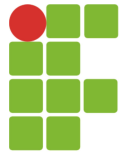
Grafos: Percurso

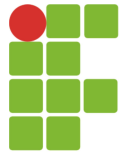
Tópicos Especiais em Algoritmos - Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

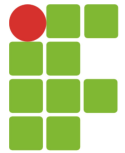
IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário

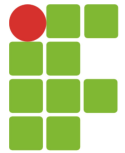


Sumário

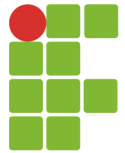


Percurso em Grafos

- Talvez o problema mais elementar em grafos seja percorrê-los.
- O percurso deve certificar de não visitar os mesmos nós várias vezes.
- Precisamos marcar o nó após visitá-lo, para não ficarmos presos em um ciclo.
- Duas estratégias elementares:
  1. Busca em largura (BFS - Breath-First-Search).
  2. Busca em profundidade (DFS - Depth-First-Search).



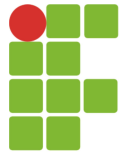
Sumário



Busca em Largura

BFS

- Consiste em, a partir de um nó, descobrir todos os seus vizinhos.
- O procedimento é repetido para cada vizinho do nó original em ordem de descoberta.
- Cada nó visitado é marcado para evitar loops.
- Implementável com uma fila!

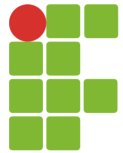


Busca em Largura

Algorithm 1: BFS(G, v)

Input: G, v

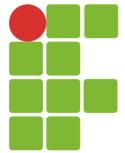
```
1  foreach u ∈ V do
2      u.color ← WHITE
3      u.π ← NIL
4  Q ← ∅
5  Q.PUSH(v)
6  v.color ← GREY
7  while ¬Q.EMPTY() do
8      u ← Q.POP()
9      foreach v ∈ (u, v) do
10         if( v.color = WHITE )
11             v.color ← GREY
12             v.π ← u
13             Q.PUSH(v)
14     u.color ← BLACK
```



Busca em Largura

Complexidade

- $\Theta(|V| + |E|)$ com listas de adjacências.
- $\Theta(|V|^2)$ com matrizes de adjacências.



Busca em Largura

Complexidade

- $\Theta(|V| + |E|)$ com listas de adjacências.
- $\Theta(|V|^2)$ com matrizes de adjacências.
- Por que?

Busca em Largura

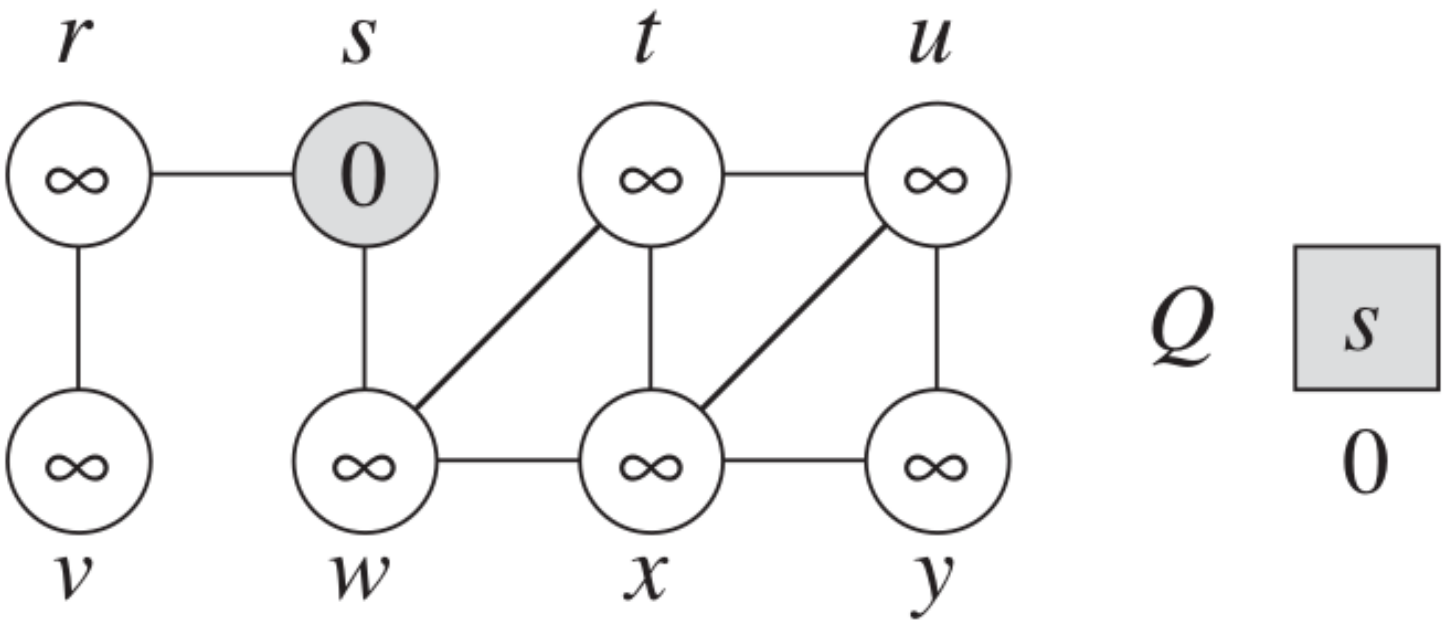


Figura: Como ficaria a busca em largura para este grafo?

