INTELIGÊNCIA ARTIFICIAL

PARTE 2

Grafos e Algoritmos de Busca Exaustiva

- *Breadth-First Search*
- *Depth-First Search*

Prof. Dr. Celso Gallão - 2017

1

Introdução aos Grafos

1 - Introdução aos Grafos

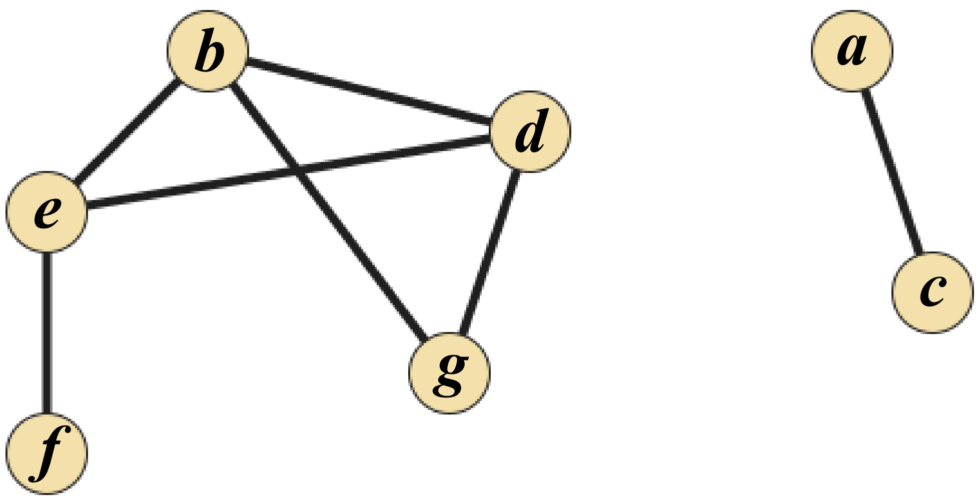
Definições:

- Grafos são estruturas representadas por $\boldsymbol{G(V, E)}$, onde $\boldsymbol{V}$ é um conjunto não vazio de **vértices** e $\boldsymbol{E}$ é um conjunto de pares não orientados de $\boldsymbol{V}$, chamado **arestas** *(edges)*. Cada aresta é formada por um par de vértices.

Exemplo:

$V = \{a, b, c, d, e, f, g\}$

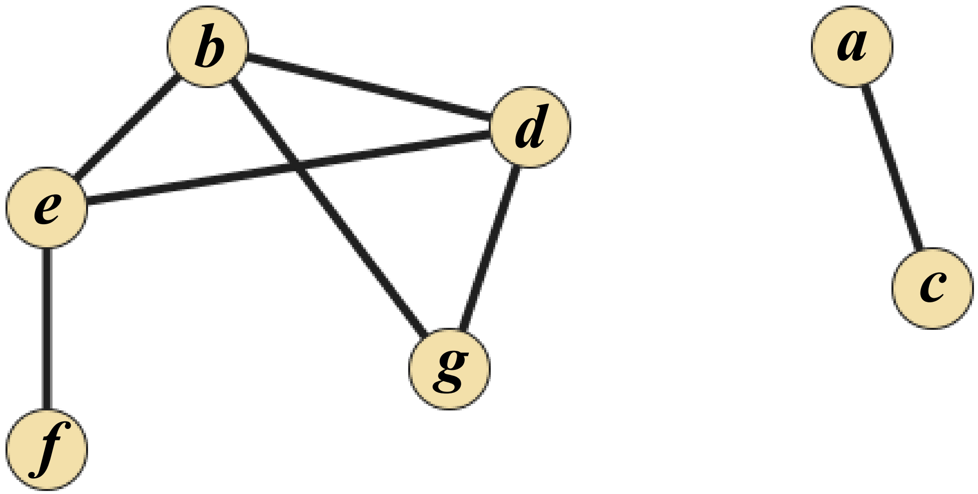
$E = \{(a, c), (b, d), (b, e), (b, g), (d, e), (d, g), (e, f)\}$



1 - Introdução aos Grafos

Propriedades:

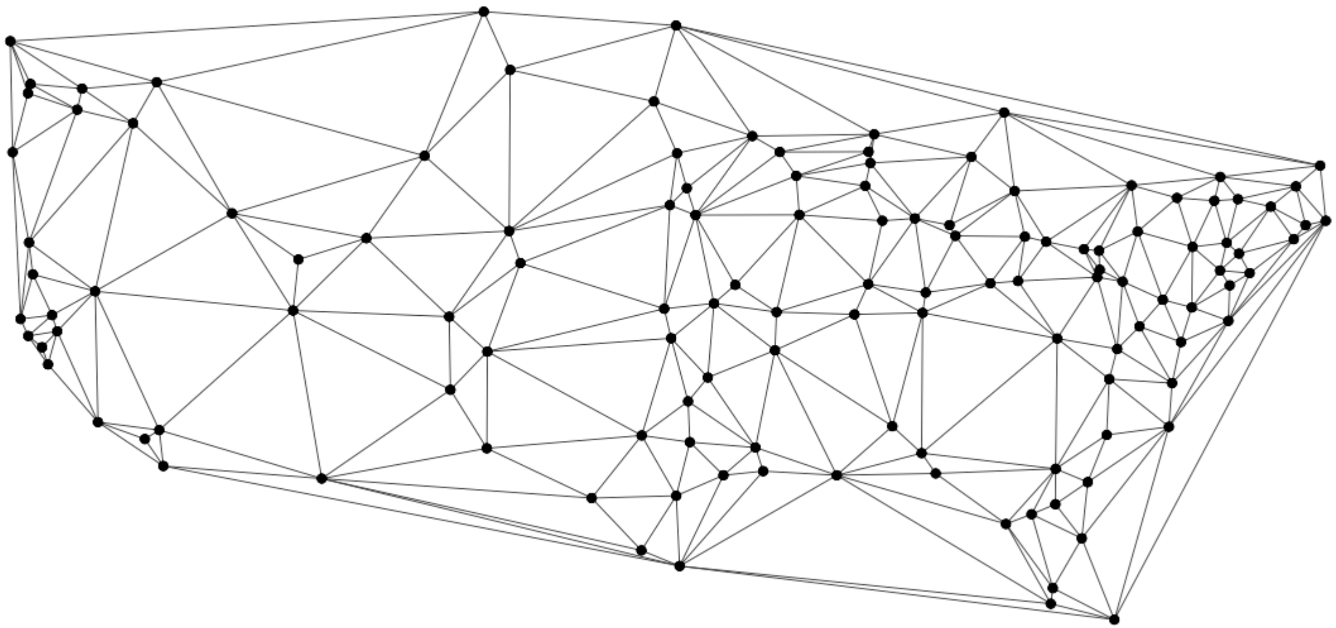
- Vértices ***a*** e ***c*** são **adjacentes**. Vértices ***f*** e ***g*** não!
- Vértice ***b*** e aresta ***(b,g)*** são **incidentes**. Vértice ***f*** e aresta ***(b,g)*** não!
- A vizinhança de ***b*** é ***N(b)={d,e,g}***
- O grau do vértice ***b*** é ***3***.



1 - Introdução aos Grafos

Grafo Não-orientado:

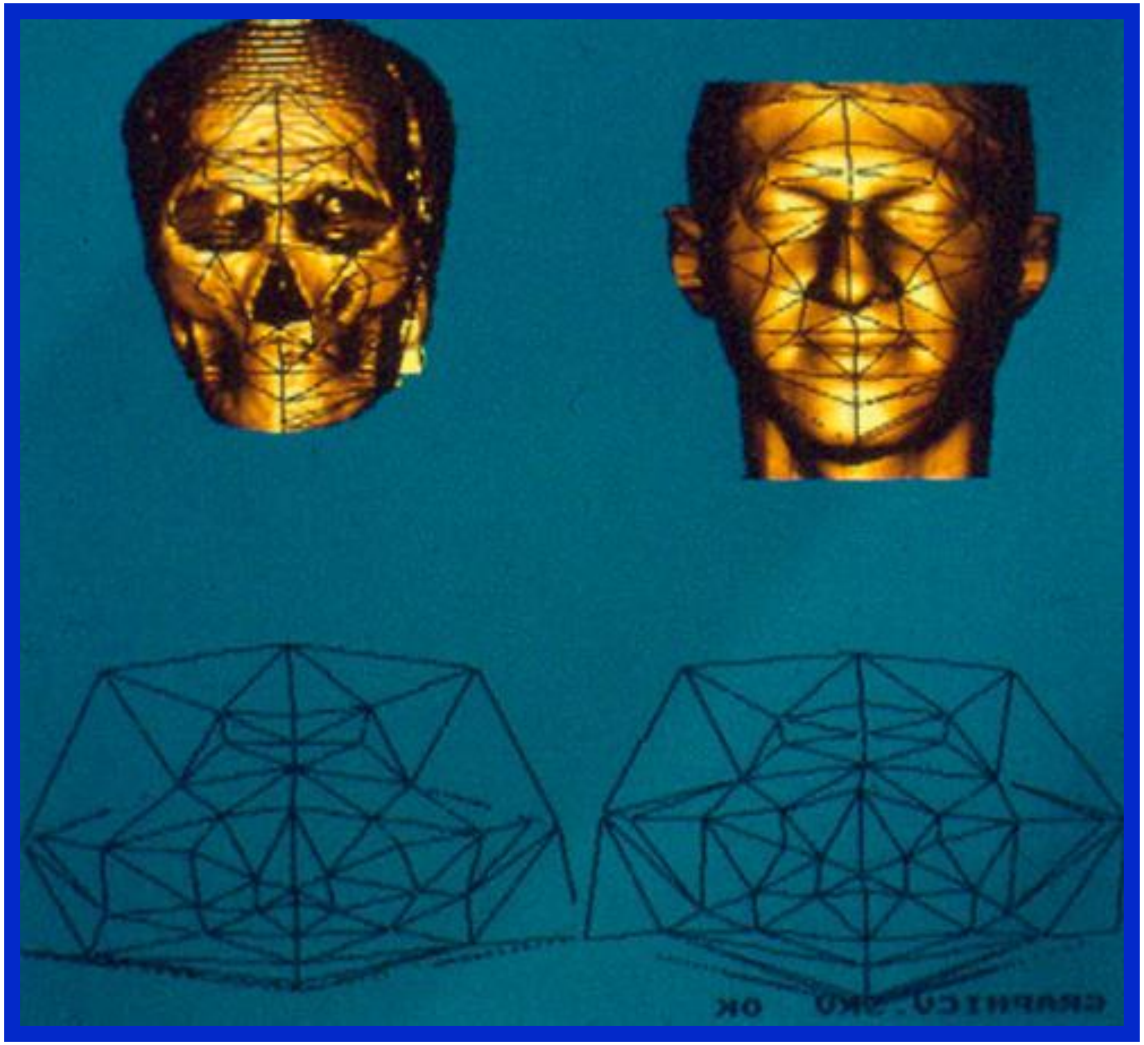
- Arestas são pares não-orientados ***(v,w)***, ou seja, a aresta ***(v,w)*** e a aresta ***(w,v)*** são a mesma aresta.
- Não há direção!



Este grafo representa 128 cidades dos EUA
(extraído do livro *The Stanford GraphBase*.)

1 - Introdução aos Grafos

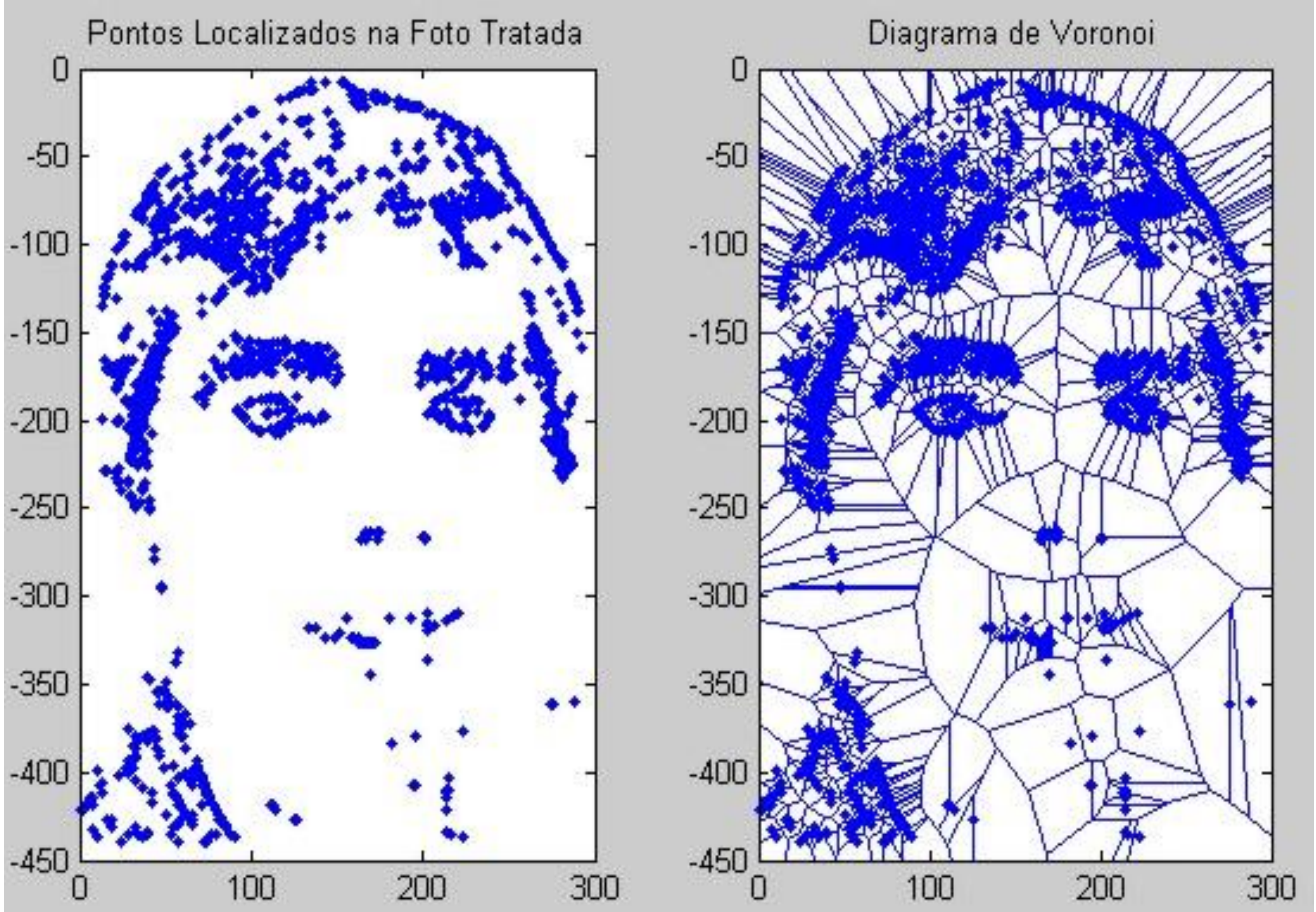
Grafo Não-orientado:



Exemplo de **Triangularização de Delaunay** extraído das aulas de Geometri a Computacional do Prof. Dr. Paulo Sérgio Rodrigues (FEI/2010)

1 - Introdução aos Grafos

Grafo Não-orientado:

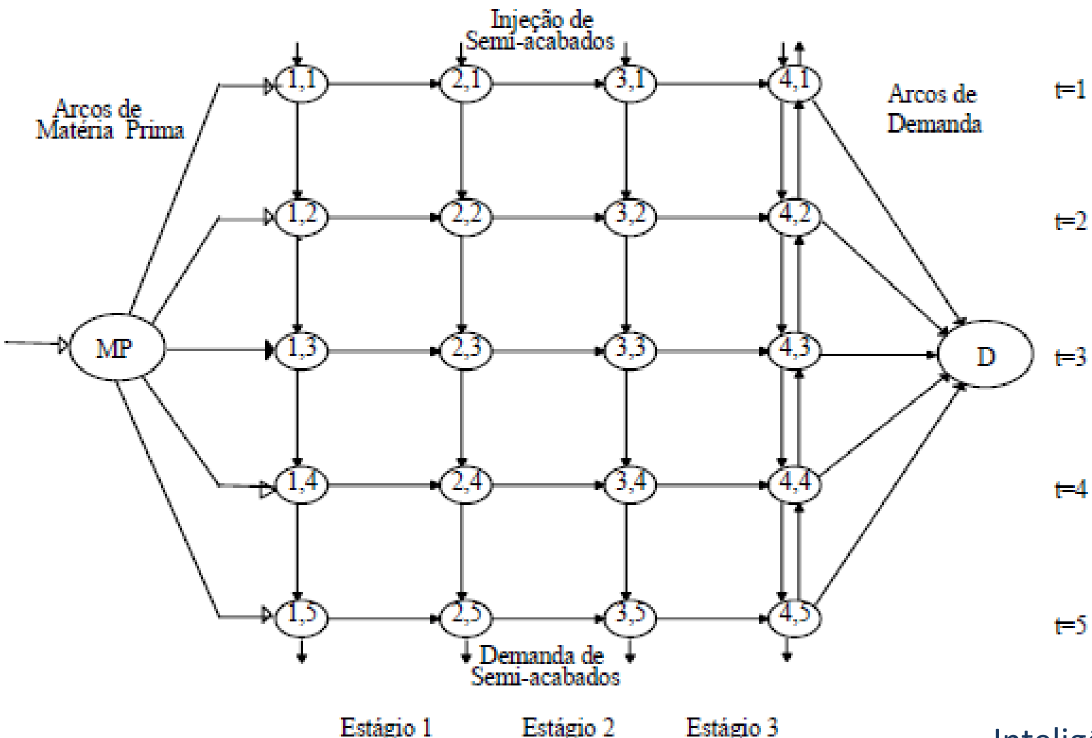


Exemplo de
Diagrama de Voronoi

1 - Introdução aos Grafos

Grafo Orientado:

- Arestas são pares orientados ***(v,w)***, ou seja, a aresta ***(v,w)*** e a aresta ***(w,v)*** são diferentes.
- Aresta ***(v,w)*** é direcionada ***v*** → ***w***, onde ***v*** é a origem (*source*) e ***w*** é o alvo (*target*).

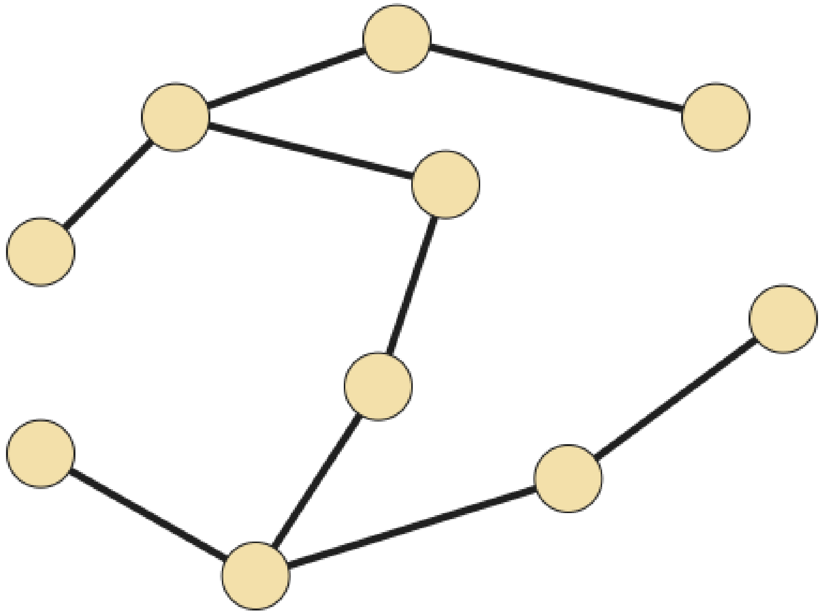


Este grafo representa um sistema produtivo através de uma linha de montagem serial, com 3 estágios e 5 períodos.

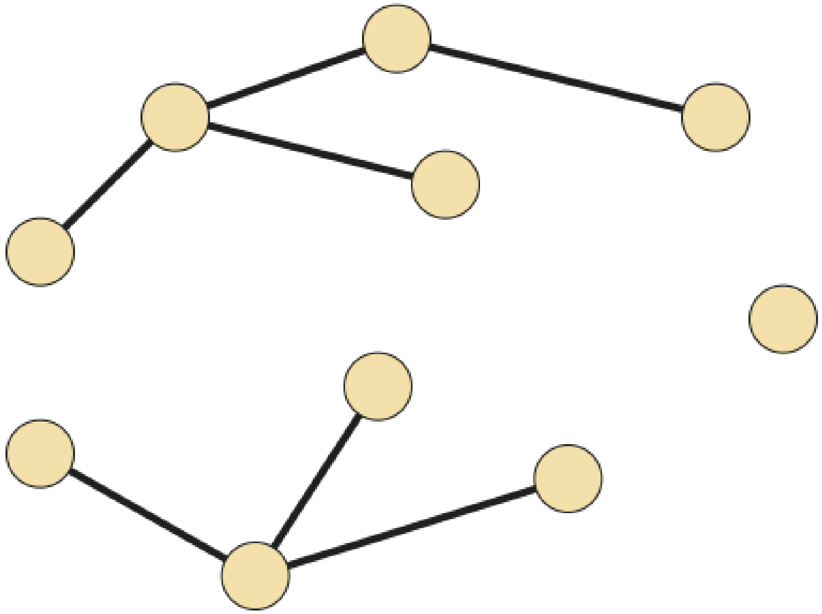
1 - Introdução aos Grafos

Árvores e Florestas:

Uma **árvore** é um grafo que é conectado e não contém ciclos.



Uma **floresta** é um grafo que não contém ciclos. Os componentes conectados de uma floresta são árvores



1 - Introdução aos Grafos

Representação por <u>MATRIZ</u> de Adjacências:

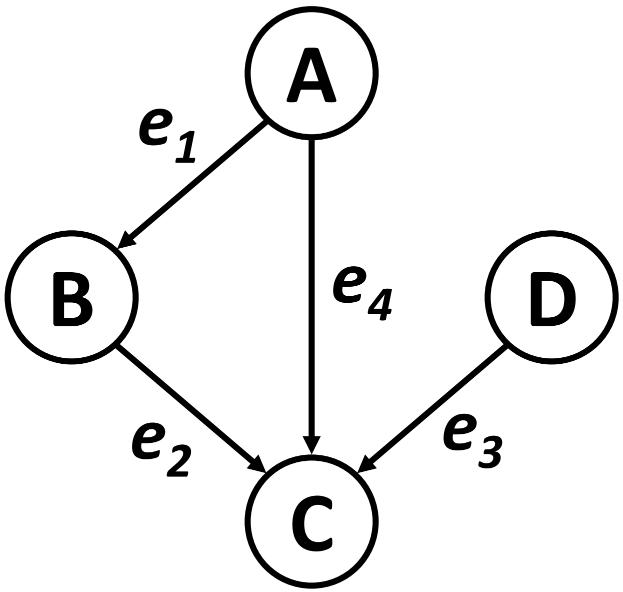
- Assuma $V = \{1, 2, \ldots, n\}$
- Uma matriz de adjacências representa um grafo como uma matriz $A_{n \times n}$:

$$A[i, j] \begin{cases} = 1 \text{ se aresta } (i, j) \in E \\ = 0 \text{ se aresta } (i, j) \notin E \end{cases}$$

1 - Introdução aos Grafos

Representação por MATRIZ de Adjacências:

- Exemplo de **grafo orientado**:



↗	A	B	C	D
A	0	**1**	**1**	0
B	0	0	**1**	0
C	0	0	0	0
D	0	0	**1**	0

1 - Introdução aos Grafos

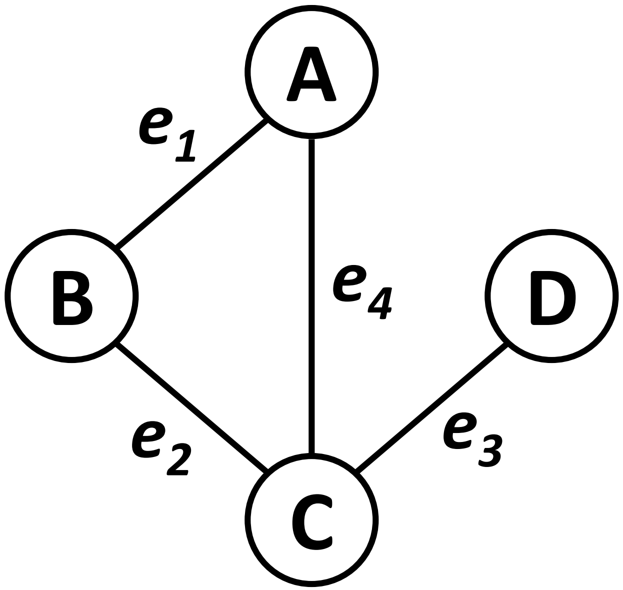
Representação por <u>MATRIZ</u> de Adjacências:

- **A Matriz de Adjacências requer quanto de espaço?**
  - Matriz tem tamanho $V \times V$ → $\Theta(V^2)$
  - A **notação assintótica** serve tanto para **tempo de execução** quanto para **espaço de armazenamento**.
  - Por exemplo, para o espaço $\Theta(g(n))$, revela o quanto a ocupação do espaço irá crescer quando n aumentar!

1 - Introdução aos Grafos

Representação por <u>MATRIZ</u> de Adjacências:

- Para um grafo **não-orientado**:
  - A Matriz de Adjacências é <u>**simétrica**</u>.
  - Podemos armazenar apenas metade da matriz.
- Exemplo de grafo <u>**não-orientado**</u>:



	A	**B**	**C**	**D**
A	0	**1**	**1**	0
B	**1**	0	**1**	0
C	**1**	**1**	0	**1**
D	0	0	**1**	0

1 - Introdução aos Grafos

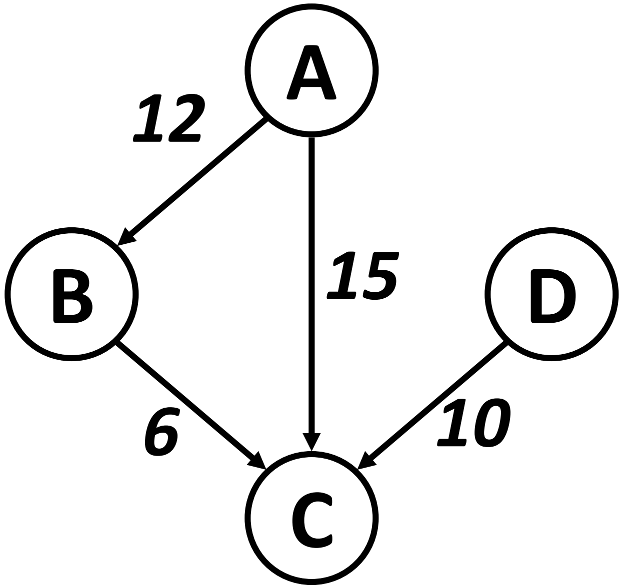
Representação por MATRIZ de Adjacências:

- A **Matriz de Adjacências**:
  - Usualmente armazena muita informação para grafos grandes.
  - Pode ser muito eficiente para grafos pequenos.
  - Geralmente usado para grafos densos.

1 - Introdução aos Grafos

Representação por <u>MATRIZ</u> de Adjacências:

- Para um grafo **ponderado**, ou seja com pesos em suas arestas, representa-se uma aresta por meio de seu peso:
- Exemplo de grafo **<u>orientado ponderado</u>**:



↗	A	B	C	D
A	0	**12**	**15**	0
B	0	0	**6**	0
C	0	0	0	0
D	0	0	**10**	0

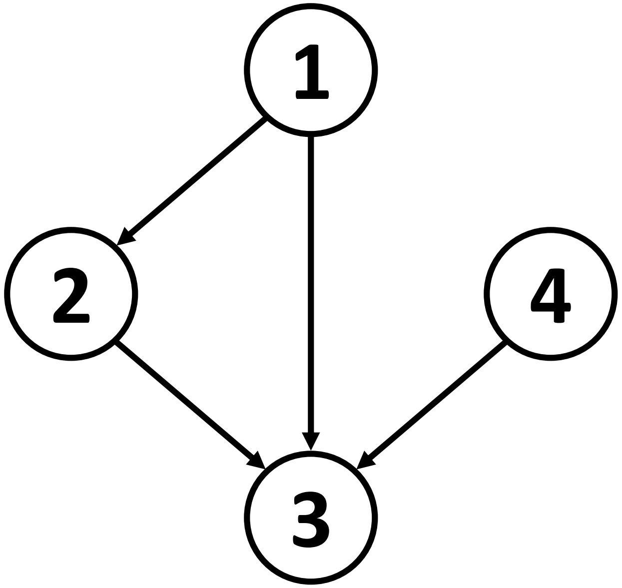
1 - Introdução aos Grafos

Representação por <u>LISTA</u> de Adjacências:

- Para cada vértice $v \in V$, armazena-se a lista de vértices adjacentes a v.

Exemplo:

- $Adj[1] = \{2,3\}$
- $Adj[2] = \{3\}$
- $Adj[3] = \{\}$
- $Adj[4] = \{3\}$



1 - Introdução aos Grafos

Representação por LISTA de Adjacências:

- **A Lista de Adjacências requer quanto de espaço?**

  O grau de cada vértice v é igual a quantidade de arestas incidentes, sendo assim, é necessário $\Theta(V+E)$ de espaço para armazenamento:

  - Para **grafos orientados**, o número de itens na lista de adjacências é a somatória do grau de saída $(v) = |E|$. É preciso armazenar cada vértice $v \in V$ mais a lista de arestas E, portanto, $\Theta(V + E)$.
  - Para **grafos não-orientados**, o número de itens na lista de adjacências é a somatória do grau $(v) = 2|E|$, que também é $\Theta(V + E)$.

1 - Introdução aos Grafos

Características dos Algoritmos de Busca em Grafos:

- Algoritmos de Busca são técnicas de Inteligência Artificial aplicadas à problemas de **alta complexidade** teórica que não são resolvidos com técnicas de programação convencionais, principalmente as de natureza puramente numérica.
- A complexidade de um problema está diretamente relacionada ao tamanho do seu **Espaço de Busca** correspondente.

1 - Introdução aos Grafos

Objetivos dos Algoritmos de Busca em Grafos:

- Seja um grafo $G = (V,E)$, **orientado ou não**, o objetivo é explorar metodologicamente cada vértice e cada aresta do grafo, ou apenas construir uma árvore.
- Para tanto, toma-se um vértice como sendo a **raiz** e escolhe-se as arestas apropriadas para produzir a árvore. Pode-se construir uma floresta se o grafo não for conectado.

