice  $s$  em um grafo  $G$  se existe um caminho de  $s$  a  $v$
- A busca em largura também é denominada como busca por nível
  - Os vértices são explorados em ordem de distância em relação à fonte  $s$
  - Por exemplo, a busca é iniciada nos vértices adjacentes a  $s$ , ou com distância  $d = 1$  e, em seguida, processamento é continuado para  $d = 2$ , onde processo é repetido, até a exploração completa (se possível) do grafo

- O algoritmo de busca em largura tem a sua execução finalizada quando todos os vértices alcançáveis a partir de  $s$  forem explorados
- Se  $v$  **não é alcançável** a partir de  $s$ , então dizemos que a distância entre ambos vértices é  $\infty$  (infinita)
  - No início da aplicação do algoritmo, o vértice  $s$  começa com o valor de distância igual a zero e os demais, com  $\infty$
  - No final da execução do algoritmo, para cada vértice há um valor de distância em relação ao  $s$

- Algumas aplicações do algoritmo de busca em largura
  - GPS
  - Motores de busca
  - Redes ponto-a-ponto
  - Transmissão em rede
  - etc.



- Um algoritmo de busca em largura recebe um grafo  $G = (V, E)$  e um vértice especificado  $s$  chamado fonte (*source*)
- Percorre todos os vértices alcançáveis a partir de  $s$  em ordem de distância
- O algoritmo constrói uma **árvore de busca em largura** (subgrafo acíclico gerador) com raiz  $s$ 
  - Cada caminho entre  $s$  e  $v$  nessa árvore corresponde a um caminho mais curto entre ambos vértices
  - O algoritmo descobre todos os vértices a uma distância  $k$  do vértice origem antes de descobrir qualquer vértice a uma distância  $k + 1$

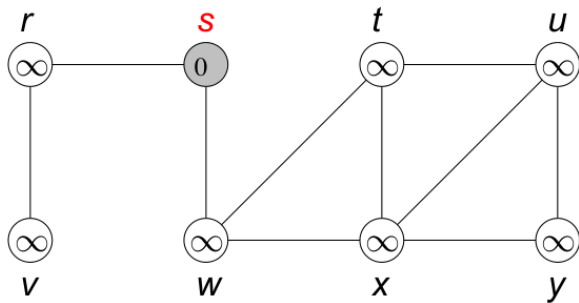
# Busca em Largura

- Inicialmente a árvore de busca em largura contém apenas o vértice fonte  $s$
- Para cada vizinho  $v$  de  $s$ , o vértice  $v$  e a aresta  $(s, v)$  são acrescentadas à árvore
- O processo é repetido para os vizinhos dos vizinhos de  $s$ 
  - Isso é feito até que todos os vértices alcançáveis por  $s$  sejam adicionados na árvore
  - O algoritmo de busca em largura também pode formar uma floresta
- O processo de busca é implementado através de uma fila  $Q$

- Durante a aplicação do algoritmo de busca em largura, cada vértice pode ser colorido por meio das seguintes cores
  - Branca: não visitado (inicialmente, todos os vértices são brancos)
  - Cinza: visitado pela primeira vez
  - Preta: todos os seus vizinhos foram visitados
- Vértices de cinza podem ter alguns vértices adjacentes brancos, e eles representam a fronteira entre vértices descobertos e não descobertos

