Algoritmos e Estrutura de Dados II (AE23CP-3CP)

Aula #08 - Grafos - Algoritmos de Busca:
-Busca em Profundidade

Profª Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Roteiro

Algoritmos de busca em grafos:

- Busca em profundidade

Busca em grafos - motivação

O que é busca em grafos?

Processo de seguir sistematicamente pelas arestas a fim de visitar os vértices do grafo.

Para que serve?

Pode-se obter várias informações sobre a estrutura do grafo, que podem ser úteis para projetar algoritmos eficientes para determinados problemas.

Algoritmos

Grafos são estruturas mais complicadas do que vetores, listas e árvores (binárias). Temos dois métodos para explorar/percorrer um grafo (orientado ou não-orientado):

- Busca em largura (*breadth-first search*)
- Busca em profundidade (*depth-first search*)

Busca em profundidade

Busca em profundidade

Depth First Search = busca em profundidade

- A estratégia consiste em pesquisar o grafo o mais "profundamente" sempre que possível.
- Aplicável tanto a grafos orientados quanto não-orientados.
- Possui um número enorme de aplicações:
  - determinar os componentes de um grafo
  - ordenação topológica
  - determinar componentes fortemente conexos
  - subrotina para outros algoritmos

Algoritmo de busca em profundidade

Entrada

Recebe um grafo $G = (V, E)$ (representado por listas de adjacências).

Processamento

A busca inicia-se em um vértice qualquer. **Busca em profundidade** é um método recursivo. A idéia básica consiste no seguinte:

- Suponha que a busca atingiu um vértice u. Escolhe-se um vizinho não visitado v de u para prosseguir a busca.
- "Recursivamente" a busca em profundidade prossegue a partir de v.
- Quando esta busca termina, tenta-se prosseguir a busca a partir de outro vizinho de u. Se não for possível, ela retorna (*backtracking*) ao nível anterior da recursão.

Algoritmo de busca em profundidade

Saída

Constrói uma **Floresta de Busca em Profundidade**.

- A busca em profundidade associa a cada vértice x um predecessor $\pi[x]$.
- O subgrafo induzido pelas arestas $\{(\pi[x], x) : x \in V[G] \quad e \quad \pi[x] \neq NIL\}$ é a **Floresta de Busca em Profundidade**.
- Cada componente desta floresta é uma **Árvore de Busca em Profundidade**.

Algoritmo de busca em profundidade

Outra forma de entender **Busca em Profundidade**

Imagine que os vértices são armazenados em uma **pilha** à medida que são visitados. Compare isto com **Busca em Largura** onde os vértices são colocados em uma fila.

- Suponha que a busca atingiu um vértice u.
- Escolhe-se um vizinho não visitado v de u para prosseguir a busca.
- Empilhe v e repete-se o passo anterior com v.
- Se nenhum vértice não visitado foi encontrado, então desempilhe um vértice da pilha, digamos u, e volte ao primeiro passo.

Algoritmo de busca em profundidade

Cores dos vértices

A medida que o grafo é percorrido, os vértices visitados vão sendo coloridos.

Cada vértice tem uma das seguintes cores:

- Cor branca = "vértice ainda não visitado". Inicialmente todos os vértices são brancos.
- Cor cinza = "vértice visitado mas ainda não finalizado".
- Cor preta = "vértice visitado e finalizado".

Algoritmo de busca em profundidade

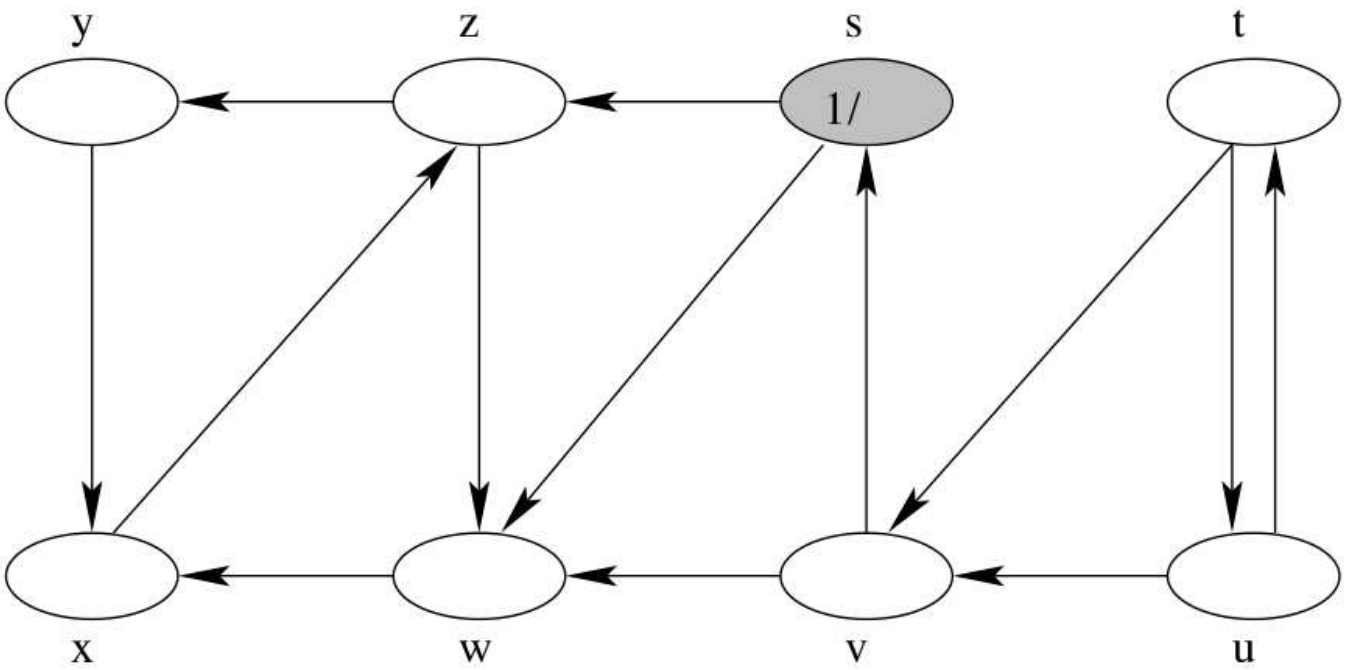
Estampas/rótulos

A busca em profundidade associa a cada vértice x dois rótulos:

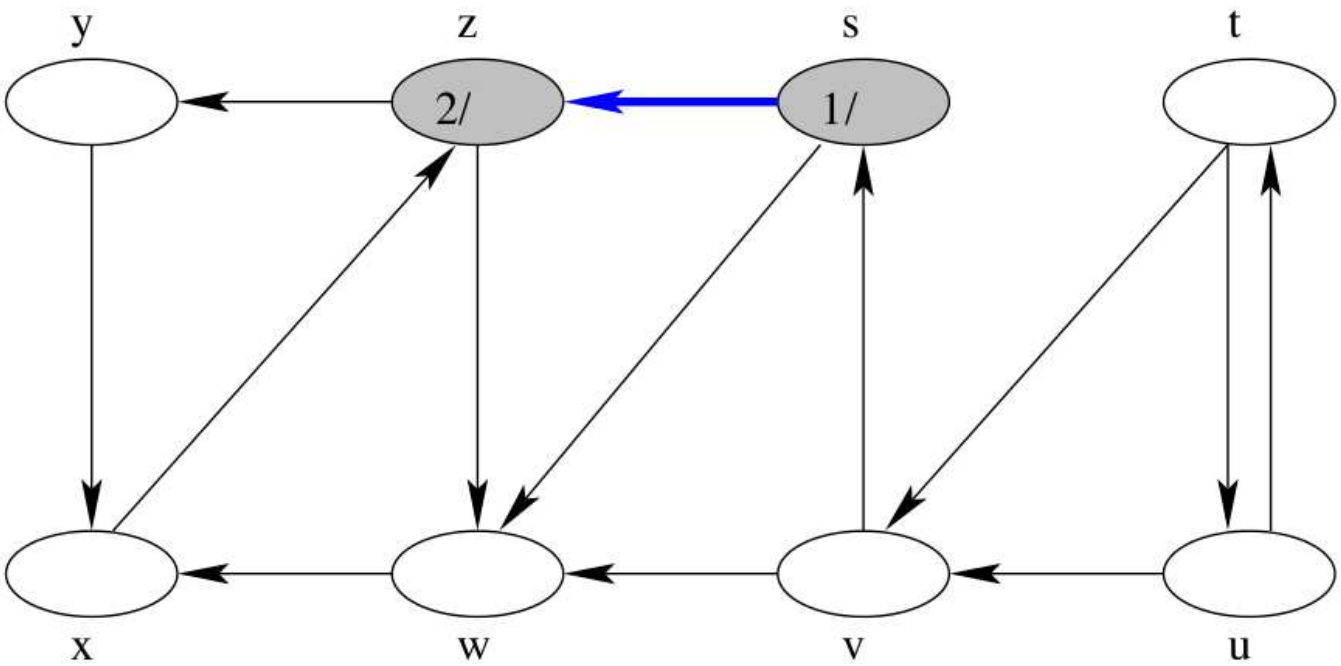
- $d[x]$: instante de descoberta de x. Neste instante x torna-se cinza.
- $f[x]$: instante de finalização de x. Neste instante x torna-se preto.

Os rótulos são inteiros entre 1 e $2|V|$.

