Projeto e Análise de Algoritmos II (MC558)
Busca em grafos

Prof. Dr. Ruben Interian

Resumo

Resumo

Revisão do conteúdo

- **Algoritmos de busca em grafos** recebem um grafo G, percorrem o grafo visitando todos os vértices alcançáveis a partir de um vértice inicial s em G, e geram uma **árvore** (ou floresta) de busca.
- O algoritmo **BFS** (breadth-first search), ou busca em largura, começa visitando o vértice inicial, depois os seus vizinhos, depois os vizinhos dos vizinhos, e assim até percorrer todos os vértices alcançáveis.
- O algoritmo **BFS** gera a **árvore de busca** e, adicionalmente, as distâncias dos vértices ao vértice inicial s.

Objetivo

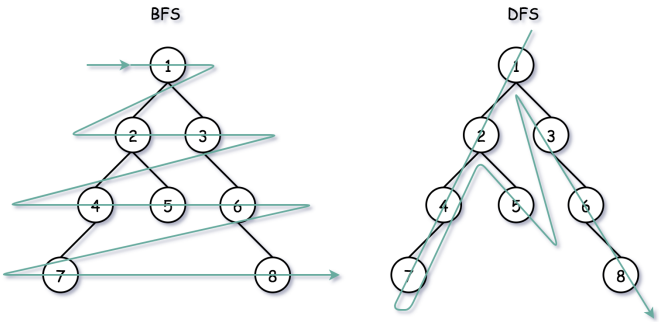
- **Busca em profundidade** (**DFS**, do inglês depth-first search): o segundo algoritmo de busca em grafos que iremos estudar.

Resumo

Busca em profundidade: ideia do algoritmo

Ideia do algoritmo DFS:

- Começar por um vértice inicial s;
- Depois, todos os vértices alcançáveis pelo 1^{ro} vizinho de s;
- Depois, todos os vértices alcançáveis pelo 2^{do} vizinho de s;
- . . . [e assim por diante]



Busca em profundidade

Aplicações:

- Aplicável tanto a grafos direcionados quanto não direcionados.
- Exemplos práticos: ordem de trabalhos ou tarefas, ordem das disciplinas quando há pré-requisitos entre elas (ordenação topológica).

Busca em profundidade

Ideias para implementar o algoritmo:

- O algoritmo recebe um grafo G, e gera uma **floresta de busca** (em profundidade).

Busca em profundidade

Ideias para implementar o algoritmo:

- O algoritmo recebe um grafo G, e gera uma **floresta de busca** (em profundidade). – Veja a diferença em relação ao BFS que vimos!

Busca em profundidade

Ideias para implementar o algoritmo:

- O algoritmo recebe um grafo G, e gera uma **floresta de busca** (em profundidade). – Veja a diferença em relação ao BFS que vimos!
- O grafo G é representado por listas de adjacências.

Busca em profundidade

Ideias para implementar o algoritmo:

- O algoritmo recebe um grafo G, e gera uma **floresta de busca** (em profundidade). – Veja a diferença em relação ao BFS que vimos!
- O grafo G é representado por listas de adjacências.
- Um vértice inicial s é escolhido.
- Para cada vizinho não visitado v de s, adicionamos a aresta (s, v) à árvore atual, e visitamos os vizinhos de v.

Busca em profundidade

Ideias para implementar o algoritmo:

- O algoritmo recebe um grafo G, e gera uma **floresta de busca** (em profundidade). – Veja a diferença em relação ao BFS que vimos!
- O grafo G é representado por listas de adjacências.
- Um vértice inicial s é escolhido.
- Para cada vizinho não visitado v de s, adicionamos a aresta (s, v) à árvore atual, e visitamos os vizinhos de v.
- Veja que o vértice v "vira" o vértice inicial - podemos usar a recursividade!

Busca em profundidade

Da mesma forma que o algoritmo BFS usa uma fila, podemos usar uma pilha para armazenar os vértices a serem visitados.

Busca em profundidade

Da mesma forma que o algoritmo BFS usa uma fila, podemos usar uma pilha para armazenar os vértices a serem visitados.

- Porém, o uso da recursividade gera um algoritmo mais **intuitivo**, "limpo", e mais fácil de entender.

Busca em profundidade

Cada árvore de busca contém só os vértices alcançáveis a partir de s.

- A **floresta de busca** é gerada repetindo o processo para os vértices ainda não visitados.

Busca em profundidade

Cada árvore de busca contém só os vértices alcançáveis a partir de s.

- A **floresta de busca** é gerada repetindo o processo para os vértices ainda não visitados.

Representando uma floresta:

- De novo, iremos utilizar um vetor de pais π: o pai do vértice x é $\pi[x]$.
- Um vértice v com $\pi[v] = \text{NIL}$ é a raiz de uma árvore de busca.
- As arestas da floresta são:

$$\{(\pi[v], v) \,:\, v \in V \text{ e } \pi[v] \neq \text{NIL}\}$$

.