Busca em Largura

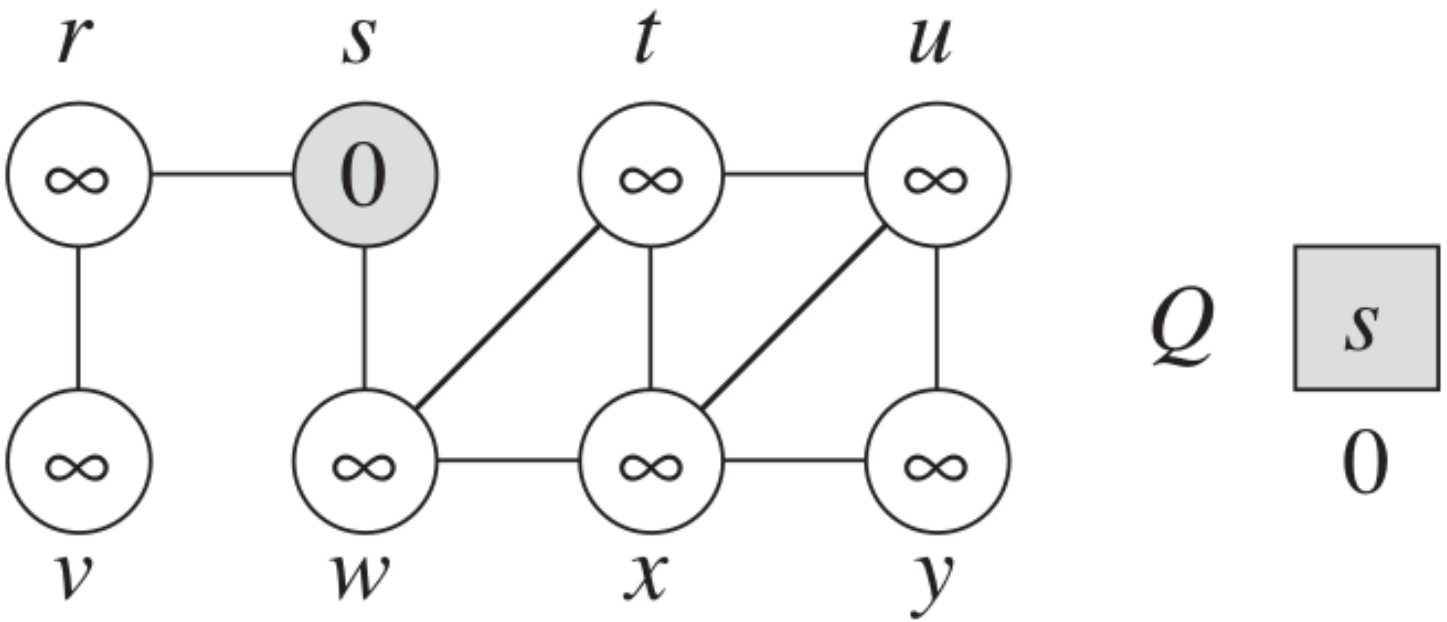
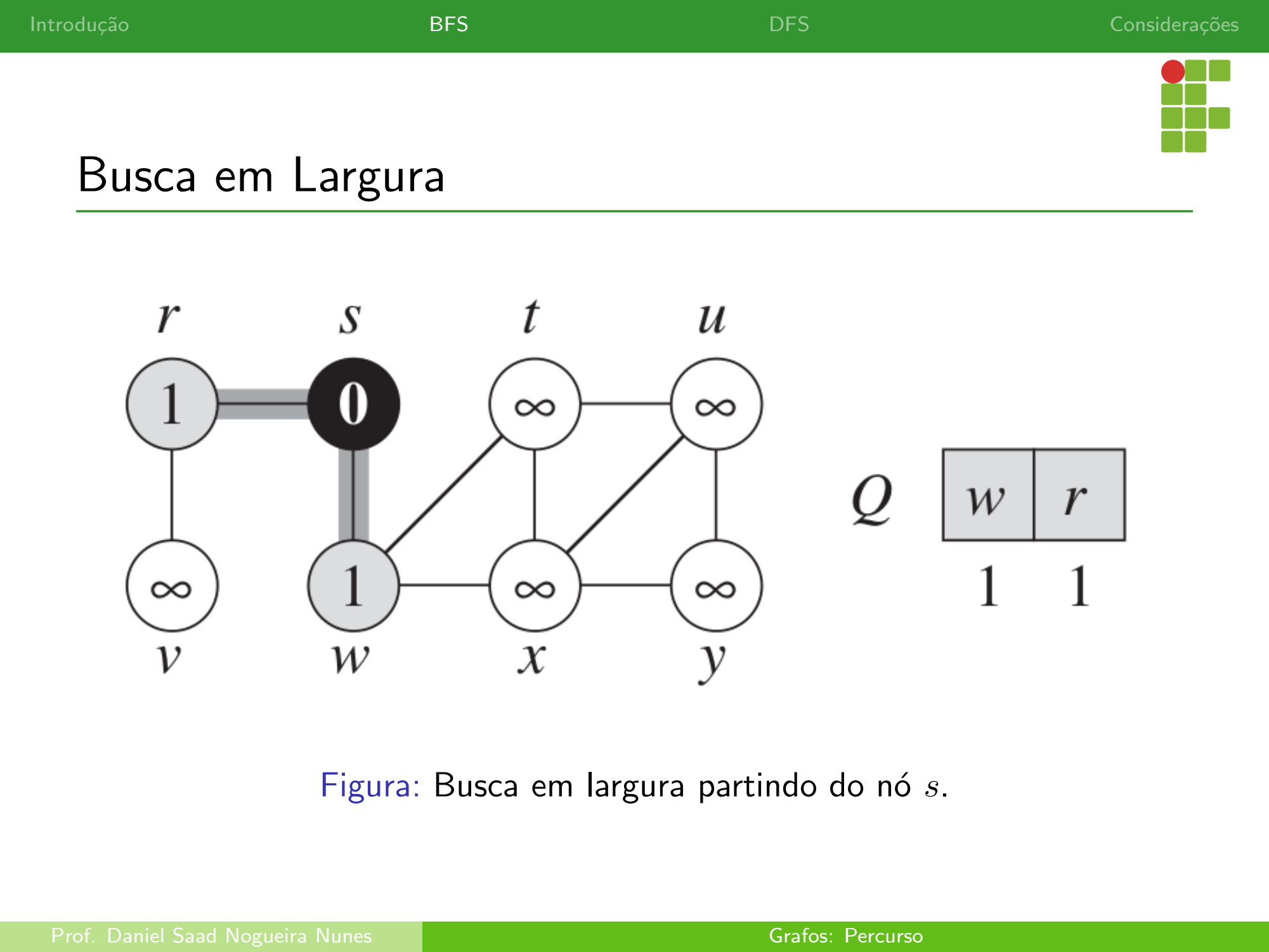