2

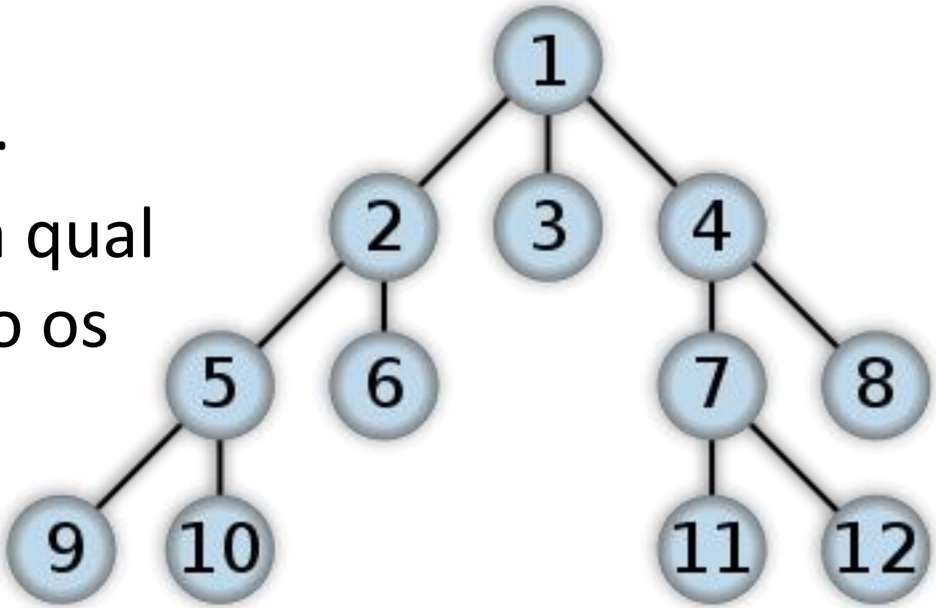
Busca em Largura ou Amplitude (*Breadth-First Search*)

2 - Busca em Largura (*Breadth-First Search*)

2.1 - Características:

Em uma **busca exaustiva em largura** a partir do vértice ***v***, espera-se que todos os vizinhos de ***v*** sejam visitados antes de continuar a busca mais profundamente:

- **BFS** encontra a menor distância até um certo nó, ou seja, o número mínimo de arestas.
- **BFS** constroi uma árvore na qual os caminhos de cada nó são os menores até a raiz.



2 - Busca em Largura (*Breadth-First Search*)

2.1 - Características:

- **Explora-se** todo o grafo, tornando-o uma árvore de busca:
  - Um vértice por vez.
  - Expande em largura a fronteira de vértices explorados.
  - Busca exaustiva (todos os nós são visitados).

- **Constrói-se** a árvore de busca sobre o grafo:
  - Toma-se um vértice pai como raiz (Geração 0),
  - Descobre-se todos os filhos (Geração 1),
  - Para cada filho (Geração 1) descobre-se todos os seus filhos (Geração 2), e assim por diante.

2 - Busca em Largura (*Breadth-First Search*)

2.2 - Metodologia BFS:

- Colorindo os vértices:
  - **Vértices brancos** ainda não foram considerados

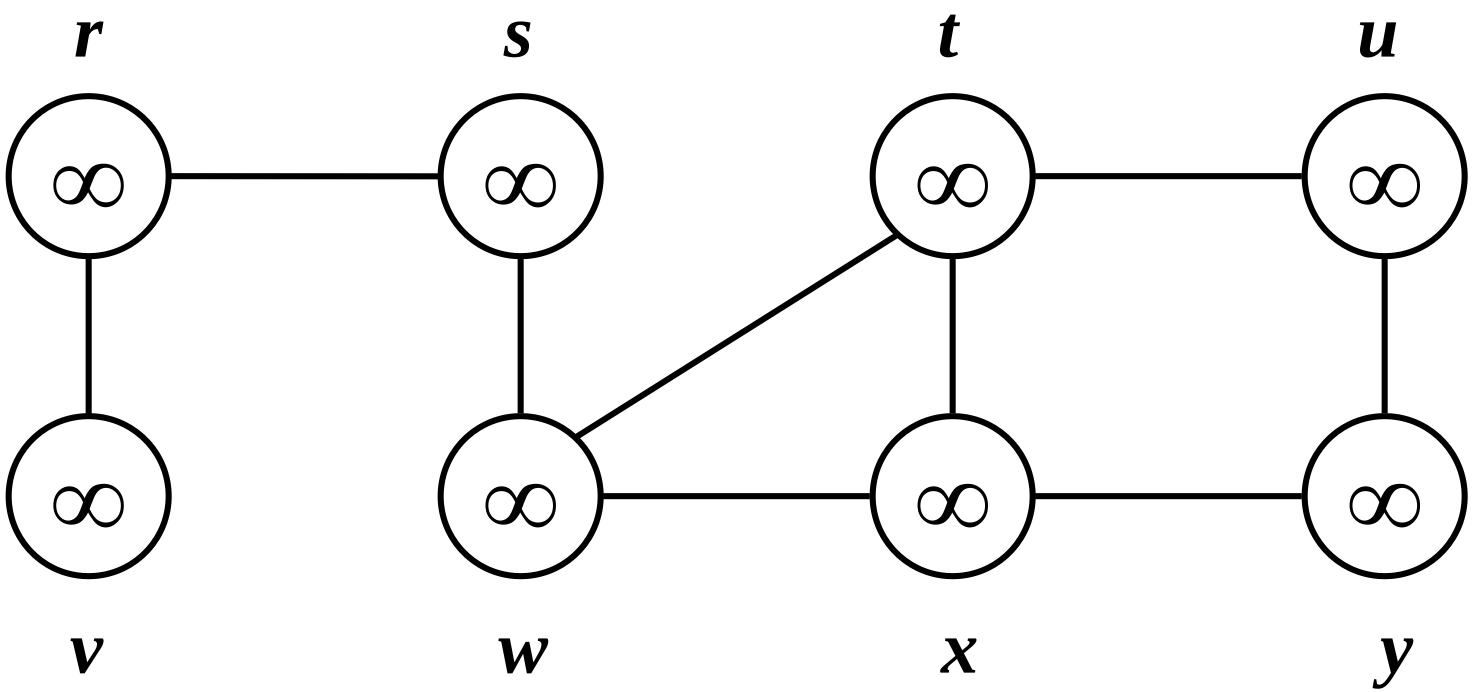
    Todos começam como branco.
  - **Vértices cinza** foram considerados mas não totalmente explorados

    Eles podem ser adjacentes de vértices brancos.
  - **Vértices pretos** foram totalmente explorados

    São vértices adjacentes a vértices pretos ou cinzas.
- Explorando vértices escaneando a lista de adjacências de vértices cinzas.

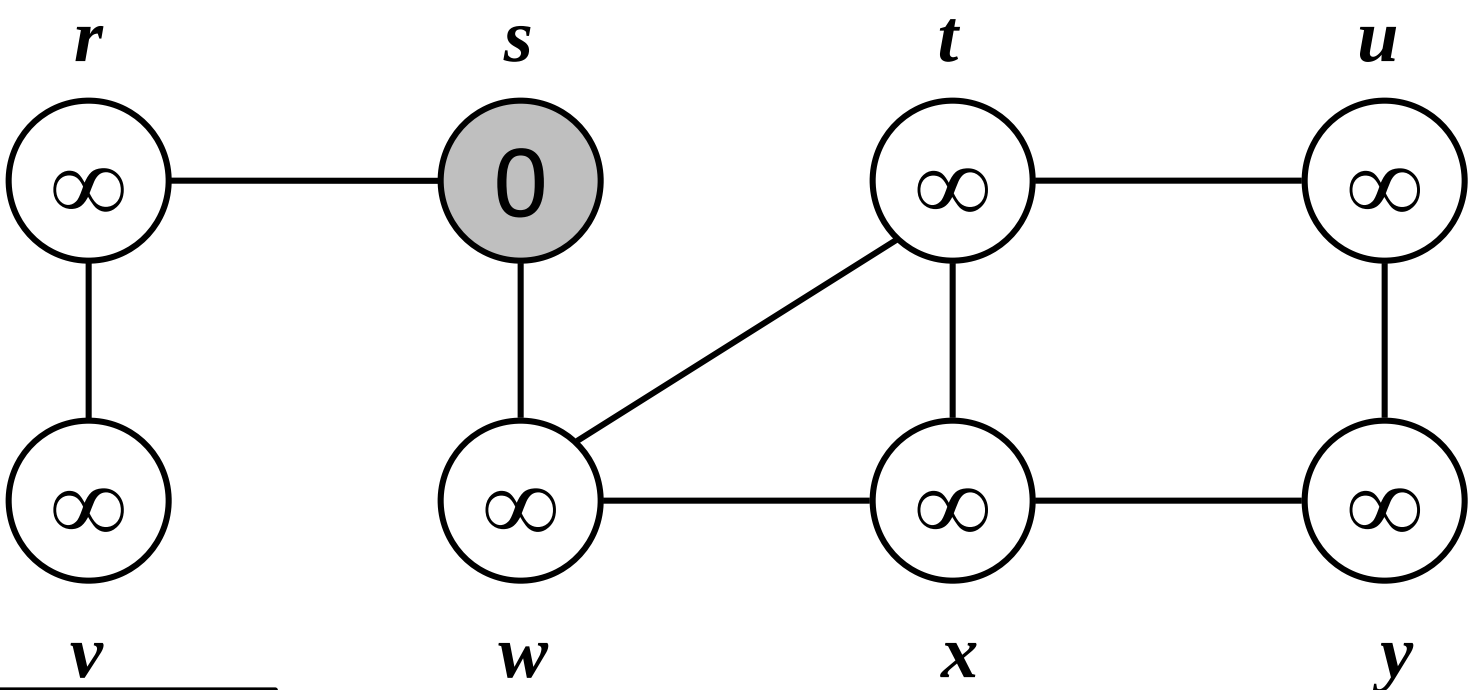
2 - Busca em Largura (*<u>B</u>readth-<u>F</u>irst <u>S</u>earch*)

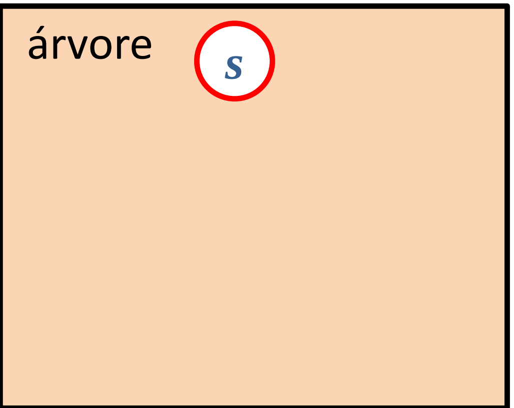
Exemplo BFS (iniciando no vértice *s*):



2 - Busca em Largura (*Breadth-First Search*)

Exemplo BFS:

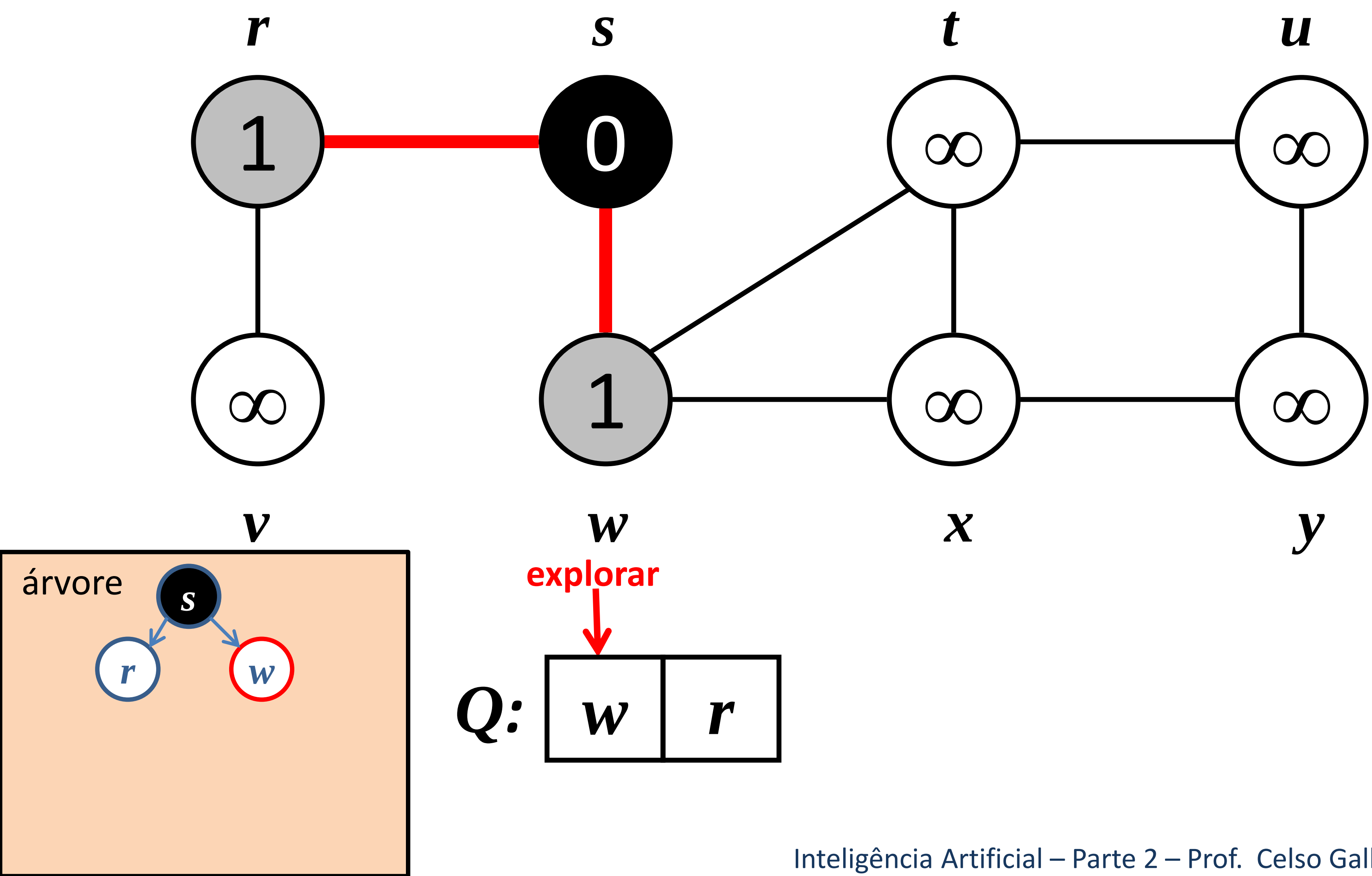




Q: $\boxed{s}$

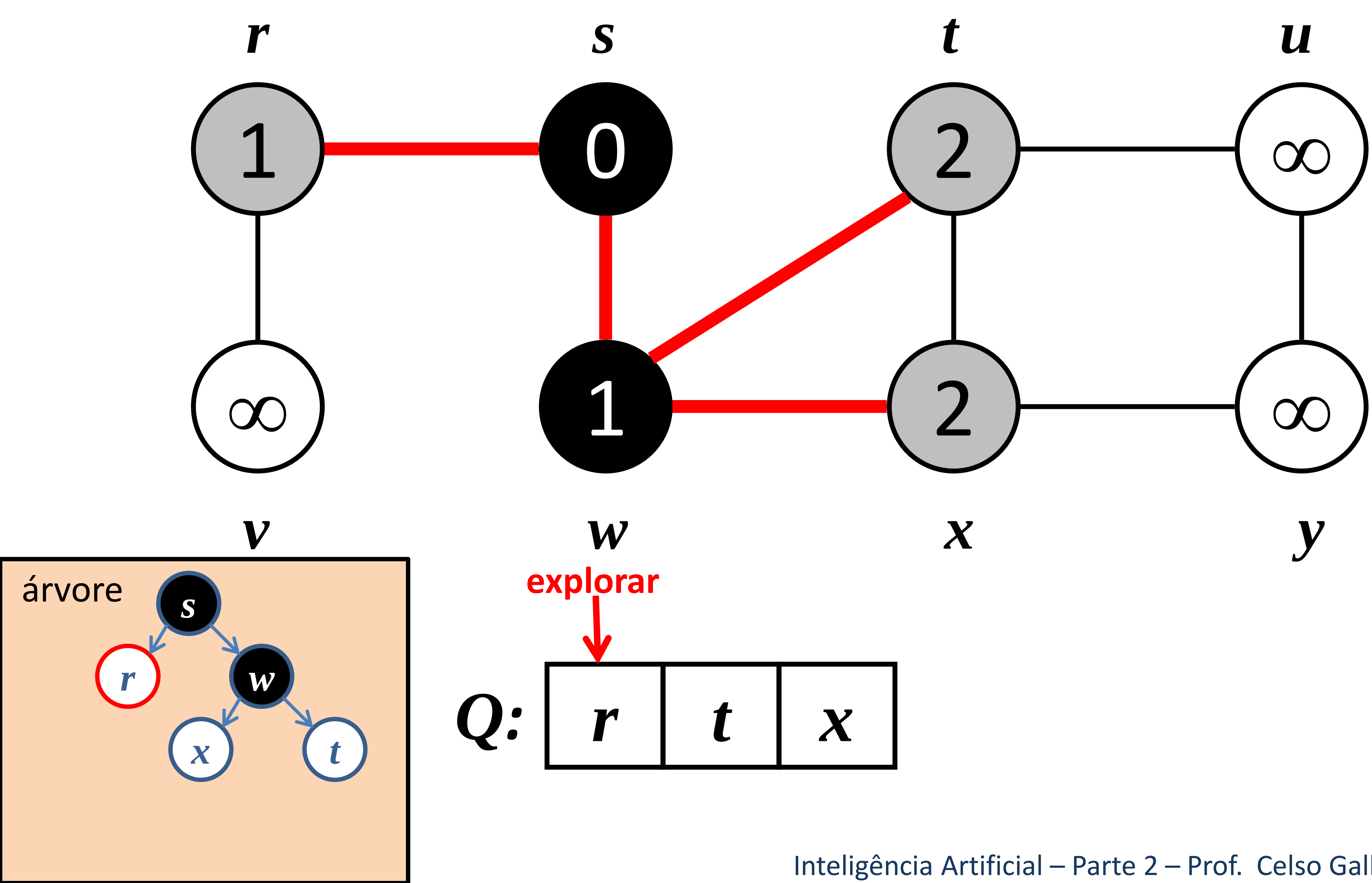
2 - Busca em Largura (*Breadth-First Search*)

Exemplo BFS:



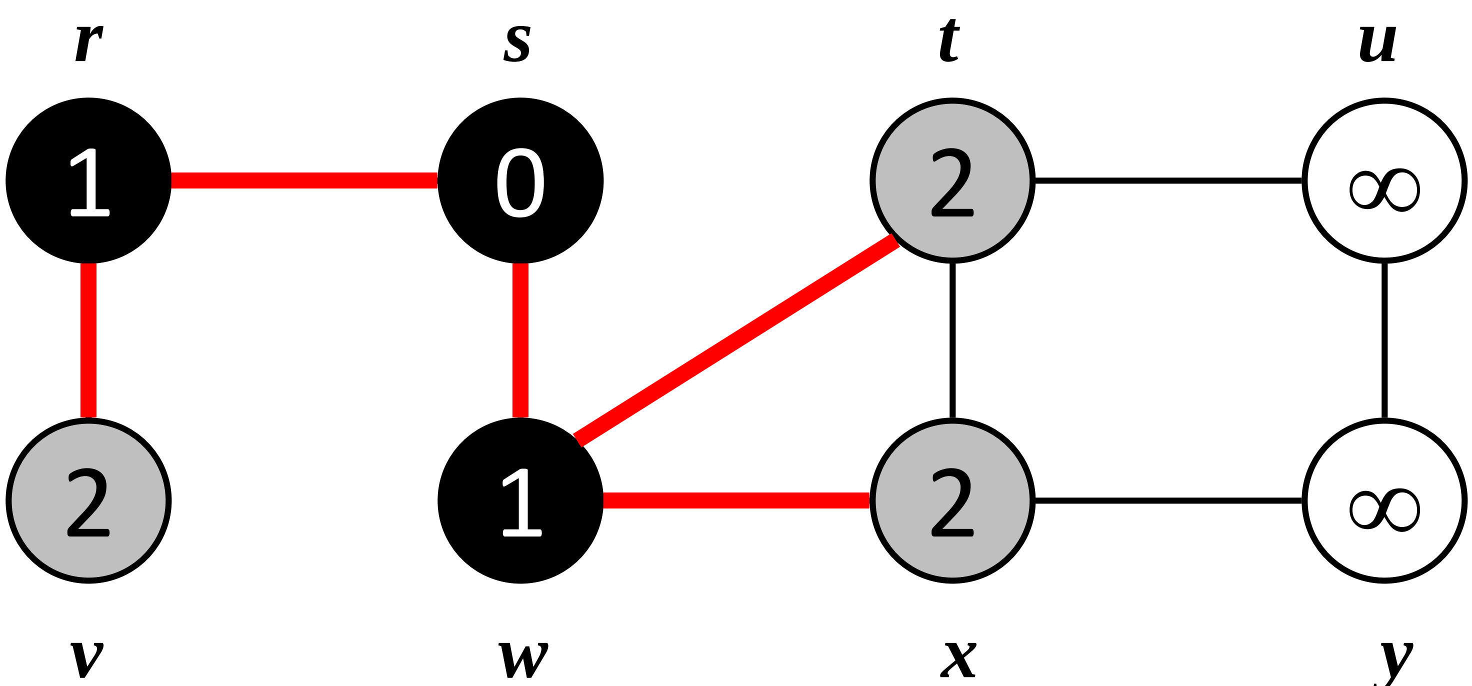
2 - Busca em Largura (*Breadth-First Search*)

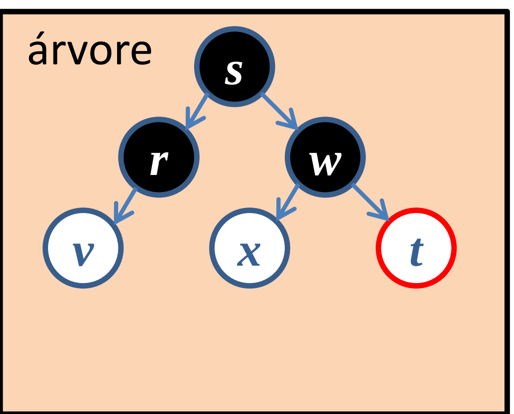
Exemplo BFS:



2 - Busca em Largura (*Breadth-First Search*)

Exemplo BFS:





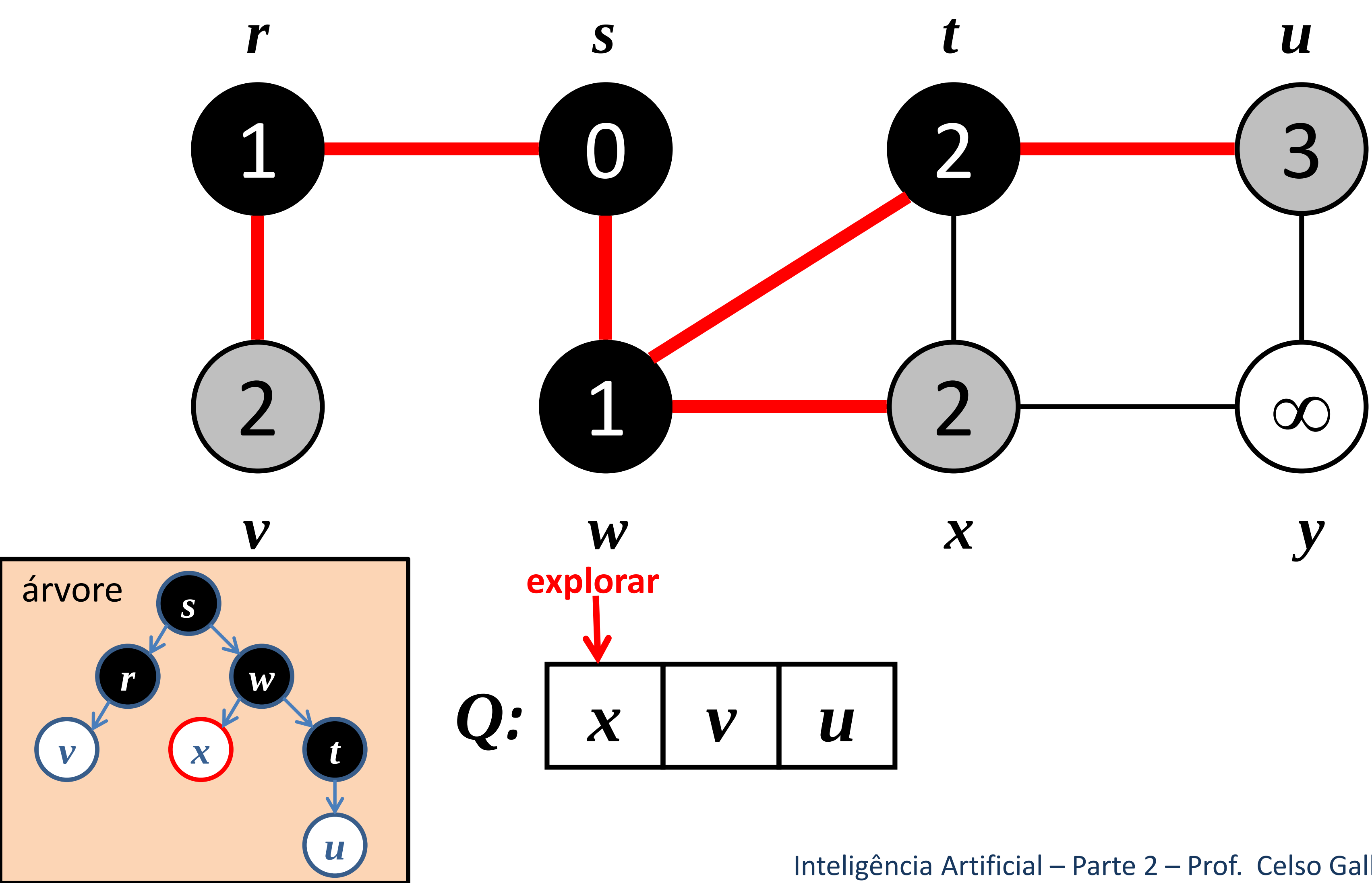
explorar

Q:

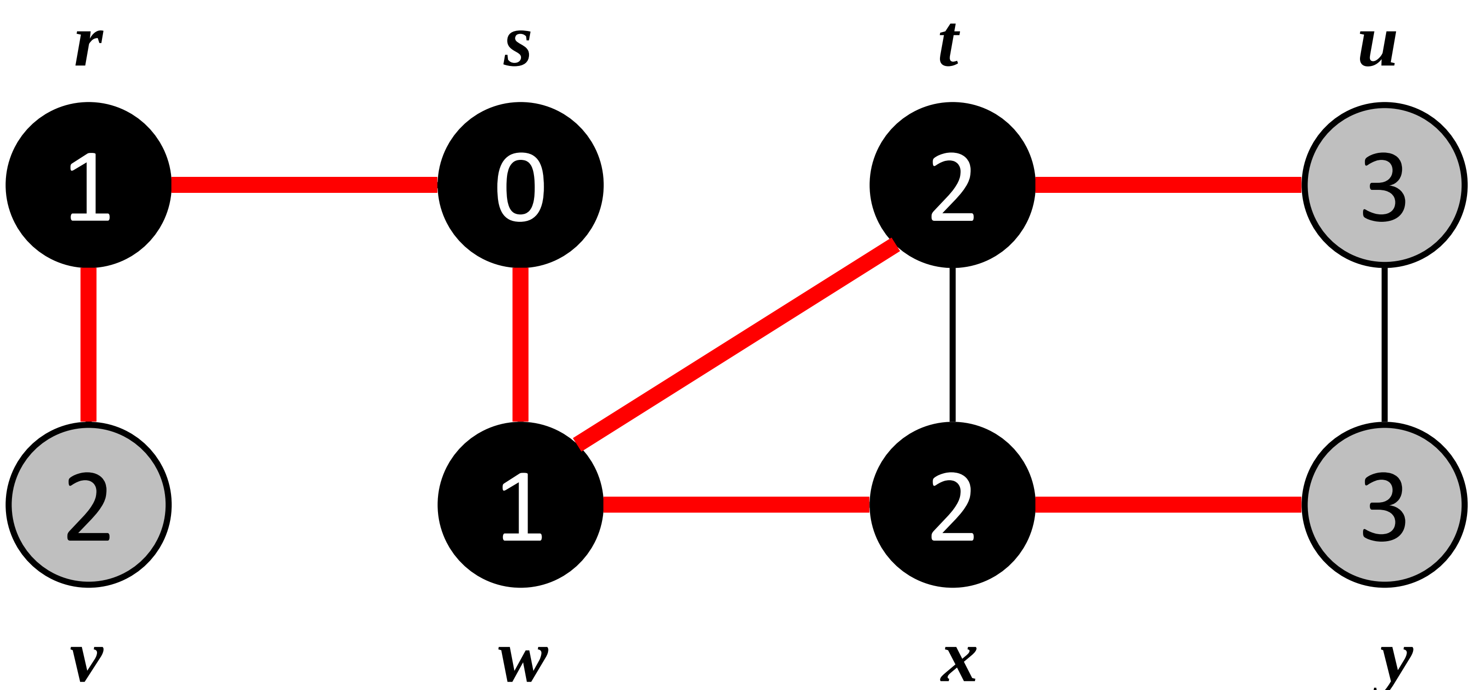
t	*x*	*v*

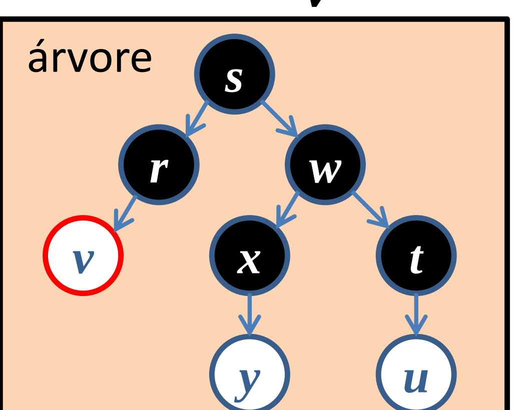
2 - Busca em Largura (*Breadth-First Search*)

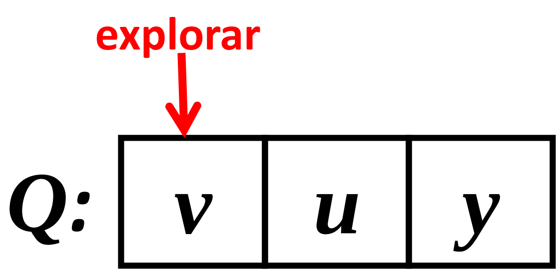
Exemplo BFS:



Exemplo BFS:

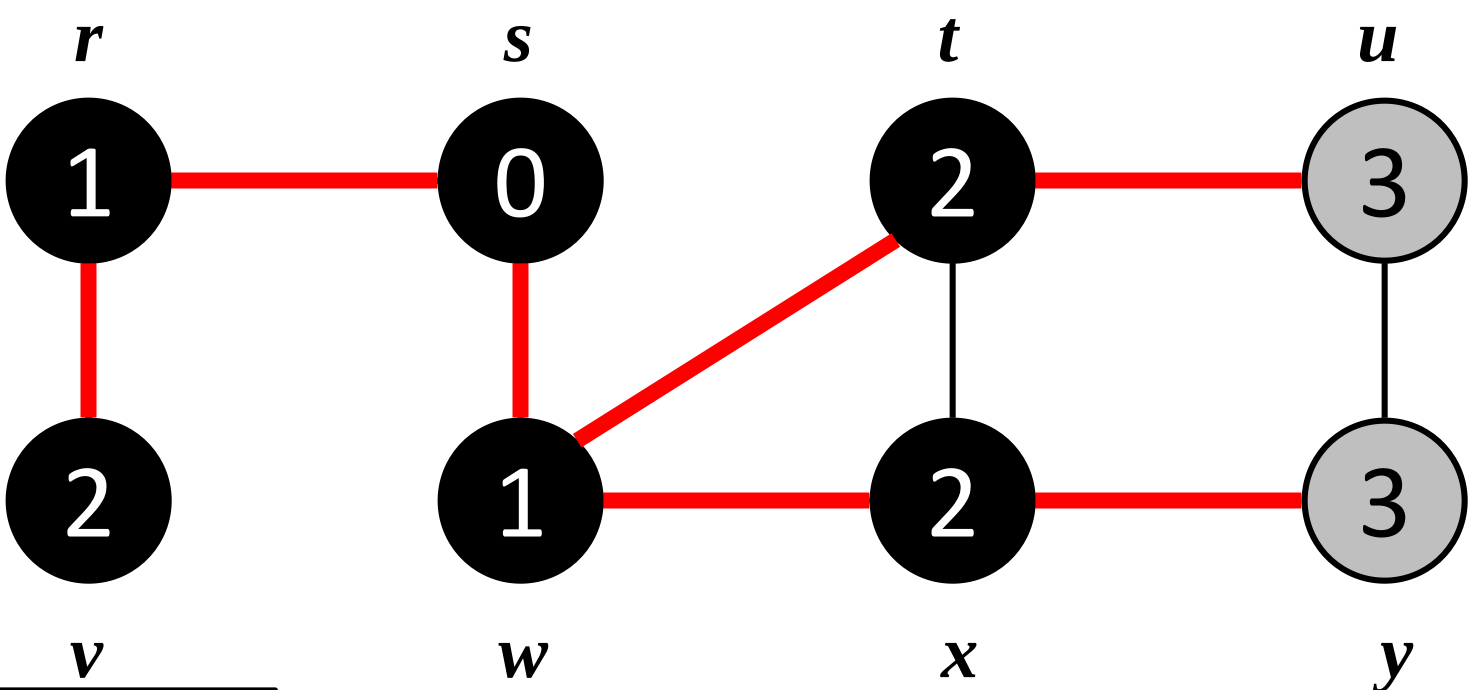


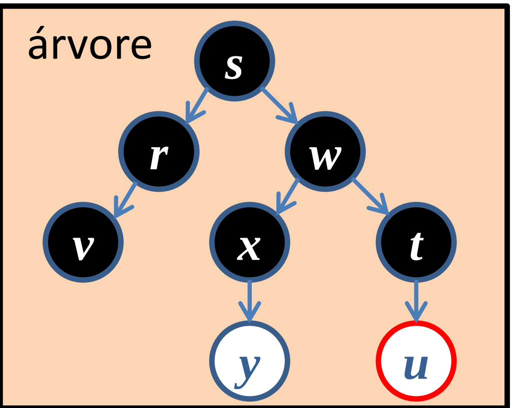




2 - Busca em Largura (*Breadth-First Search*)

Exemplo BFS:

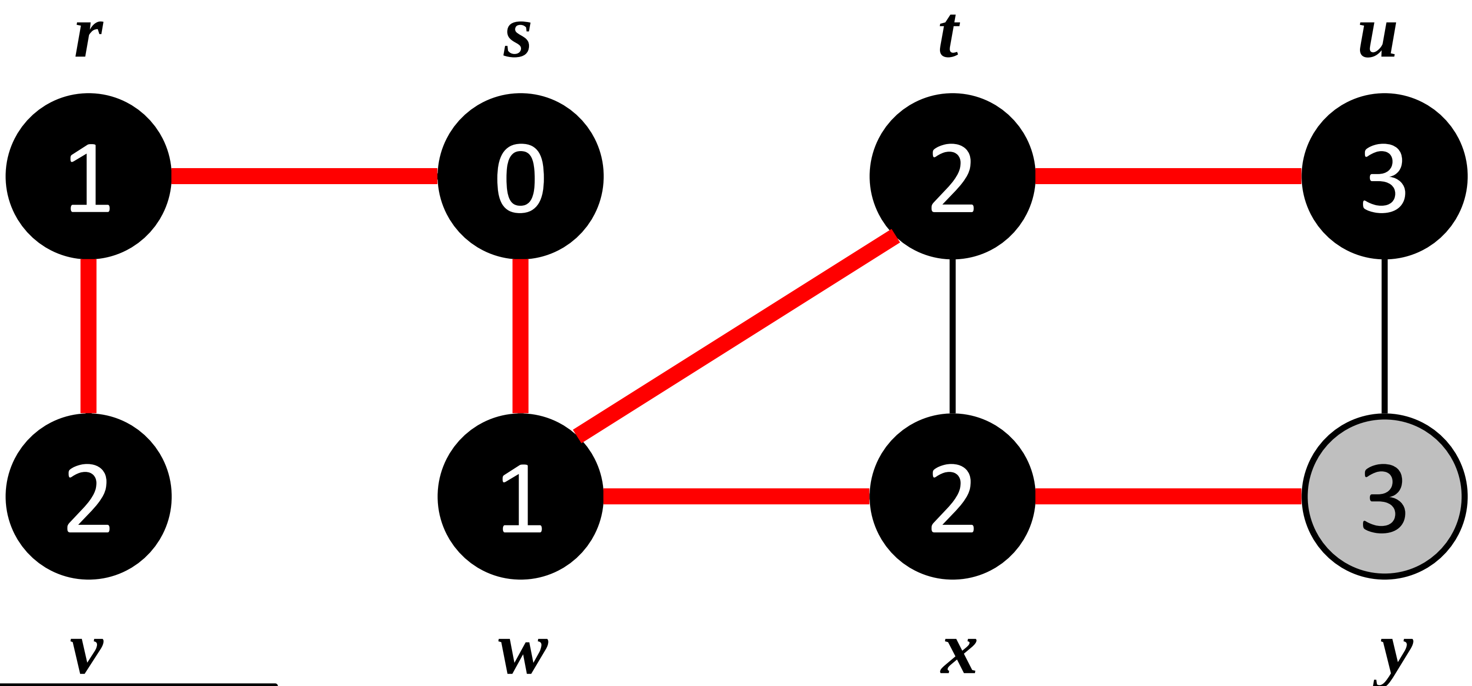


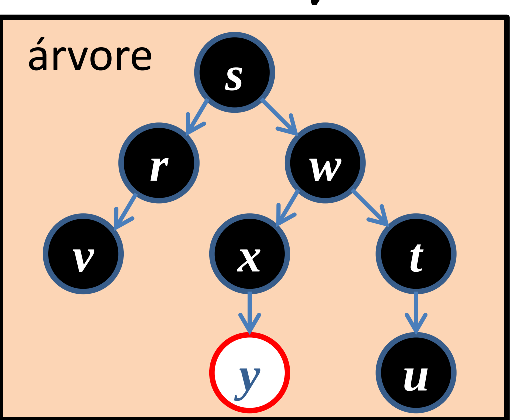


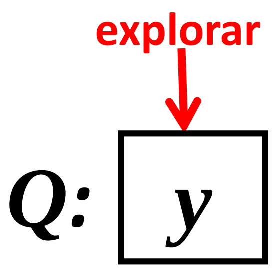
explorar

Q:		
	u	*y*

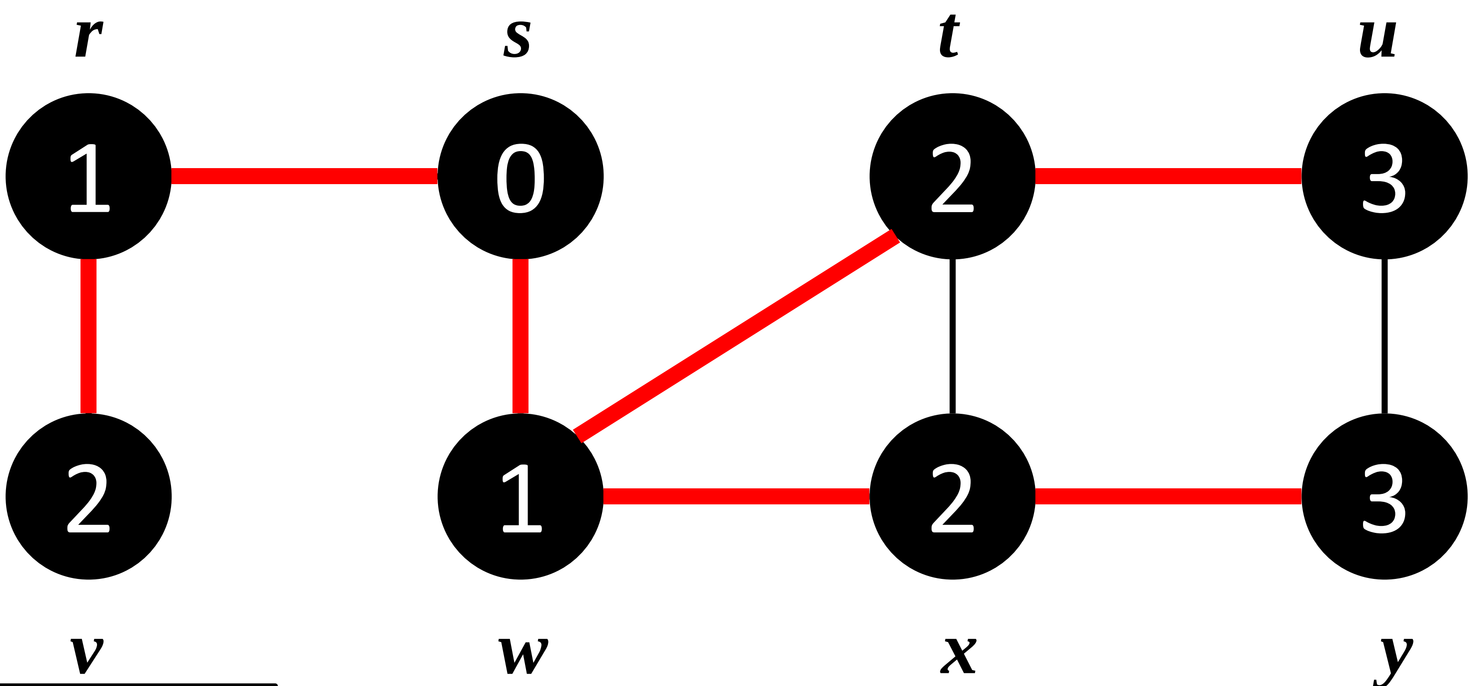
Exemplo BFS:

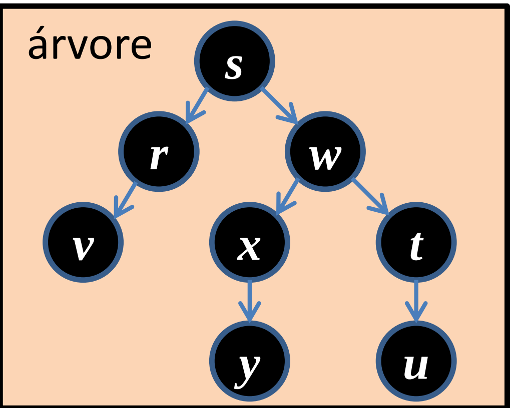






Exemplo BFS:





Q: Ø

2 - Busca em Largura (*Breadth-First Search*)

2.3 - O Método BFS:

1. Definir o vértice inicial ***s.***
2. Explorar as arestas do grafo para descobrir todos os vértices alcançáveis a partir de ***s.***
3. Computar a distância (menor número de arestas) de ***s*** para todos os vértices alcançáveis.
4. Produzir uma árvore de amplitude cuja raiz é ***s*** e contém todos os vértices alcançáveis.
5. Para todo vértice ***v*** alcançável a partir de ***s,*** o caminho na árvore de amplitude corresponde ao menor caminho de ***s*** para ***v*** no grafo.

2 - Busca em Largura (*Breadth-First Search*)

2.4 - O Algoritmo BFS:

- Assumir que o grafo $G = (V, E)$ é representado com lista de adjacências.
- Para cada vértice no grafo, o algoritmo mantém estruturas auxiliares:
  - A variável $cor[v']$ mantém a informação sobre a cor de cada vértice.
  - A variável $pai[v']$ mantém a informação do predecessor de cada vértice. Quando não existe predecessor $pai[v'] = NIL$.
  - A variável $d[v']$ mantém o valor da distância entre o vértice inicial e v'.

2 - Busca em Largura (*Breadth-First Search*)

2.4 - O Algoritmo BFS:

- Assumir que o grafo $G = (V, E)$ é representado com lista de adjacências.
- Para cada vértice no grafo, o algoritmo mantém estruturas auxiliares:
  - Uma fila Q com política FIFO para gerenciar a lista de vértices de cor cinza.
  - O vértice inicial é s.
  - O vértice observado é v'.

2 - Busca em Largura (*Breadth-First Search*)

2.4 - O Algoritmo BFS:

BFS(G, s)

for $\forall v' \in V[G] - \{s\}$ do

$\quad cor[v'] \leftarrow$ BRANCO

$\quad d[v'] \leftarrow \infty$

$\quad pai[v'] \leftarrow$ NIL

$cor[s] \leftarrow$ CINZA

$d[s] \leftarrow 0$

$pai[s] \leftarrow$ NIL

$Q \leftarrow \{s\}$

2 - Busca em Largura (*Breadth-First Search*)

2.4 - O Algoritmo BFS:

BFS(G, s)

for ∀v' ∈ V[G] – {s} do

cor[v'] ← BRANCO

d[v'] ← ∞

pai[v'] ← NIL

cor[s] ← CINZA

d[s] ← 0

pai[s] ← NIL

Q ← {s}

Inicia variáveis auxiliares para cada um dos vértices, com exceção da origem.

2 - Busca em Largura (*Breadth-First Search*)

2.4 - O Algoritmo BFS:

BFS(G, s)

for $\forall$v' $\in$ V[G] – {s} do

    cor[v'] $\leftarrow$ BRANCO

    d[v'] $\leftarrow$ ∞

    pai[v'] $\leftarrow$ NIL

cor[s] $\leftarrow$ CINZA

d[s] $\leftarrow$ 0

pai[s] $\leftarrow$ NIL

Q $\leftarrow$ {s}

Inicia variáveis auxiliares da origem *s* e a fila *Q*.

2 - Busca em Largura (*Breadth-First Search*)

2.4 - O Algoritmo BFS:

while $Q \neq \emptyset$ **do**

v' ← Desenfileira[Q]

for ∀v ∈ Adjacente[v'] **do**

if cor[v] = BRANCO **then**

cor[v] ← CINZA

d[v] ← d[v'] + 1

pai[v] ← v'

Enfileira(Q, v)

cor[v'] ← PRETO

Se o vértice adjacente é branco, significa que ele nunca foi visitado. Deve ser pintado de CINZA e enfileirado para posterior processamento.

2 - Busca em Largura (*Breadth-First Search*)

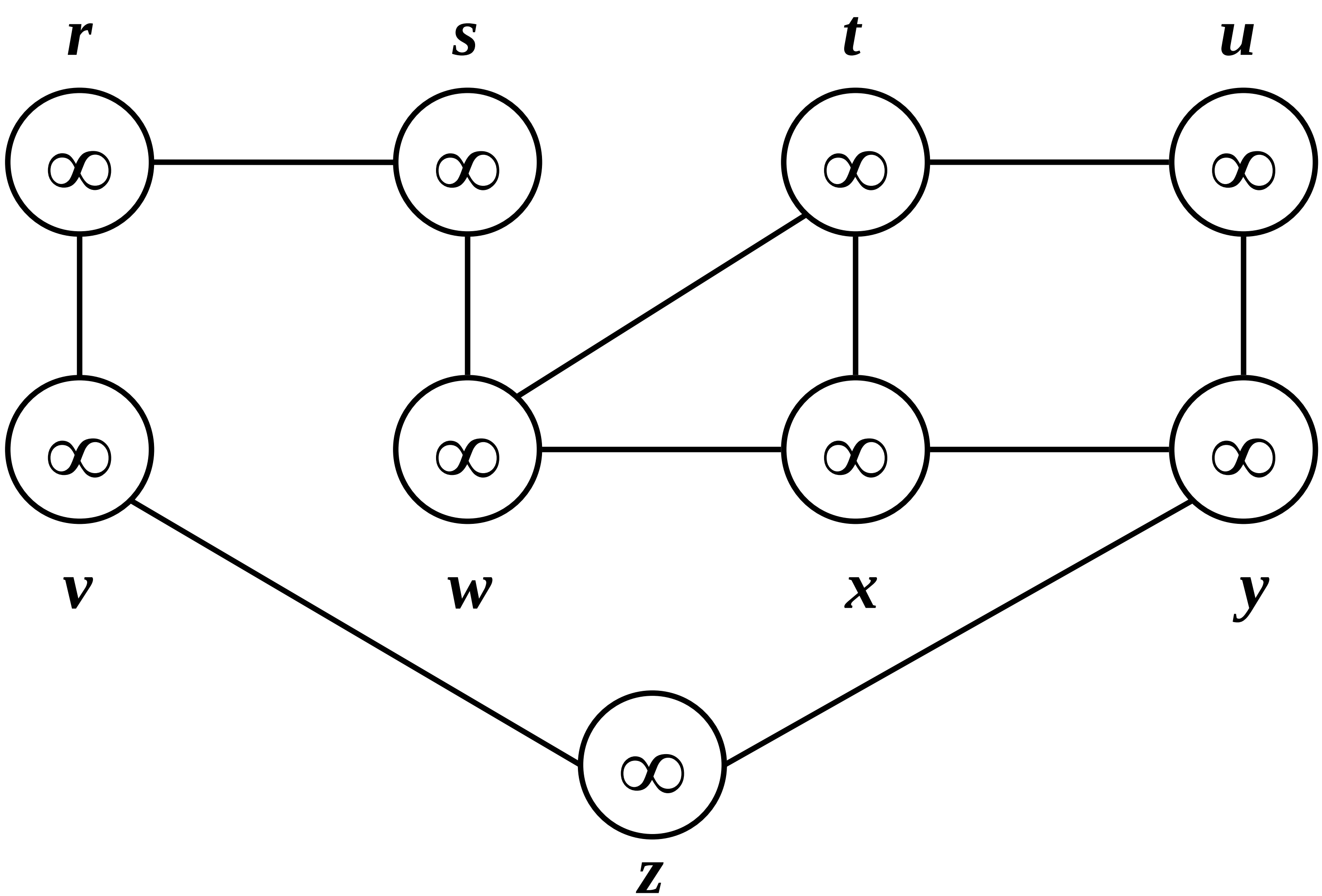
2.4 - O Algoritmo BFS:

while Q ≠ Ø **do**
    v' ← Desenfileira[Q]
    for ∀v ∈ Adjacente[v'] **do**
        if cor[v] = BRANCO **then**
            cor[v] ← CINZA
            d[v] ← d[v'] + 1
            pai[v] ← v'
            Enfileira(Q, v)
    $\boxed{\text{cor[v']} \leftarrow \text{PRETO}}$

Quando todos os adjacentes de *v'* forem processados, ele passa a ser PRETO.

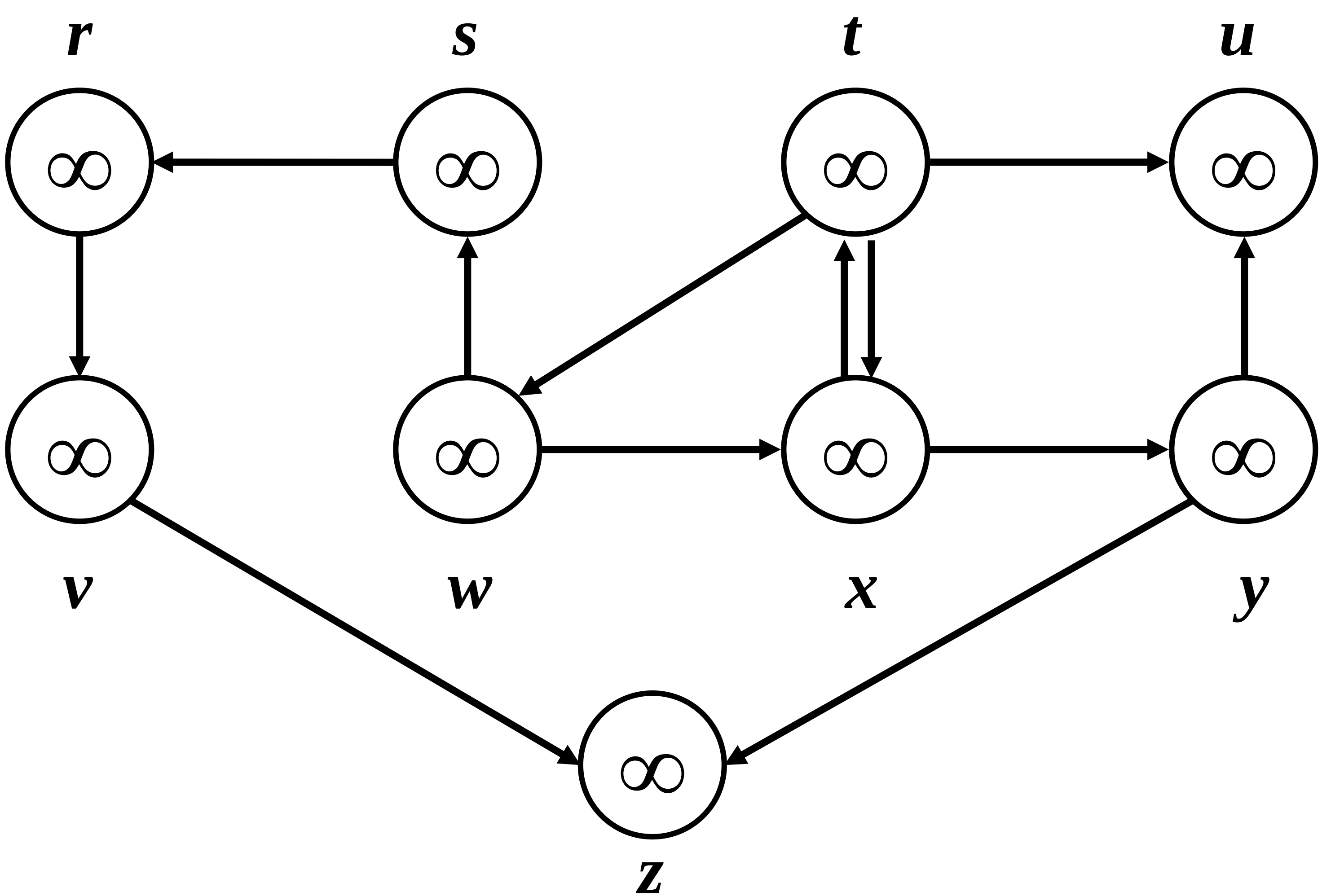
2 - Busca em Largura (*Breadth-First Search*)

2.5 – Exercícios de BFS: 1) iniciar em *x*.



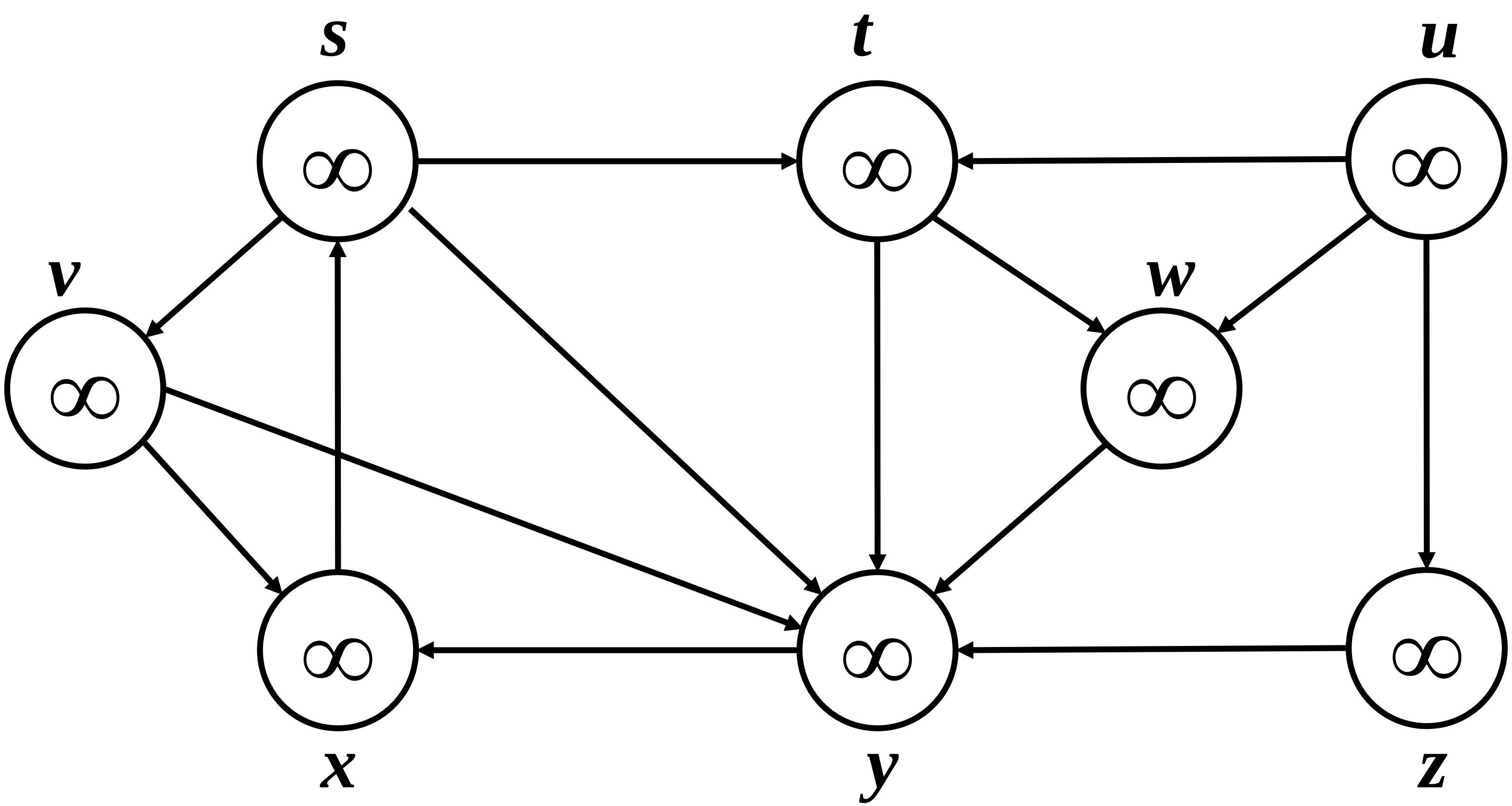
2 - Busca em Largura (*Breadth-First Search*)

2.5 – Exercícios de BFS: 2) iniciar em *x*.



2 - Busca em Largura (*Breadth-First Search*)

2.5 – Exercícios de BFS: 3) iniciar em *x*.



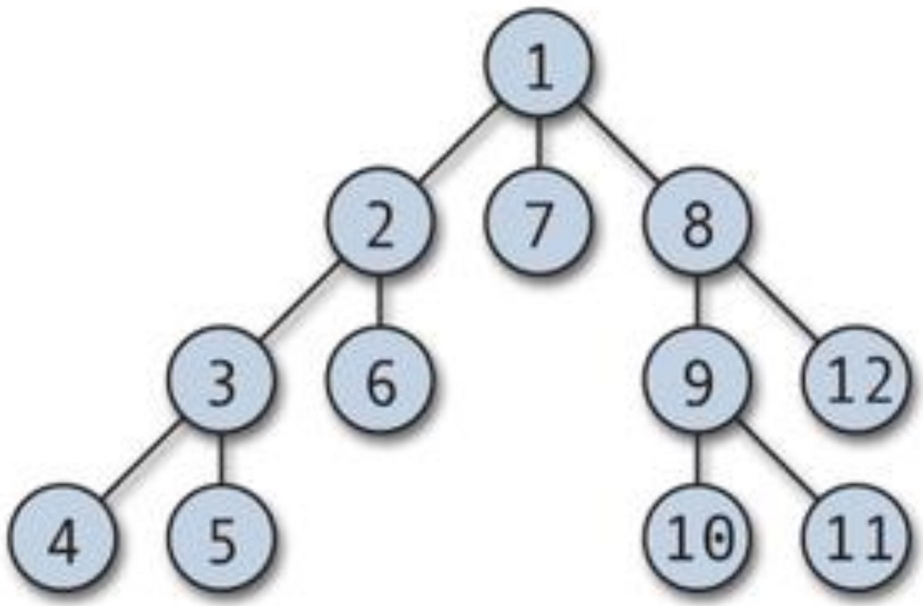
3

Busca em Profundidade
(*Depth-First Search*)

3 - Busca em Profundidade (*Depth-First Search*)

3.1 - Características DFS:

Em uma **busca exaustiva em profundidade** a partir de um vértice ***v***, espera-se que todos os filhos, e os filhos dos filhos de ***v*** sejam visitados antes de continuar a busca no vizinho:



3 - Busca em Profundidade (*Depth-First Search*)

3.1 - Características DFS:

- **Explora-se** todo o grafo, tornando-o árvore de busca:
  - Um vértice por vez, expandindo em profundidade a fronteira de vértices explorados.
  - Busca exaustiva (todos os nós são visitados).
- **Constrói-se** a árvore de busca sobre o grafo:
  - Toma-se um vértice pai como raiz (Geração 0),
  - Descobre-se apenas 1 filho (Geração 1),
  - Para este filho (Geração 1) descobre-se apenas 1 filho (Geração 2), e assim sucessivamente, até não haver mais Gerações.
  - Quando todas as arestas de um vértice já foram exploradas retorna (*backtracking)* à geração anterior e descobre-se outro filho, e assim por diante.