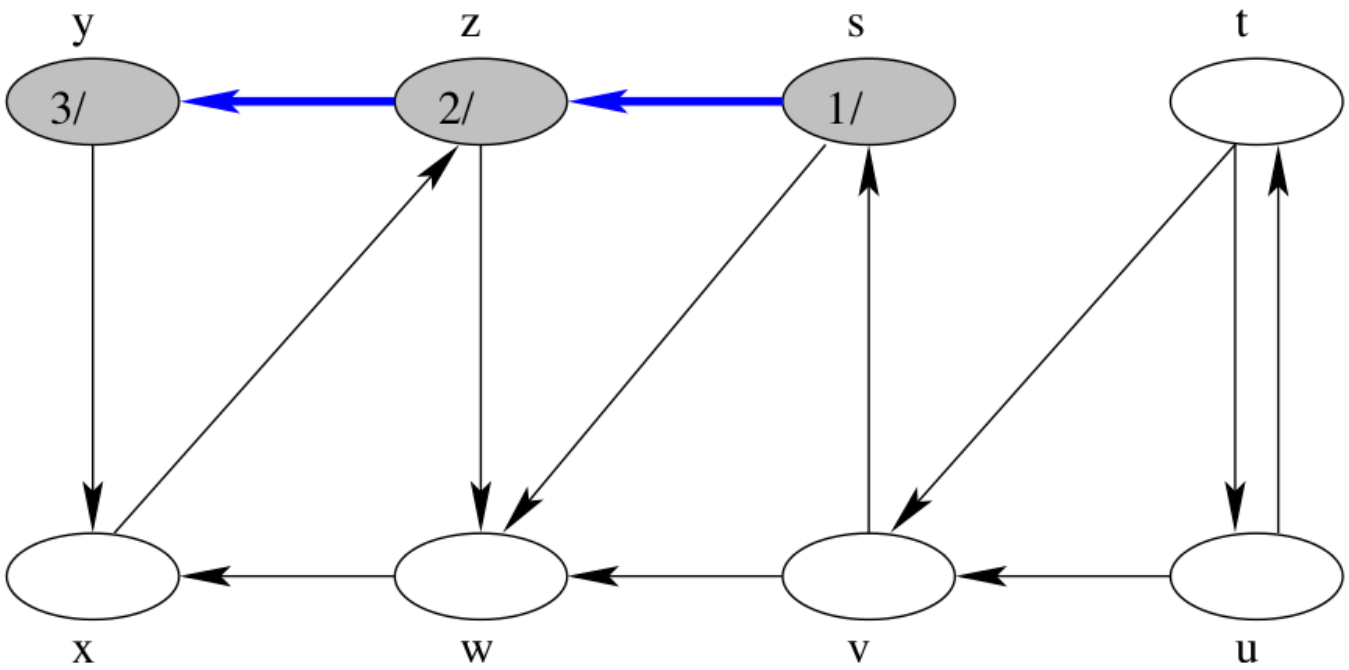
Algoritmo em profundidade - exemplo



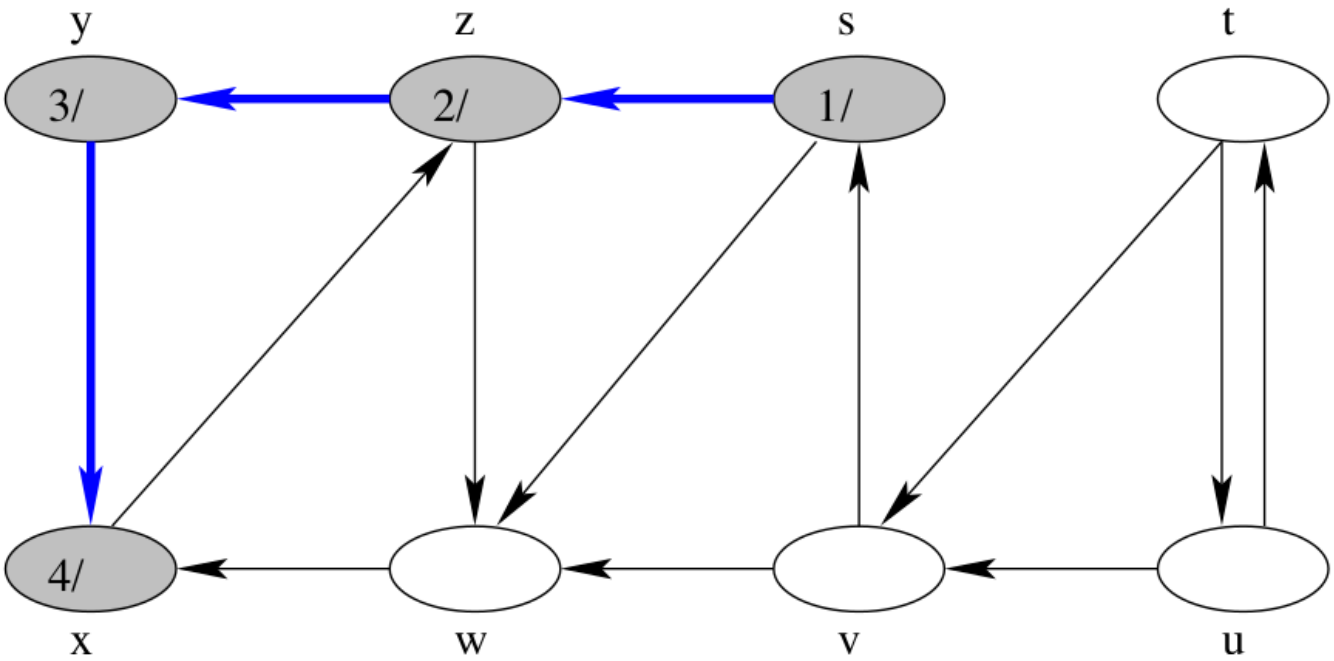
Algoritmo em profundidade - exemplo



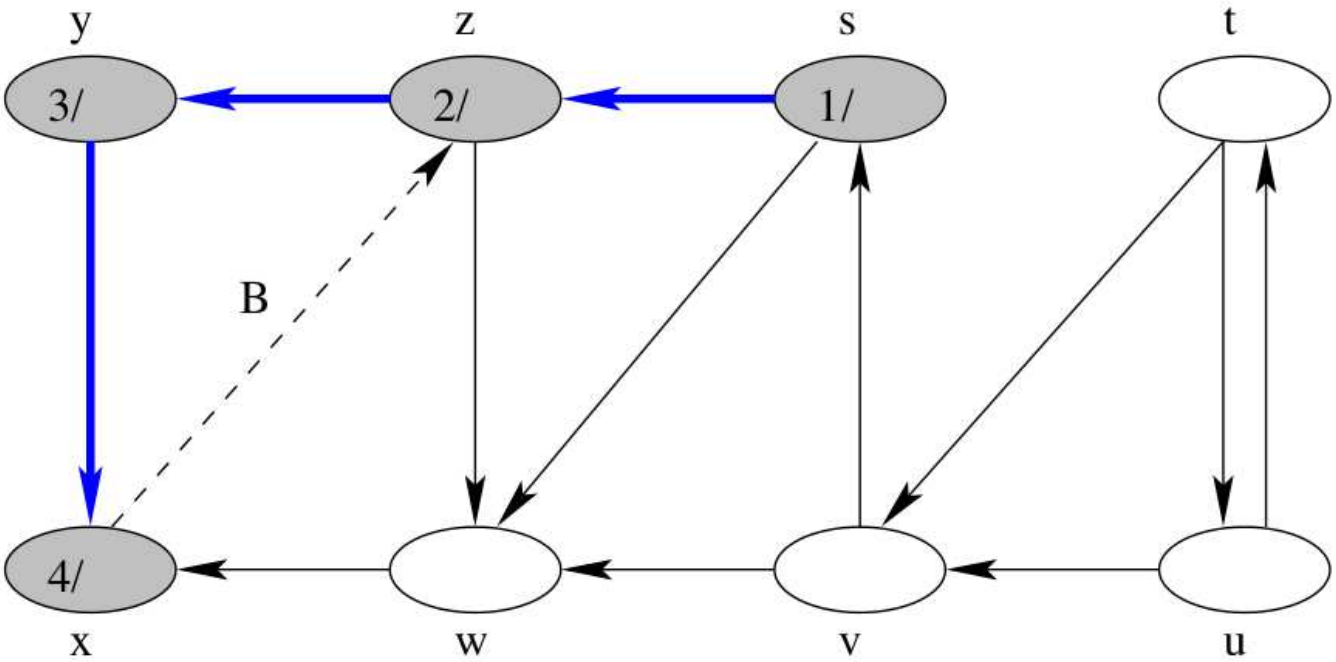
Algoritmo em profundidade - exemplo



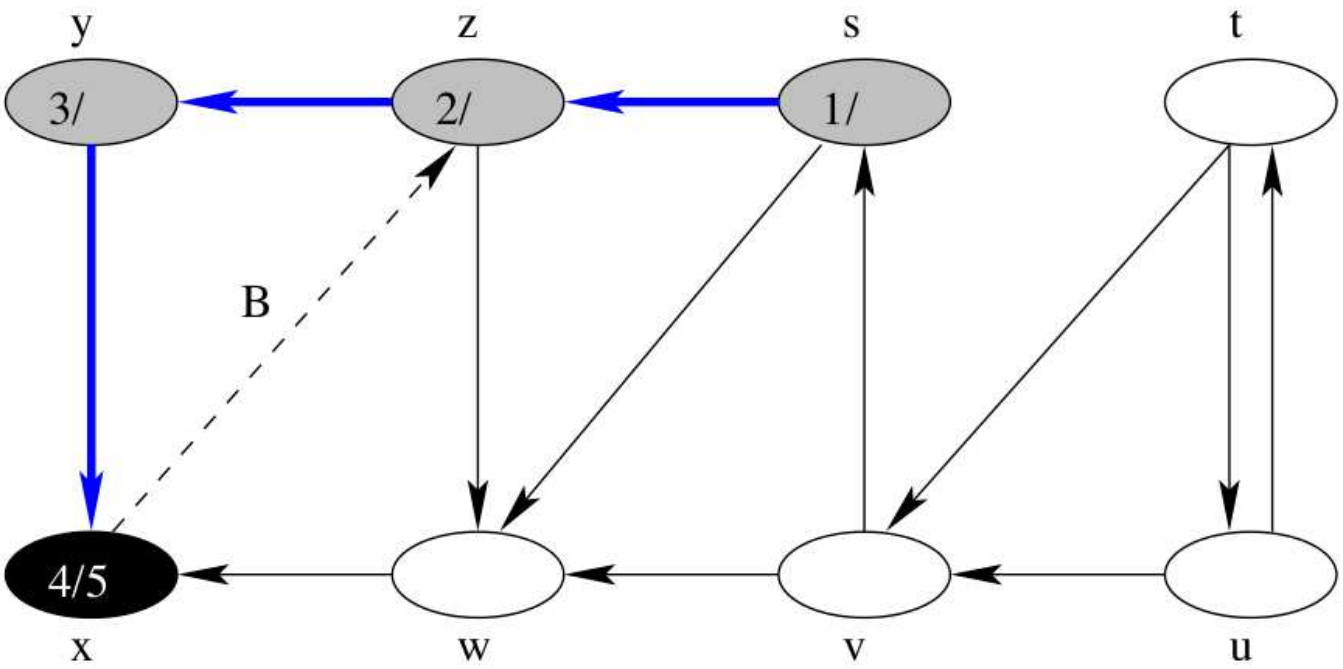
Algoritmo em profundidade - exemplo



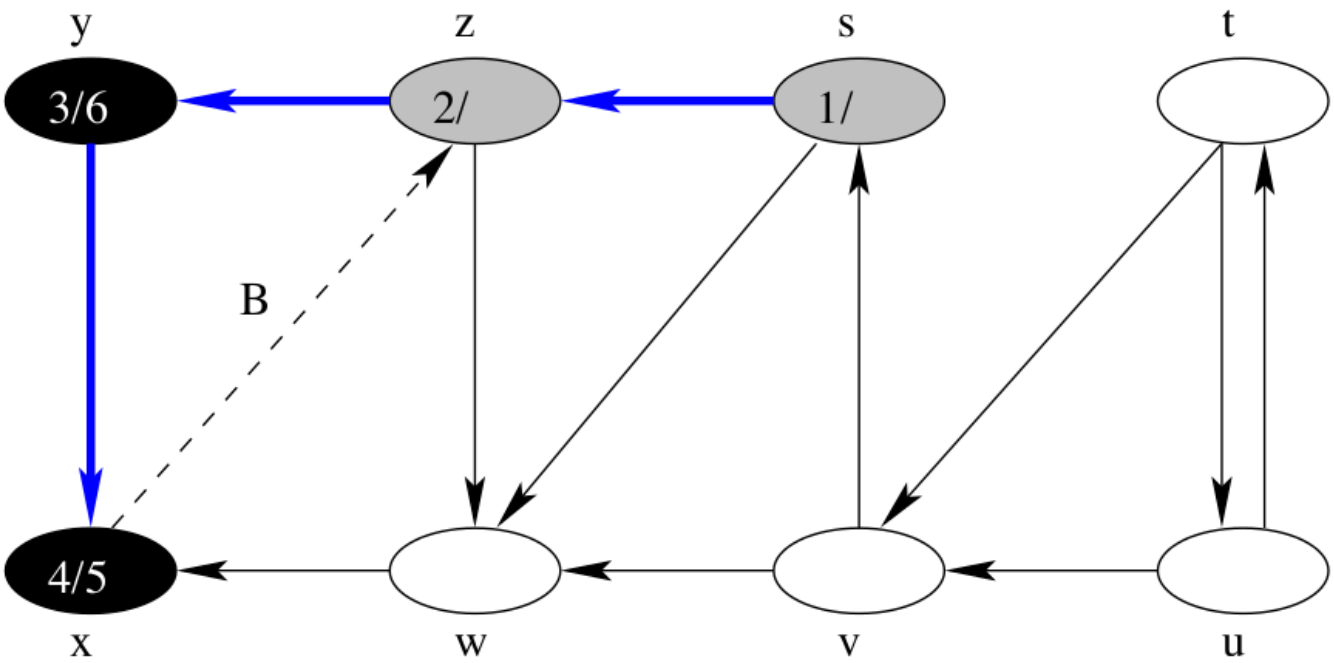
Algoritmo em profundidade - exemplo



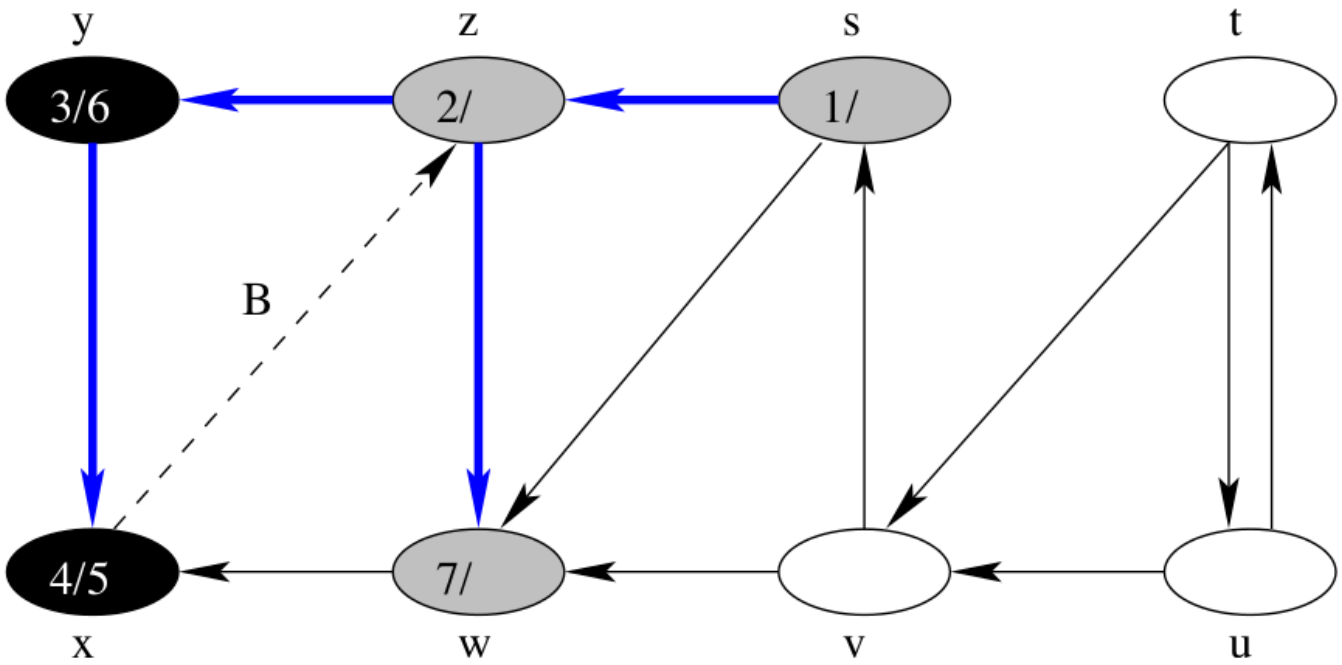
Algoritmo em profundidade - exemplo



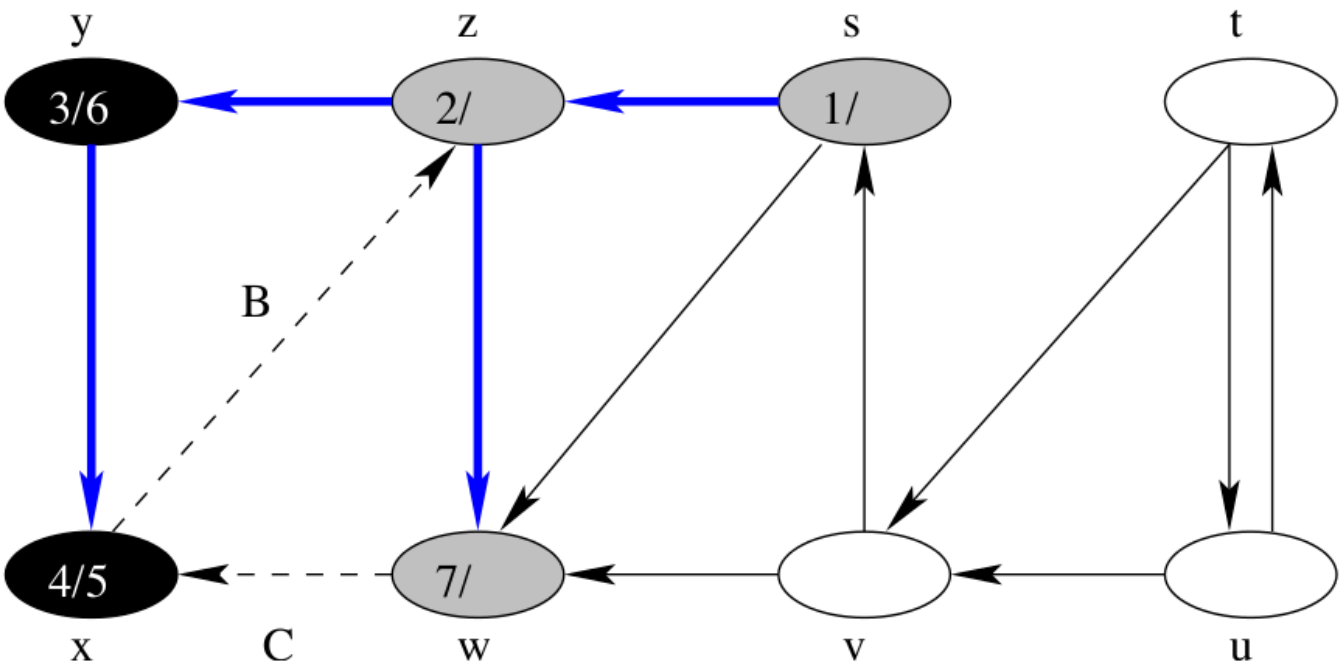
Algoritmo em profundidade - exemplo



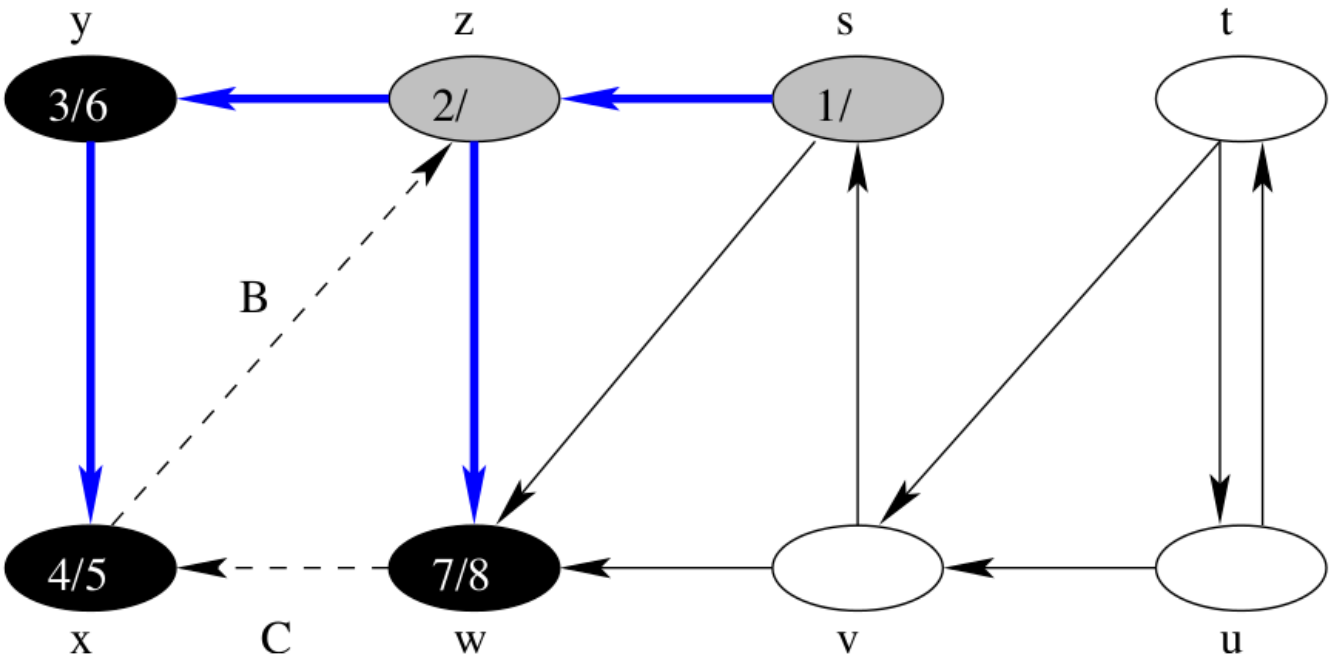
Algoritmo em profundidade - exemplo