Busca em profundidade

Cores dos vértices (semelhante ao **BFS**):

- Branco = "ainda não visitado" (inicialmente todos os vértices são brancos).
- Cinza = "visitado, ainda não finalizado" (ainda não analisei seus vizinhos).
- Preto = "visitado e finalizado" (vizinhos já analisados).

Particularidade:

- Os vértices cinza são os que têm chamadas recursivas ativas.

Busca em profundidade

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

Busca em profundidade

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

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

Busca em profundidade

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.

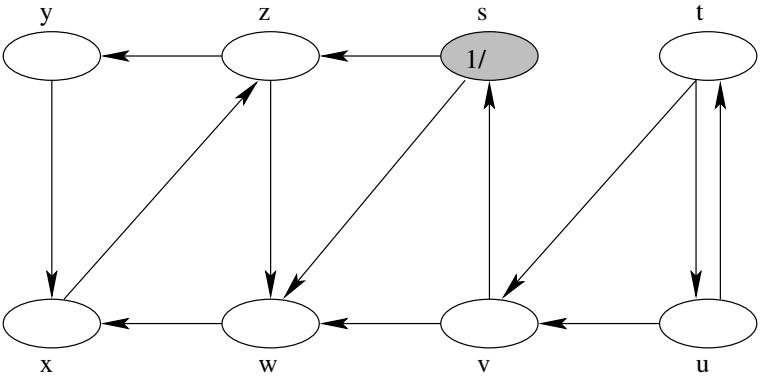
Busca em profundidade

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

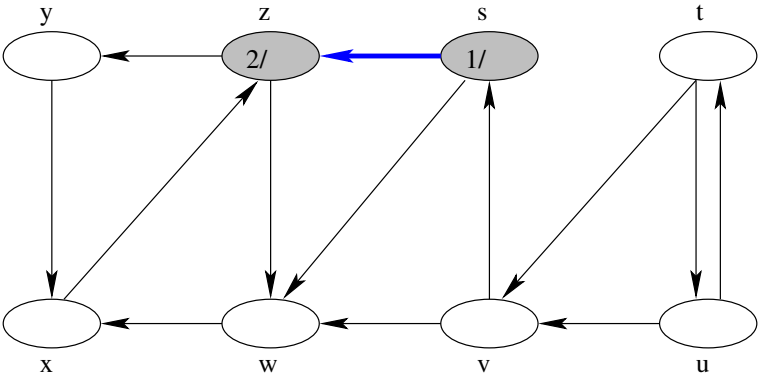
- $d[x]$: instante de descoberta de x. Neste instante x torna-se cinza.
- $f[x]$: instante de finalização de x. Neste instante x torna-se preto.

Os rótulos são inteiros entre 1 e $2|V|$, e refletem os instantes em que x muda de cor.