3 - Busca em Profundidade (*Depth-First Search*)

3.2 - Metodologia DFS:

- Colorindo os vértices:
  - **Vértices brancos** ainda não foram considerados

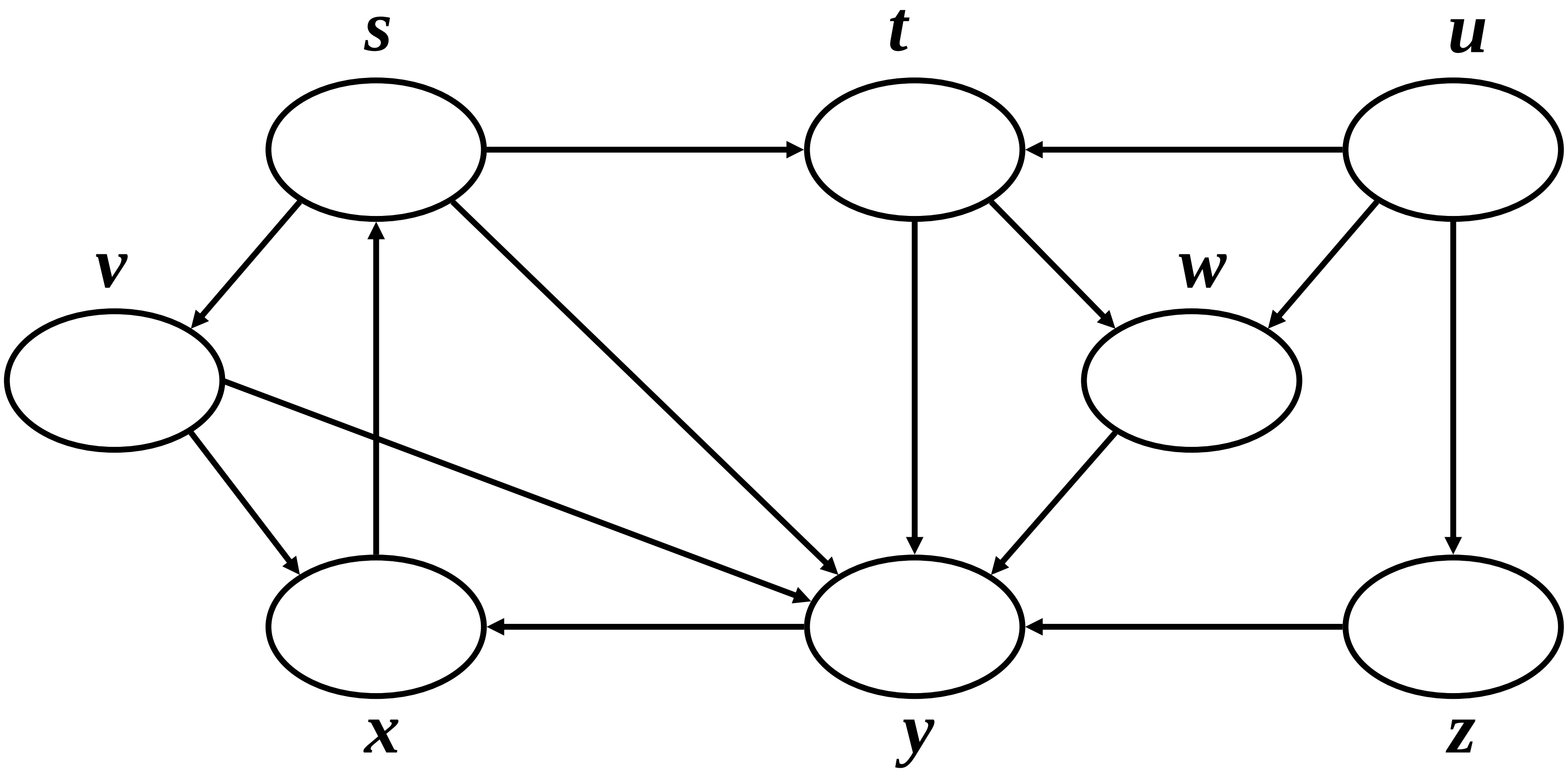
    Todos começam como branco.
  - **Vértices cinza** foram considerados mas não totalmente explorados

    Eles podem ser adjacentes de vértices brancos.
  - **Vértices pretos** foram totalmente explorados

    São vértices adjacentes a vértices pretos ou cinzas.
- Explorando vértices escaneando a lista de adjacências de vértices cinzas.

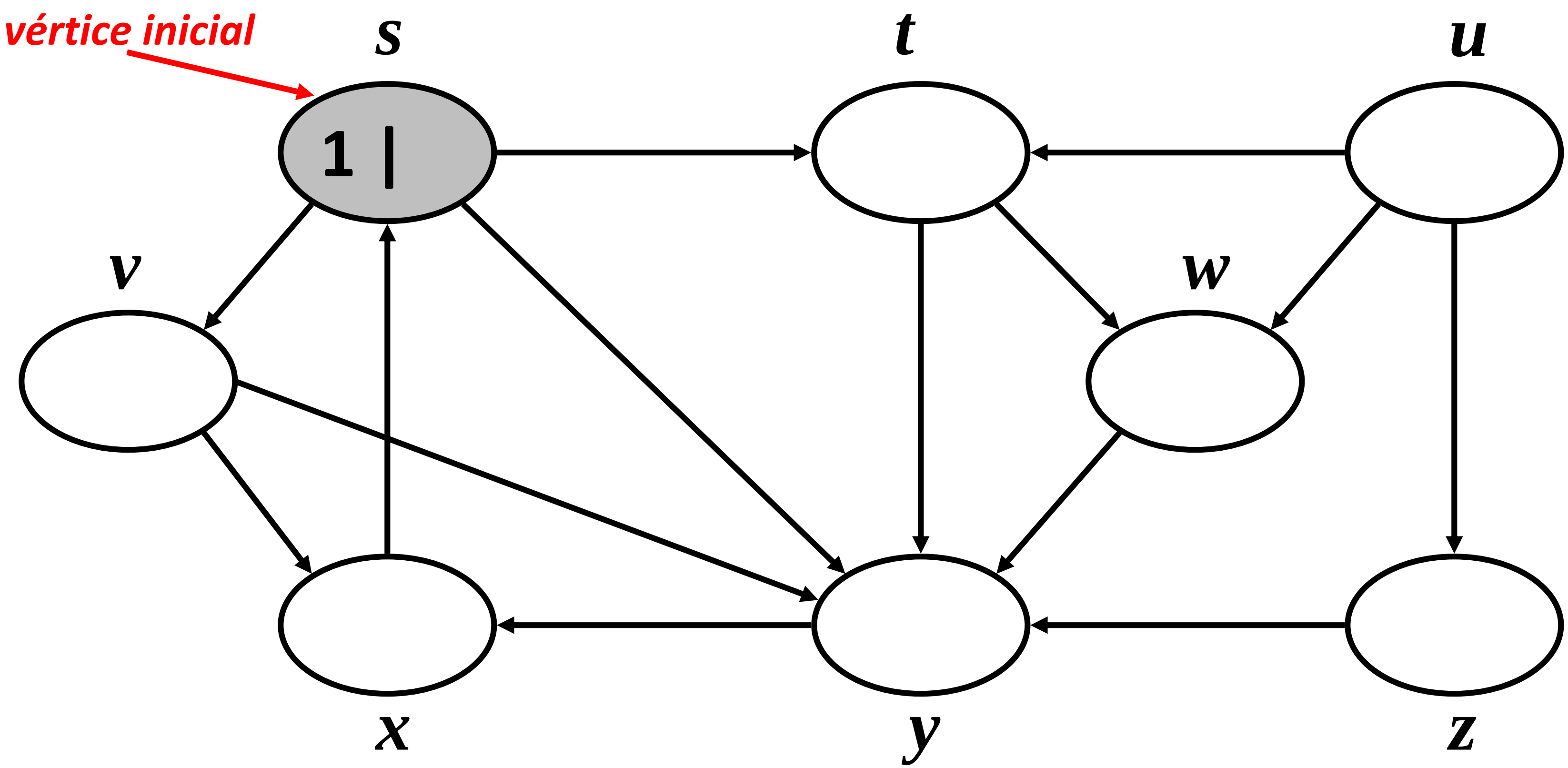
3 - Busca em Profundidade (*Depth-First Search*)

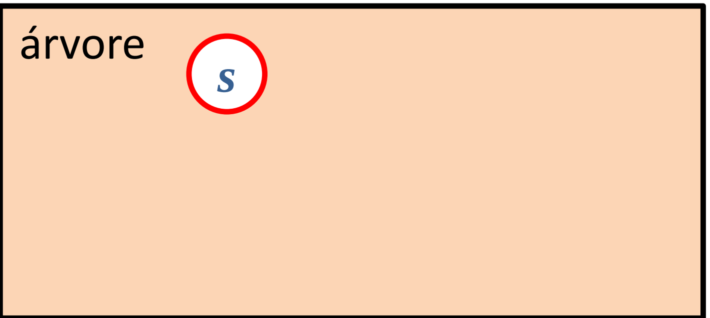
Exemplo DFS (iniciando no vértice *s*):



3 - Busca em Profundidade (*Depth-First Search*)

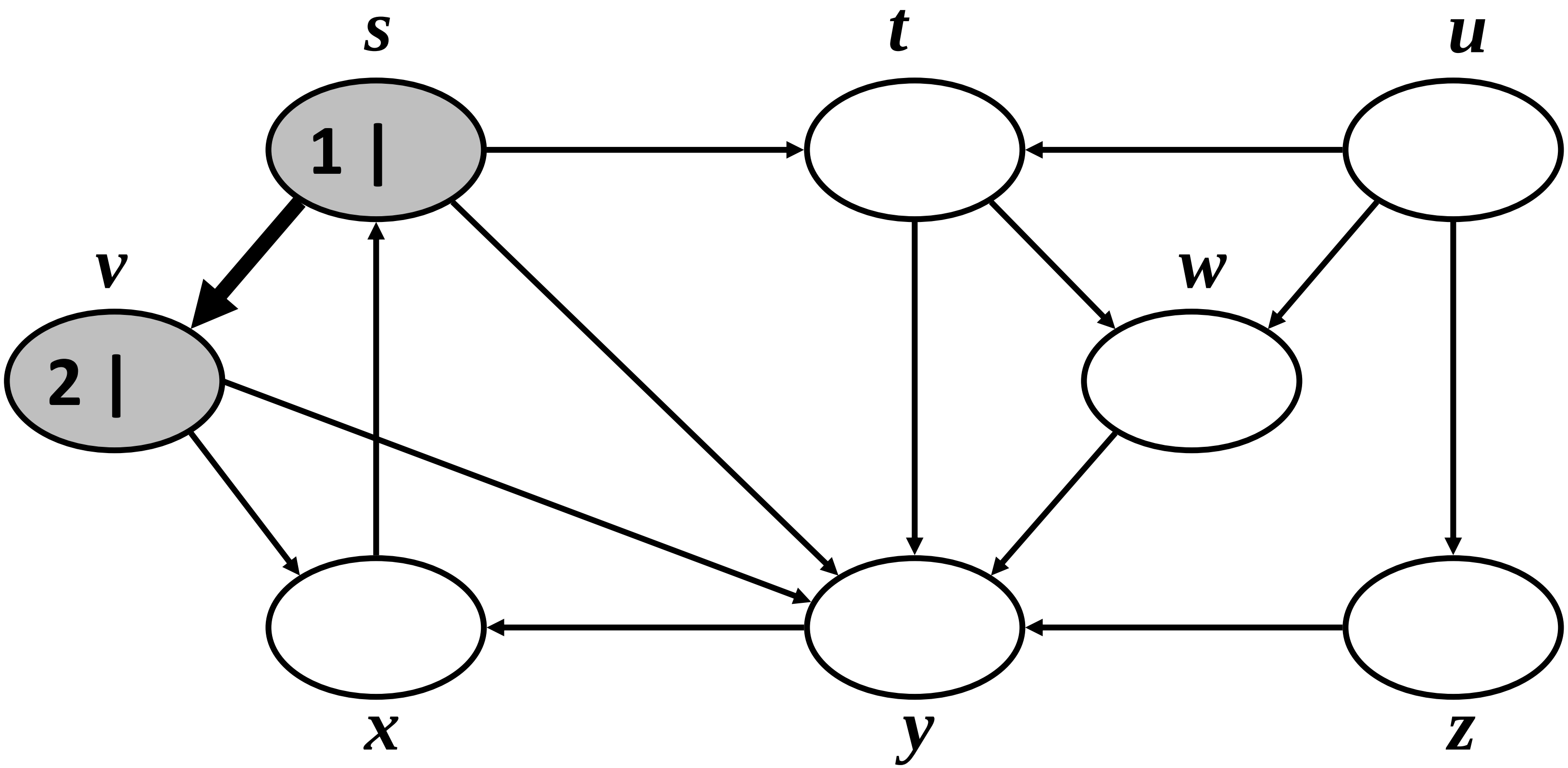
Exemplo DFS:

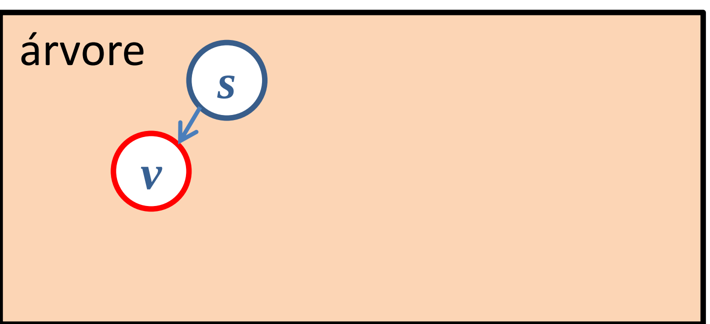




3 - Busca em Profundidade (*Depth-First Search*)

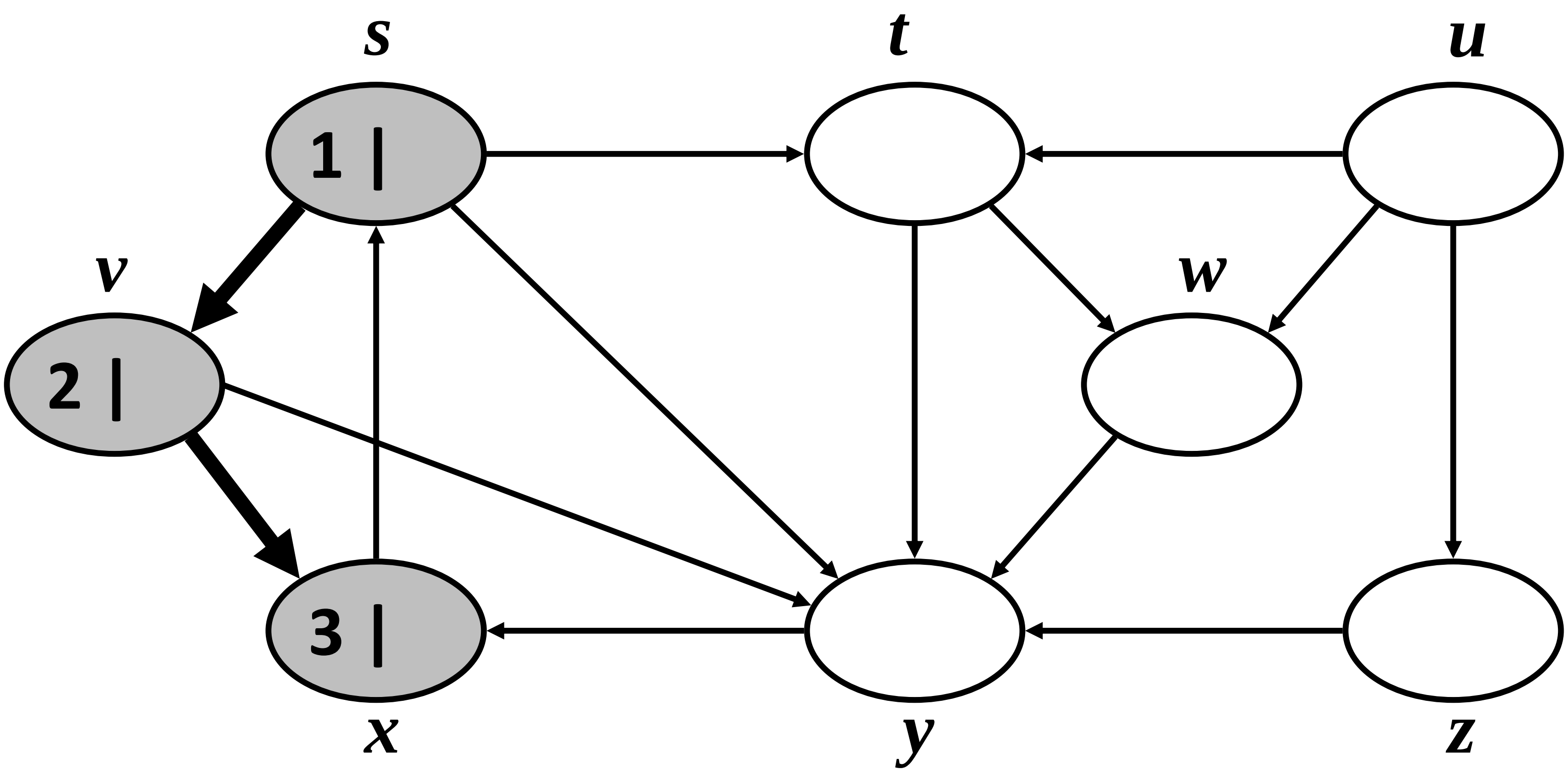
Exemplo DFS:

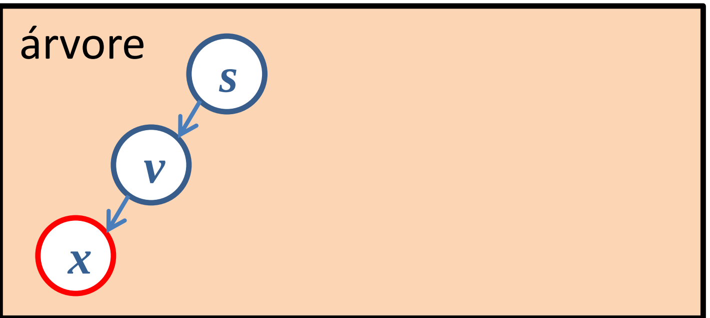




3 - Busca em Profundidade (*Depth-First Search*)

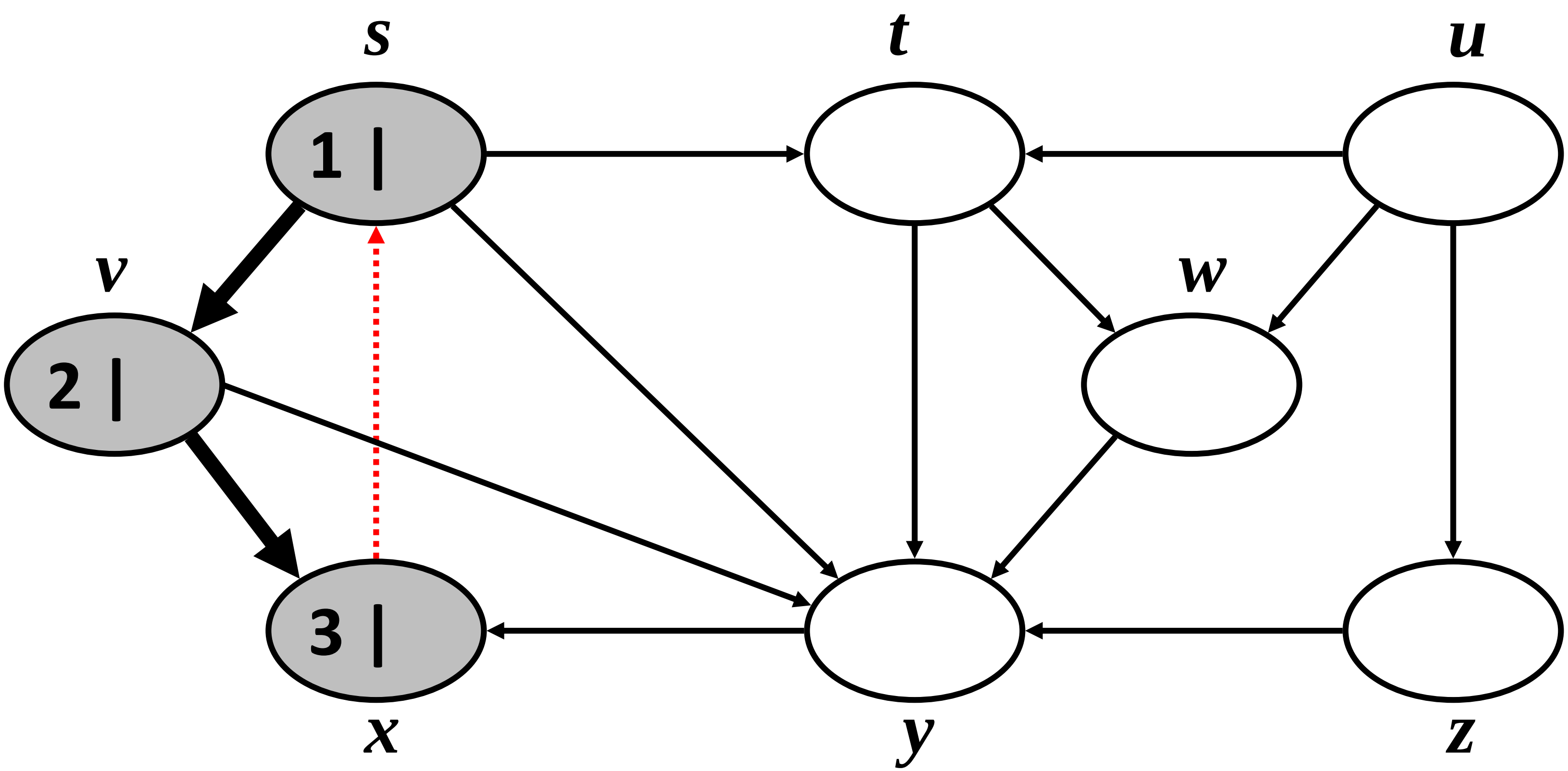
Exemplo DFS:

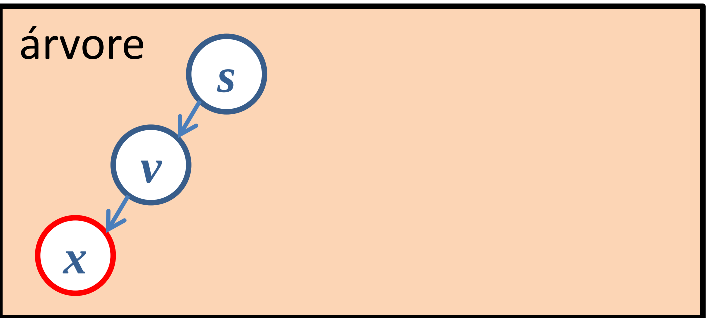




3 - Busca em Profundidade (*Depth-First Search*)

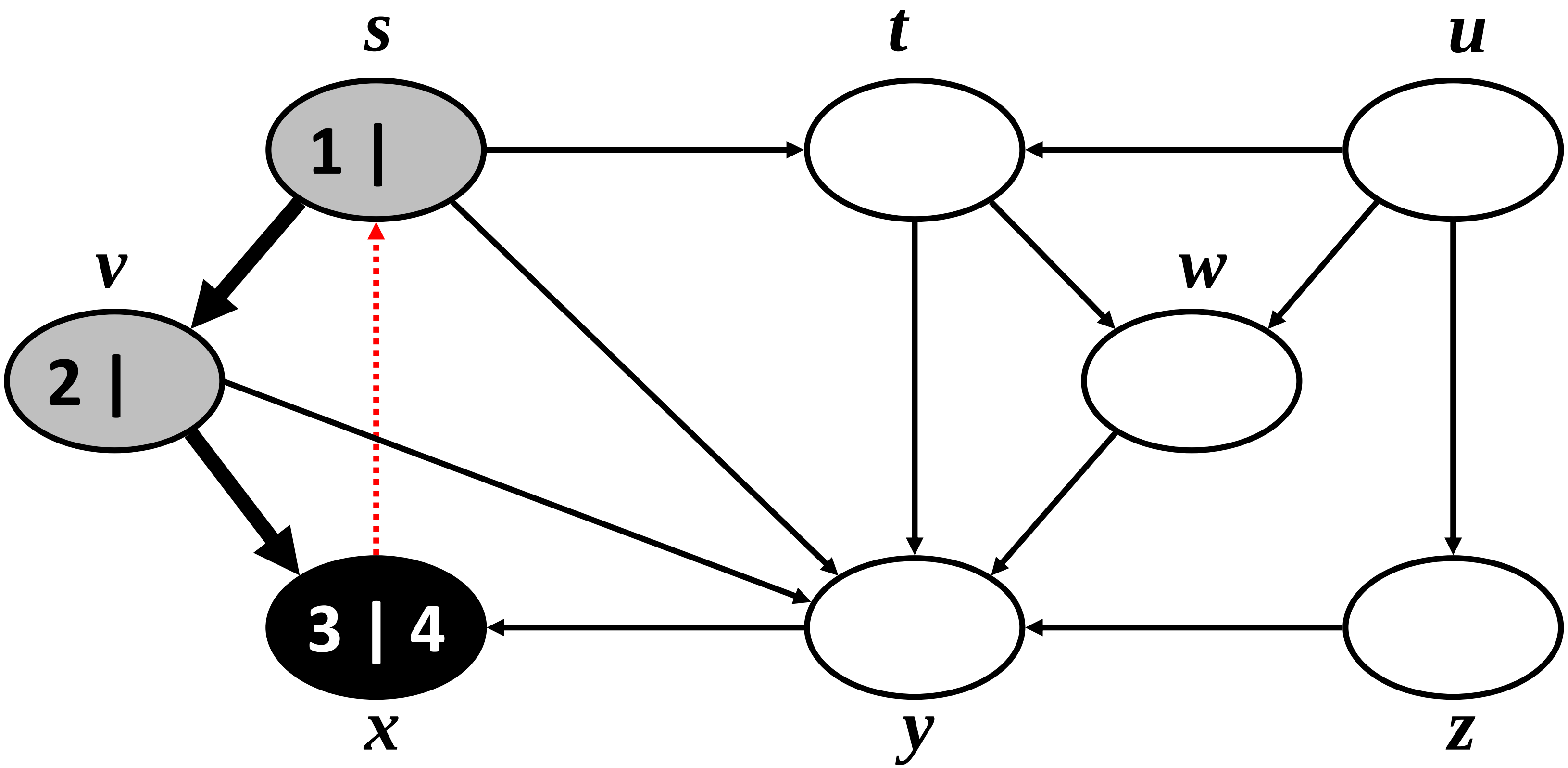
Exemplo DFS:

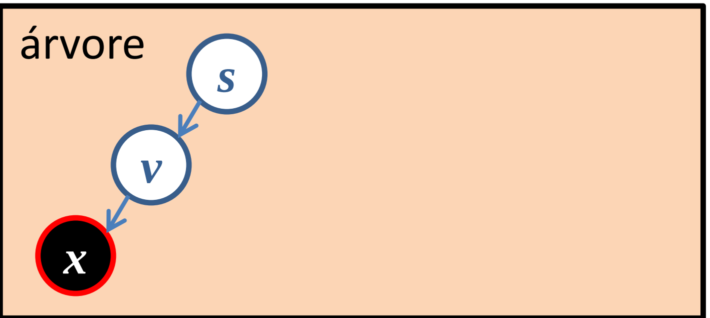




3 - Busca em Profundidade (*Depth-First Search*)

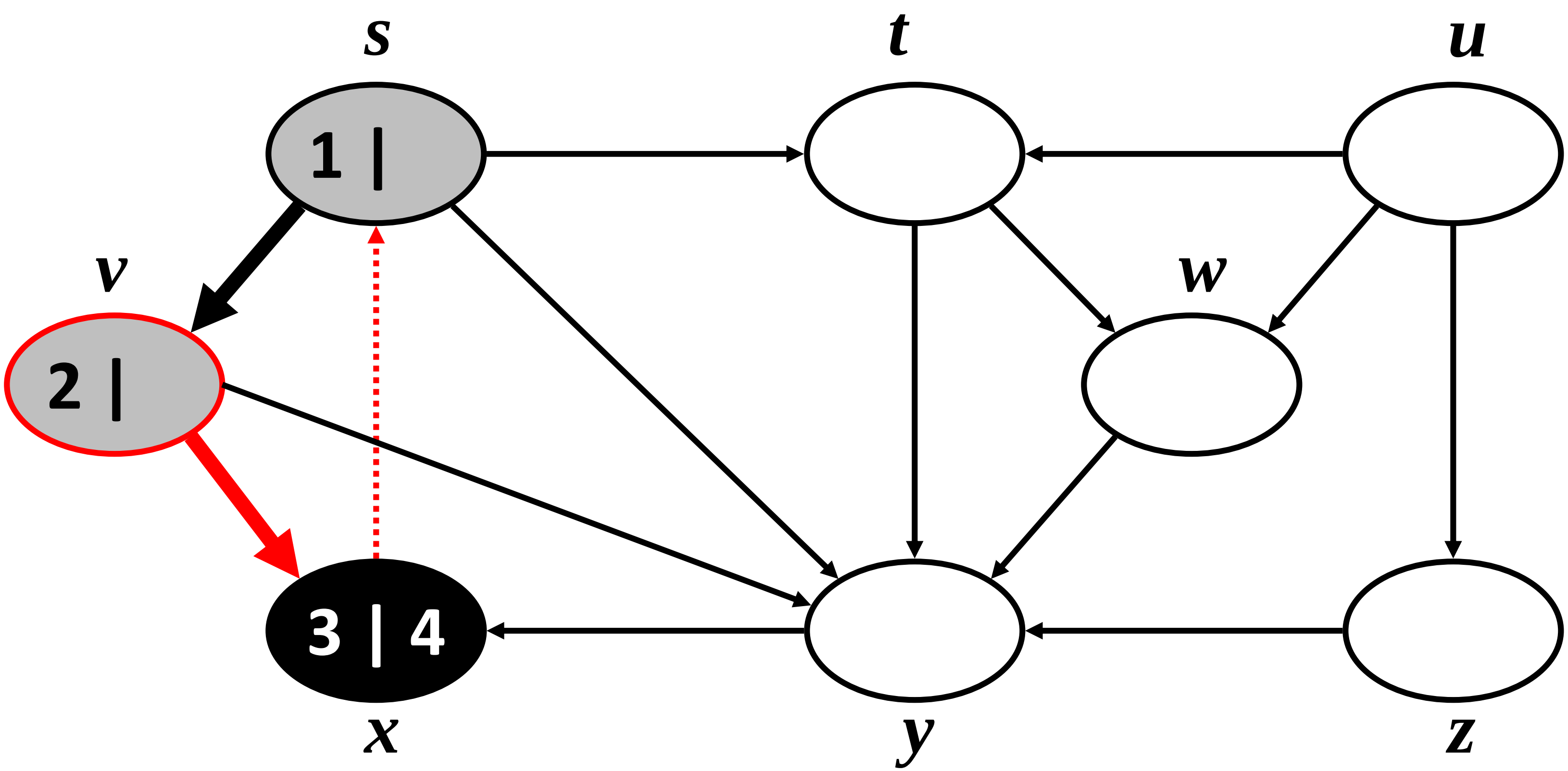
Exemplo DFS:

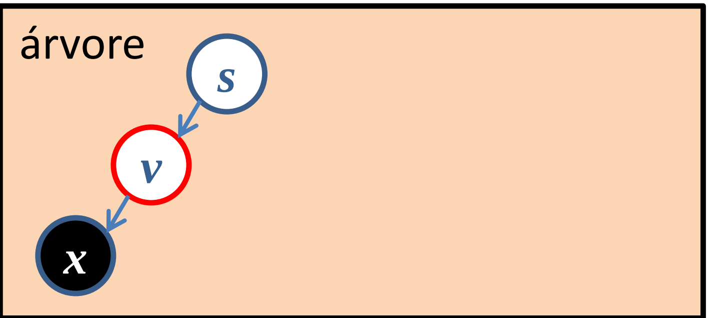




3 - Busca em Profundidade (*Depth-First Search*)

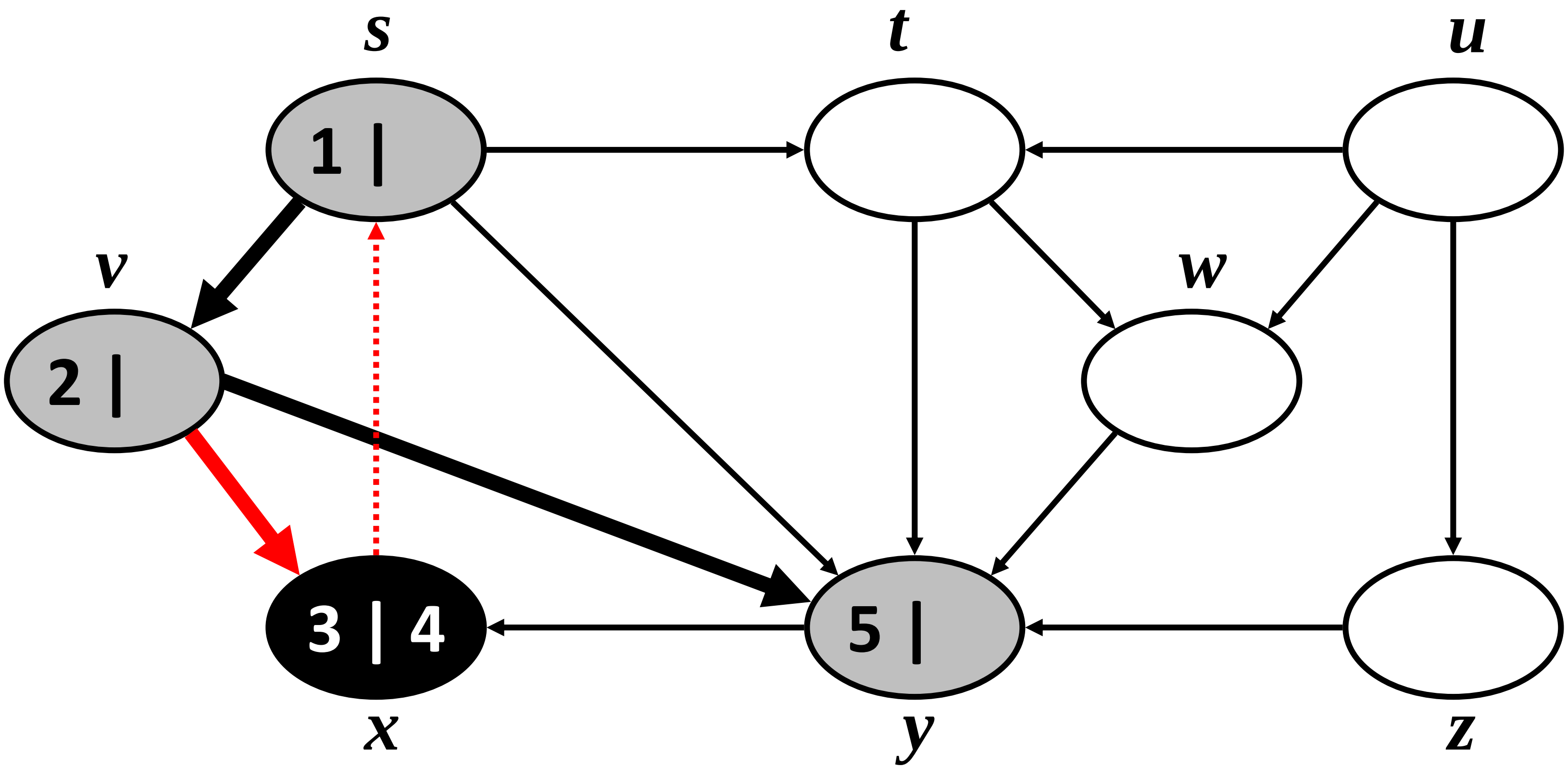
Exemplo DFS:

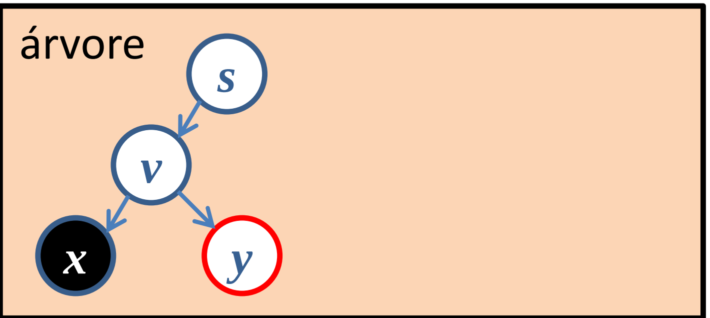




3 - Busca em Profundidade (*Depth-First Search*)

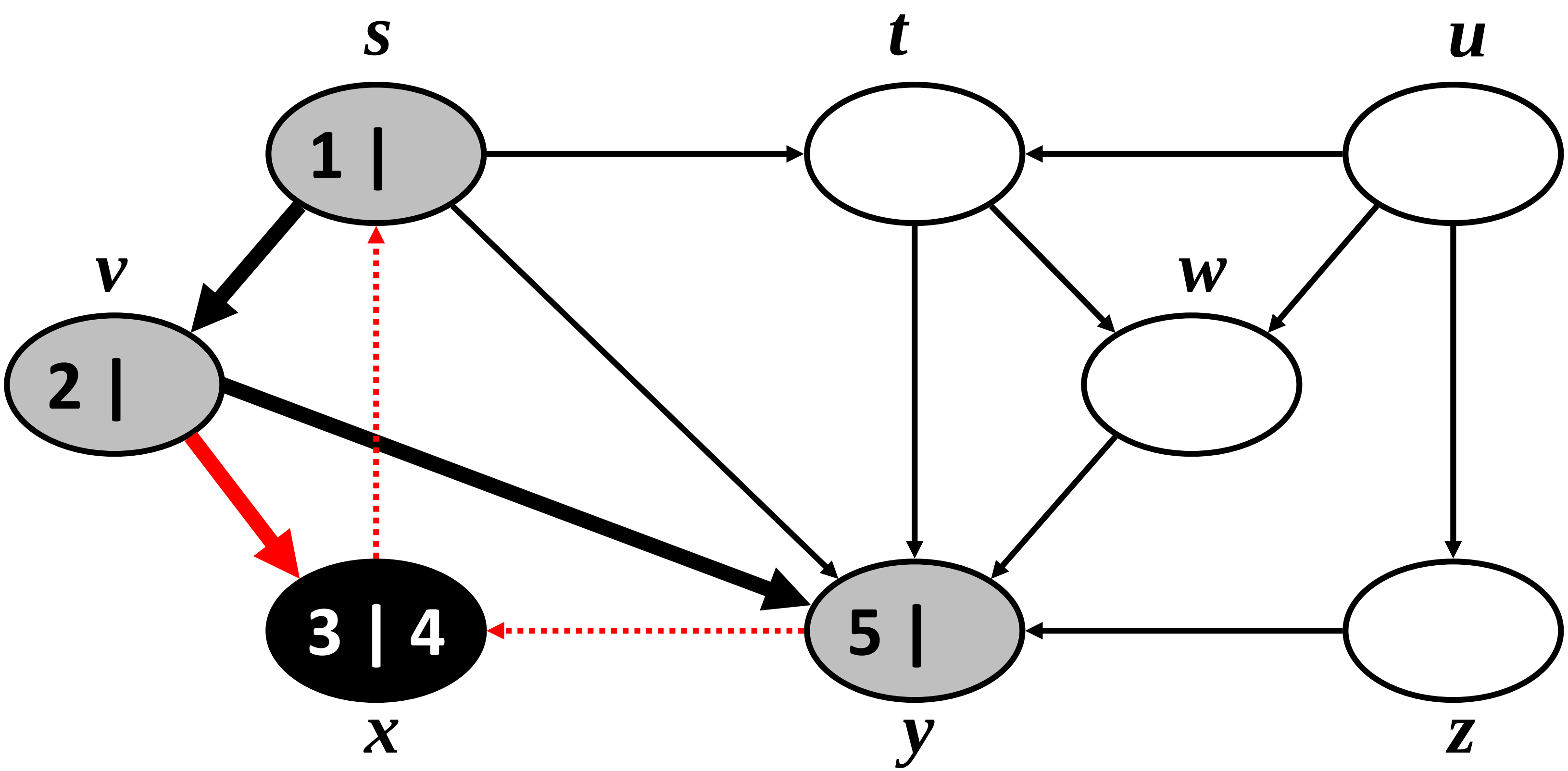
Exemplo DFS:

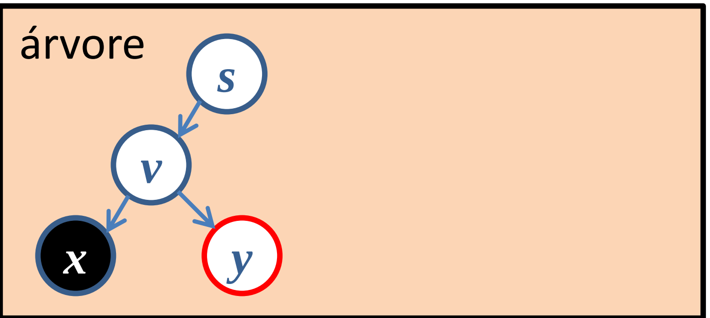




3 - Busca em Profundidade (*Depth-First Search*)

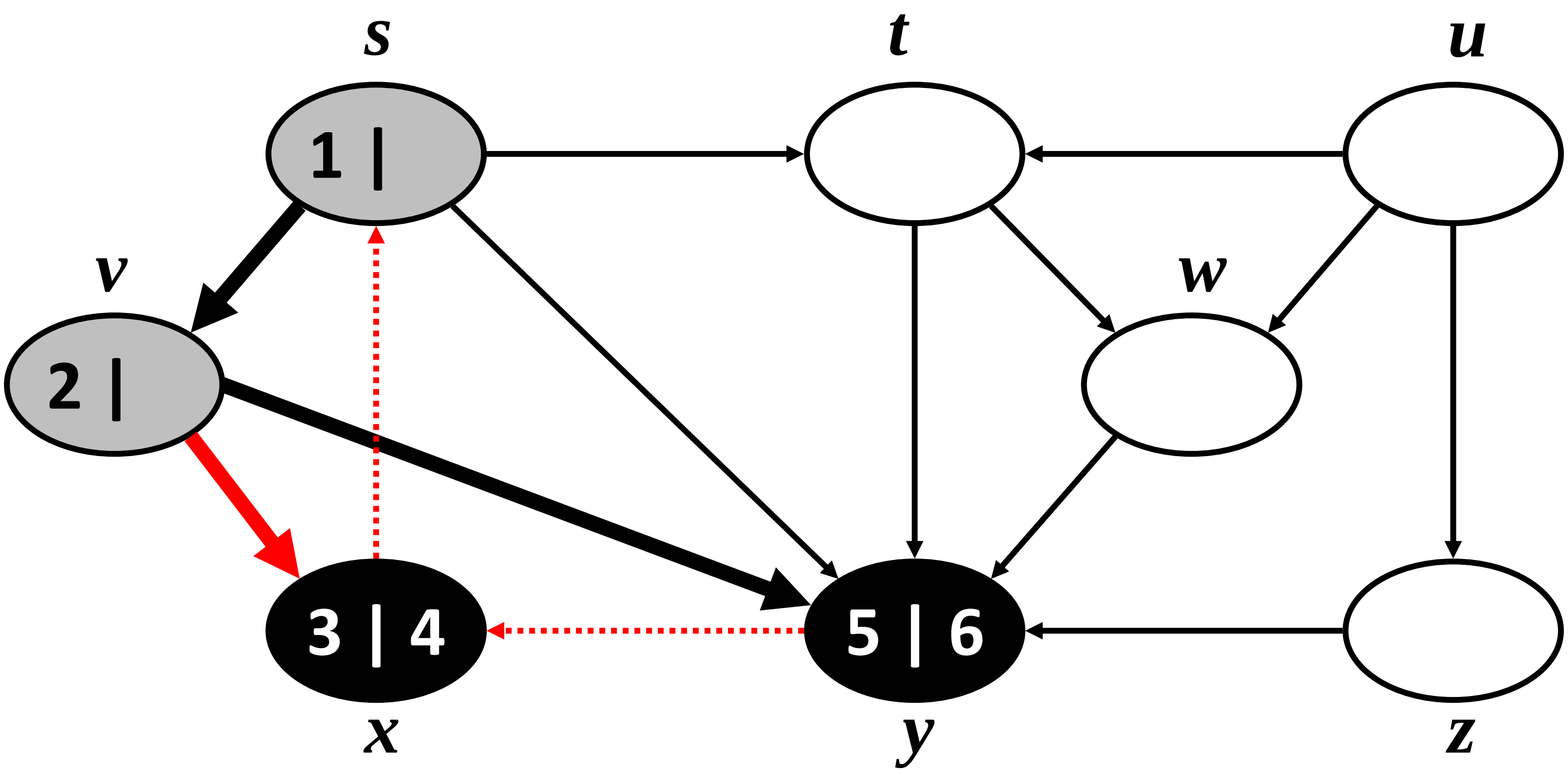
Exemplo DFS:

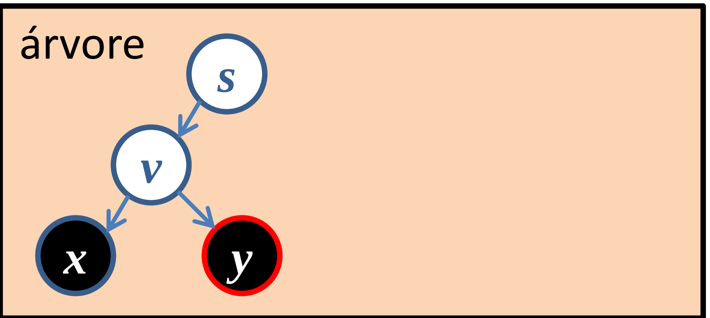




3 - Busca em Profundidade (*<u>D</u>epth-<u>F</u>irst <u>S</u>earch*)

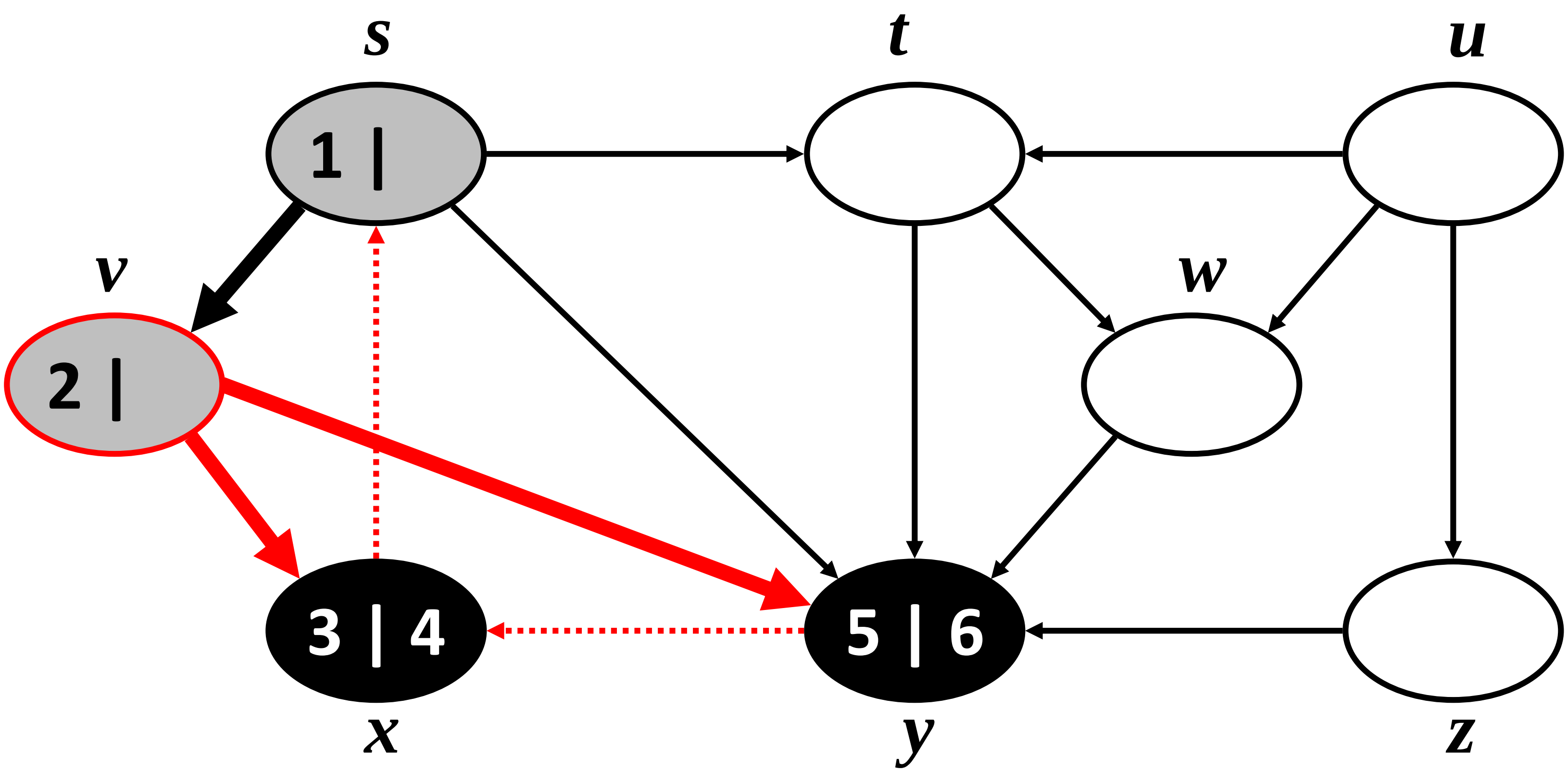
Exemplo DFS:

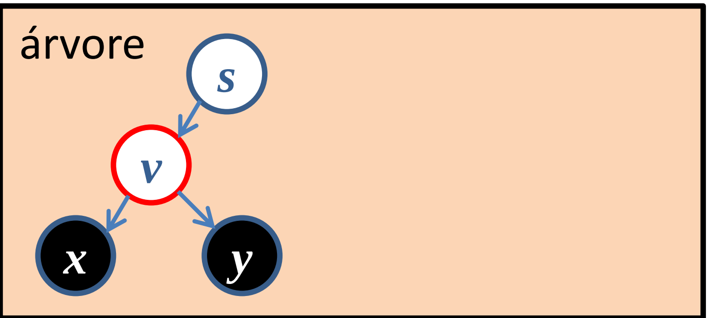




3 - Busca em Profundidade (*Depth-First Search*)

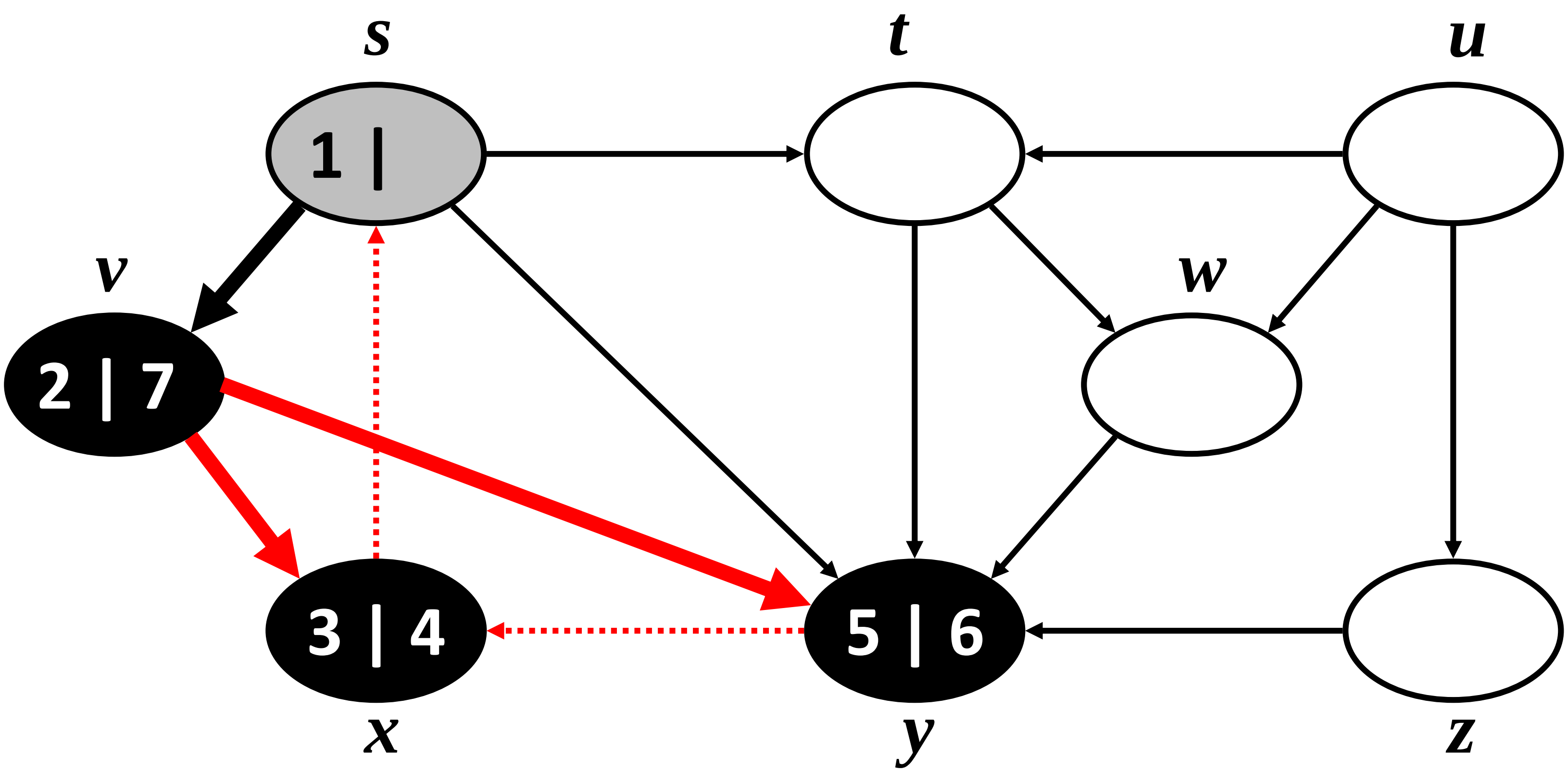
Exemplo DFS:

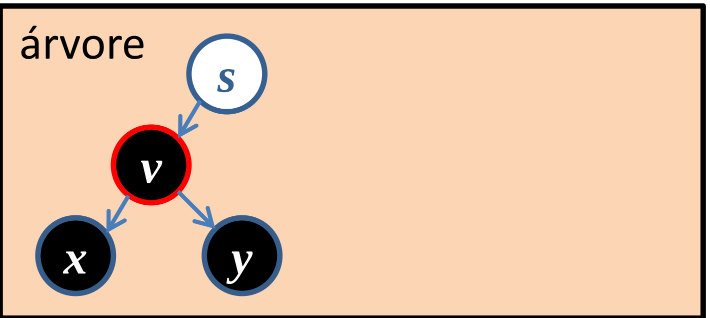




3 - Busca em Profundidade (*Depth-First Search*)

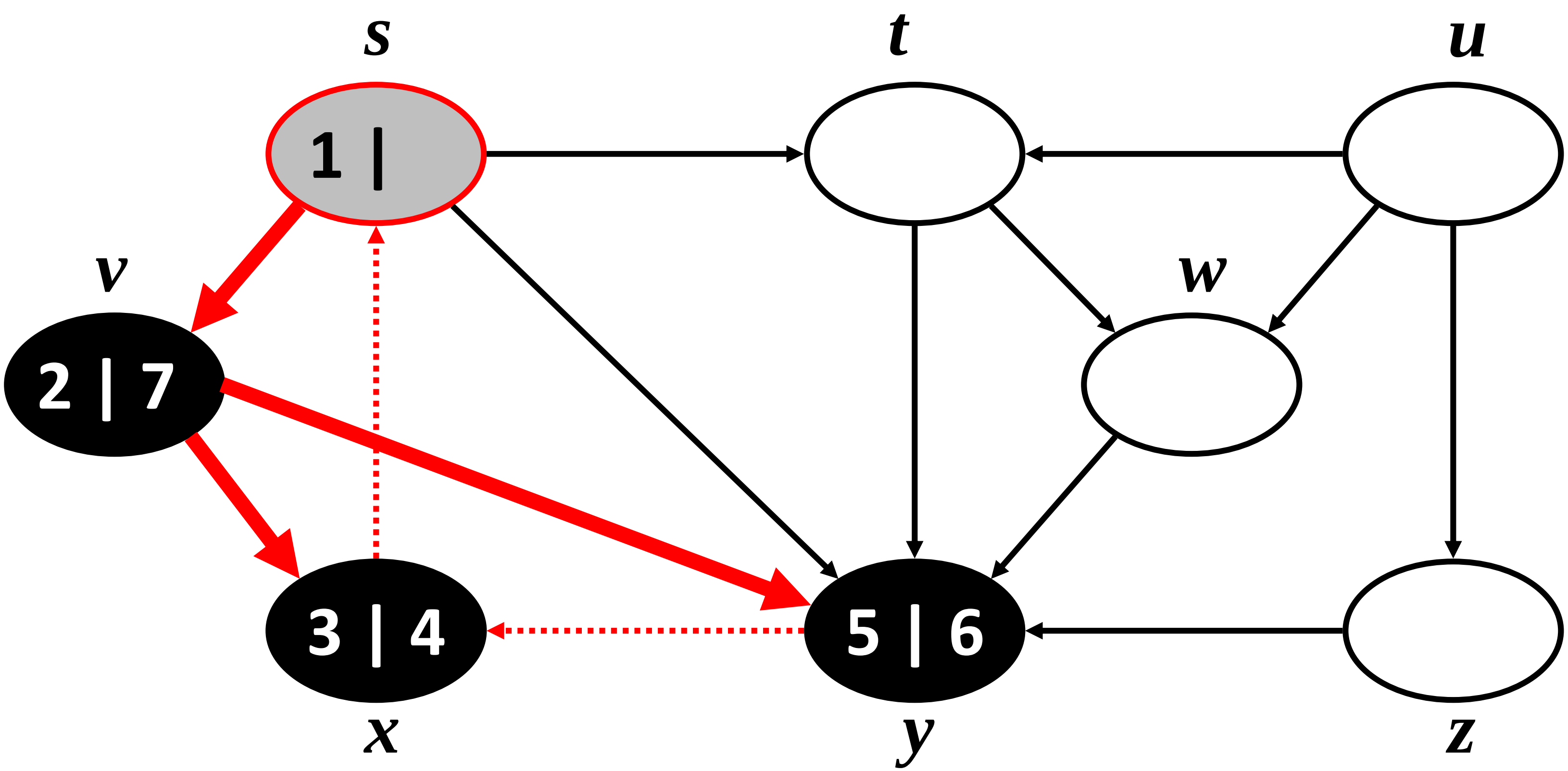
Exemplo DFS:

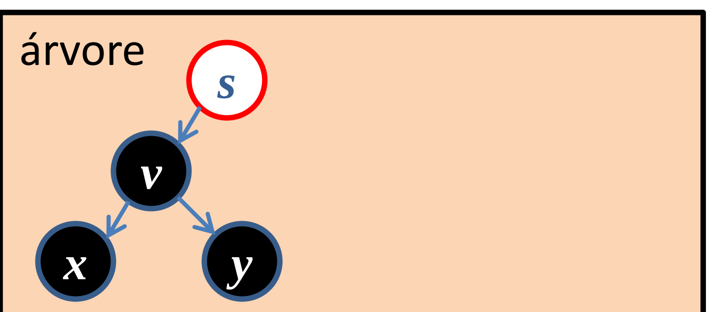




3 - Busca em Profundidade (*Depth-First Search*)