Figura: Busca em largura partindo do nó s.



Busca em Largura

Figura: Busca em largura partindo do nó s.

Busca em Largura

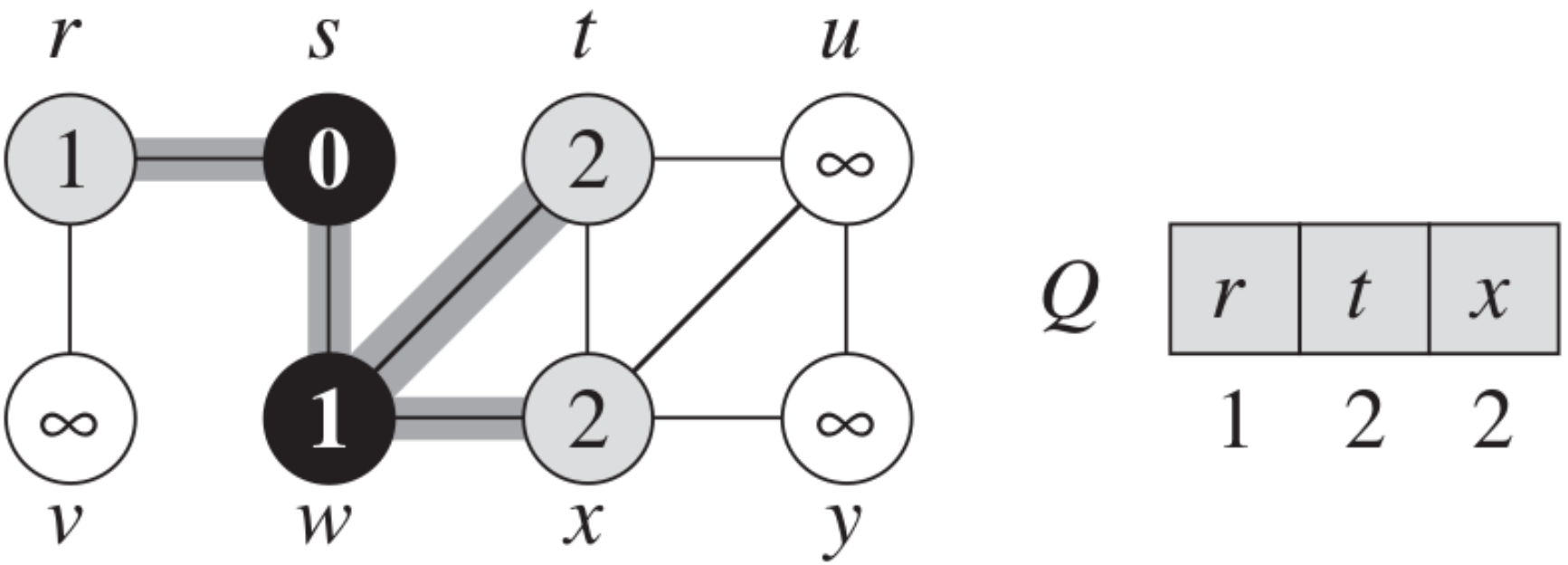


Figura: Busca em largura partindo do nó s.