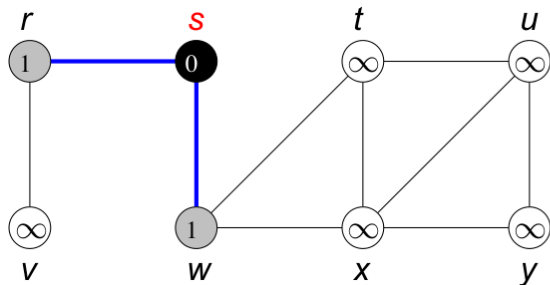
# Busca em Largura

## Exemplo



# Busca em Largura

## Exemplo

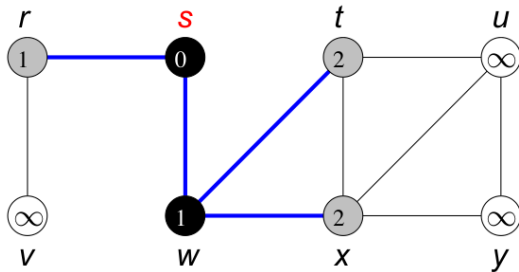


Q

w	r
1	1

# Busca em Largura

## Exemplo

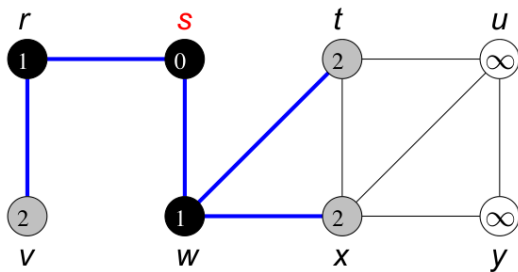


Q

$r$	$t$	$x$
1	2	2

# Busca em Largura

## Exemplo

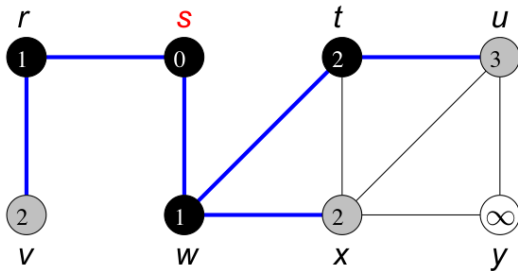


Q

$t$	$x$	$v$
2	2	2

# Busca em Largura

## Exemplo



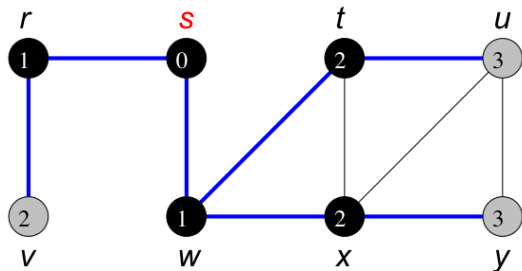
Q

$x$	$v$	$u$
2	2	3



# Busca em Largura

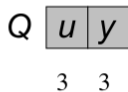
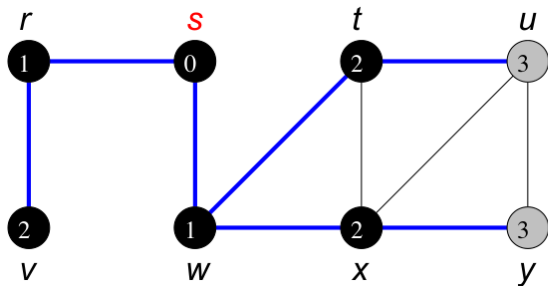
## Exemplo



Q	<table><tr><td><math>v</math></td><td><math>u</math></td><td><math>y</math></td></tr></table>	$v$	$u$	$y$
$v$	$u$	$y$		
	2   3   3			