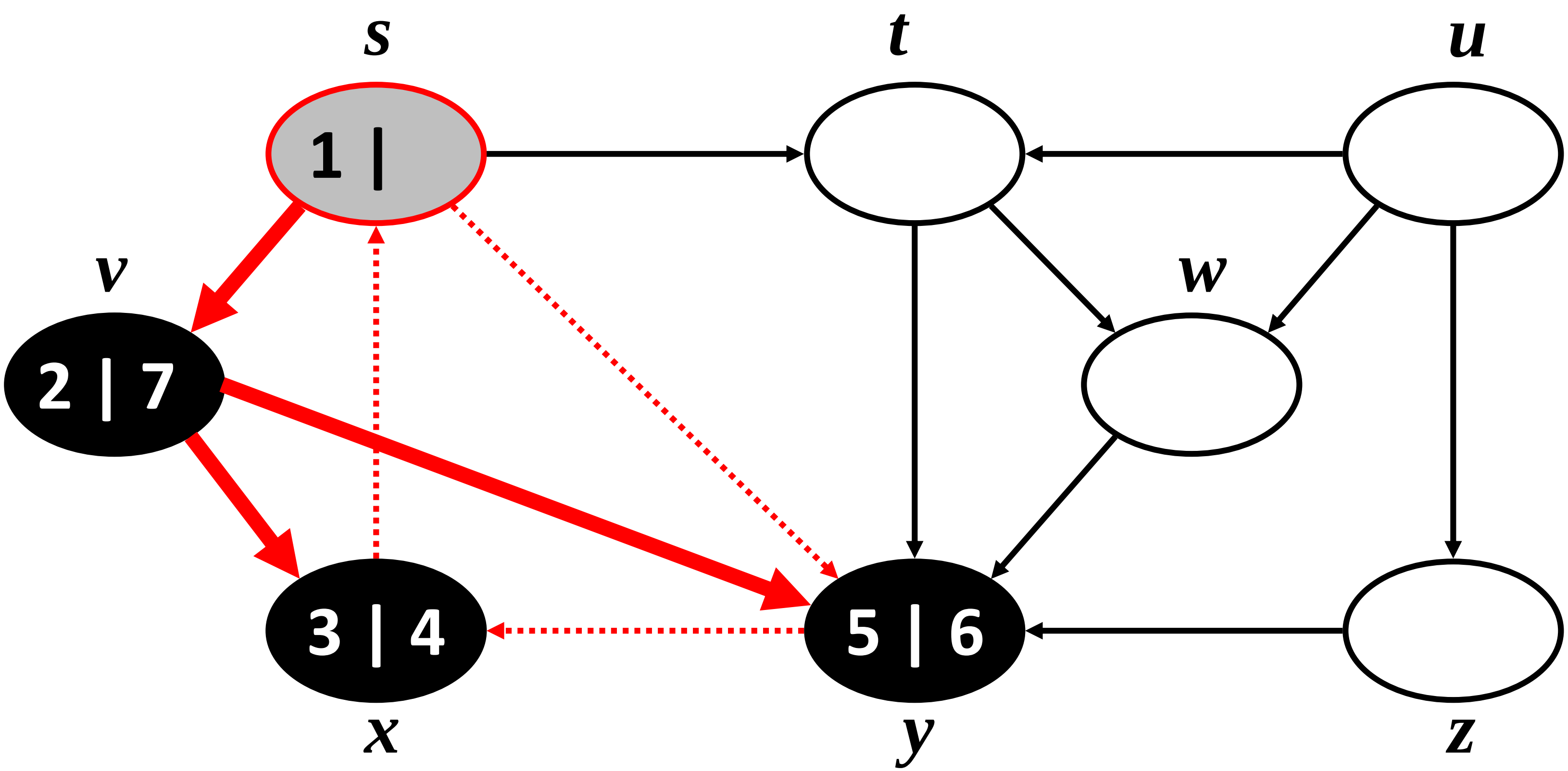
Exemplo DFS:

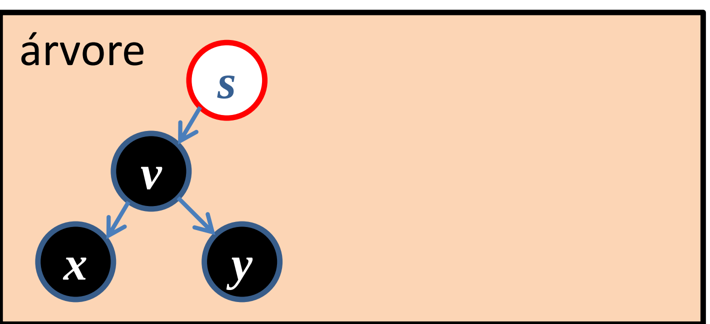




3 - Busca em Profundidade (*Depth-First Search*)

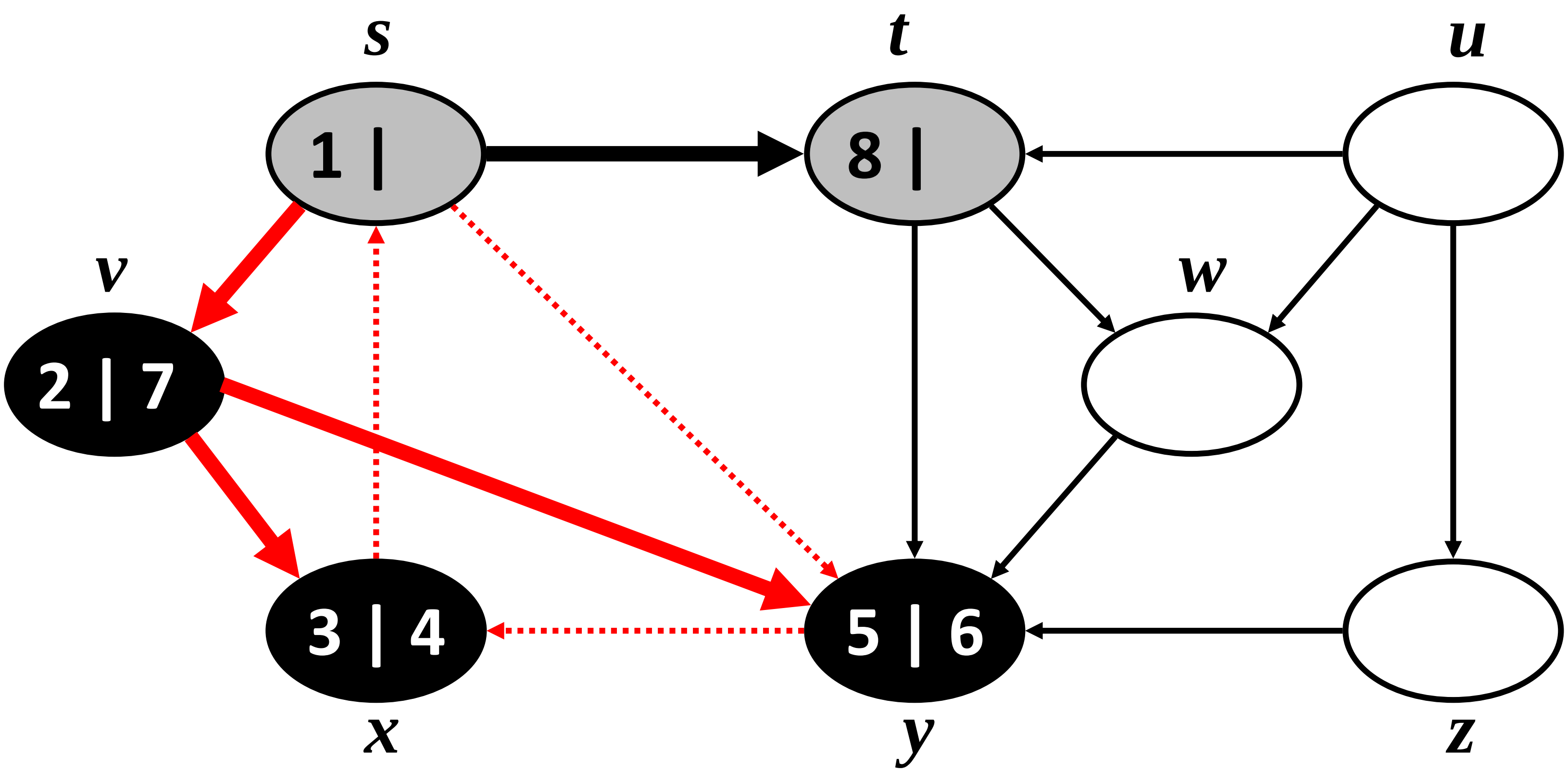
Exemplo DFS:

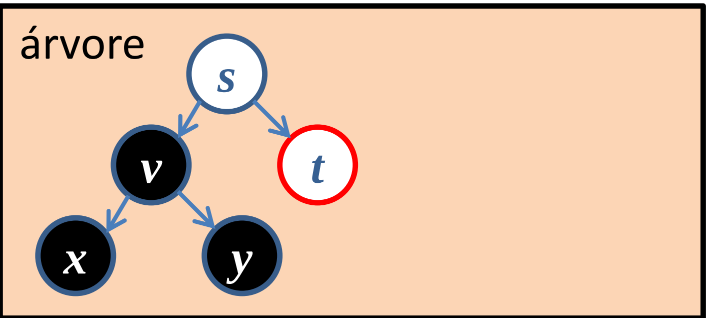




3 - Busca em Profundidade (*Depth-First Search*)

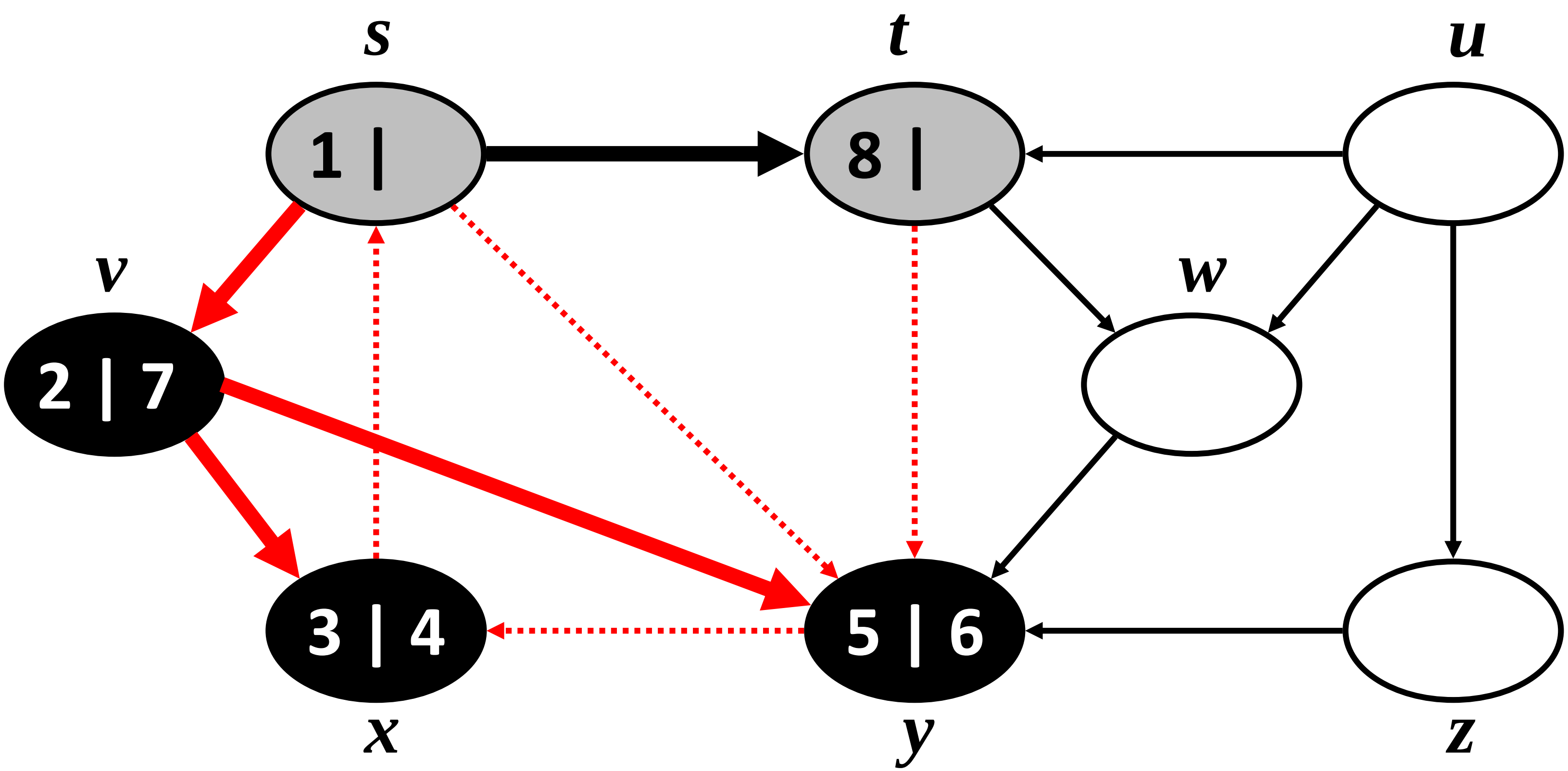
Exemplo DFS:

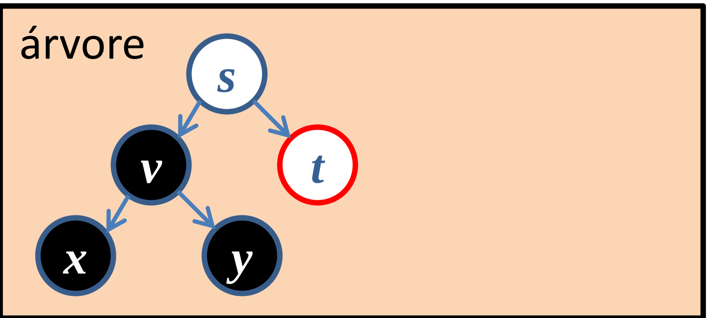




3 - Busca em Profundidade (*Depth-First Search*)

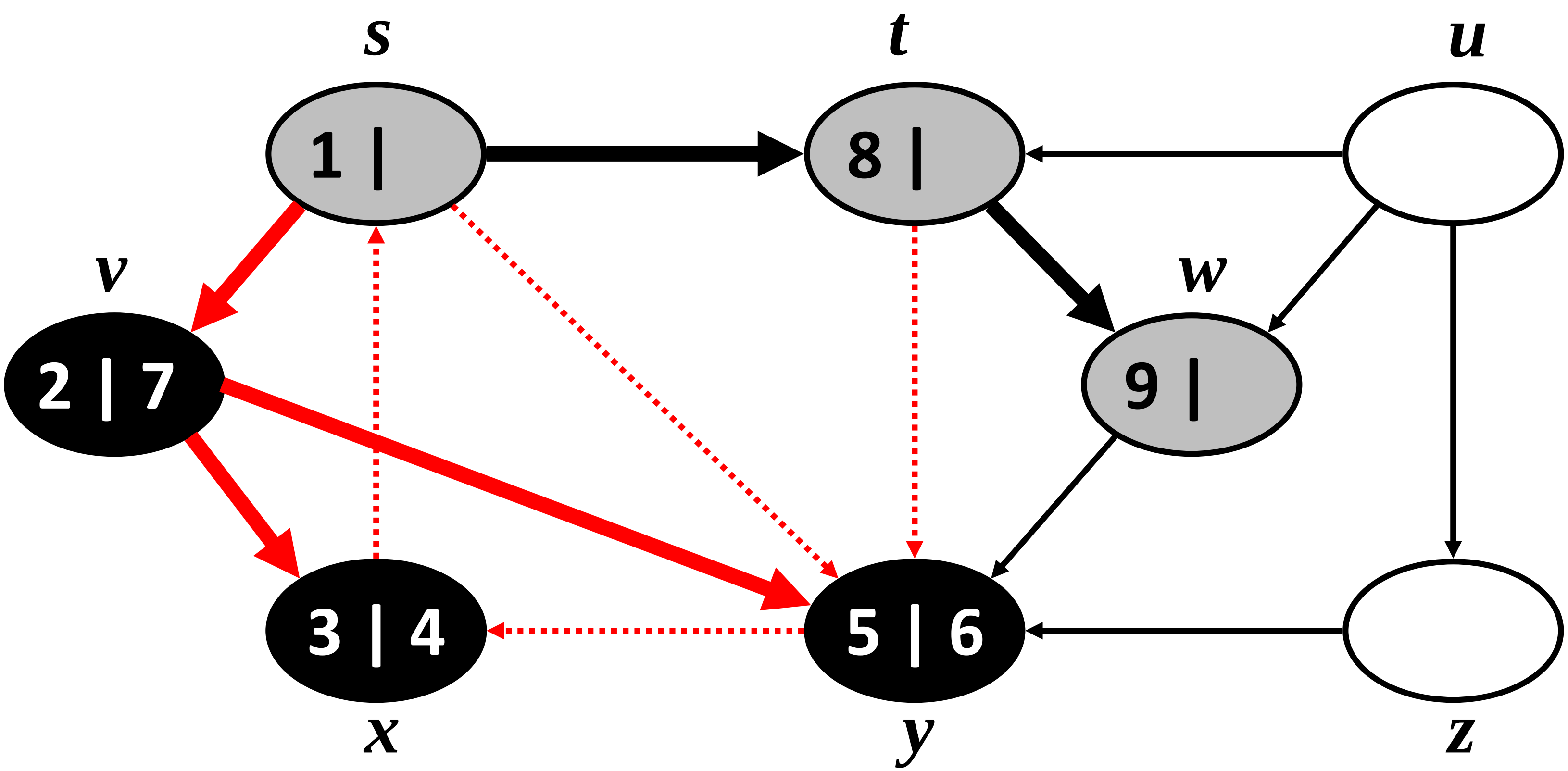
Exemplo DFS:

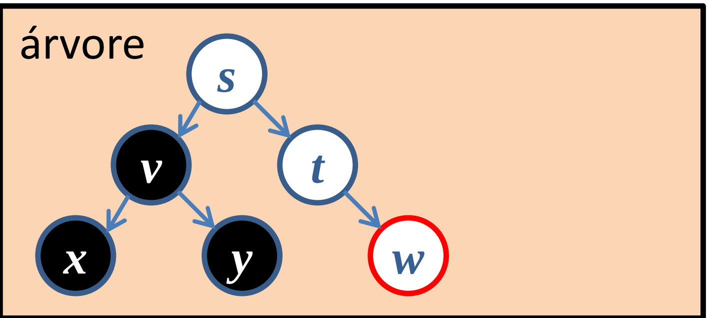




3 - Busca em Profundidade (*Depth-First Search*)

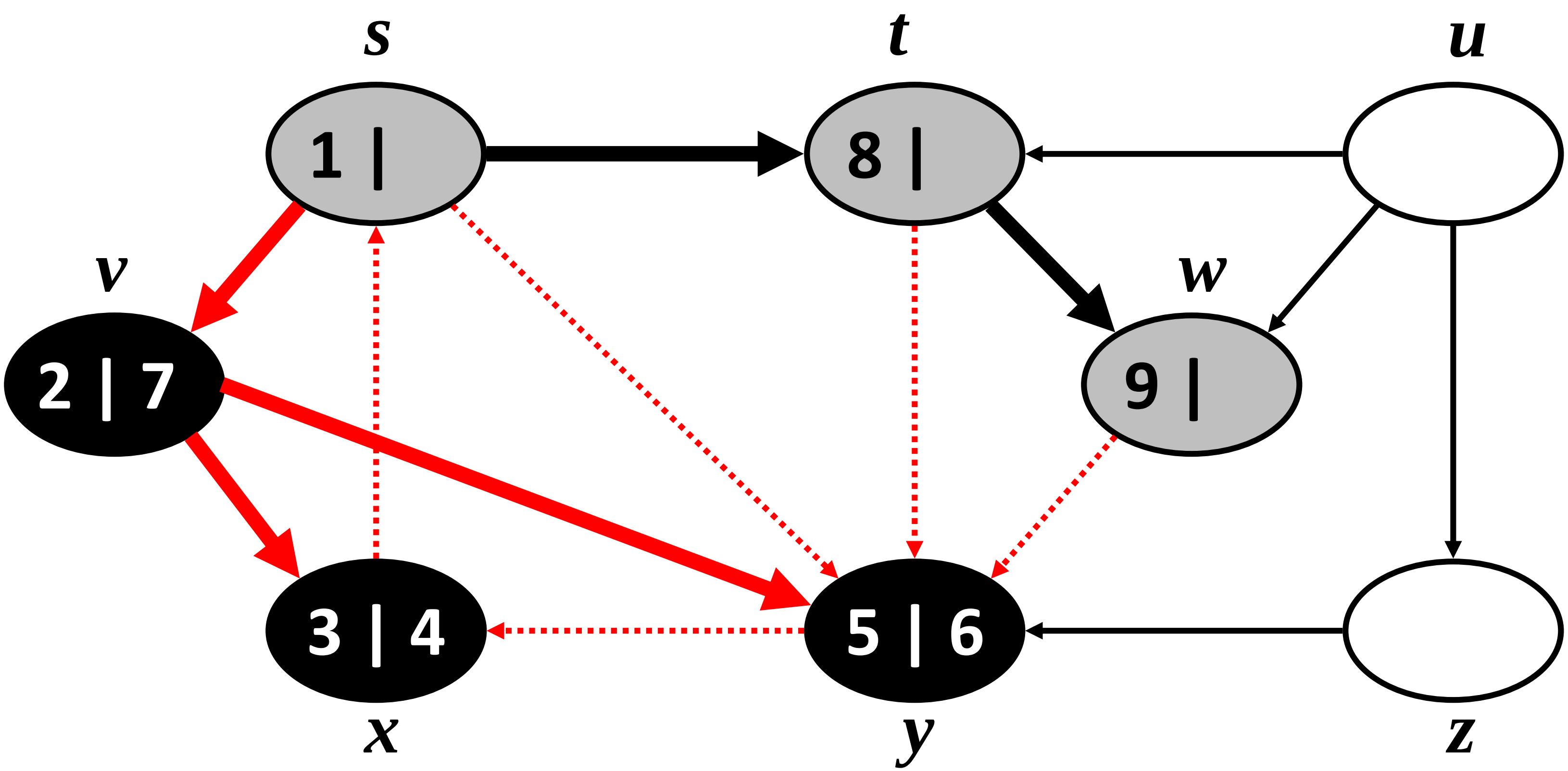
Exemplo DFS:

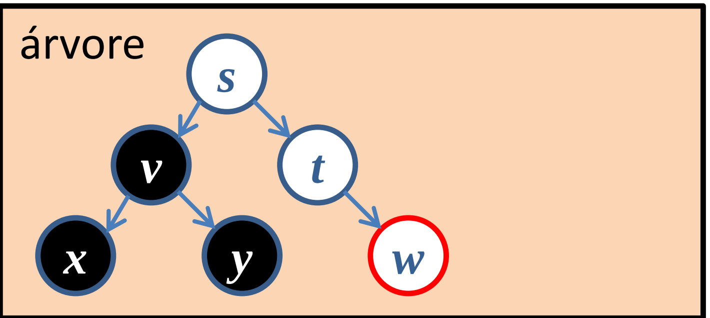




3 - Busca em Profundidade (*Depth-First Search*)

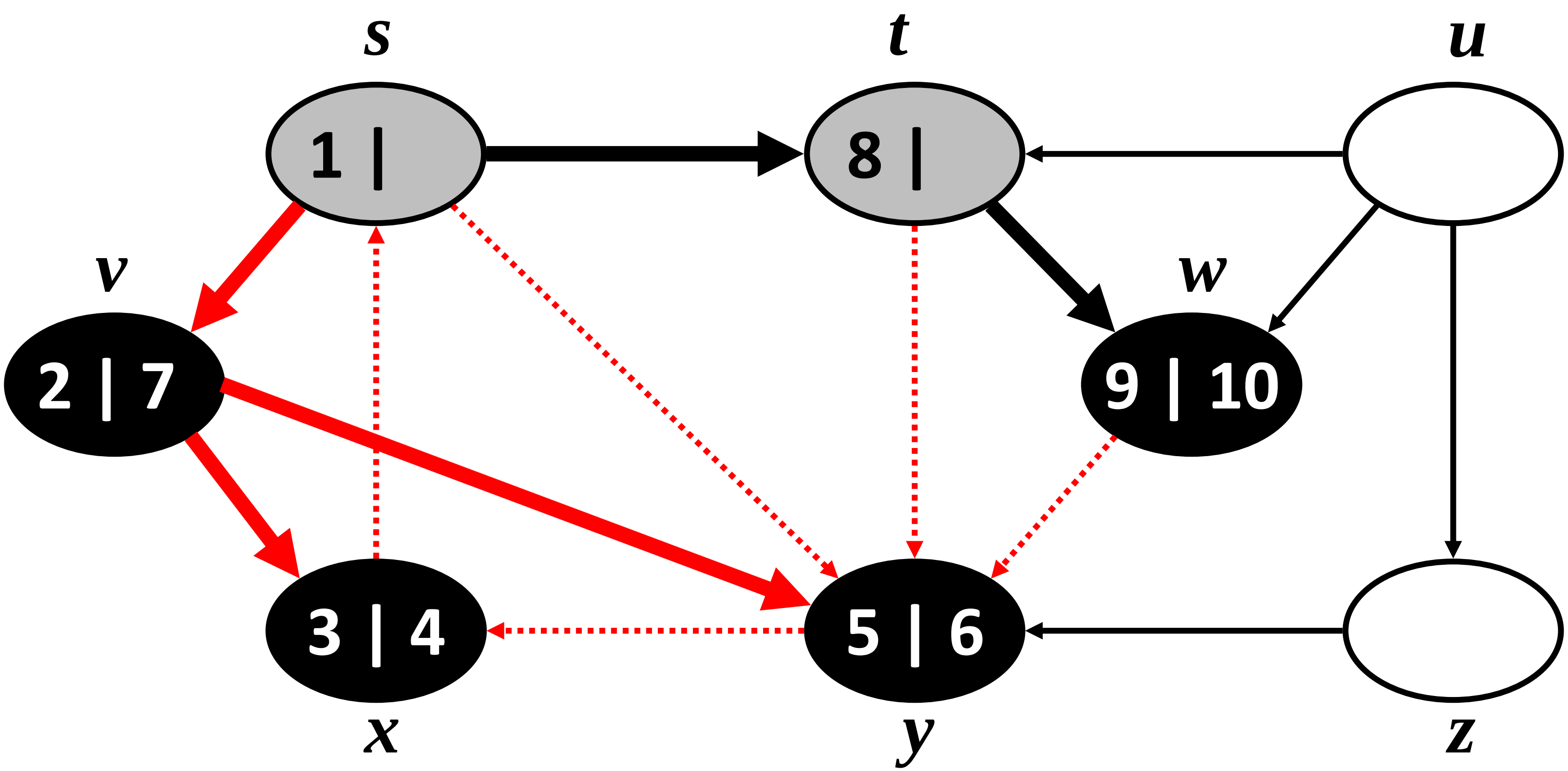
Exemplo DFS:

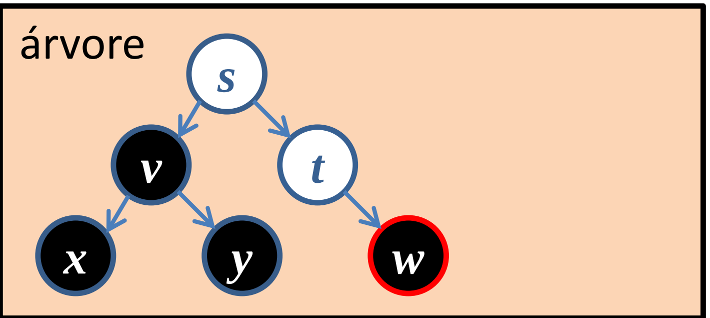




3 - Busca em Profundidade (*Depth-First Search*)

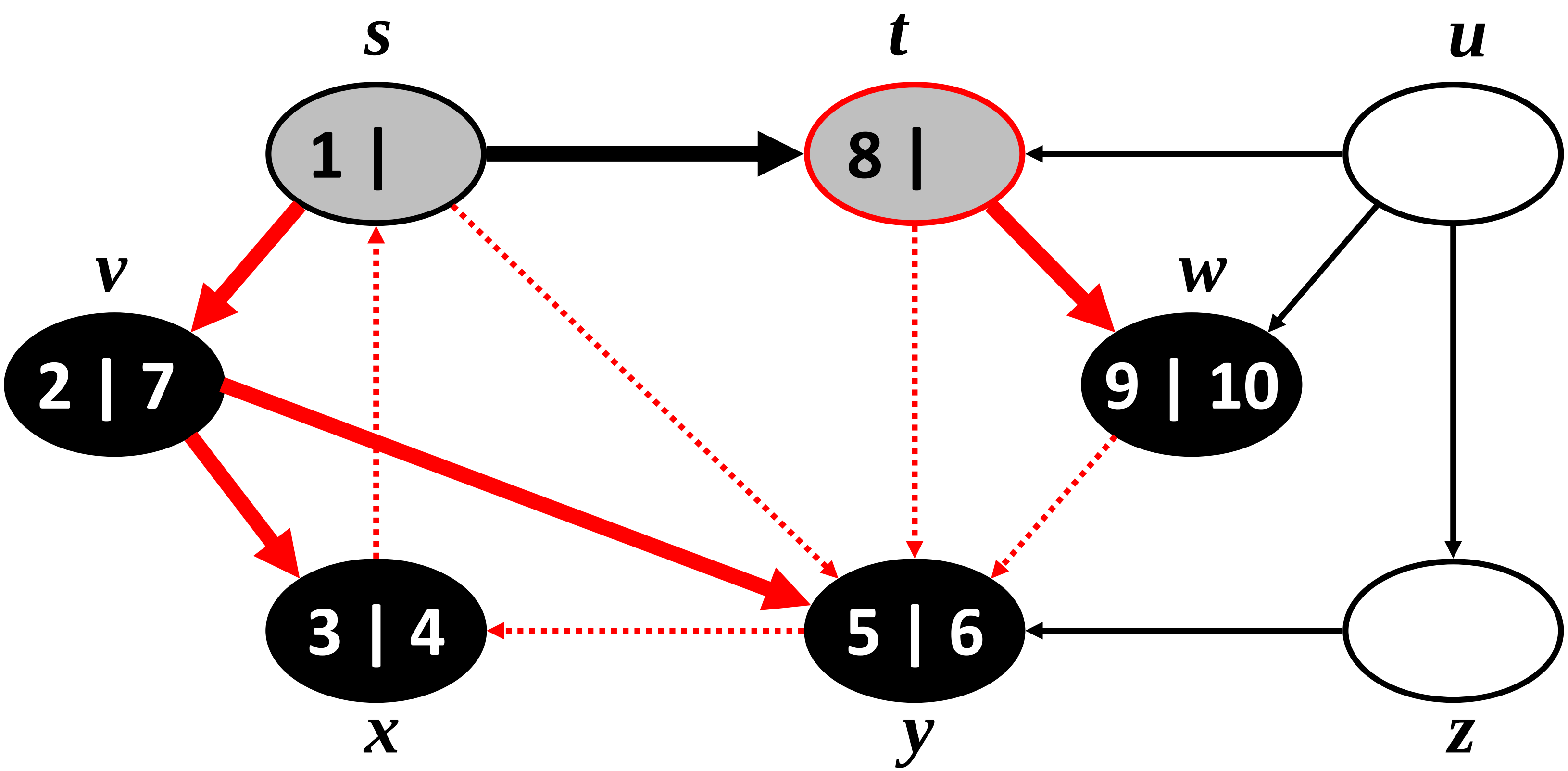
Exemplo DFS:

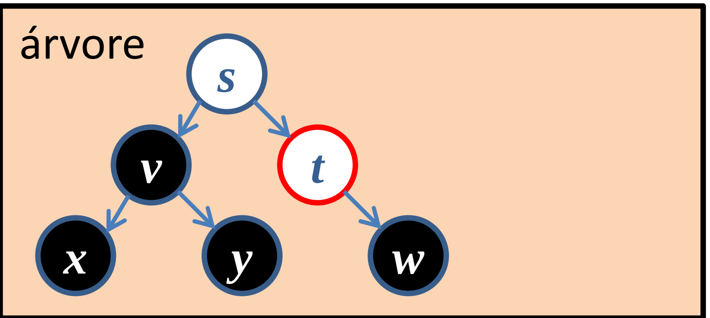




3 - Busca em Profundidade (*Depth-First Search*)