Busca em Largura

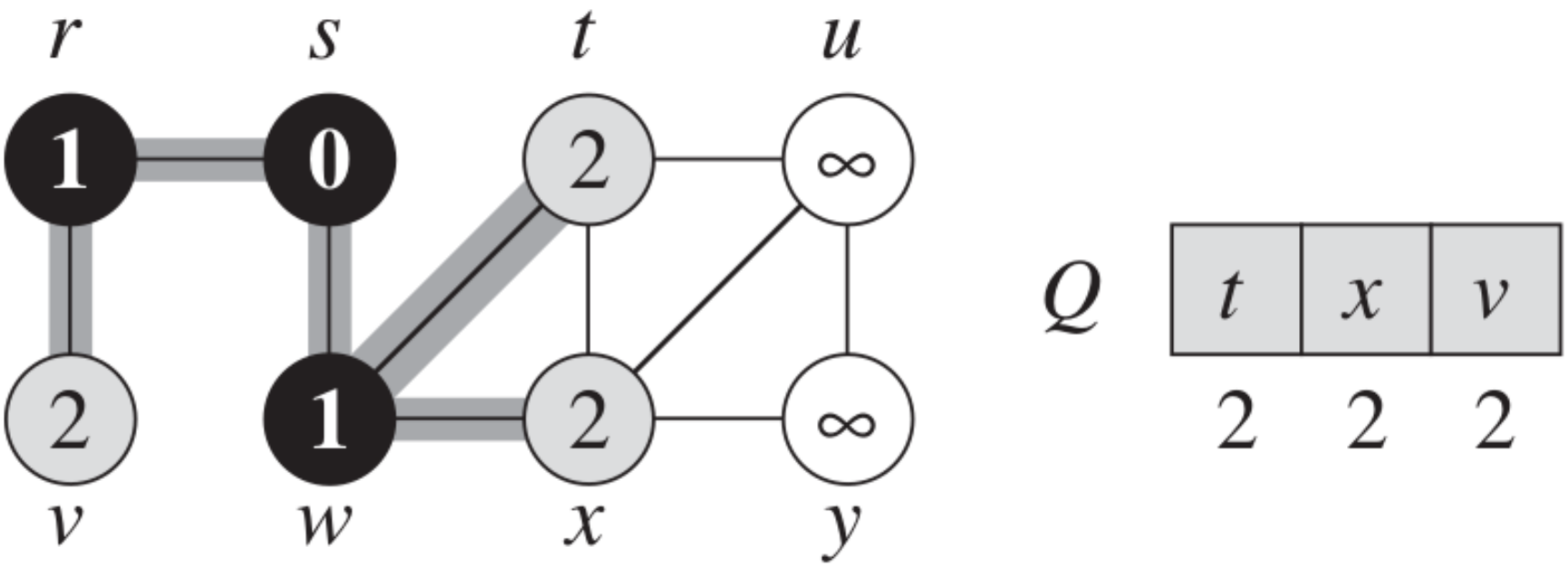


Figura: Busca em largura partindo do nó s.

Busca em Largura

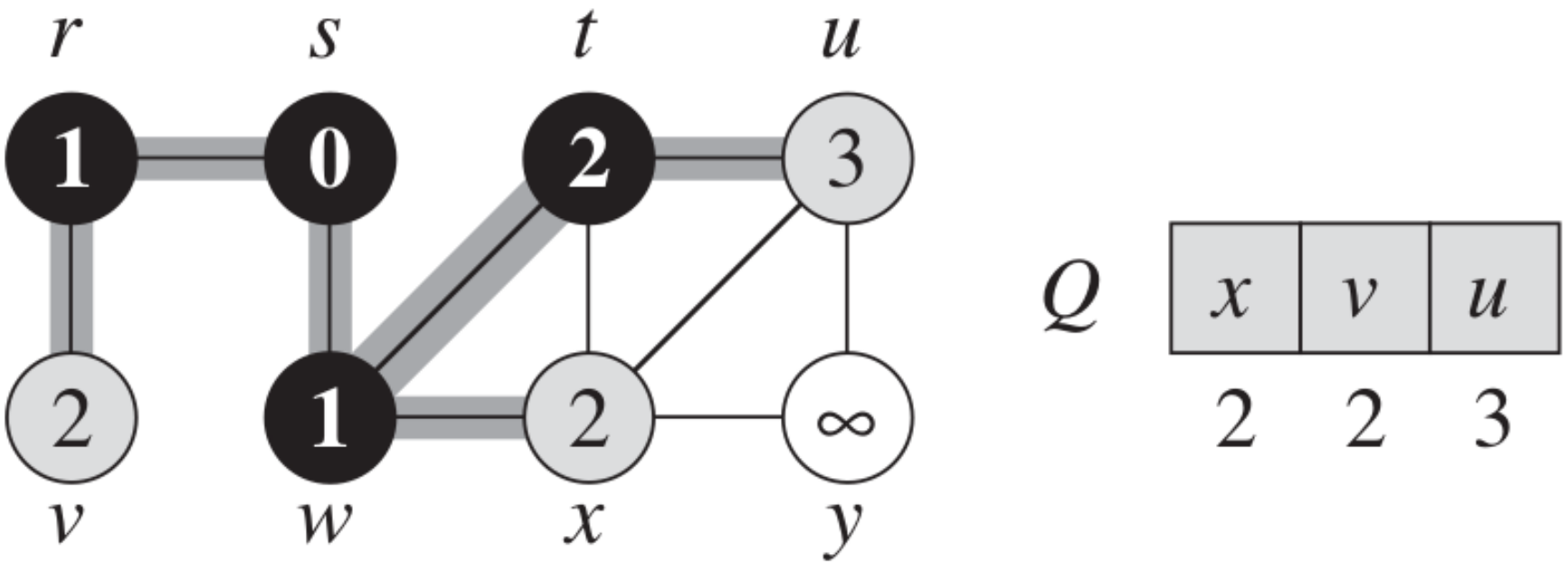


Figura: Busca em largura partindo do nó s.