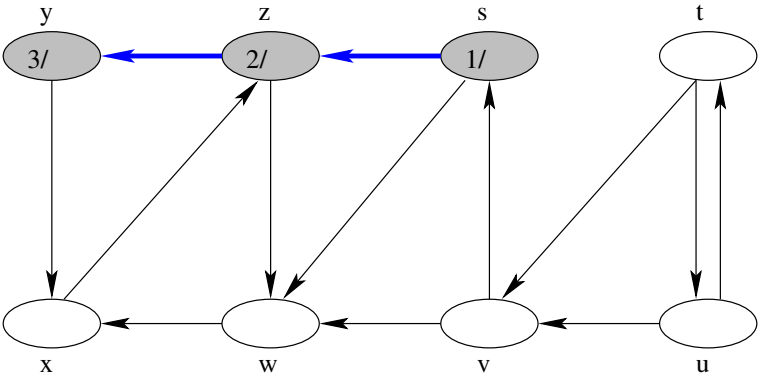
Exemplo de busca em profundidade



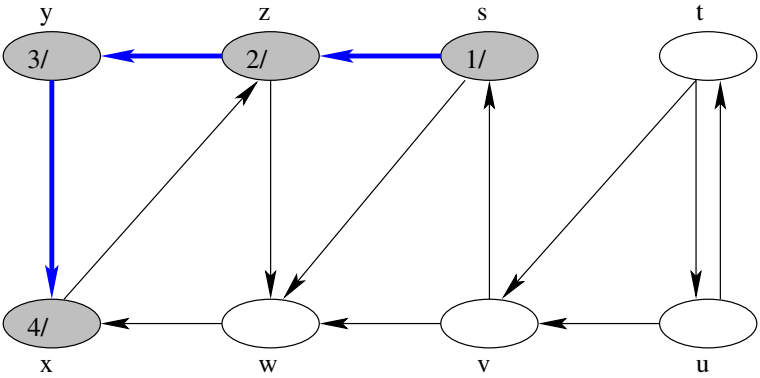
Exemplo de busca em profundidade



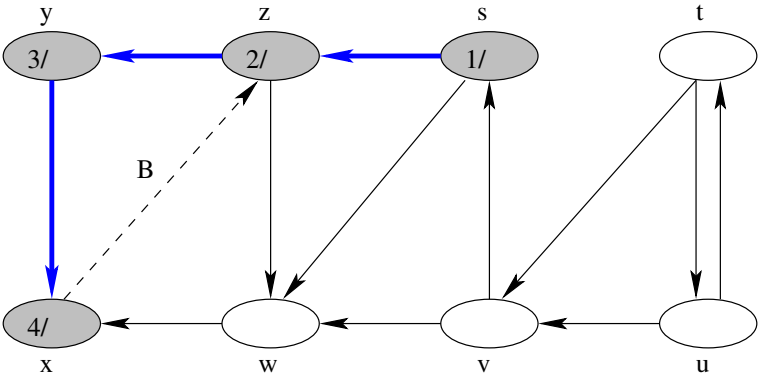
Exemplo de busca em profundidade



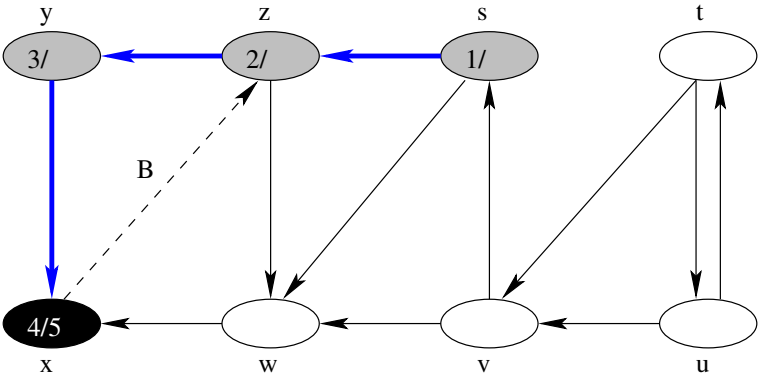
Exemplo de busca em profundidade



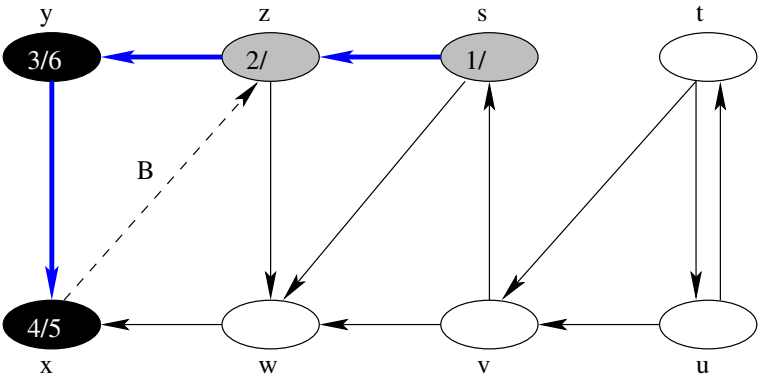
Exemplo de busca em profundidade



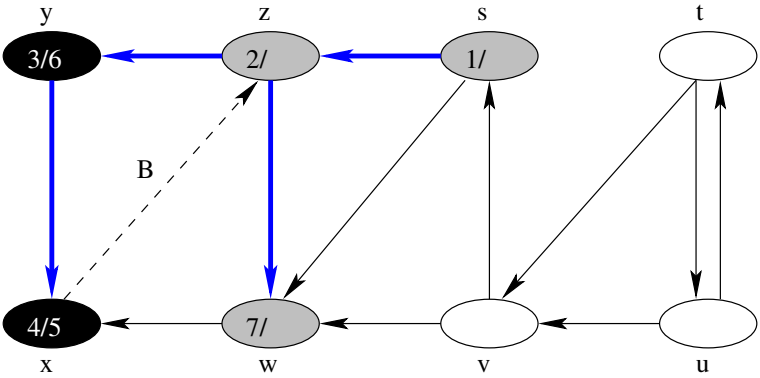