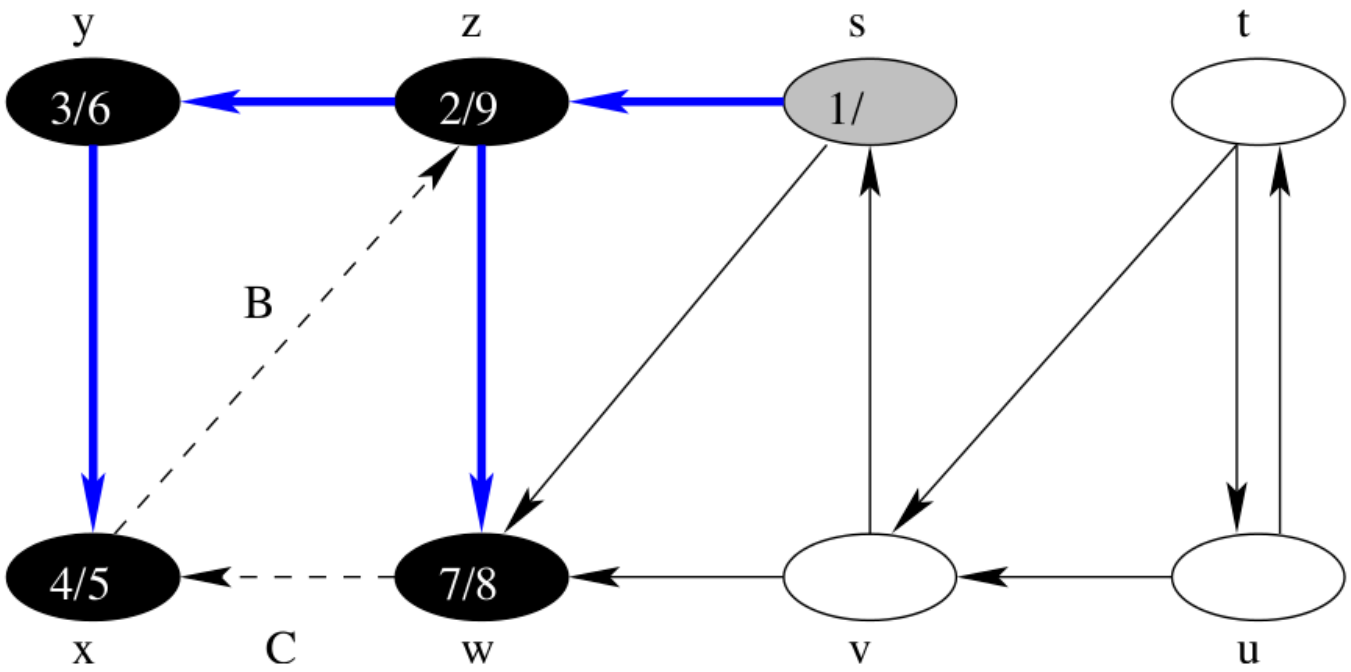
Algoritmo em profundidade - exemplo



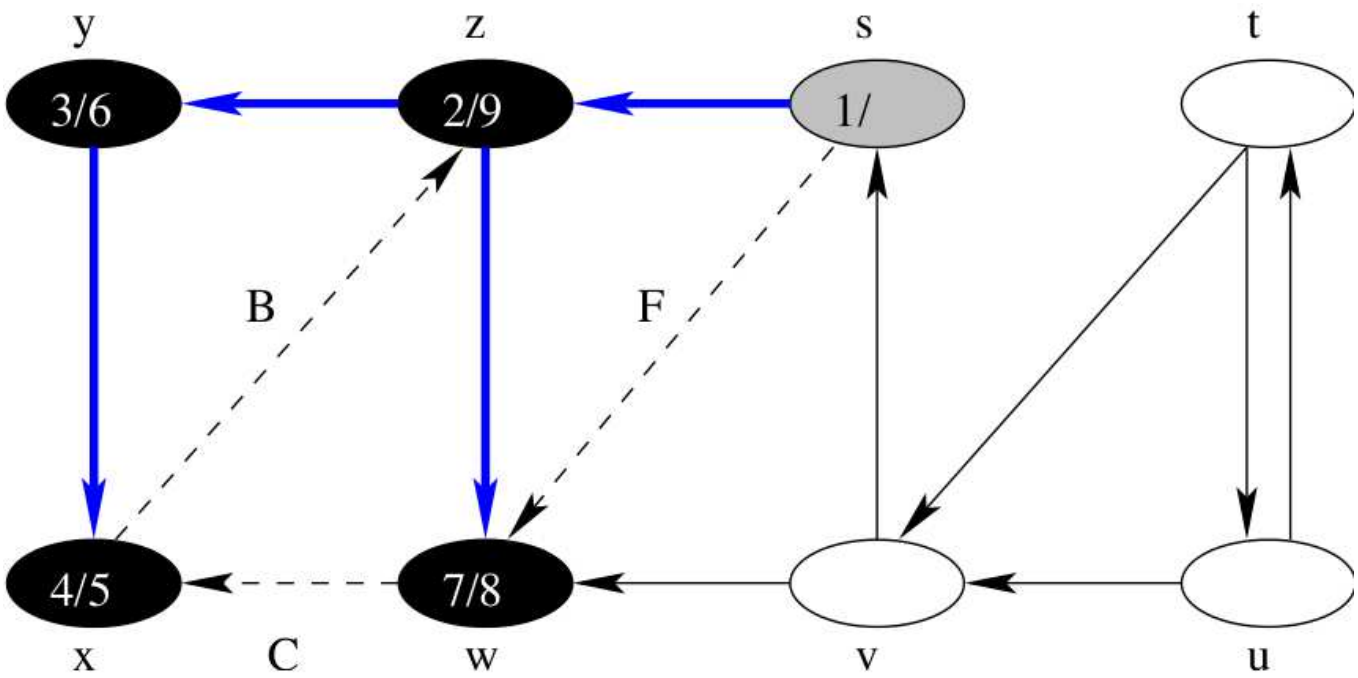
Algoritmo em profundidade - exemplo



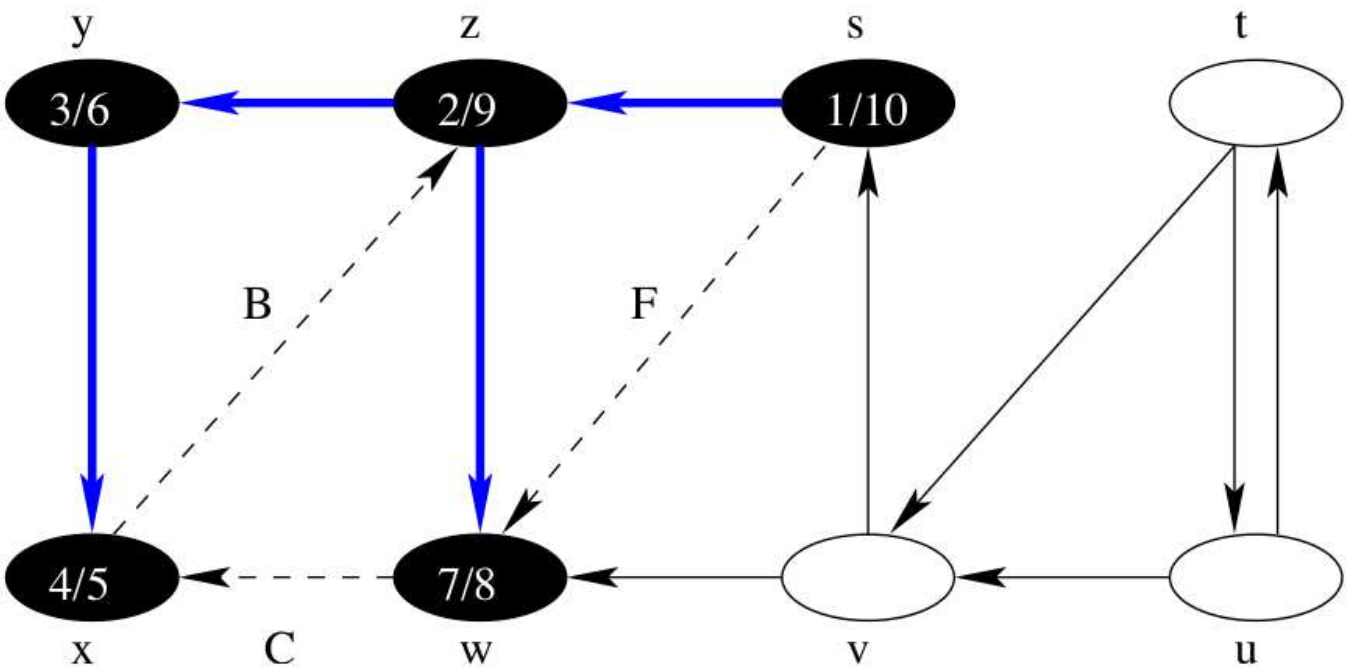
Algoritmo em profundidade - exemplo



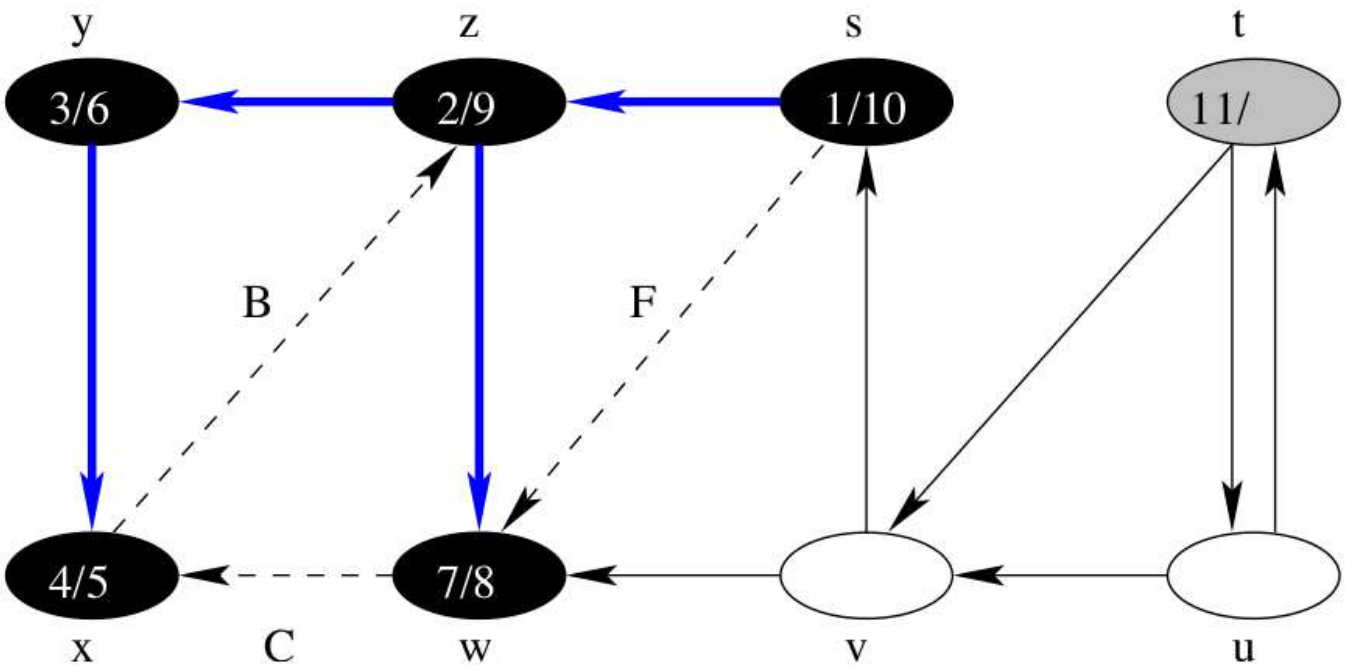
Algoritmo em profundidade - exemplo



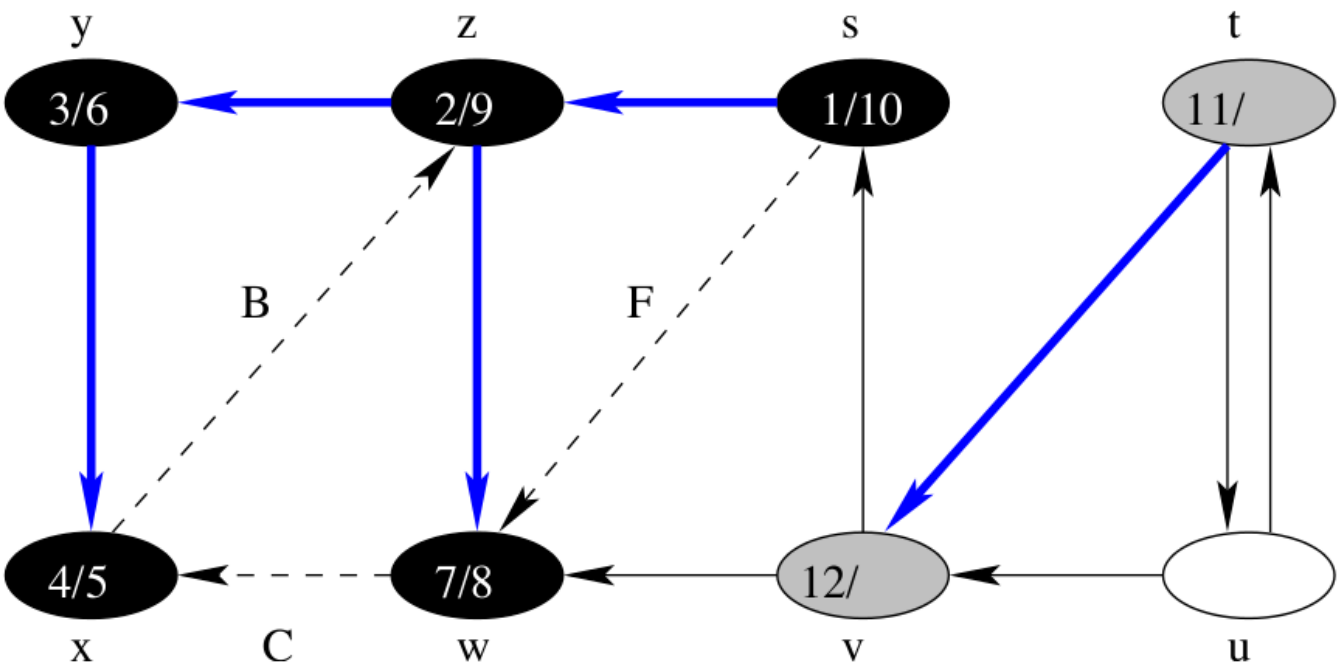
Algoritmo em profundidade - exemplo



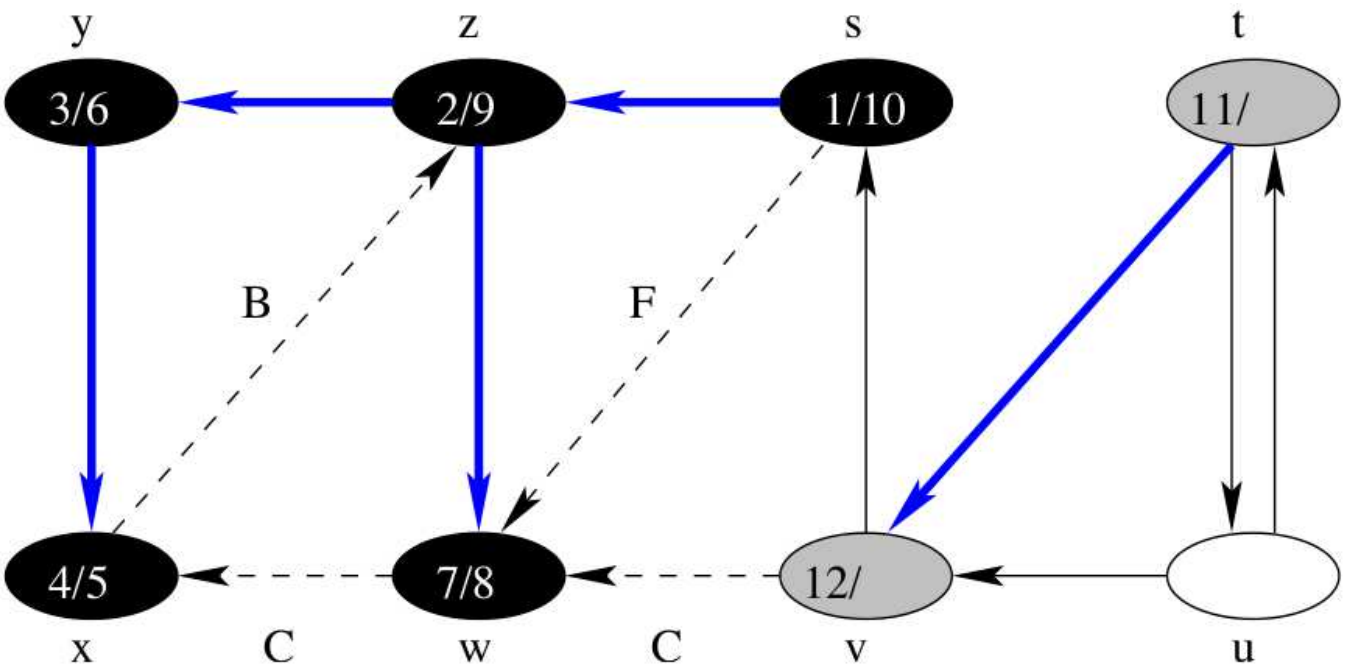
Algoritmo em profundidade - exemplo



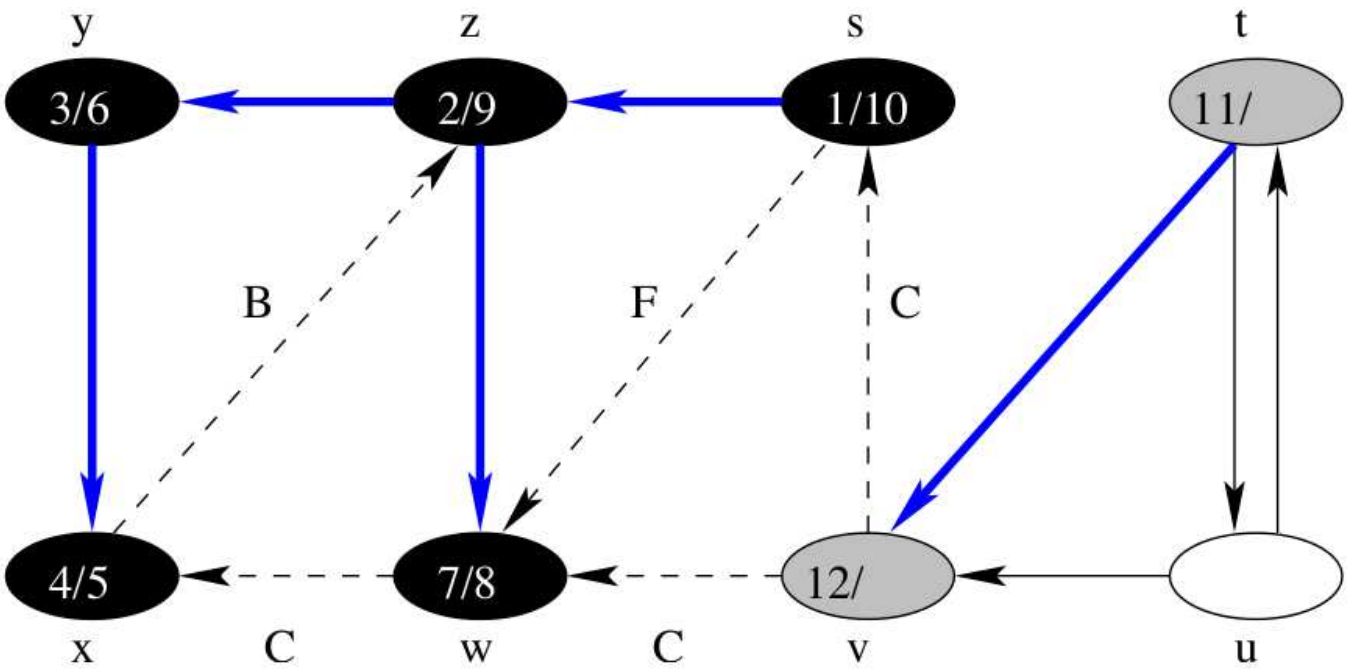
Algoritmo em profundidade - exemplo



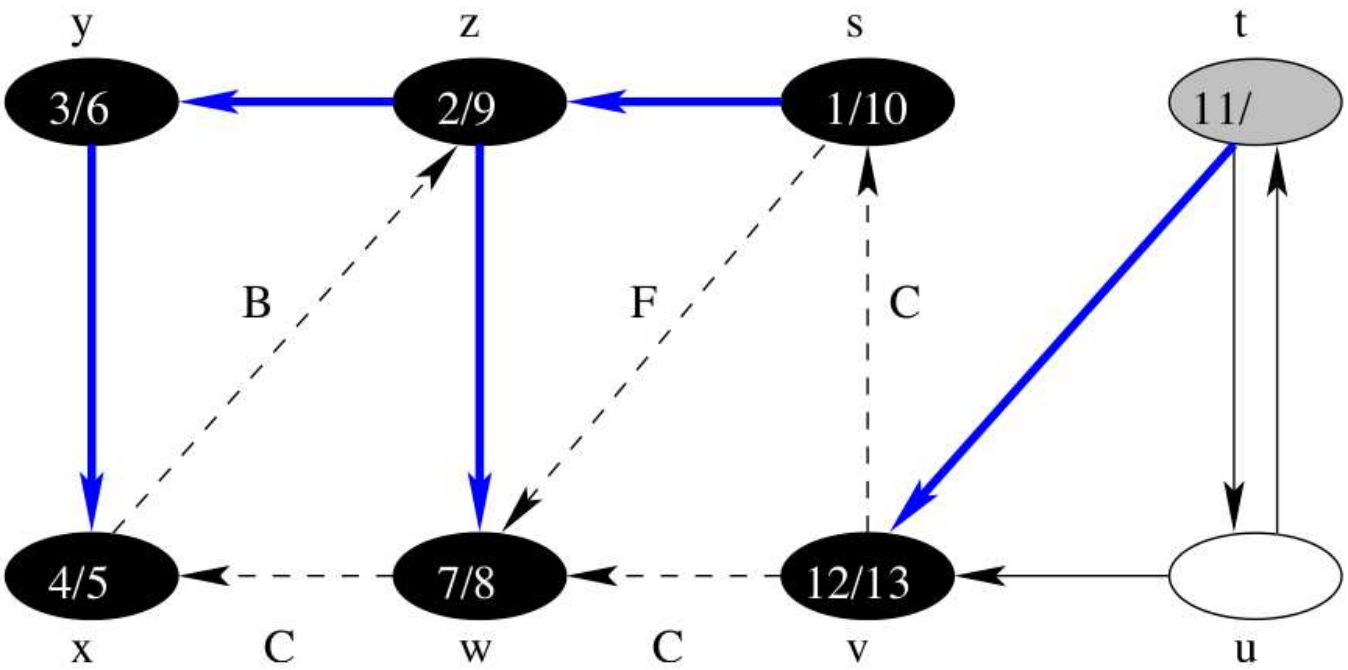