Busca em Largura

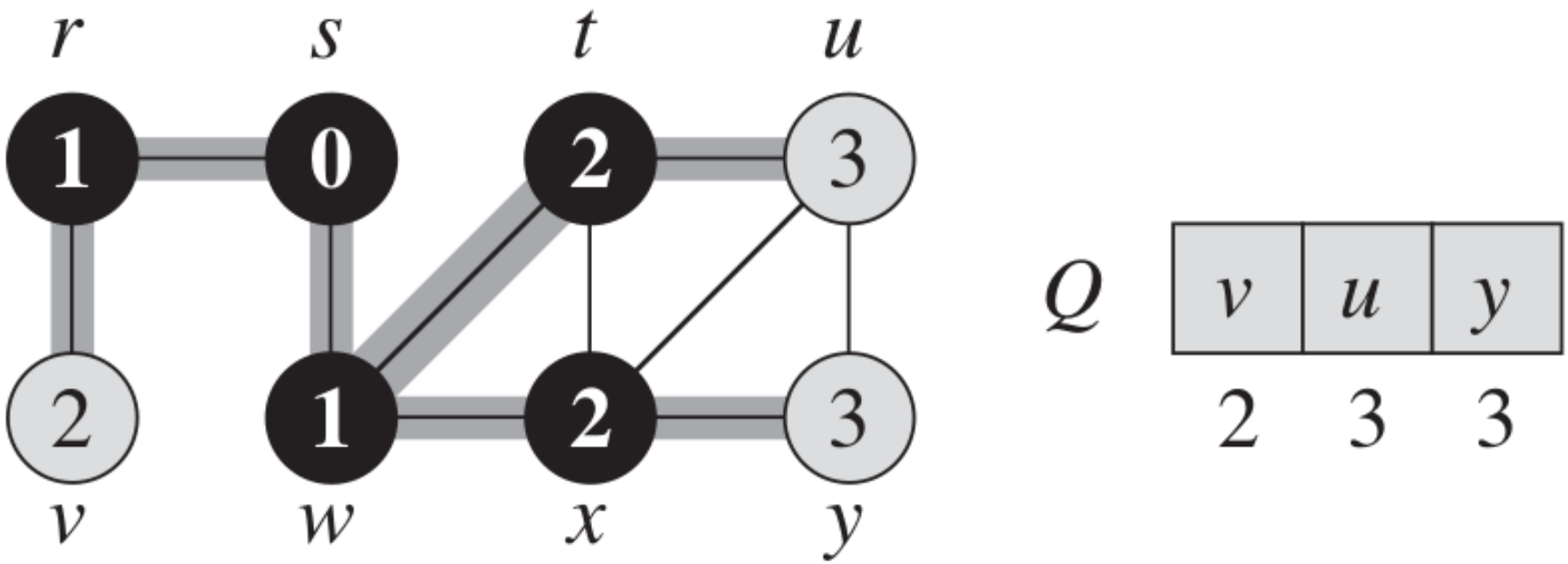


Figura: Busca em largura partindo do nó s.

Busca em Largura

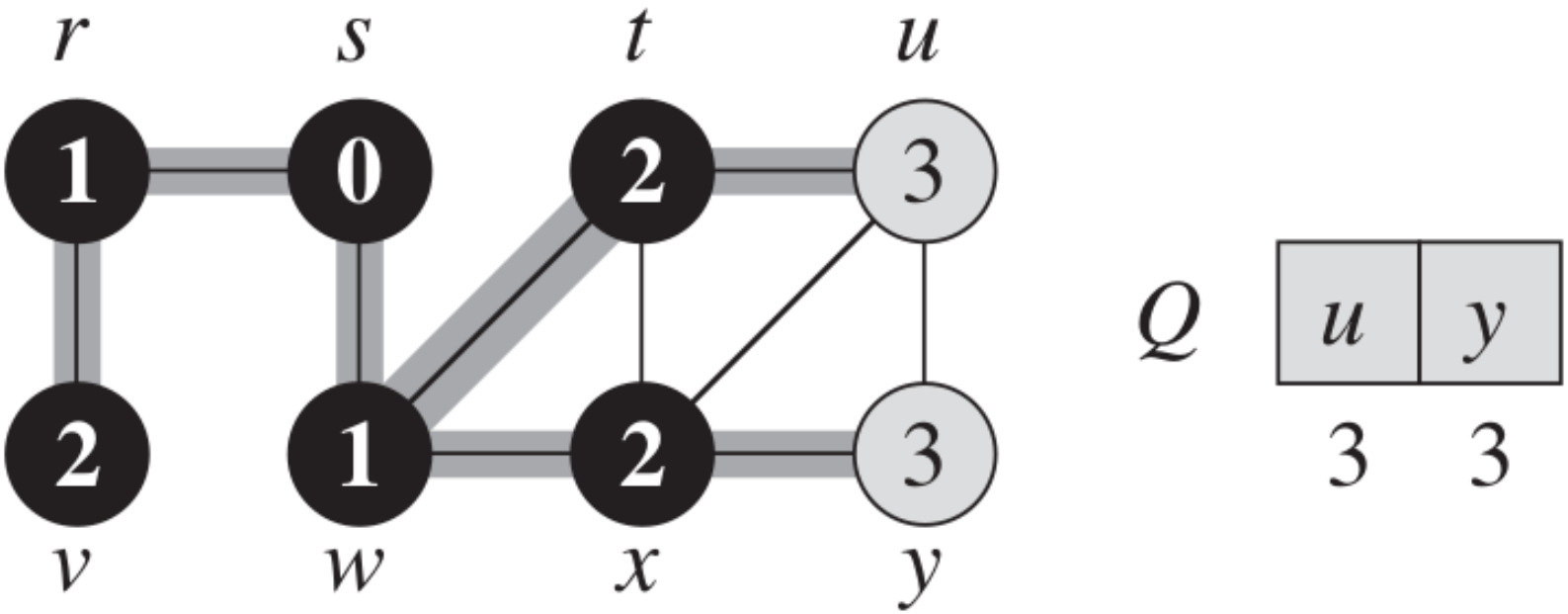


Figura: Busca em largura partindo do nó s.