Exemplo de busca em profundidade



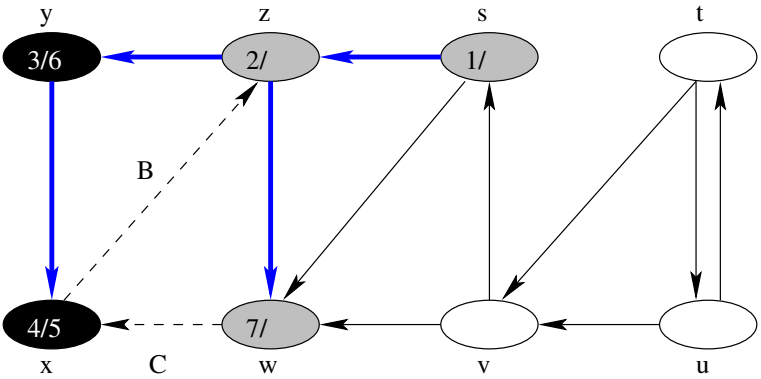
Exemplo de busca em profundidade



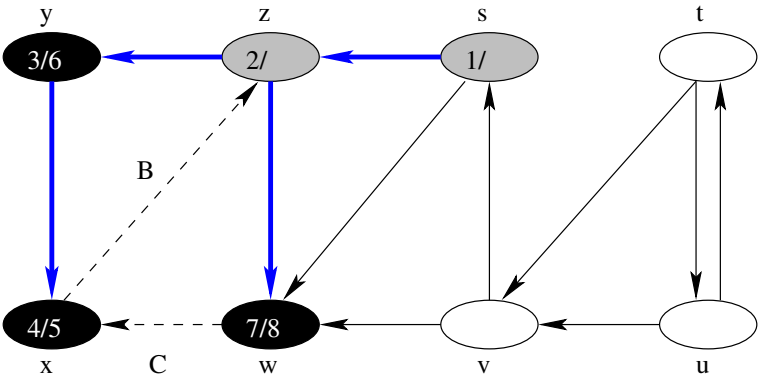
Exemplo de busca em profundidade



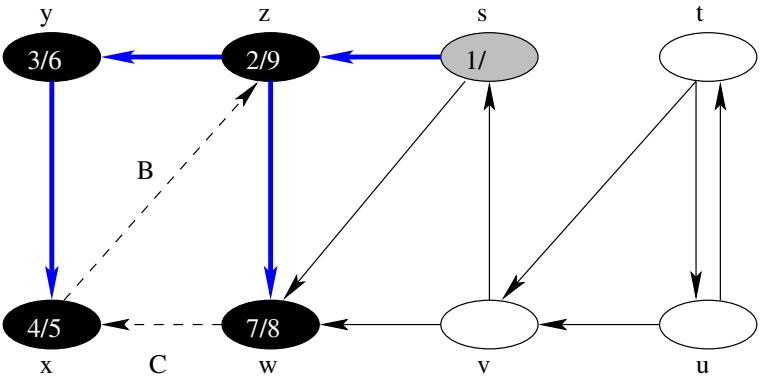
Exemplo de busca em profundidade



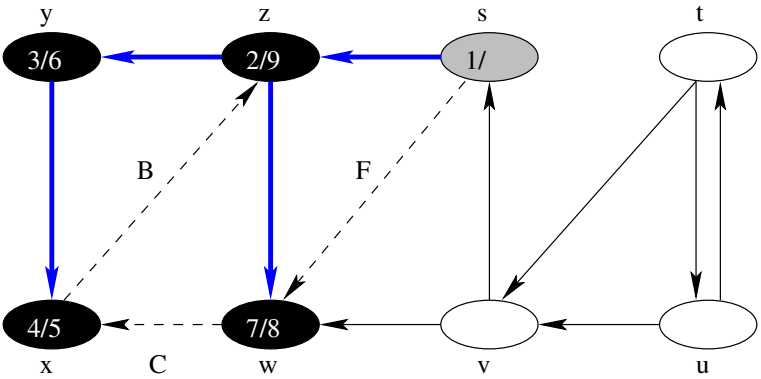
Exemplo de busca em profundidade



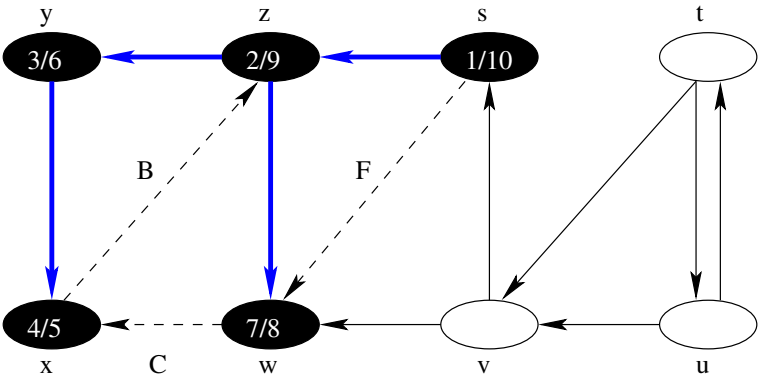
Exemplo de busca em profundidade



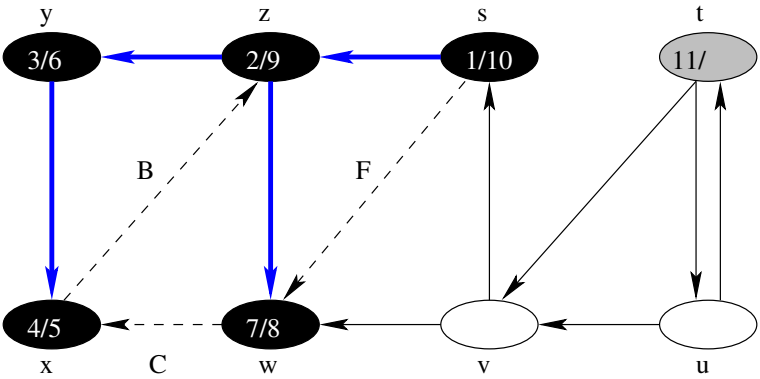