Algoritmo em profundidade - exemplo



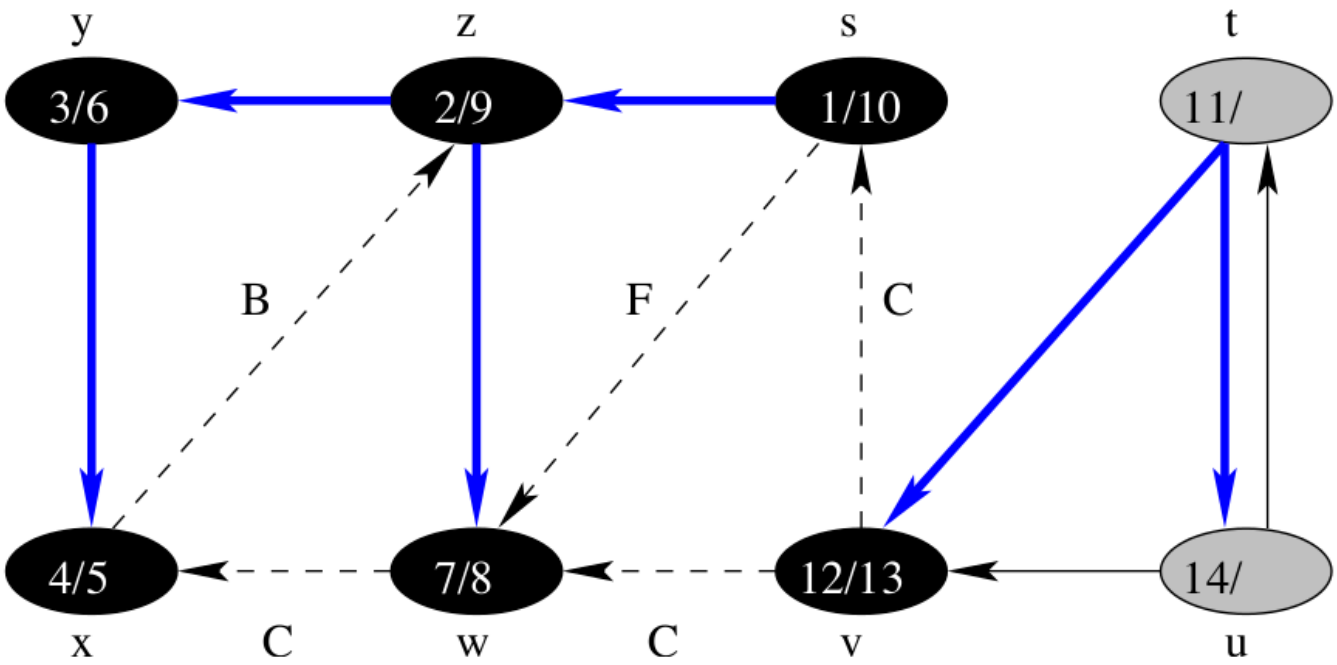
Algoritmo em profundidade - exemplo



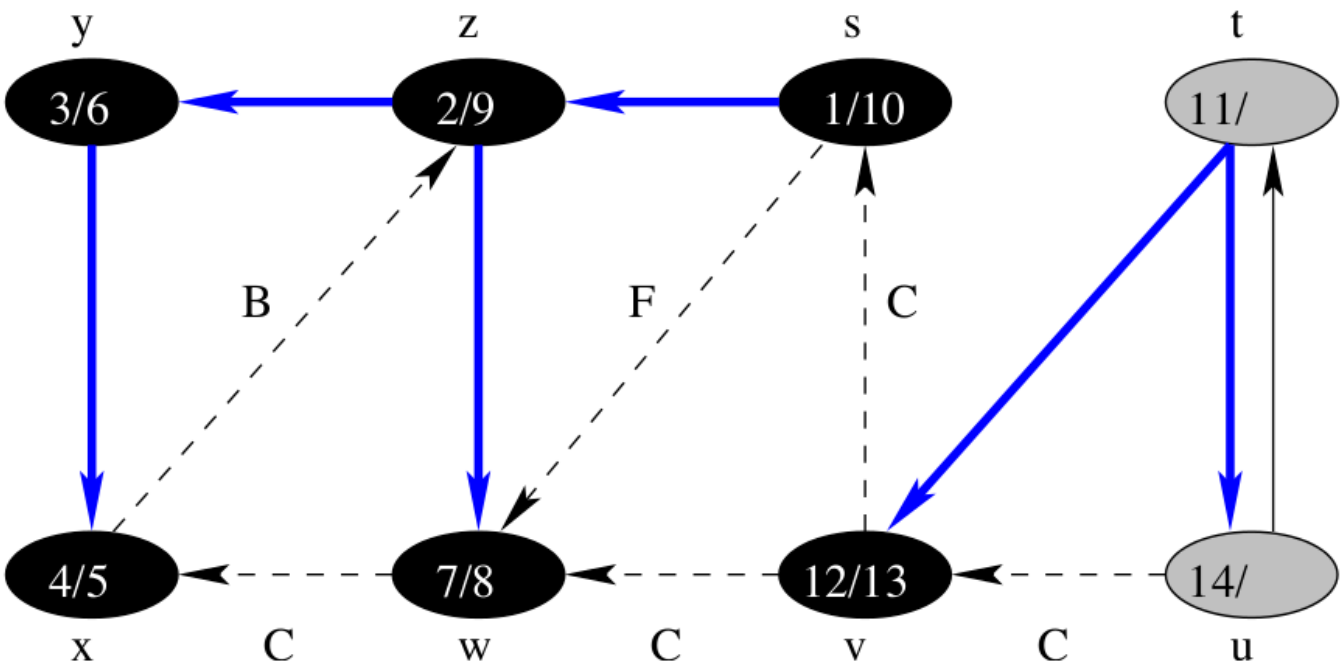
Algoritmo em profundidade - exemplo



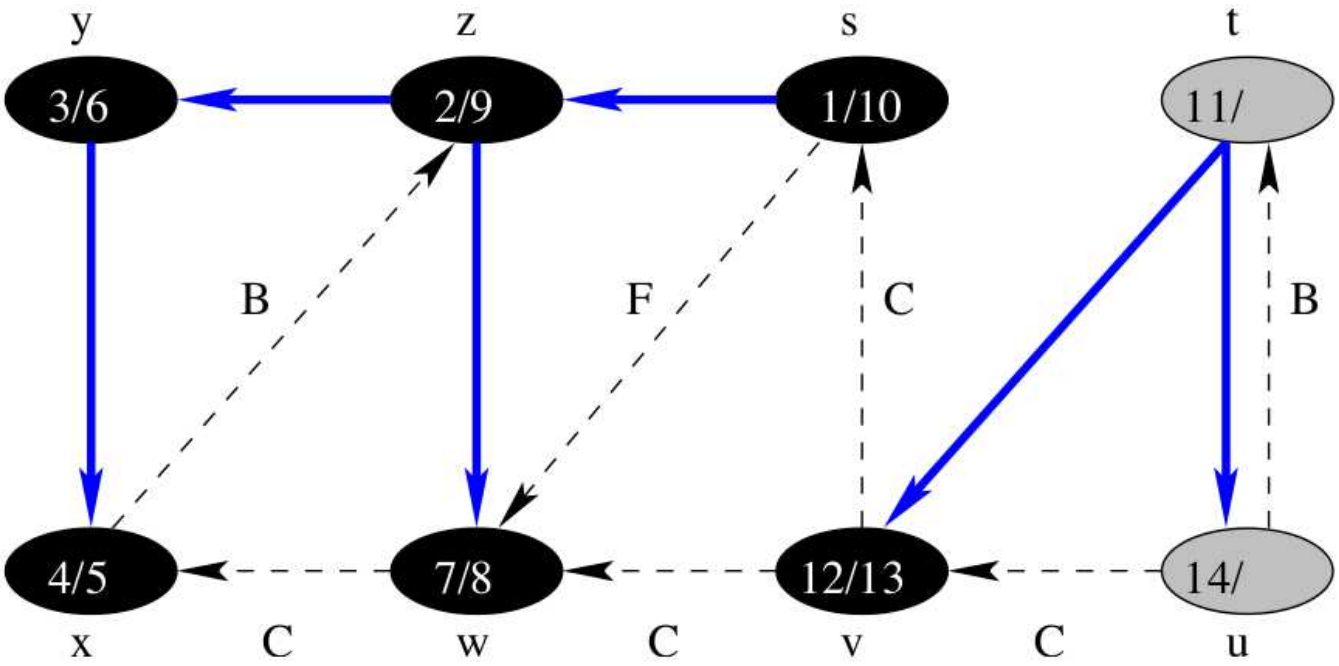
Algoritmo em profundidade - exemplo



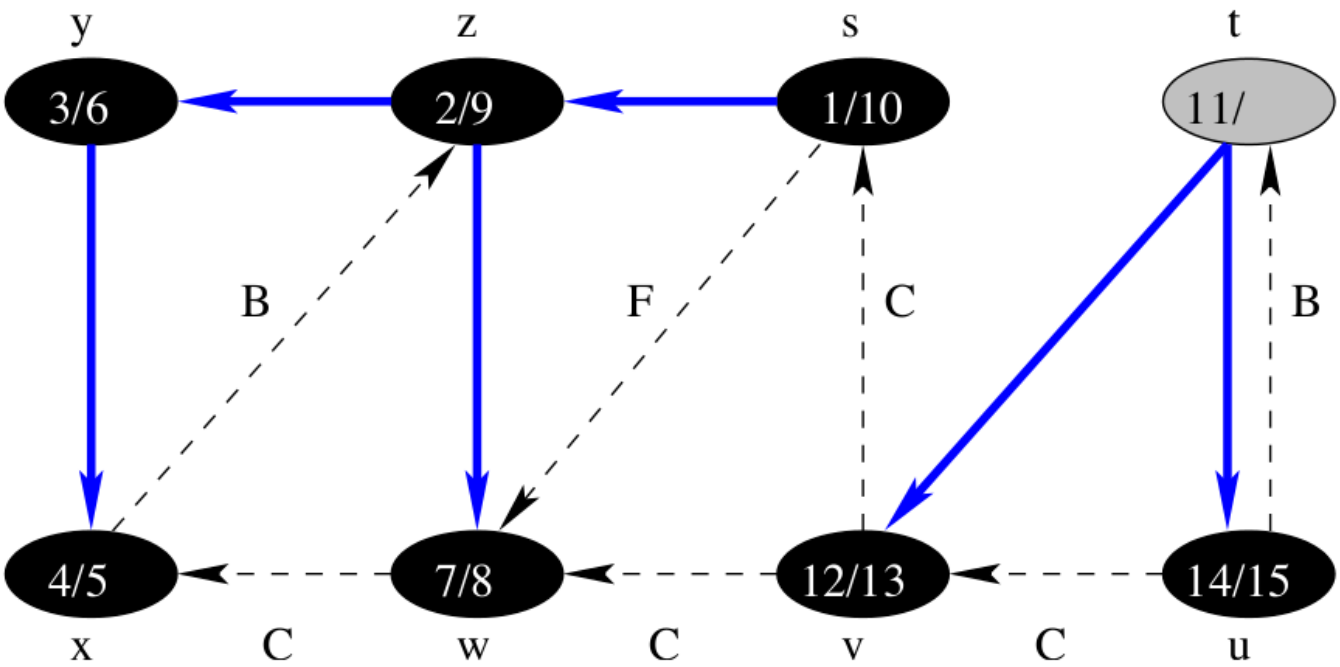
Algoritmo em profundidade - exemplo



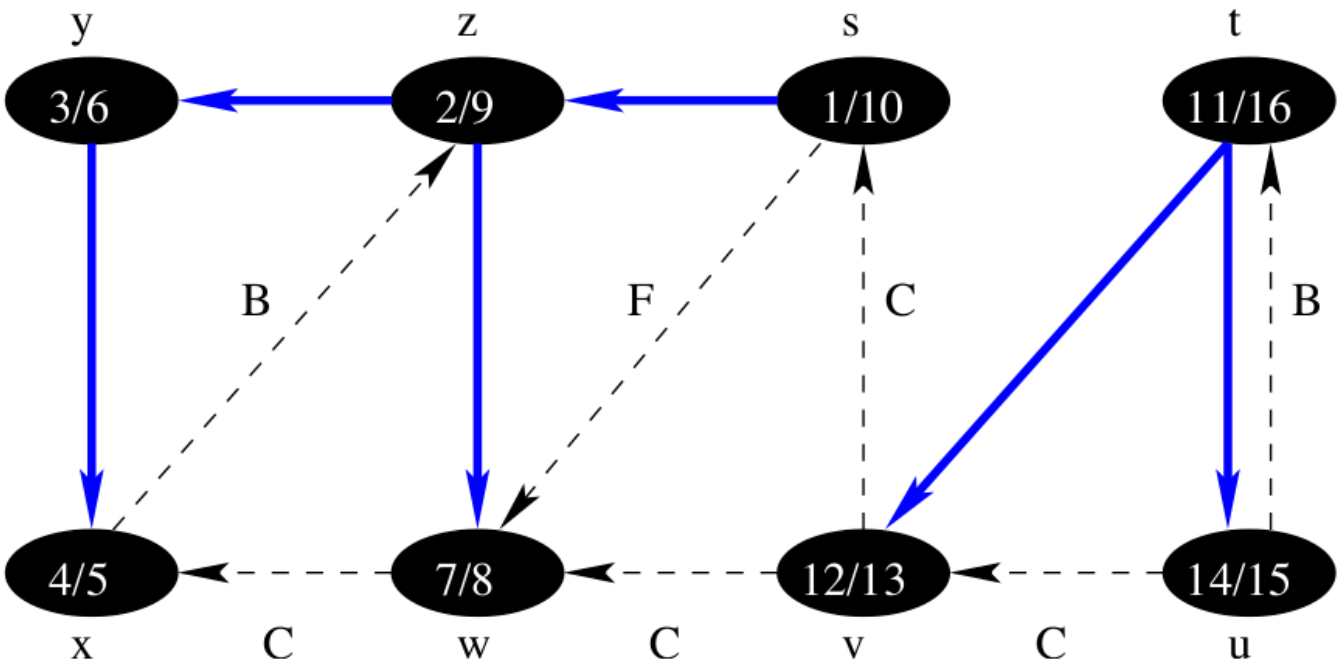
Algoritmo em profundidade - exemplo



Algoritmo em profundidade - exemplo



Algoritmo em profundidade - exemplo



Algoritmo de busca em profundidade

Rótulos versus cores

Para todo $x \in V[G]$ vale que $d[x] < f[x]$.