Busca em Largura

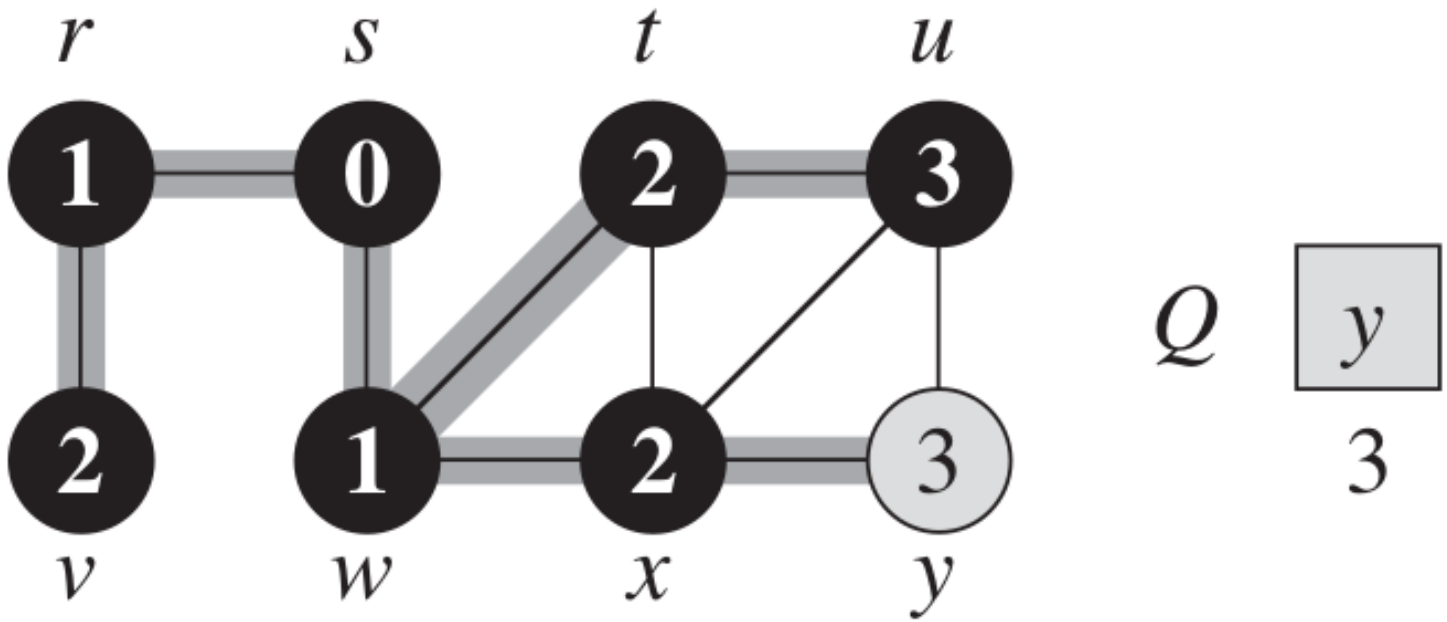


Figura: Busca em largura partindo do nó s.

Busca em Largura

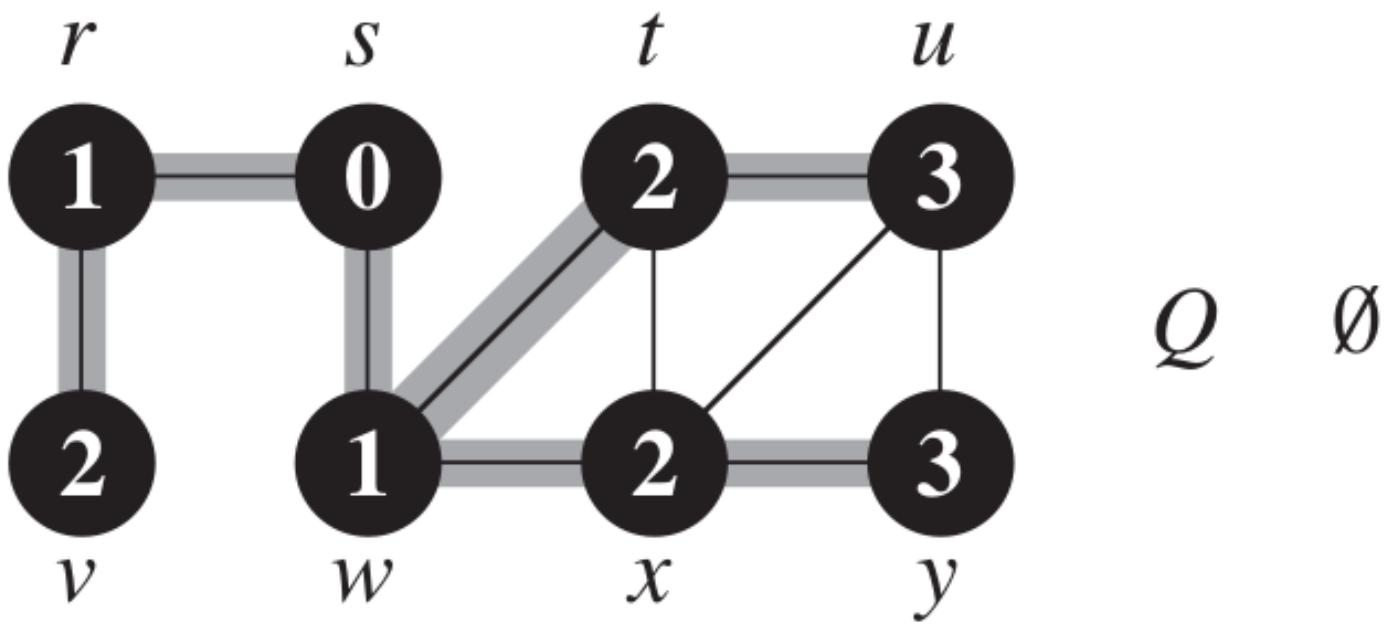
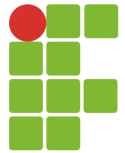
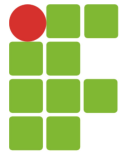


Figura: Busca em largura partindo do nó s.