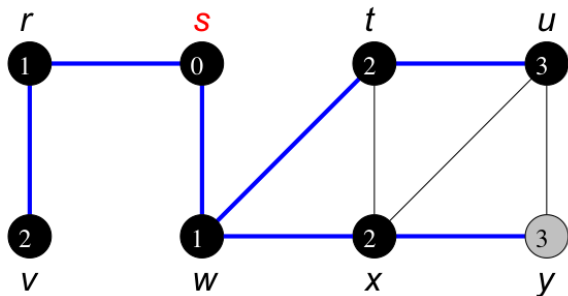
# Busca em Largura

## Exemplo



# Busca em Largura

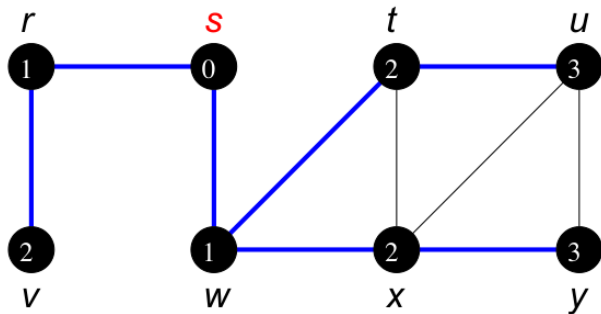
## Exemplo



Q y  
3

# Busca em Largura

## Exemplo



$Q \emptyset$

## Implementação da Busca em Largura

# Implementação da Busca em Largura

- A variável  $d[v]$  é usada para armazenar a **distância** de  $s$  a  $v$ 
  - Determinada durante o processo de busca
  - Na inicialização de  $d$ ,  $d[s] = 0$  e os demais são iniciados com "infinito" ( $d[v] = \infty$ )
- Cores
  - Para cada vértice  $v$ , a cor atual é guardada no vetor  $cor[v]$ , que pode ser branco, cinza ou preto
  - Todos os elementos de  $cor$  são inicializados com branco
  - Para o efeito de implementação, o uso da cor não é realmente necessário, mas facilita a compreensão do algoritmo
- A raiz da árvore de busca em largura é  $s$

# Implementação da Busca em Largura

- Cada vértice  $v$  (exceto a raiz) possui um pai  $\pi[v]$ 
  - Todos os elementos de  $\pi$  são inicializados com "nulo" ou um outro valor para representar que cada vértice ainda não tem pai
  - No final da aplicação, apenas  $s$  e vértices inalcançáveis por  $s$  não possuem pai
  - O caminho de  $s$  até  $v$  é dado por
$$v, \pi[v], \pi[\pi[v]], \pi[\pi[\pi[v]]], \dots, s$$
- O algoritmo de busca em largura recebe um grafo  $G$  (na forma de listas de adjacências), e um vértice  $s \in V[G]$  e devolve
  - 1 Para cada vértice  $v$ , a distância de  $s$  a  $v$  em  $G$
  - 2 Árvore ou floresta de busca em largura

# Implementação da Busca em Largura

**BUSCA-EM-LARGURA**( $G, s$ )

0   ▷ Inicialização

1   **para cada**  $u \in V[G] - \{s\}$  **faça**

2        $\text{cor}[u] \leftarrow \text{branco}$

3        $d[u] \leftarrow \infty$

4        $\pi[u] \leftarrow \text{NIL}$

5    $\text{cor}[s] \leftarrow \text{cinza}$

6    $d[s] \leftarrow 0$

7    $\pi[s] \leftarrow \text{NIL}$

8    $Q \leftarrow \emptyset$

9   **ENQUEUE**( $Q, s$ )



# Implementação da Busca em Largura

```
10  enquanto  $Q \neq \emptyset$  faça
11       $u \leftarrow \text{DEQUEUE}(Q)$ 
12      para cada  $v \in \text{Adj}[u]$  faça
13          se  $\text{cor}[v] = \text{branco}$  então
14               $\text{cor}[v] \leftarrow \text{cinza}$ 
15               $d[v] \leftarrow d[u] + 1$ 
16               $\pi[v] \leftarrow u$ 
17               $\text{ENQUEUE}(Q, v)$ 
18       $\text{cor}[u] \leftarrow \text{preto}$ 
```

- $\pi$  é a árvore de busca em largura

# Implementação da Busca em Largura

## Exemplo

**BUSCA-EM-LARGURA**( $G, s$ )

0  $\triangleright$  Inicialização

1 **para cada**  $u \in V[G] - \{s\}$  **faça**

2      $cor[u] \leftarrow$  branco

3      $d[u] \leftarrow \infty$

4      $\pi[u] \leftarrow \text{NIL}$

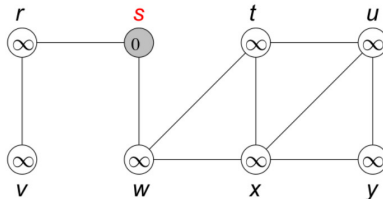
5  $cor[s] \leftarrow$  cinza

6  $d[s] \leftarrow 0$

7  $\pi[s] \leftarrow \text{NIL}$

8  $Q \leftarrow \emptyset$

9 **ENQUEUE**( $Q, s$ )