Além disso:

- x é branco antes do instante $d[x]$.
- x é cinza entre os instantes $d[x]$ e $f[x]$.
- x é preto após o instante $f[x]$.

Algoritmo busca em profundidade

Recebe um grafo G (na forma de listas de adjacências) e devolve:

(i) os instantes $d[v], f[v]$ para cada $v \in V$ e

(ii) uma Floresta de Busca em Profundidade.

```
DFS(G)
1  para cada u ∈ V[G] faça
2       cor[u] ← branco
3       π[u] ← NIL
4  tempo ← 0
5  para cada u ∈ V[G] faça
6       se cor[u] = branco
7          então DFS-VISIT(u)
```

```
DFS-VISIT(u)
1  cor[u] ← cinza
2  tempo ← tempo + 1
3  d[u] ← tempo
4  para cada v ∈ Adj[u] faça
5       se cor[v] = branco
6          então π[v] ← u
7                DFS-VISIT(v)
8  cor[u] ← preto
9  f[u] ← tempo ← tempo + 1
```

- DSF-visit: Constrói recursivamente uma Árvore de Busca em Profundidade com raiz u.

Algoritmo busca em profundidade - complexidade

DFS(G) - consumo de tempo

$O(V) + V$ chamadas a DFS-visit().

DFS-visit(u) - consumo de tempo

linhas 4-7: executado $|Adj[u]|$ vezes.

```
DFS(G)
1  para cada u ∈ V[G] faça
2       cor[u] ← branco
3       π[u] ← NIL
4  tempo ← 0
5  para cada u ∈ V[G] faça
6       se cor[u] = branco
7          então DFS-VISIT(u)
```

```
DFS-VISIT(u)
1  cor[u] ← cinza