Sumário



Busca em Profundidade

DFS

- A busca em profundidade parte de um determinado nó e avança para o seu vizinho imediato.
- Recursivamente, repetimos a mesma ideia.
- Apenas após ter ido à profundidade máxima, passamos para o próximo vizinho.

Depth-First-Search

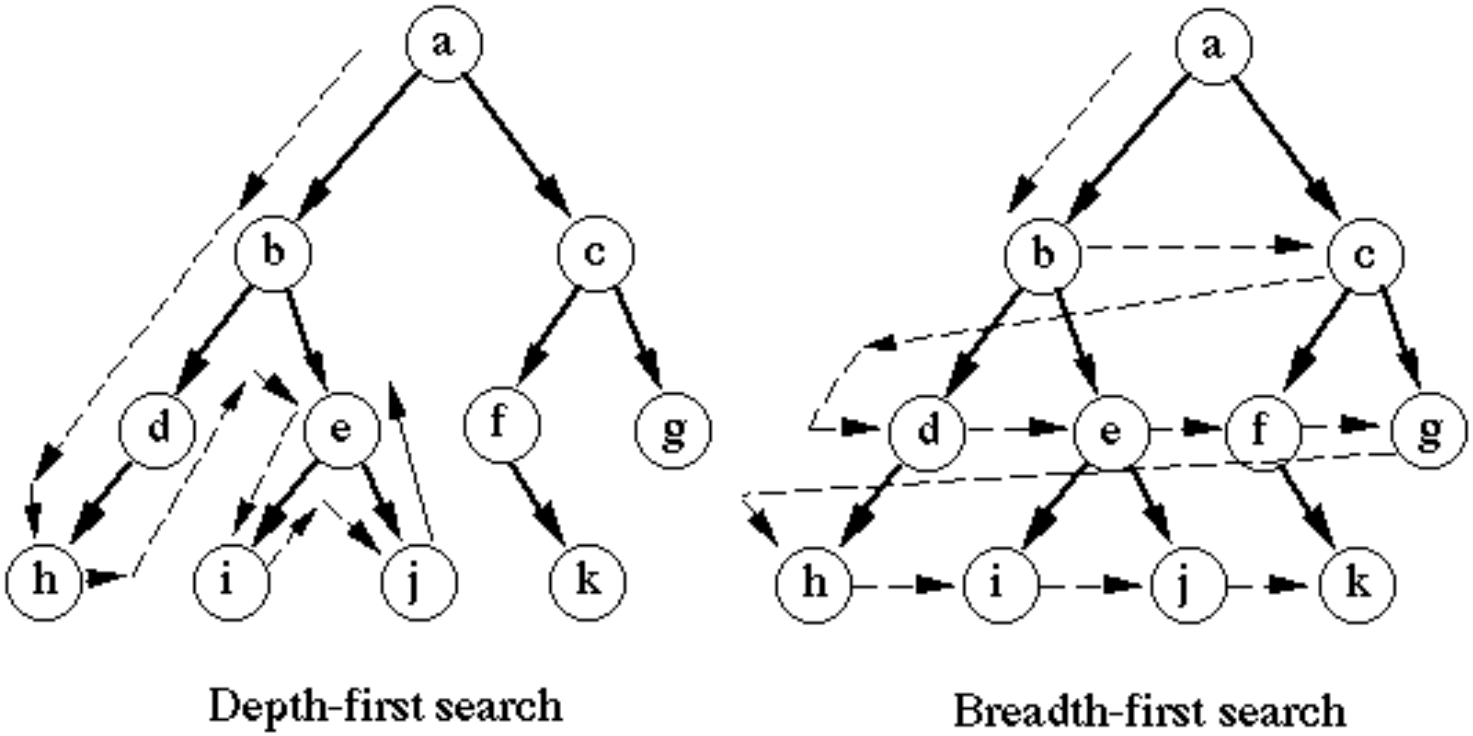
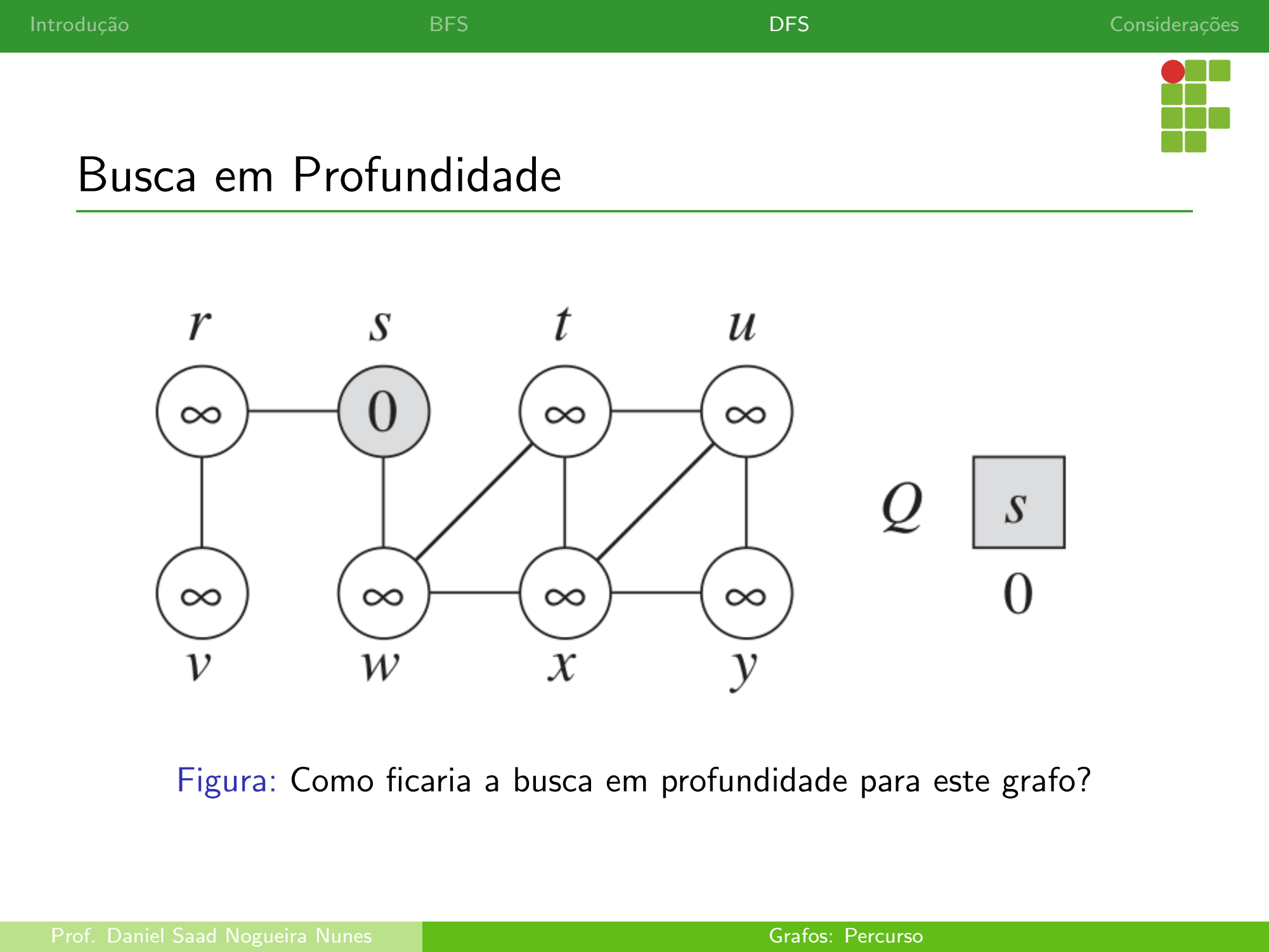
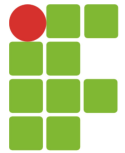