	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	b	0	b	b	b	b	b	b
$\pi$	a	a	a	a	a	a	a	a

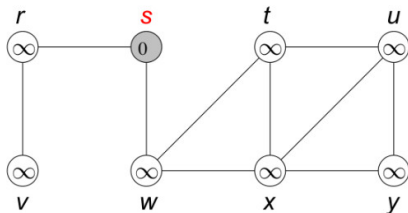


# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	b	0	b	b	b	b	b	b
$\pi$	a	a	a	a	a	a	a	a



$Q$   $S$   
0

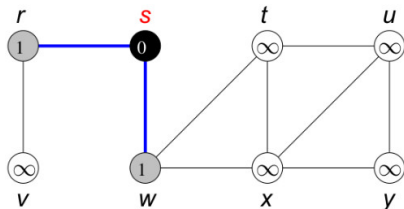
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = s$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	b	b	b	b	b	1
$\pi$	s	a	a	a	a	a	a	s



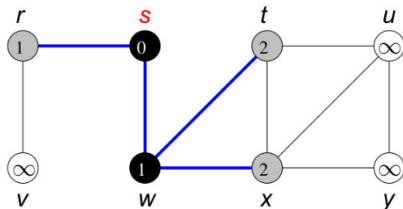
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = w$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	b	b	2	b	1
$\pi$	s	a	w	a	a	w	a	s



$Q$	$r$	$t$	$x$
	1	2	2

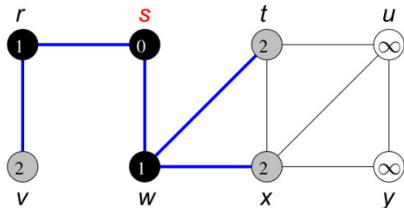
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = r$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	b	2	2	b	1
$\pi$	s	a	w	a	r	w	a	s



$Q$	$t$	$x$	$v$
	2	2	2

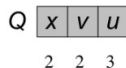
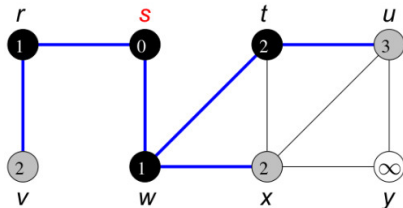
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = t$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	3	2	2	b	1
$\pi$	s	a	w	t	r	w	a	s



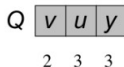
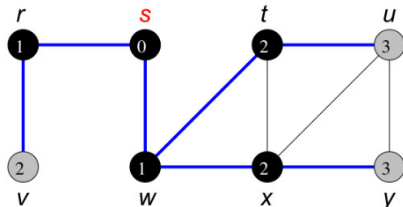
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = x$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	3	2	2	3	1
$\pi$	s	a	w	t	r	w	x	s





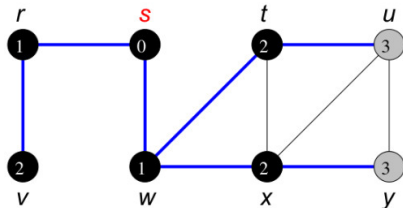
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = v$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	3	2	2	3	1
$\pi$	s	a	w	t	r	w	x	s



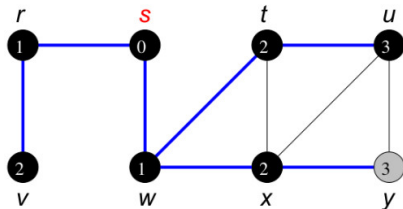
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = u$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	3	2	2	3	1
$\pi$	s	a	w	t	r	w	x	s



$Q$   $y$   
3

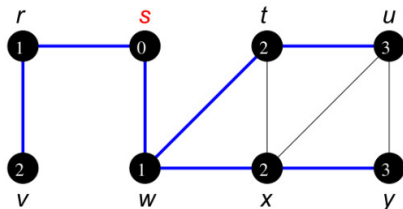
# Implementação da Busca em Largura

## Exemplo

```
10 enquanto  $Q \neq \emptyset$  faça
11    $u \leftarrow \text{DEQUEUE}(Q)$ 
12   para cada  $v \in \text{Adj}[u]$  faça
13     se  $\text{cor}[v] = \text{branco}$  então
14        $\text{cor}[v] \leftarrow \text{cinza}$ 
15        $d[v] \leftarrow d[u] + 1$ 
16        $\pi[v] \leftarrow u$ 
17        $\text{ENQUEUE}(Q, v)$ 
18    $\text{cor}[u] \leftarrow \text{preto}$ 
```