2  tempo ← tempo + 1
3  d[u] ← tempo
4  para cada v ∈ Adj[u] faça
5       se cor[v] = branco
6          então π[v] ← u
7                DFS-VISIT(v)
8  cor[u] ← preto
9  f[u] ← tempo ← tempo + 1
```

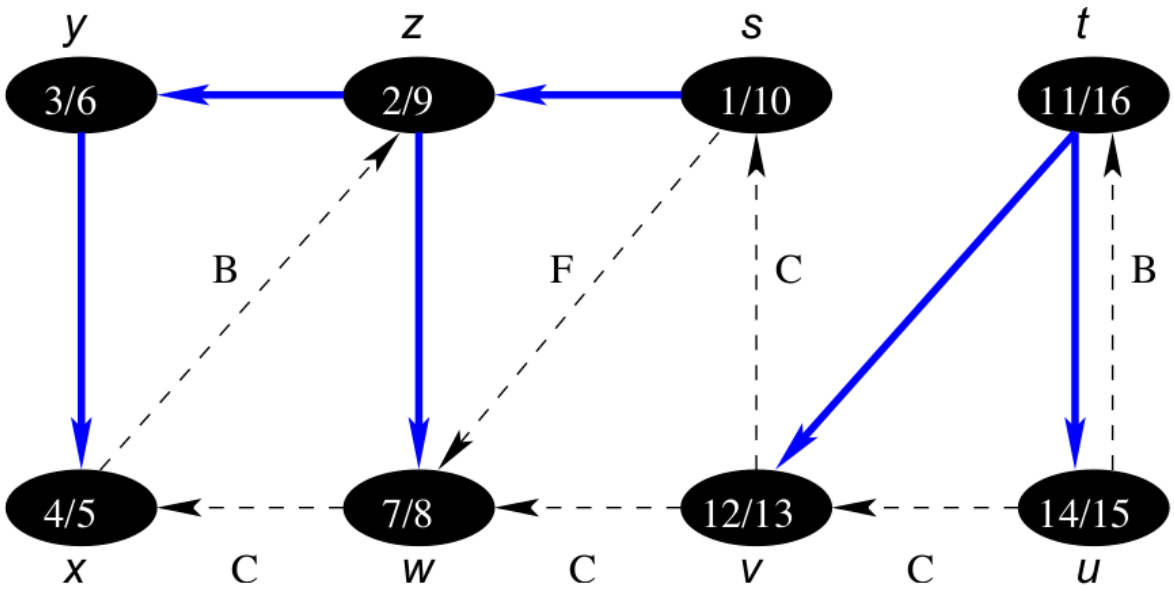
Algoritmo busca em profundidade - complexidade

- DFS-visit(v) é executado exatamente uma vez para cada $v \in V$.
- Em uma execução de DFS-visit(v), o laço das linhas 4-7 é executado $|Adj[u]|$ vezes.
  Assim, o custo total de todas as chamadas é:
  $\sum_{v \in V} |Adj(v)| = \Theta(E)$.

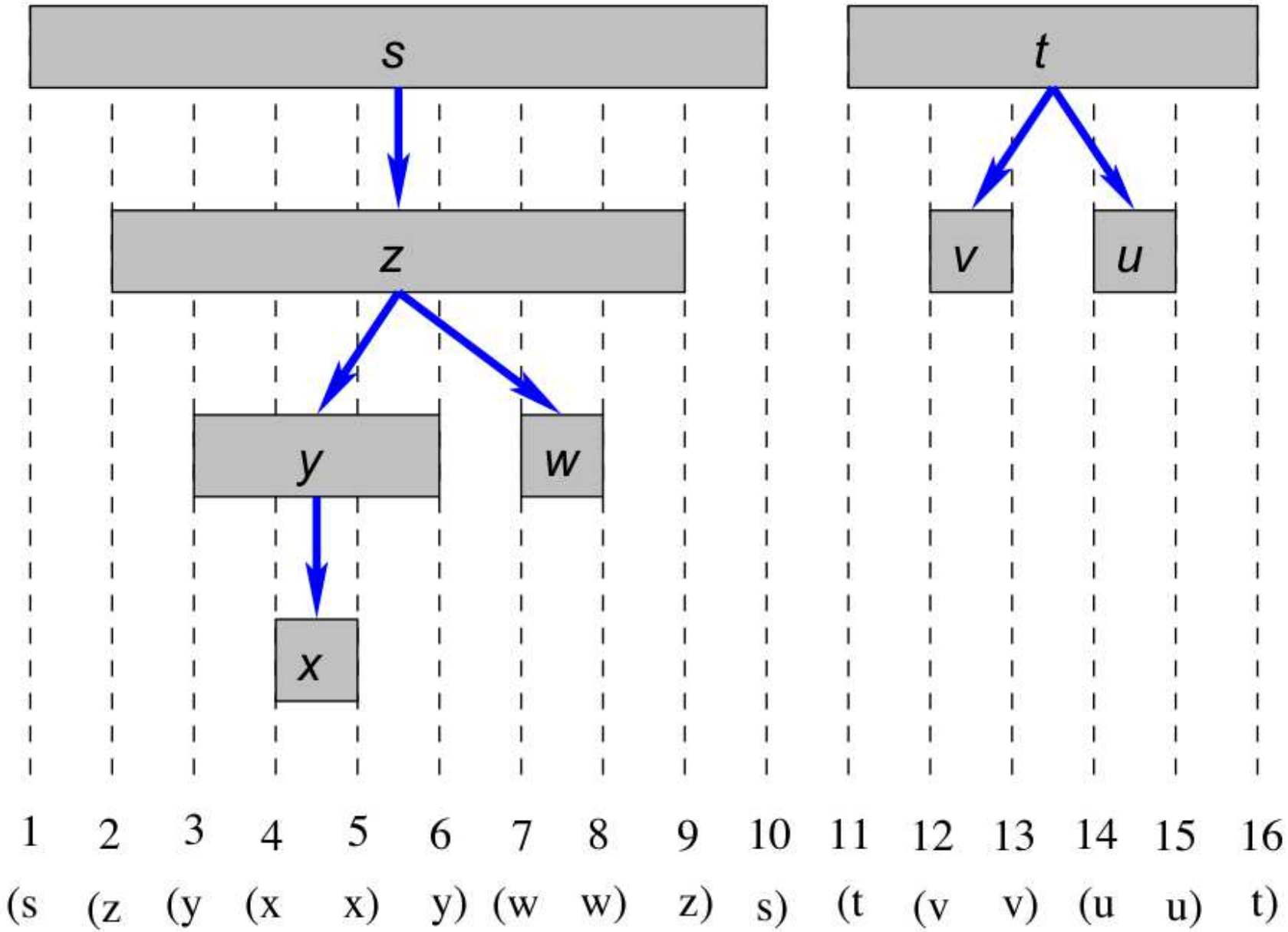
Conclusão: A complexidade de tempo de DFS é $O(V + E)$.