Figura: Busca em Largura vs Busca em Profundidade.



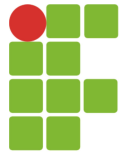
Busca em Profundidade

Figura: Como ficaria a busca em profundidade para este grafo?



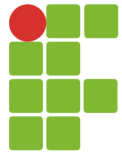
Busca em Profundidade

- Como implementar a busca em profundidade?



Busca em Profundidade

- Como implementar a busca em profundidade?
- Busca em largura trocando fila por pilha!

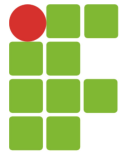


Busca em Profundidade

Algorithm 2: $\mathrm{DFS}(G, v)$

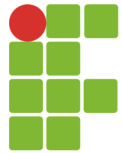
Input: G, v

```
1  foreach u ∈ V do
2      u.color ← WHITE
3      u.π ← NIL
4  S ← ∅
5  S.PUSH(v)
6  v.color ← GREY
7  while ¬S.EMPTY() do
8      u ← S.POP()
9      foreach v ∈ (u, v) do
10         if( v.color = WHITE )
11             v.color ← GREY
12             v.π ← u
13             S.PUSH(v)
14     u.color ← BLACK
```



Busca em Profundidade

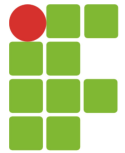
- Podemos implementar recursivamente também.
- Mais simples e mais elegante.
- Pilha implícita.



Busca em Profundidade

Complexidade

- $\Theta(|V| + |E|)$ com listas de adjacências.
- $\Theta(|V|^2)$ com matrizes de adjacências.

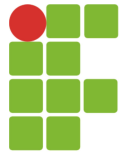


Busca em Profundidade

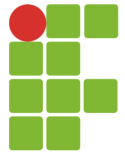
Algorithm 3: $\mathrm{DFS}(G, v)$

Input: G, v

1 $v.color \leftarrow GREY$
2 **foreach** $w \in (v, w)$ **do**
3 **if**($w.color = WHITE$)
4 $w.\pi \leftarrow v$
5 $\mathrm{DFS}(G, w)$
6 $v.color \leftarrow BLACK$



Sumário



Considerações

- Os algoritmos de busca são essenciais para resolver problemas sobre grafos.
- Aplicáveis em qualquer representação: listas ou matrizes de adjacências.
- Vários problemas podem ser resolvidos com uma simples adaptação da busca em largura ou profundidade.