Exemplo de busca em profundidade



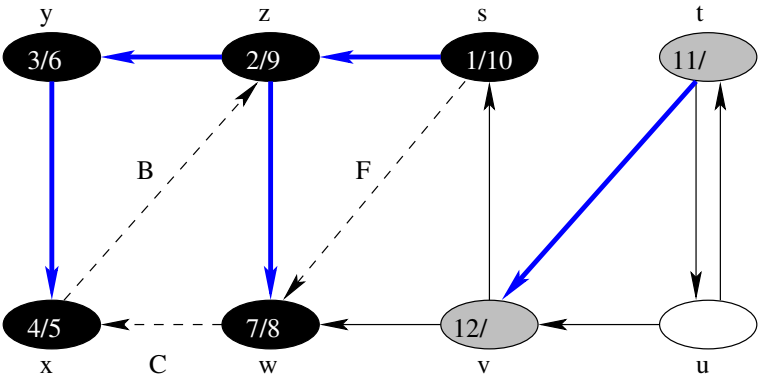
Exemplo de busca em profundidade



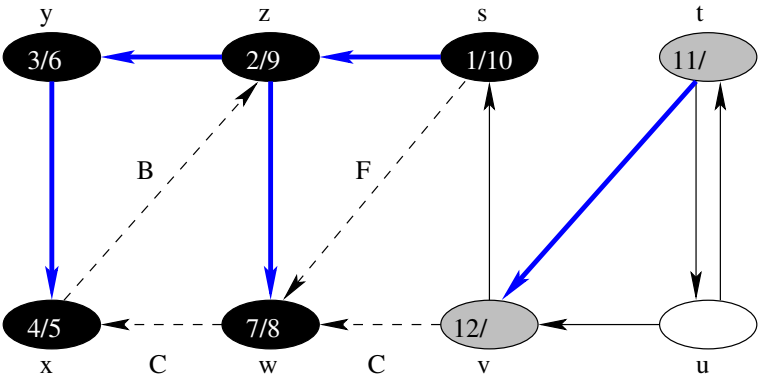
Exemplo de busca em profundidade



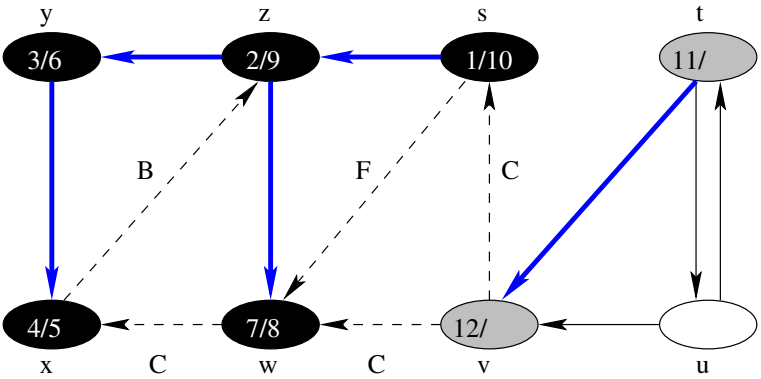
Exemplo de busca em profundidade



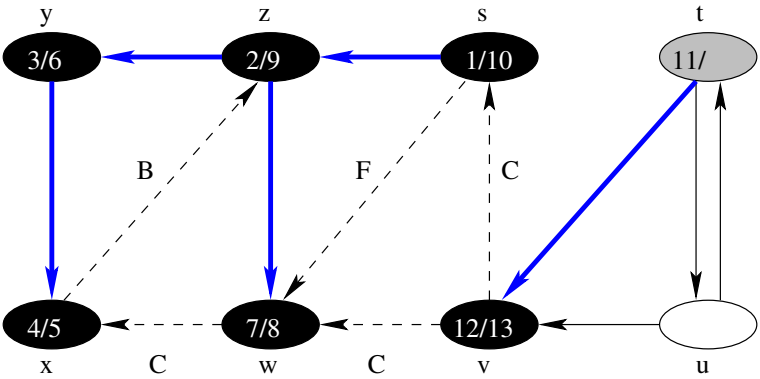
Exemplo de busca em profundidade



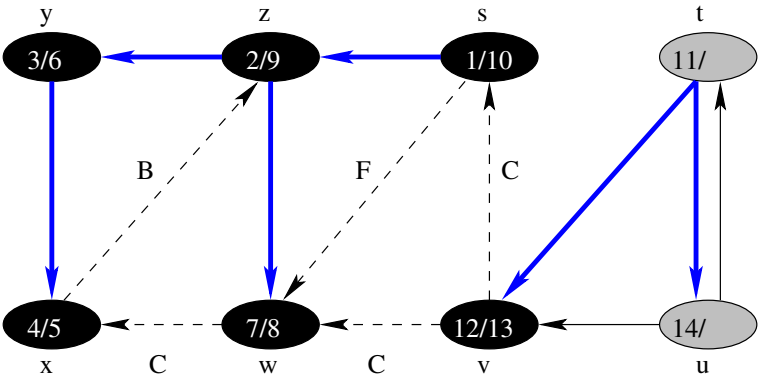
Exemplo de busca em profundidade