Estrutura de parênteses

- Os rótulos $d[x]$, $f[x]$ têm propriedades muito úteis para serem usadas em outros algoritmos.
- Eles refletem a ordem em que a **busca em profundidade** foi executada.
- Eles fornecem informação de como é a "cara" (estrutura) do grafo.



Estrutura de parênteses



Estrutura de parênteses

Teorema dos Parênteses

Em uma busca em profundidade sobre um grafo $G = (V, E)$, para quaisquer vértices u e v, ocorre exatamente uma das situações abaixo:

- $[d[u], f[u]]$ e $[d[v], f[v]]$ são disjuntos.
- $[d[u], f[u]]$ está contido em $[d[v], f[v]]$ e u é descendente de v na **Árvore de BP**.
- $[d[v], f[v]]$ está contido em $[d[u], f[u]]$ e v é descendente de u na **Árvore de BP**.

Classificação de arestas

Busca em profundidade pode ser usada para classificar arestas de um grafo $G = (V, E)$.
Ela classifica as arestas em quatro tipos:

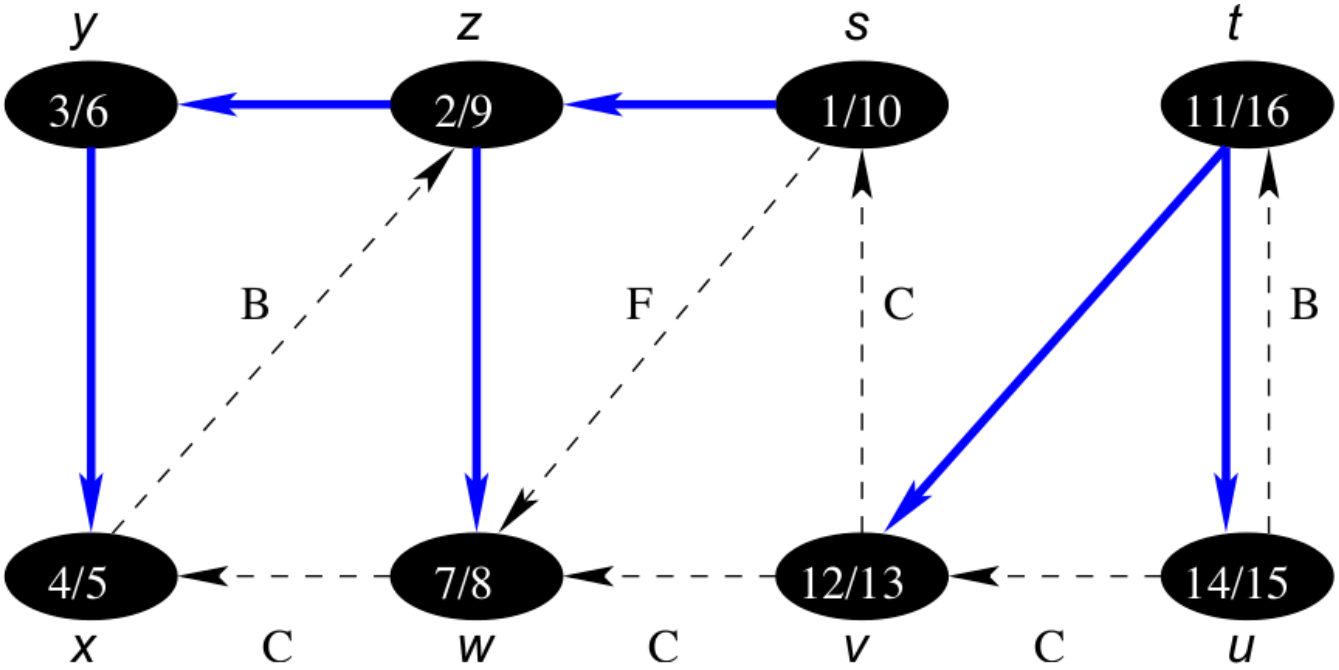
Arestas da árvore: arestas que pertencem á **Floresta de BP**.

Arestas de retorno: arestas (u, v) ligando um vértice u a um ancestral v na **Árvore de BP**.

Arestas de avanço: arestas (u, v) ligando um vértice u a um descendente próprio v na **Árvore de BP**.

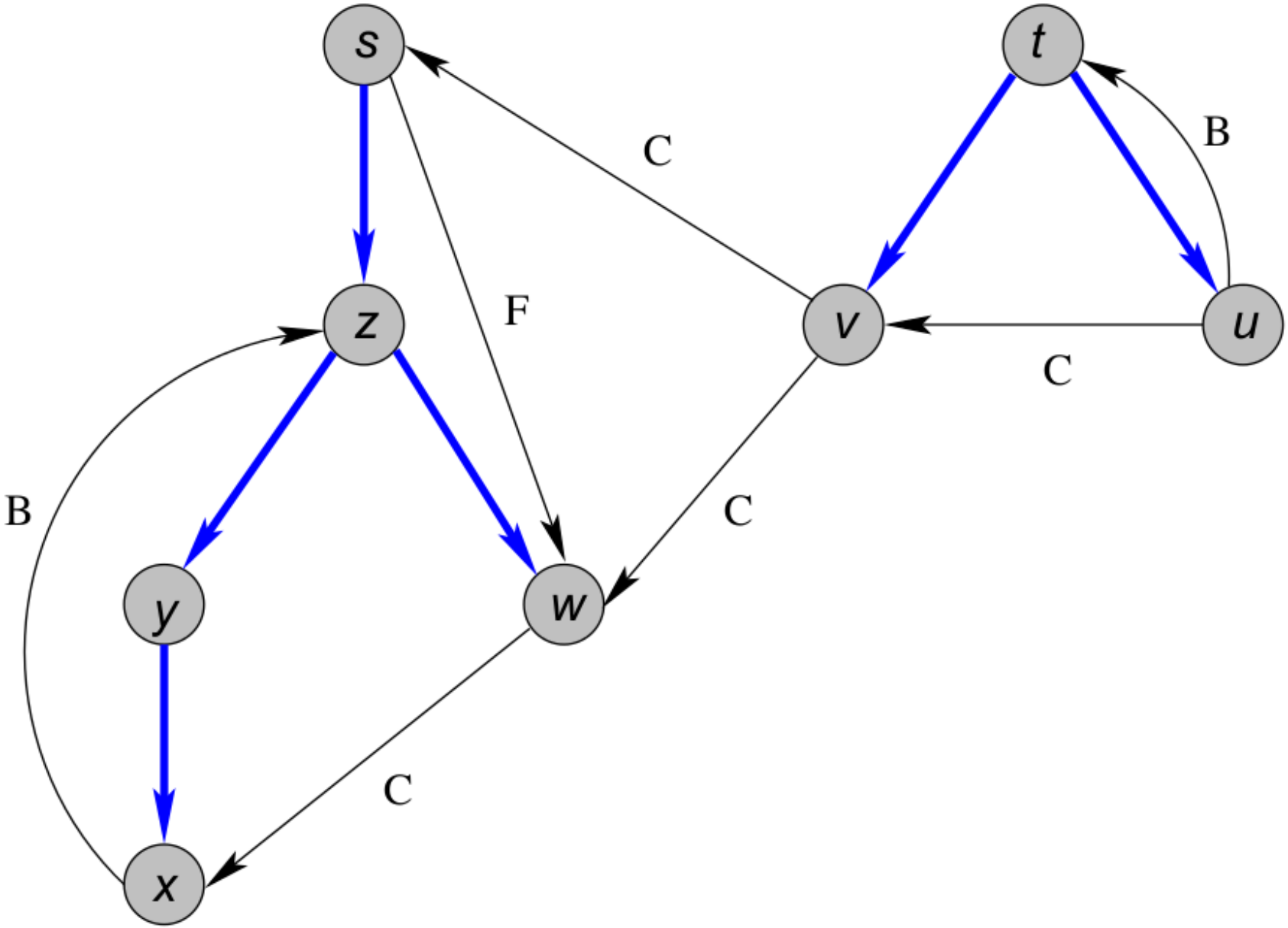
Arestas de cruzamento: todas as outras arestas.

Classificação de arestas



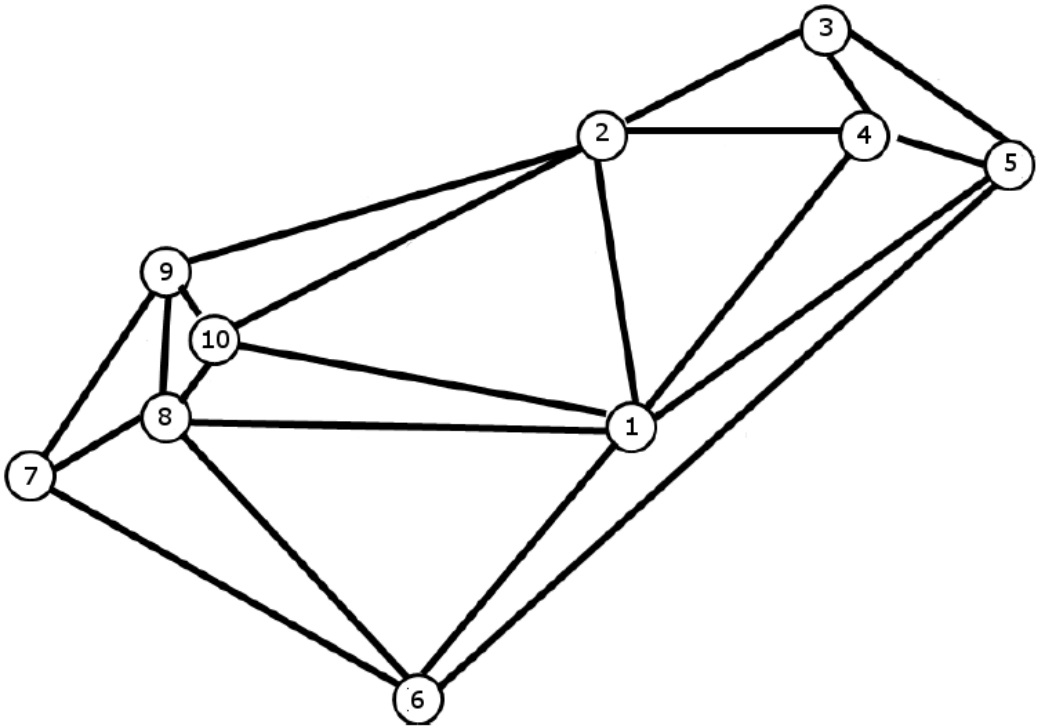
É fácil modificar o algoritmo DFS(G) para que ele também classifique as arestas de G. (**Exercício**)

Classificação de arestas



Exercício

Considere o grafo abaixo, em seguida faça:



- Execute a **busca em profundidade** sobre o grafo acima, considerando quaisquer um dos nós como origem.
- Implemente o algoritmo de busca em profundidade em linguagem C.
- Classifique as arestas do grafo.