$u = y$

	$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$d$	1	0	2	3	2	2	3	1
$\pi$	s	a	w	t	r	w	x	s



## Análise de Complexidade da Busca em Largura

# Análise de Complexidade da Busca em Largura

- A inicialização consome tempo  $O(|V|)$
- Depois que um vértice deixa de ser branco, ele não volta a ser branco novamente
  - Cada vértice é adicionado na fila apenas uma vez
  - Cada operação sobre a fila consome tempo  $O(1)$ , resultando em um total de  $O(|V|)$
- Em uma lista de adjacência, cada vértice é percorrido apenas uma vez
  - A soma dos comprimentos das listas de adjacência é  $O(|E|)$
  - Logo, o tempo gasto para percorrer as listas é  $O(|E|)$

- Conclusão

A complexidade de tempo do algoritmo  
***BUSCA-EM-LARGURA*** é  $O(|V| + |E|)$

## Caminho Mais Curto

Print-Path( $G, s, v$ )

1   **se**  $v = s$  **então**

2       imprime  $s$

3   **senão**

4       **se**  $\pi[v] = \text{NIL}$  **então**

4           imprime não existe caminho de  $s$  a  $v$ .

5       **senão**

6           Print-Path( $G, s, \pi[v]$ )

7           imprime  $v$ .

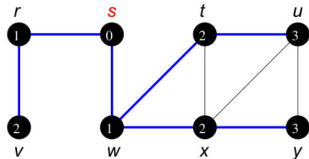


# Caminho Mais Curto

- Imprime um caminho mais curto de  $s$  a  $u$

Print-Path( $G, s, v$ )

```
1  se  $v = s$  então
2    imprime  $s$ 
3  senão
4    se  $\pi[v] = \text{NIL}$  então
4      imprime não existe caminho de  $s$  a  $v$ .
5    senão
6      Print-Path( $G, s, \pi[v]$ )
7    imprime  $v$ .
```



Print-Path( $G, s, u$ )

Print-Path( $G, s, t$ )

Print-Path( $G, s, w$ )

Print-Path( $G, s, s$ )

imprime  $s$

imprime  $w$

imprime  $t$

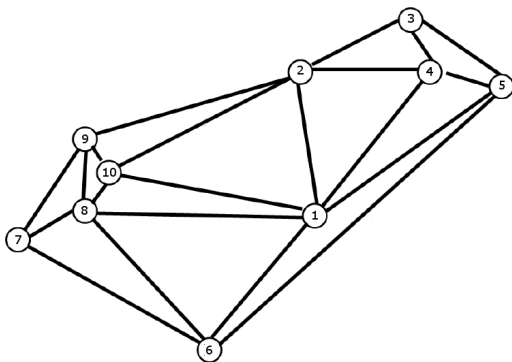
imprime  $u$

pi 

$r$	$s$	$t$	$u$	$v$	$x$	$y$	$w$
$s$	$a$	$w$	$t$	$r$	$w$	$x$	$s$

# Exercício

- Considere o grafo abaixo e faça:



- Execute a busca em largura considerando o vértice 1 como origem. Para este caso, considere que todas as listas de adjacência estejam ordenadas crescentemente.



Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.  
*Introduction to Algorithms.*  
Third edition, The MIT Press, 2009.



Marin, L. O.  
Grafos – Algoritmos de Busca: Busca em Largura. AE23CP –  
Algoritmos e Estrutura de Dados II.  
*Slides.* Engenharia de Computação. Dainf/UTFPR/Pato  
Branco, 2017.



Ziviani, N.  
*Projeto de Algoritmos - com implementações em Java e C++.*  
Thomson, 2007.