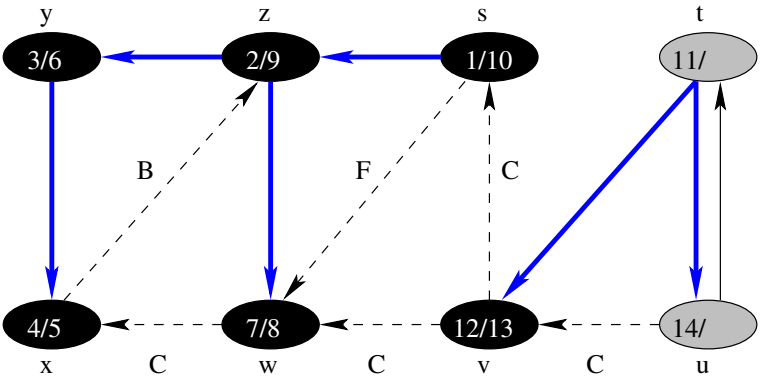
Exemplo de busca em profundidade



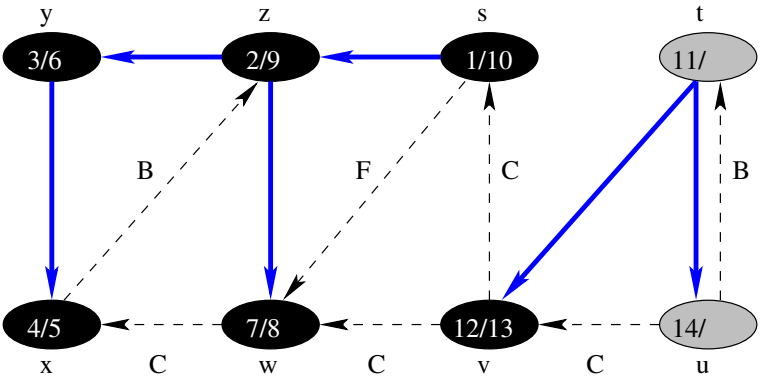
Exemplo de busca em profundidade



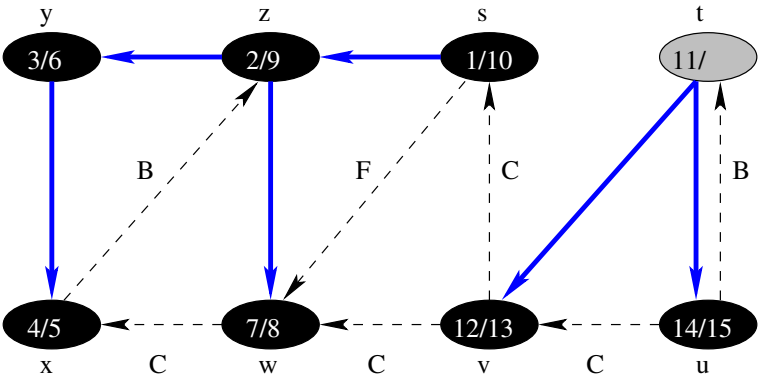
Exemplo de busca em profundidade



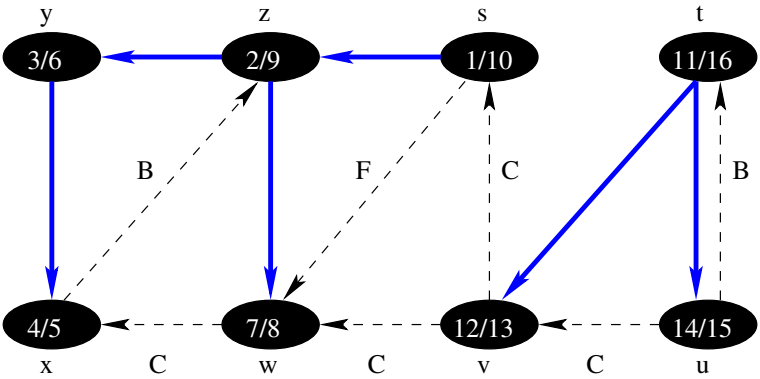
Exemplo de busca em profundidade



Exemplo de busca em profundidade



Exemplo de busca em profundidade



Busca em profundidade

Veja que para todo $x \in V$, $d[x] < f[x]$.