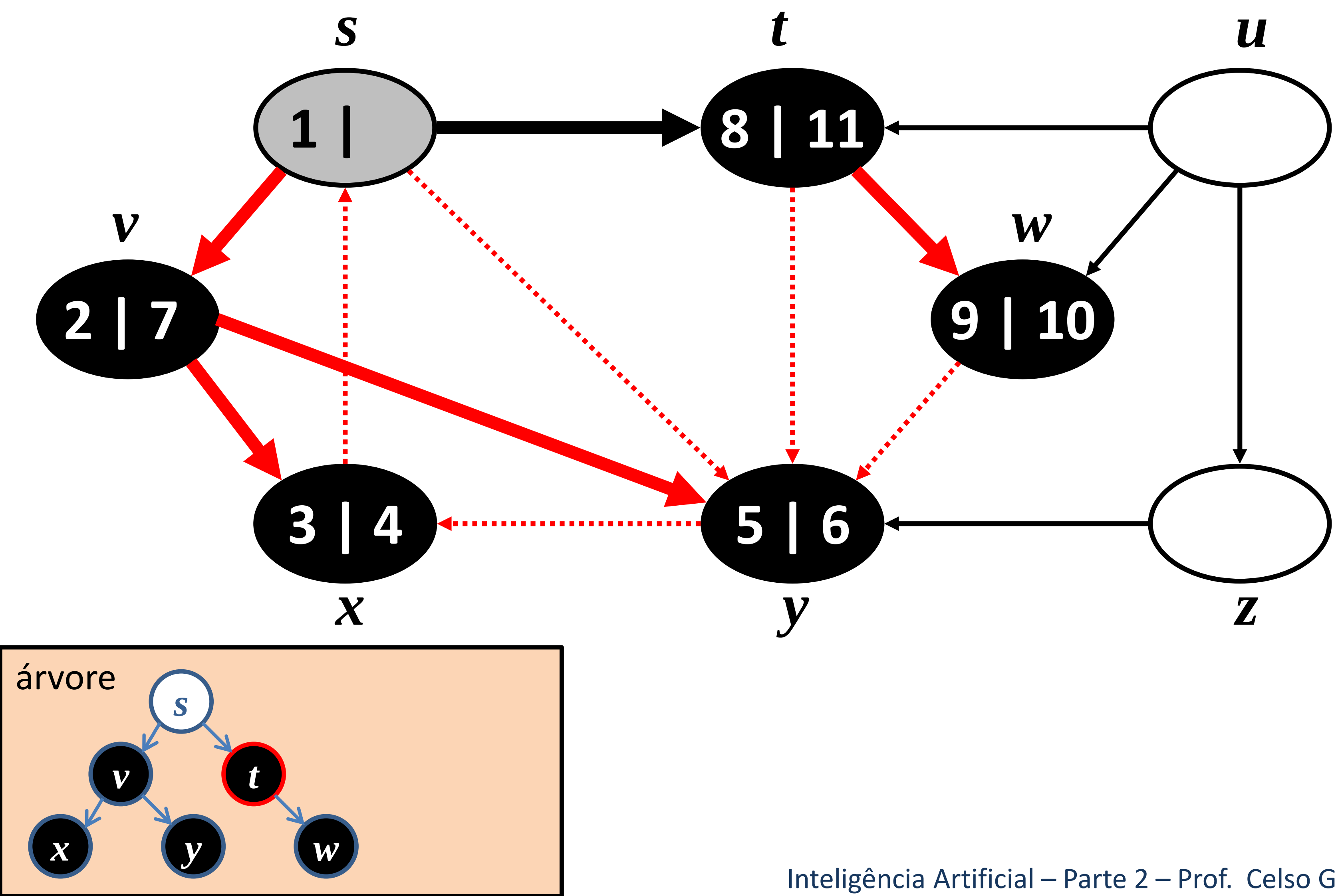
Exemplo DFS:





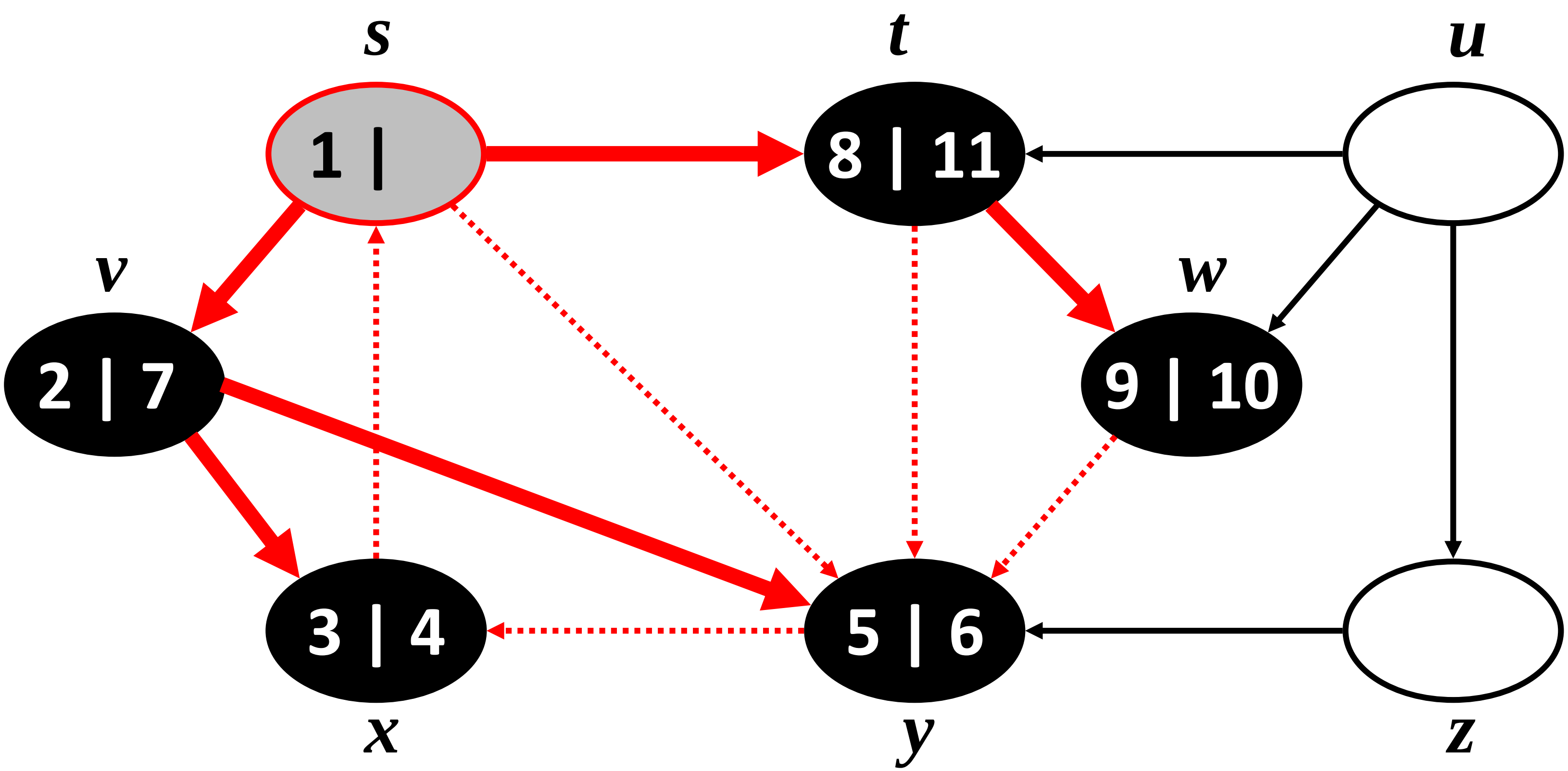
3 - Busca em Profundidade (*<u>D</u>epth-<u>F</u>irst <u>S</u>earch*)

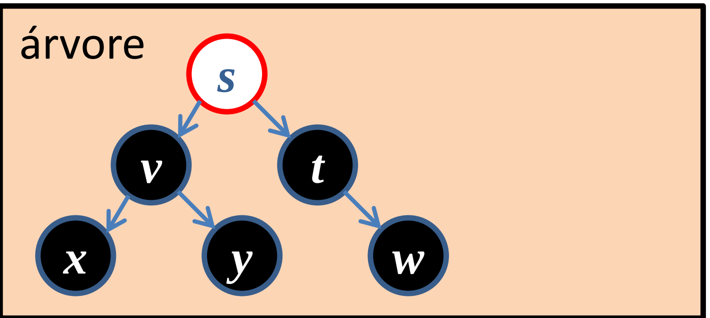
Exemplo DFS:



3 - Busca em Profundidade (*Depth-First Search*)

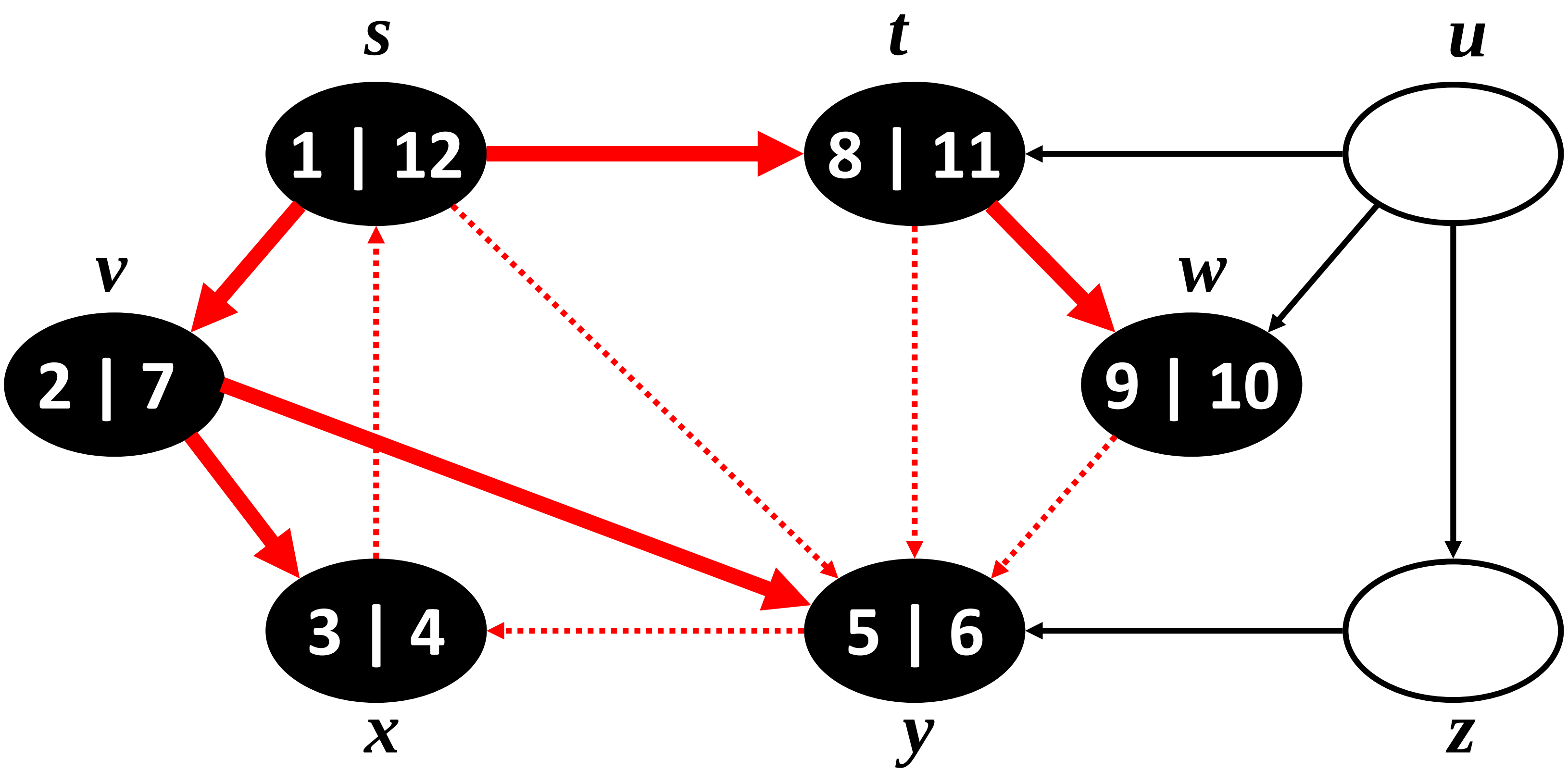
Exemplo DFS:

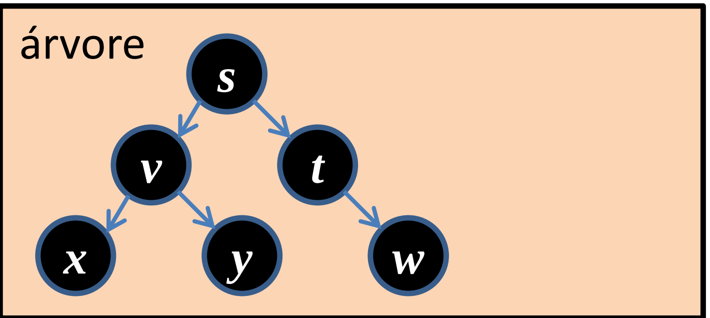




3 - Busca em Profundidade (*Depth-First Search*)

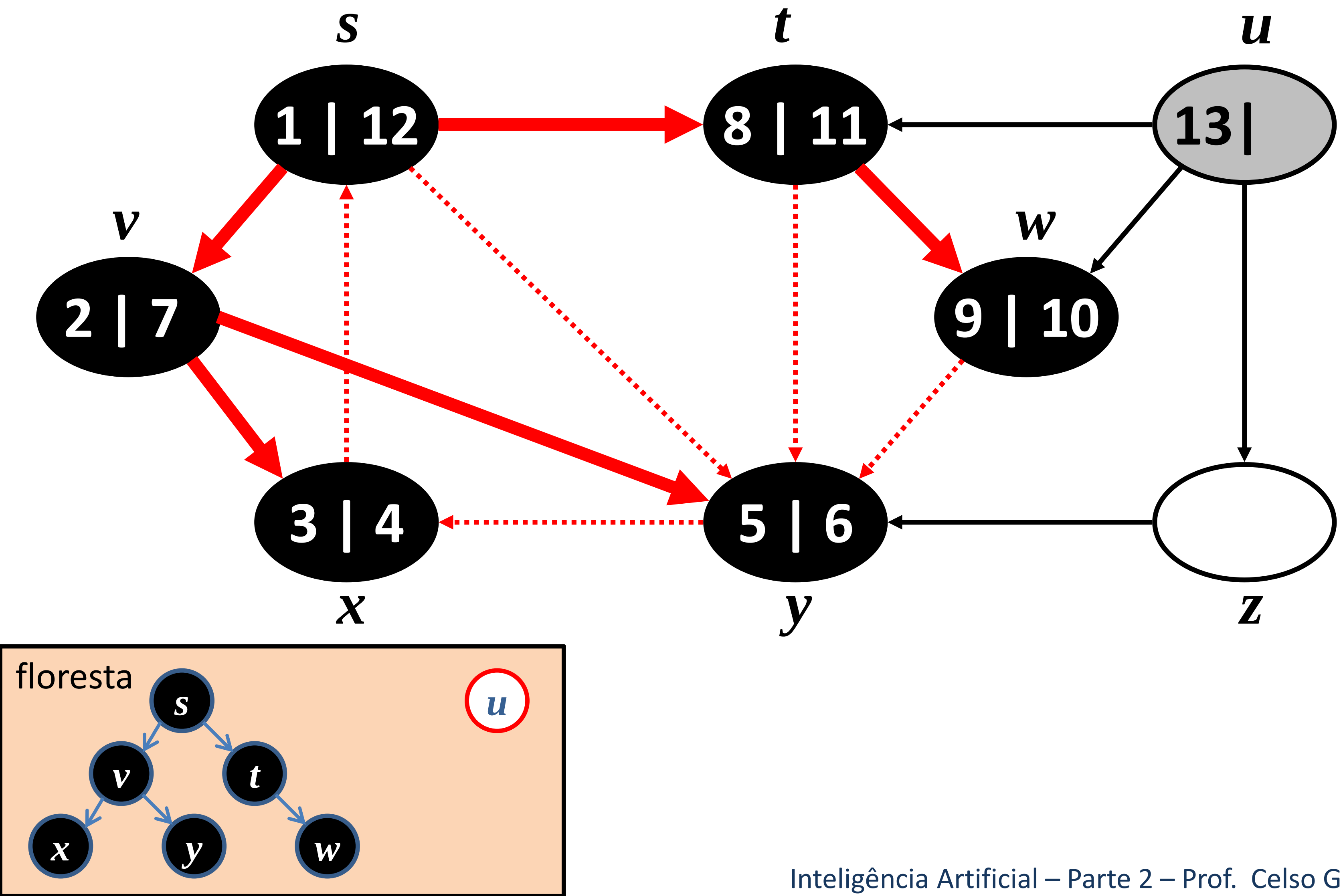
Exemplo DFS:





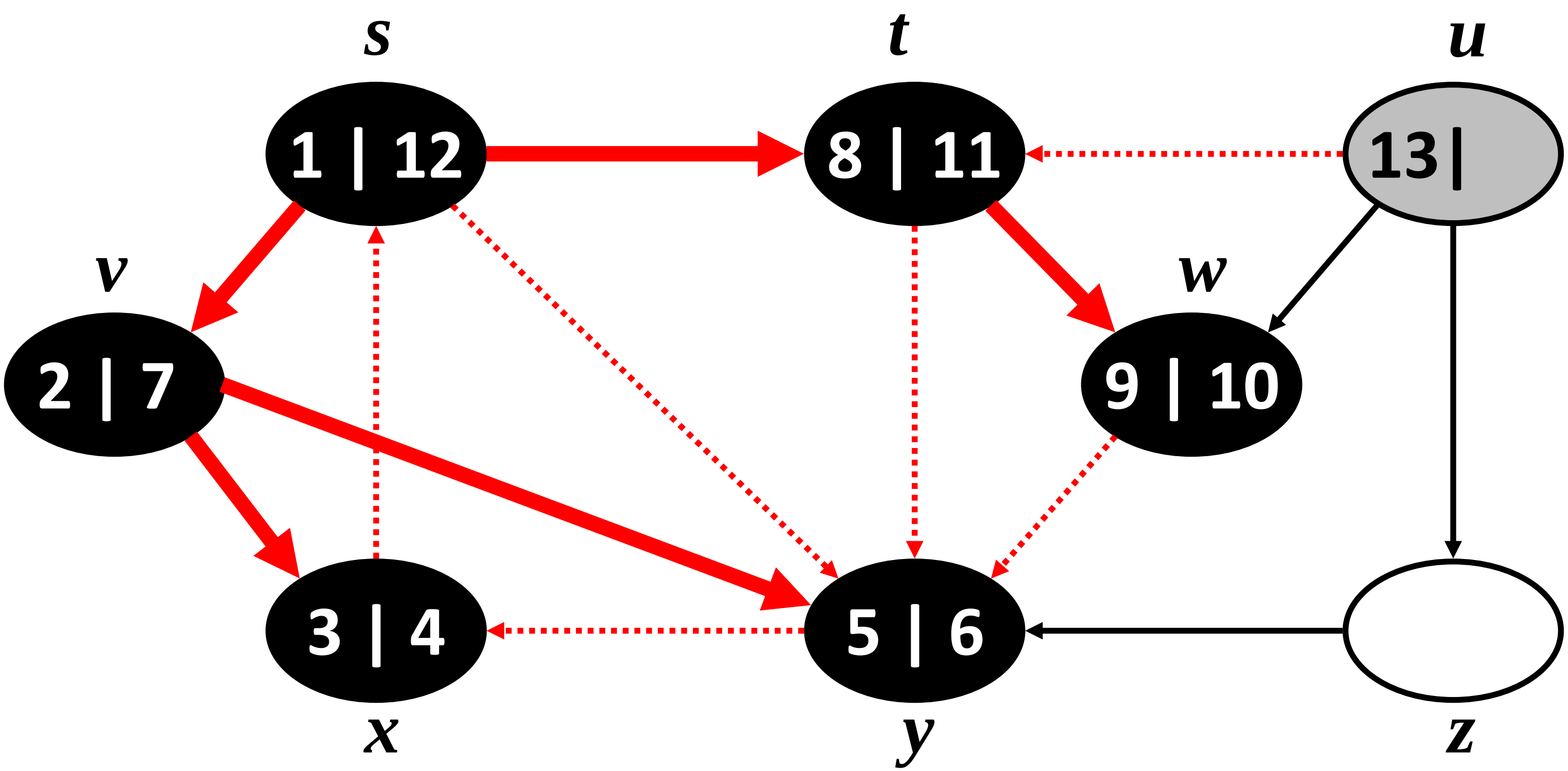
3 - Busca em Profundidade (*Depth-First Search*)

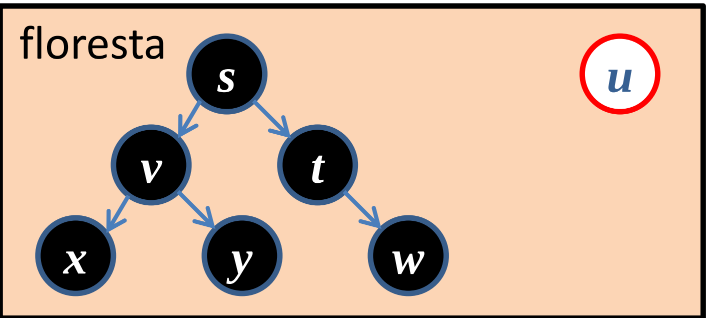
Exemplo DFS:



3 - Busca em Profundidade (*Depth-First Search*)

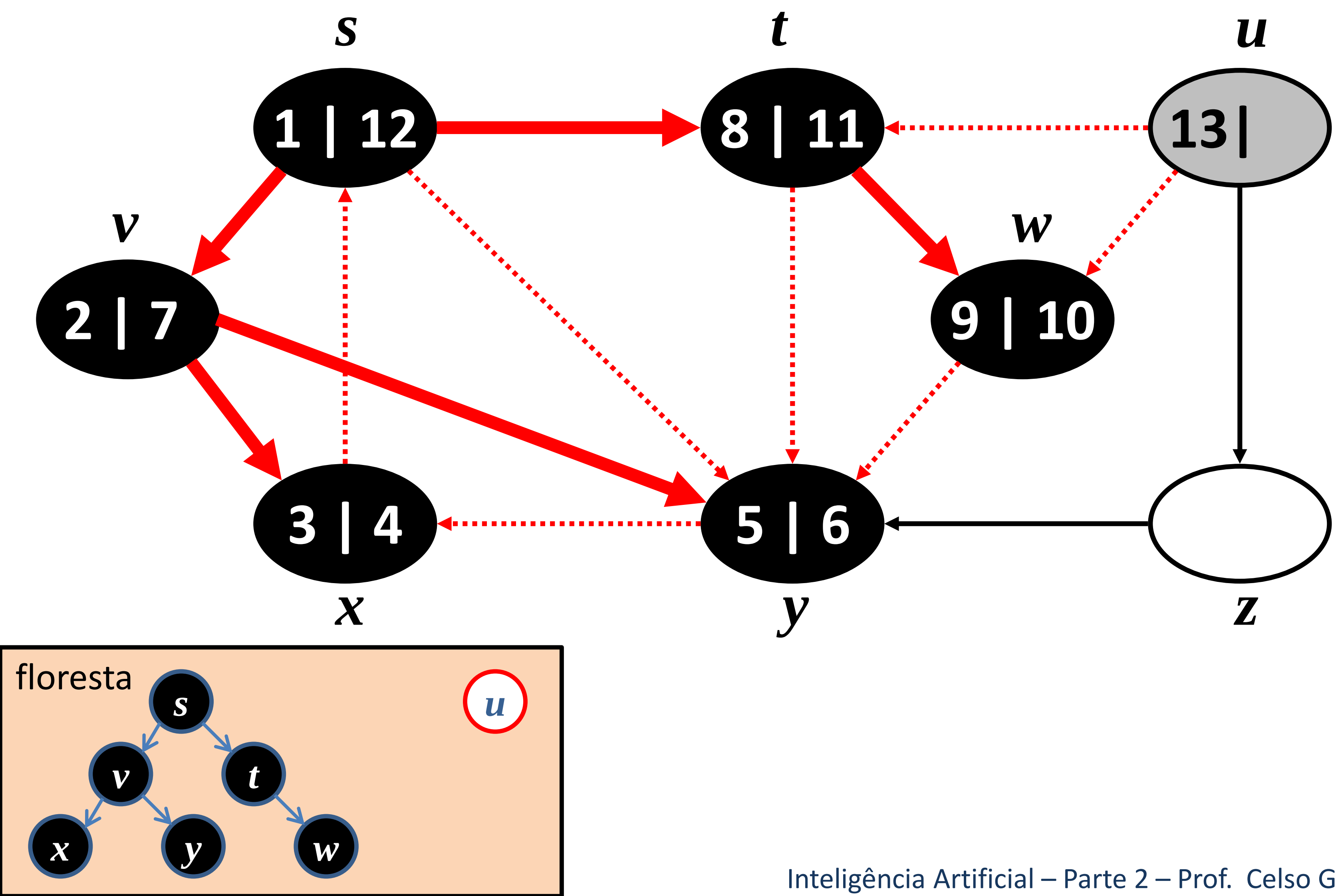
Exemplo DFS:





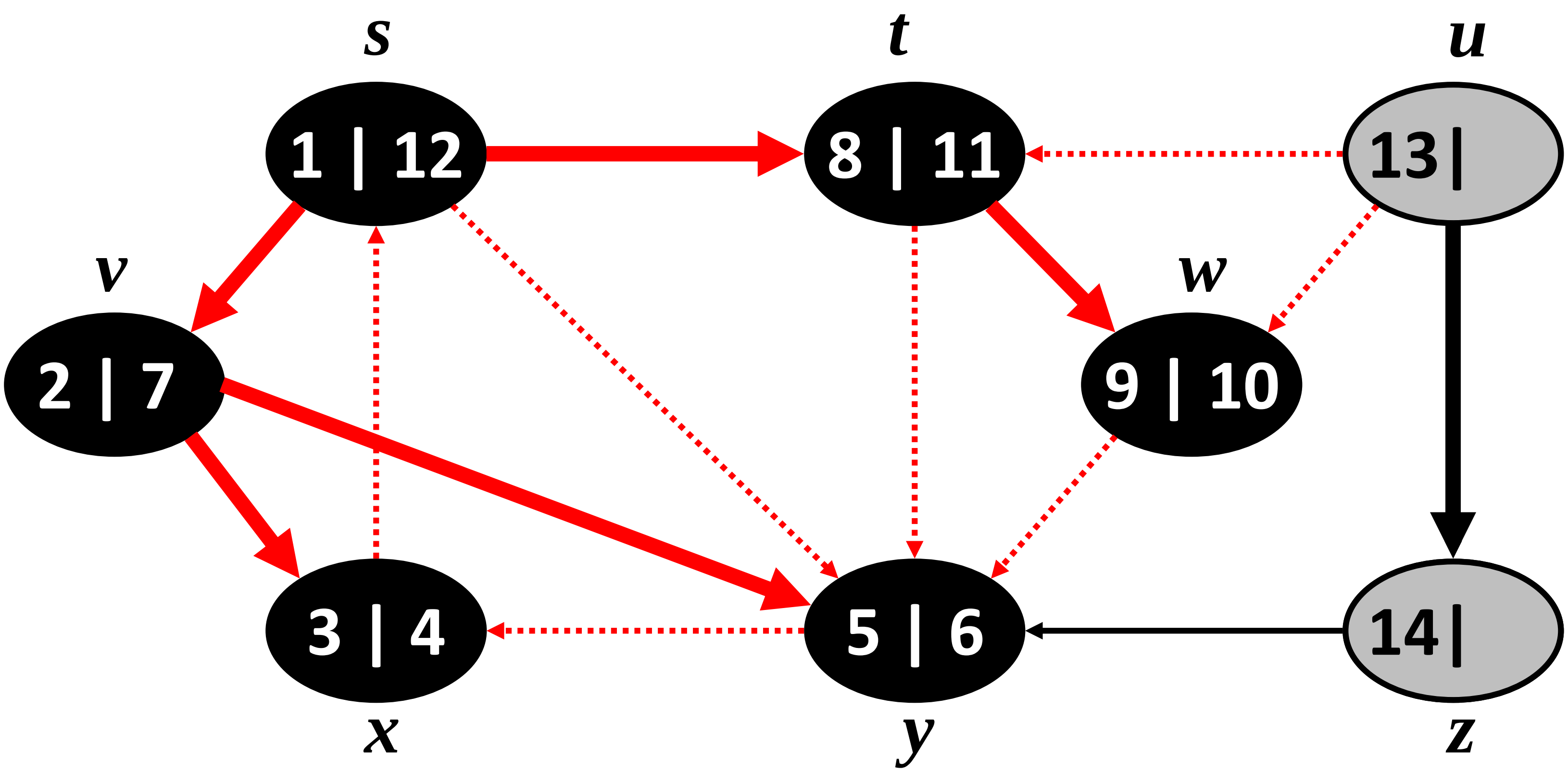
3 - Busca em Profundidade (*Depth-First Search*)

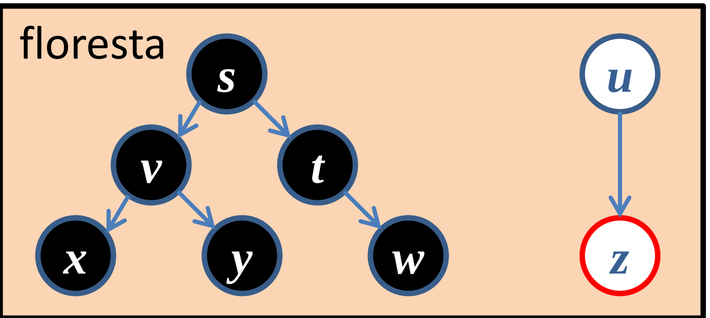
Exemplo DFS:



3 - Busca em Profundidade (*Depth-First Search*)