Busca em profundidade

Veja que para todo $x \in V$, $d[x] < f[x]$.

Durante a execução do algoritmo:

- 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 $f[x]$.

Busca em profundidade

O **algoritmo DFS**:

- Recebe um grafo G na forma de listas de adjacências.
- Devolve os instantes $d[v], f[v]$ para cada $v \in V$ e uma floresta de busca em profundidade.

Busca em profundidade: O algoritmo

DFS (G)

```
1: ▷ Inicialização:
2: para cada u ∈ V faça
3:     cor[u] ← branco
4:     π[u] ← NIL
5: tempo ← 0
6: ▷ Execução:
7: para cada u ∈ V faça
8:     se cor[u] = branco então
9:         DFS-Visit(G, u)
```

Busca em profundidade: O algoritmo

DFS (G)

```
1: ▷ Inicialização:
2: para cada u ∈ V faça
3:     cor[u] ← branco
4:     π[u] ← NIL
5: tempo ← 0
6: ▷ Execução:
7: para cada u ∈ V faça
8:     se cor[u] = branco então
9:         DFS-Visit(G, u)
```

DFS-Visit (G, u)

```
1: ▷ Cria árvore de busca com origem u:
2: cor[u] ← cinza
3: d[u] ← ++tempo
4: para cada v ∈ Adj[u] faça
5:     se cor[v] = branco então
6:         π[v] ← u
7:         DFS-Visit (G, v)
8: cor[u] ← preto
9: f[u] ← ++tempo
```

Busca em profundidade: complexidade de tempo

Complexidade de tempo do DFS:

- A inicialização consome tempo $\Theta(V)$.

Busca em profundidade: complexidade de tempo

Complexidade de tempo do DFS:

- A inicialização consome tempo $\Theta(V)$.
- **Ao todo**, fazemos $|V|$ chamadas a **DFS-Visit**.

Busca em profundidade: complexidade de tempo

Complexidade de tempo do DFS:

- A inicialização consome tempo $\Theta(V)$.
- **Ao todo**, fazemos $|V|$ chamadas a **DFS-Visit**.
- Qual é o tempo de execução de todas as chamadas a **DFS-Visit**?
  - Cada chamada percorre a lista de adjacências de um vértice u.
  - O laço da linha 4 é executado $|Adj(u)|$ vezes.
  - Para todas as chamadas, o custo é $O(\sum_{v \in V} |Adj(v)|) = O(\sum_{v \in V} d(v)) = O(E)$.

Busca em profundidade: complexidade de tempo

Complexidade de tempo do DFS:

- A inicialização consome tempo $\Theta(V)$.
- **Ao todo**, fazemos $|V|$ chamadas a **DFS-Visit**.
- Qual é o tempo de execução de todas as chamadas a **DFS-Visit**?
  - Cada chamada percorre a lista de adjacências de um vértice u.
  - O laço da linha 4 é executado $|Adj(u)|$ vezes.
  - Para todas as chamadas, o custo é $O(\sum_{v \in V} |Adj(v)|) = O(\sum_{v \in V} d(v)) = O(E)$.
- A complexidade de tempo do **DFS** é $O(V + E)$.

Busca em profundidade: propriedades

Durante a execução, os vértices de cor cinza correspondem a um caminho na floresta.

- Os rótulos $d[x]$, $f[x]$ têm propriedades úteis que outros algoritmos baseados no **DFS** exploram.
- Eles refletem a ordem em que a busca em profundidade foi executada, e fornecem informação de como é a estrutura do grafo.

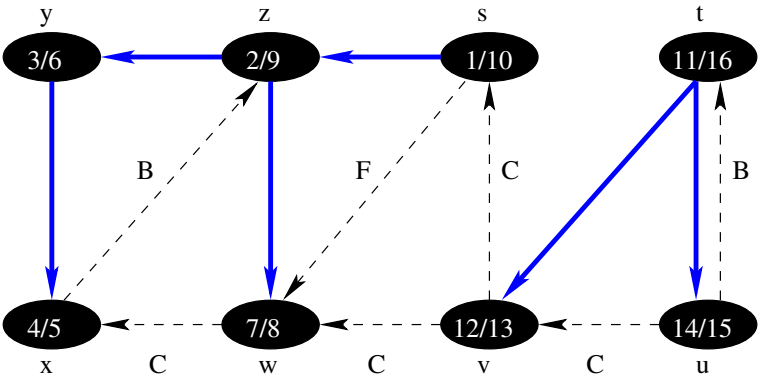
Busca em profundidade: estrutura de parênteses