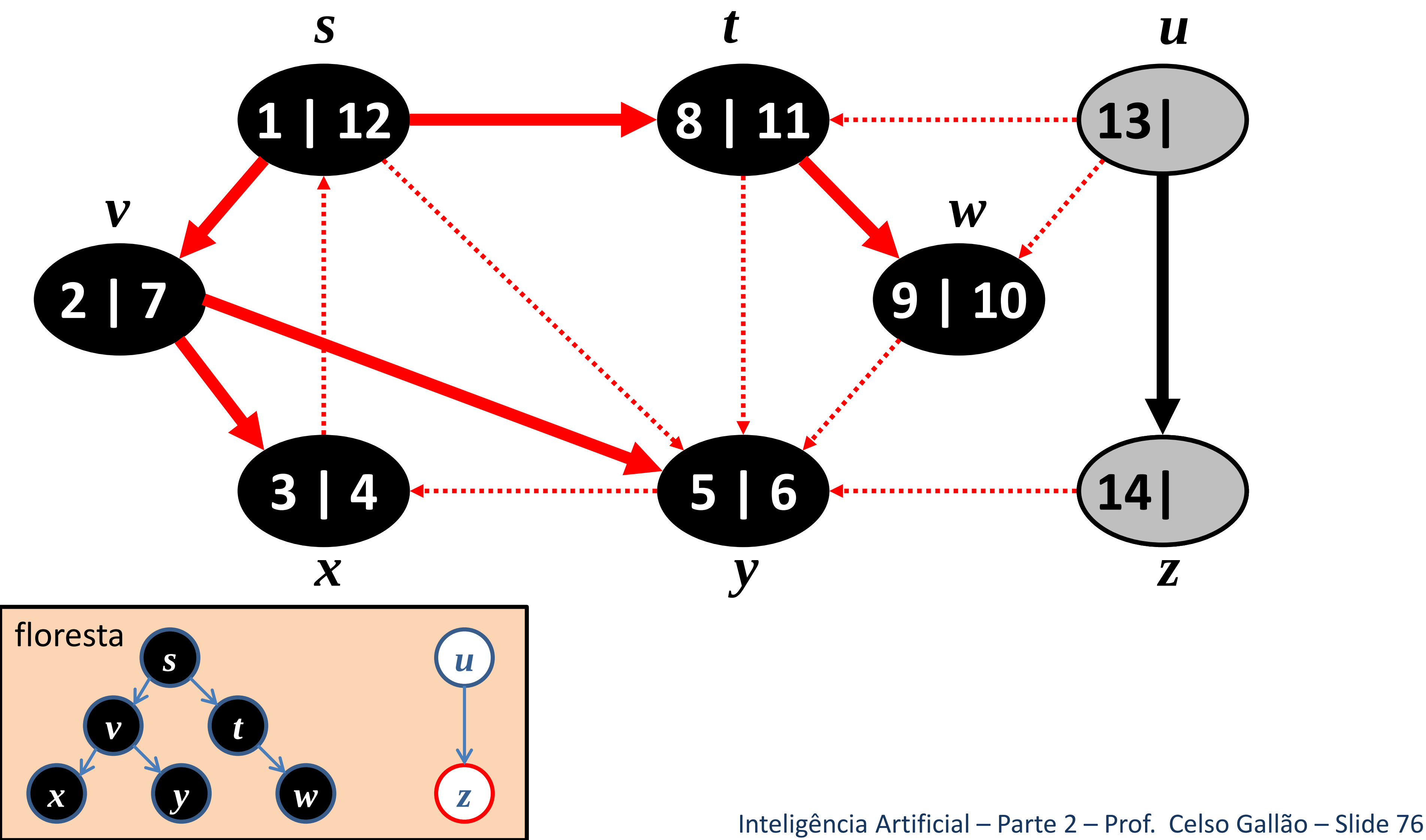
Exemplo DFS:





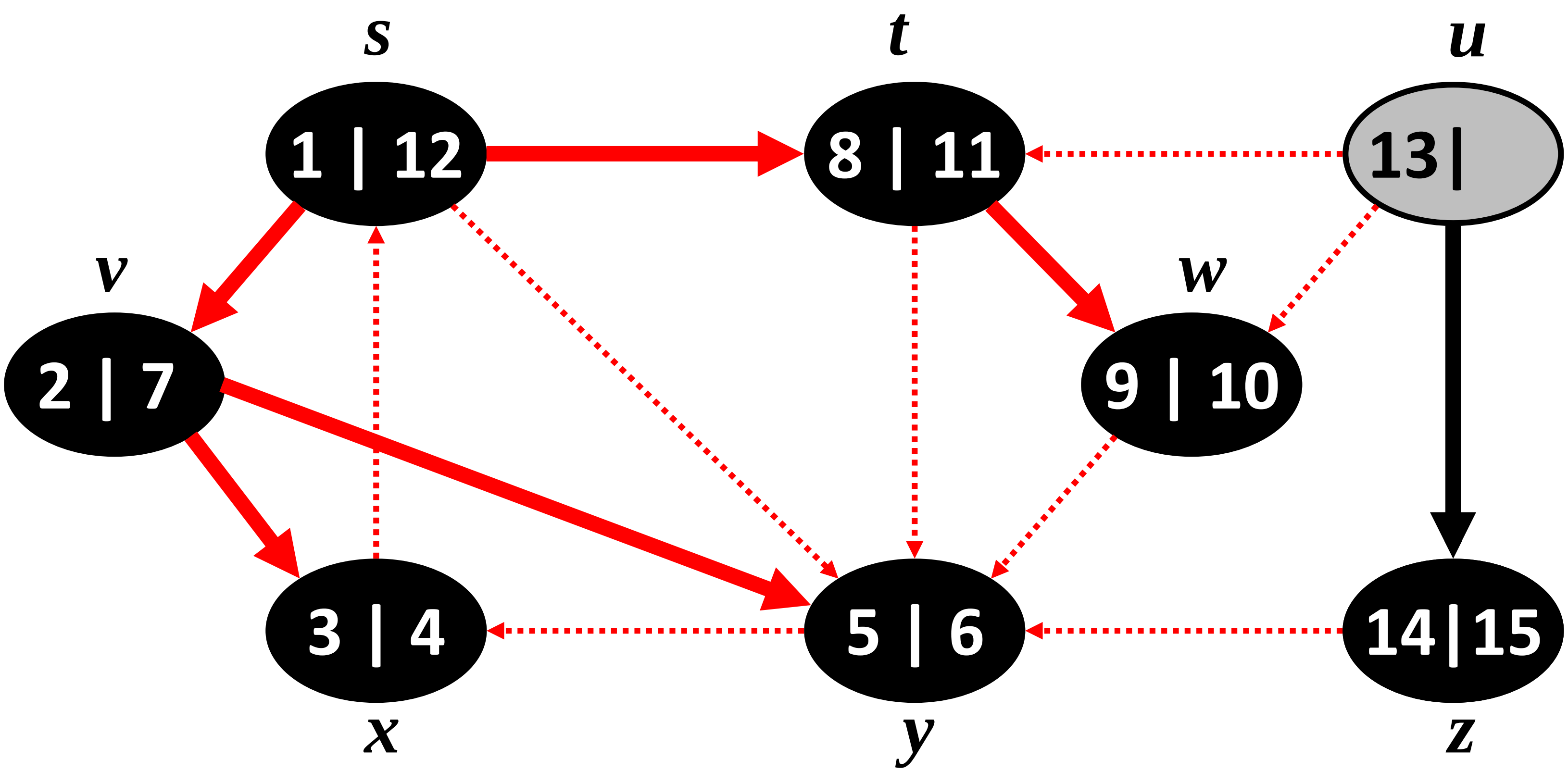
3 - Busca em Profundidade (*Depth-First Search*)

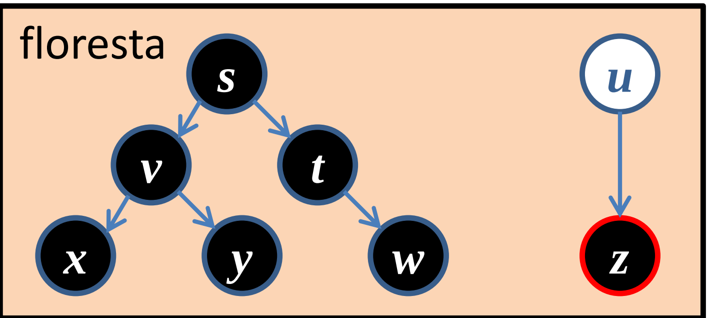
Exemplo DFS:



3 - Busca em Profundidade (*Depth-First Search*)

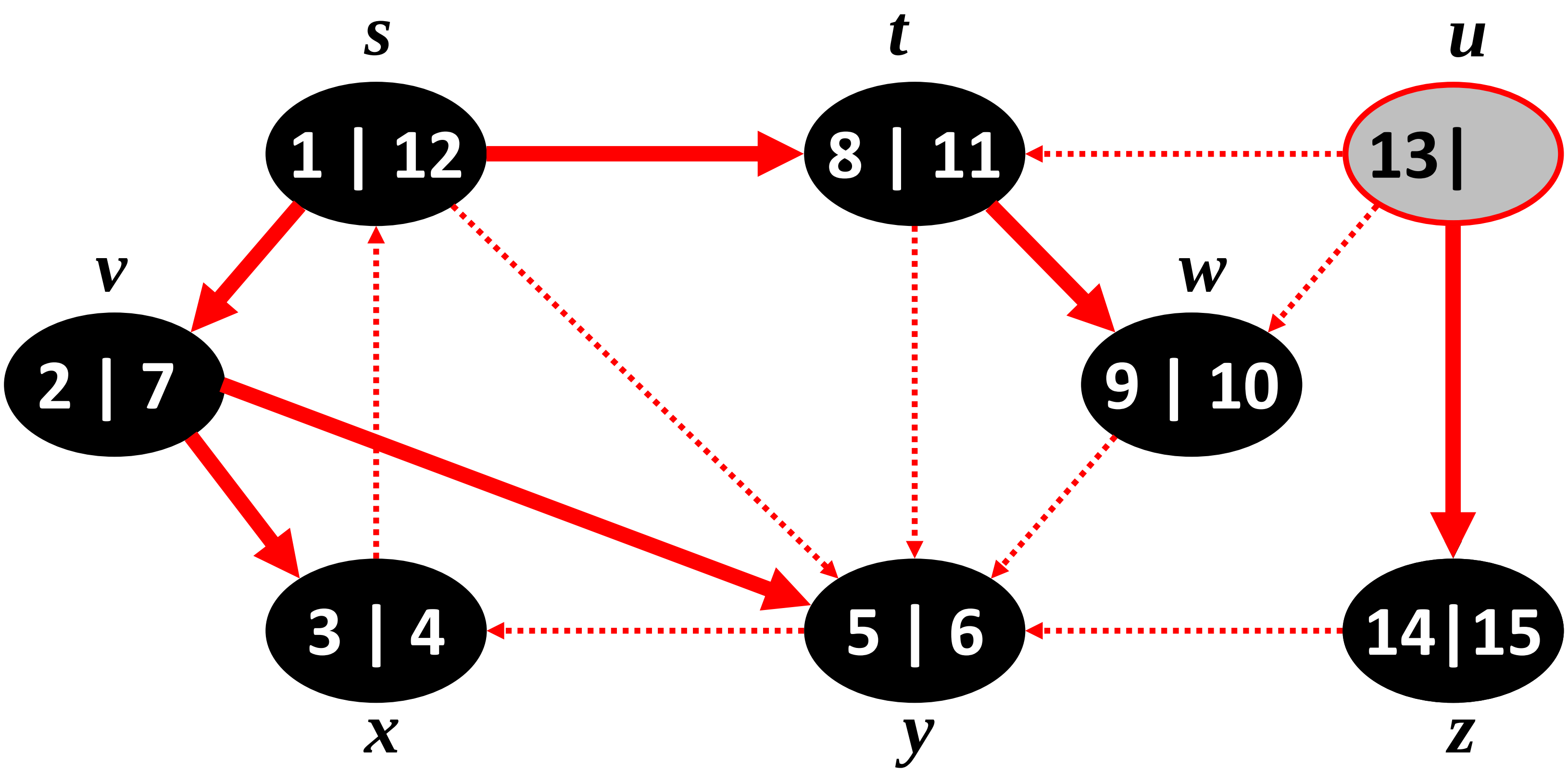
Exemplo DFS:

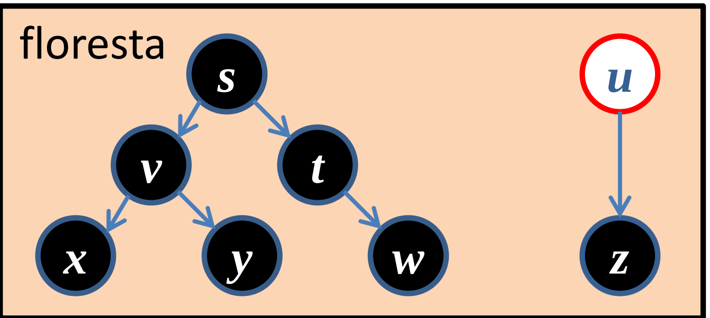




3 - Busca em Profundidade (*Depth-First Search*)

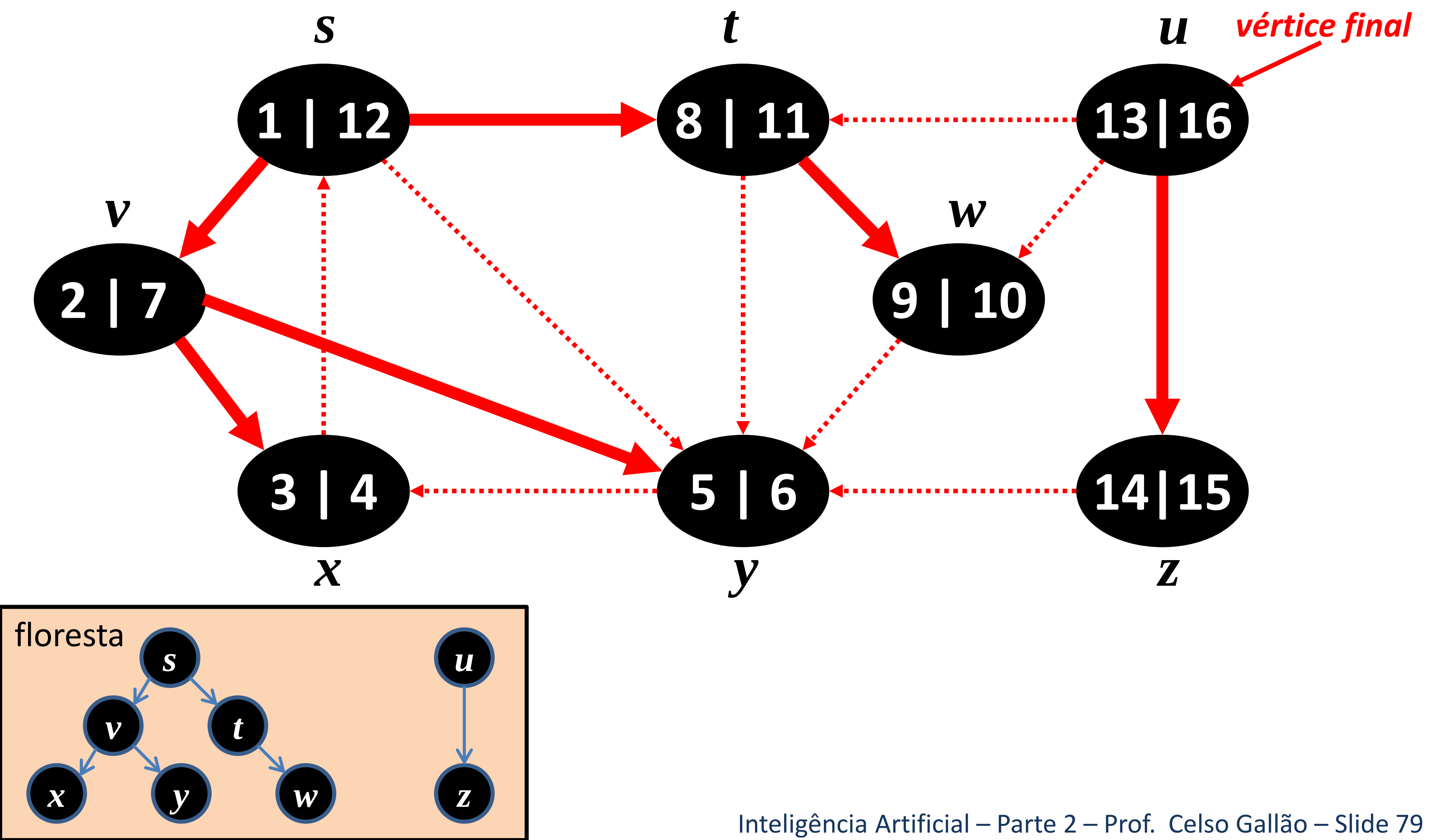
Exemplo DFS:





3 - Busca em Profundidade (*Depth-First Search*)

Exemplo DFS:



3 - Busca em Profundidade (*<u>D</u>epth-<u>F</u>irst <u>S</u>earch*)

3.3 – O Algoritmo DFS:

- Assumir que o grafo $G = (V, E)$ é representado com lista de adjacências.
- Para cada vértice no grafo, o algoritmo mantém estruturas auxiliares:
  - A variável ***cor[v']*** mantém a informação sobre a cor de cada vértice.
  - A variável ***pai[v']*** mantém a informação do predecessor de cada vértice. Quando não existe predecessor ***pai[u] = NIL***.

3 - Busca em Profundidade (*Depth-First Search*)

3.3 – O Algoritmo DFS:

- Assumir que o grafo $G = (V, E)$ é representado com lista de adjacências.
- Para cada vértice no grafo, o algoritmo mantém estruturas auxiliares:
  - A variável *d[v']* mantém o valor do tempo quando *v' foi visitado pela primeira vez.*
  - A variável *F[v']* mantém o valor do tempo quando *v' foi totalmente explorado.*
  - O vértice inicial é *s*.
  - O vértice observado é *v'*.

3 - Busca em Profundidade (*Depth-First Search*)

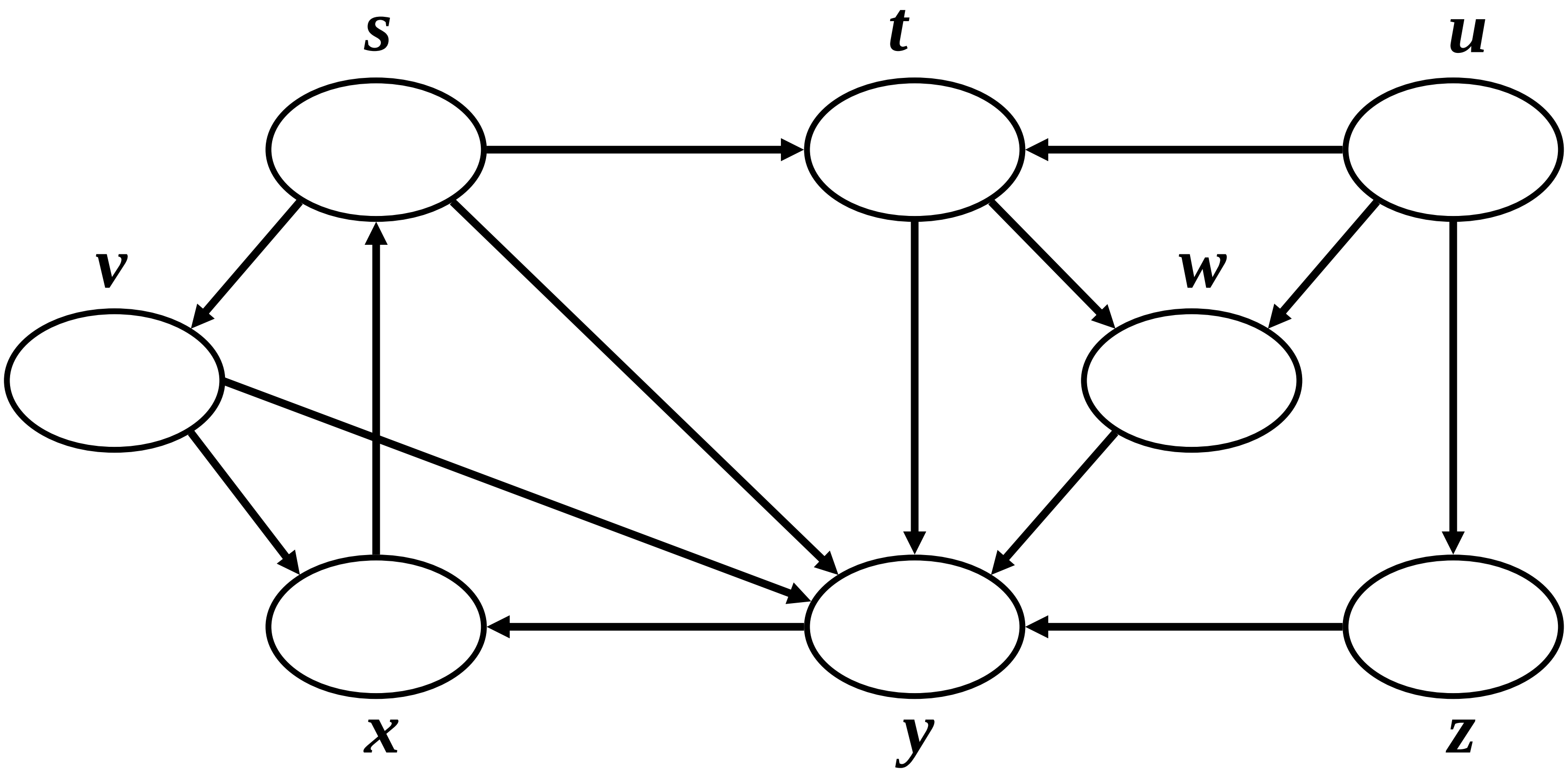
3.3 – O Algoritmo DFS:

```
DFS(G)
for ∀u ∈ V[G] do
        cor[v’] ← BRANCO
        pai[v’] ← NIL
tempo ← 0
for ∀v’ ∈ V[G] do
        if cor[v’] = BRANCO then
                VisitaDFS(u)
```

```
VisitaDFS(u)
        cor[v’] ← CINZA
        d[v’] ← tempo ← tempo+1
for ∀v ∈ Adjacente[v’] do
        if cor[v] = BRANCO then
                pai[v] ← v’
                VisitaDFS(v)
cor[v’] ← PRETO
F[v’] ← tempo ← tempo+1
```

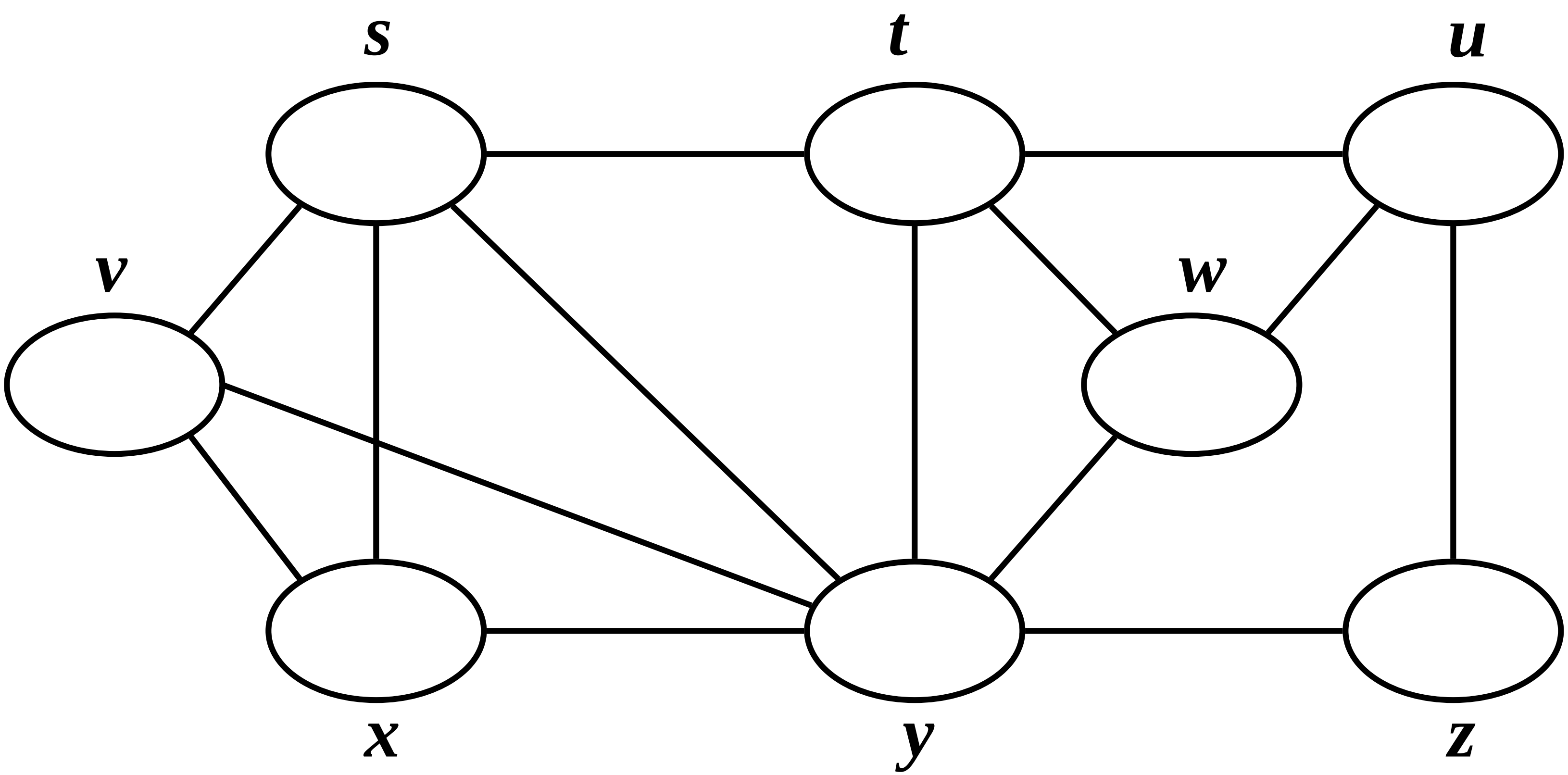
3 - Busca em Profundidade (*Depth-First Search*)

3.4 – Exercícios de DFS: 1) início em x:



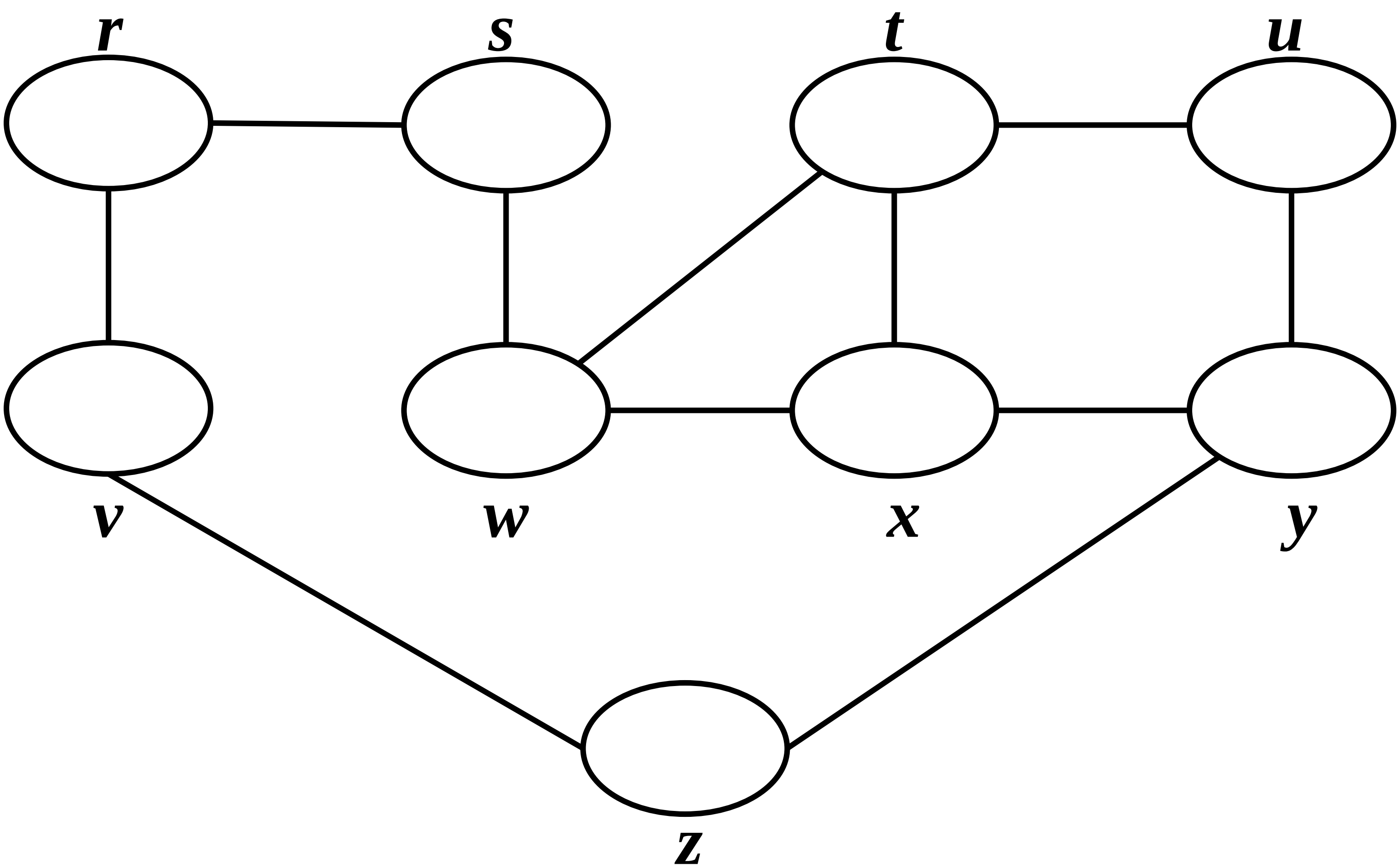
3 - Busca em Profundidade (*<u>D</u>epth-<u>F</u>irst <u>S</u>earch*)

3.4 – Exercícios de DFS: 2) início em *x*:



3 - Busca em Profundidade (*Depth-First Search*)

3.4 – Exercícios de DFS: 3) início em *x*:



Bibliografias

Obrigatórias:

1. CORMEN, LEISERSON, RIVEST, STEIN, **Algoritmos: Teoria e Prática**. 2ª ed. Rio de Janeiro: Campus, 2002: Capítulo 22 e Parte VIII Apêndice B.4.
2. RUSSELL, Stuart J; NORVIG, Peter. **Inteligência Artificial.** 2ª ed. Rio de Janeiro: Campus, 2004, Capítulos 1 e 2.
3. LUGER, George. **Inteligência Artificial: Estruturas e Estratégias para a Resolução de Problemas Complexos**. 4ª ed. Porto Alegre: Bookman, 2004, Capítulo 1.

Bibliografias

Recomendadas:

1. CORMEN, LEISERSON, RIVEST, STEIN, **Algoritmos: Teoria e Prática**. 2ª ed. Rio de Janeiro: Campus, 2002: Capítulo 3.
2. ARTERO, Almir Olivette, **Inteligência Artificial: Teórica e Prática**. 1ª ed. São Paulo: Livraria da Física, 2009: Capítulo 4.
3. ROSA, João Luis Garcia, **Fundamentos da Inteligência Artificial**. 1ª ed. Rio de Janeiro: LTC, 2011: Capítulo 2.