Teorema dos Parênteses

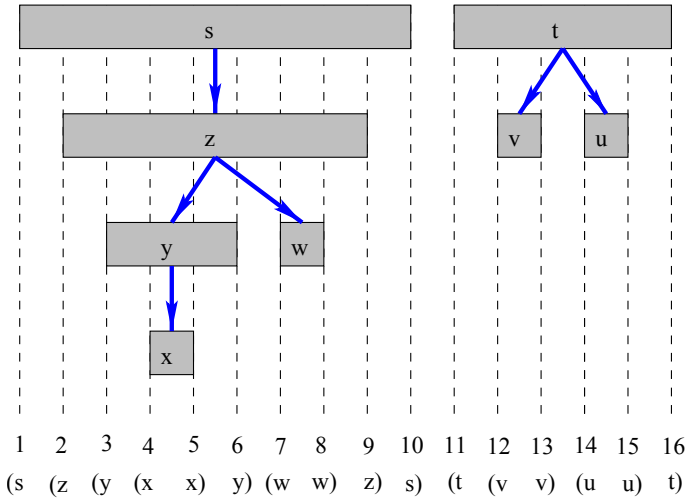
Em uma busca em profundidade sobre um grafo $G = (V, E)$, para quaisquer vértices u e v, ocorre exatamente uma das situações abaixo:

- $[d[u], f[u]]$ e $[d[v], f[v]]$ são disjuntos e nenhum é descendente do outro na floresta de BP.
- $[d[u], f[u]]$ está contido em $[d[v], f[v]]$ e u é descendente de v em uma árvore de BP.
- $[d[v], f[v]]$ está contido em $[d[u], f[u]]$ e v é descendente de u em uma árvore de BP.

Estrutura de parênteses



Estrutura de parênteses



Estrutura de parênteses

Ideia da prova. Podemos supor que $d[u] < d[v]$. Temos dois casos:

Estrutura de parênteses

Ideia da prova. Podemos supor que $d[u] < d[v]$. Temos dois casos:

- $d[v] < f[u]$: v foi descoberto enquanto u era cinza. Logo, v é descendente de u. Os adjacentes de v são explorados e v é finalizado antes da busca voltar a u:

$$d[u] < d[v] < f[v] < f[u].$$

  O intervalo $[d[v], f[v]]$ está contido em $[d[u], f[u]]$.

- $f[u] < d[v]$

Estrutura de parênteses

Ideia da prova. Podemos supor que $d[u] < d[v]$. Temos dois casos:

- $d[v] < f[u]$: v foi descoberto enquanto u era cinza. Logo, v é descendente de u. Os adjacentes de v são explorados e v é finalizado antes da busca voltar a u:

  $$d[u] < d[v] < f[v] < f[u].$$

  O intervalo $[d[v], f[v]]$ está contido em $[d[u], f[u]]$.

- $f[u] < d[v]$: u é finalizado antes de v ser descoberto. Logo,

  $$d[u] < f[u] < d[v] < f[v]$$

  e os intervalos $[d[u], f[u]]$ e $[d[v], f[v]]$ são disjuntos. Neste caso, nenhum desses vértices foi descoberto enquanto o outro estava cinza, e assim, nenhum é descendente do outro na floresta de BP.

Busca em profundidade: Teorema do Caminho Branco

Teorema do Caminho Branco

Em uma floresta de BP, um vértice v é descendente de u se e somente se no instante $d[u]$, quando u foi descoberto, existia um caminho de u a v formado apenas por vértices brancos, com exceção de u.

Busca em profundidade: Teorema do Caminho Branco

Teorema do Caminho Branco

Em uma floresta de BP, um vértice v é descendente de u se e somente se no instante $d[u]$, quando u foi descoberto, existia um caminho de u a v formado apenas por vértices brancos, com exceção de u.

Ideia da prova.

($\Rightarrow$) Se $u = v$ o resultado é óbvio.

Caso contrário, suponha que v é descendente de u na floresta de BP. Como $d[u] < d[v]$, v ainda é branco no instante $d[u]$, e assim cada vértice no caminho de u a v. Portanto, o caminho de u a v na floresta é branco (com exceção de u).

Busca em profundidade: Teorema do Caminho Branco

($\Leftarrow$)
Suponha que no instante $d[u]$ existe um caminho branco de u a v (com exceção de u). Vamos mostrar que todos os vértices no caminho se tornam descendentes de u. Por **contradição**, seja z o primeiro vértice não descendente de u no caminho, e seja w o vértice que precede imediatamente z nesse caminho.

Busca em profundidade: Teorema do Caminho Branco

($\Leftarrow$)
Suponha que no instante $d[u]$ existe um caminho branco de u a v (com exceção de u). Vamos mostrar que todos os vértices no caminho se tornam descendentes de u. Por **contradição**, seja z o primeiro vértice não descendente de u no caminho, e seja w o vértice que precede imediatamente z nesse caminho.

Veja que $f[w] < f[u]$. Como z não é descendente de u, $f[u] < d[z]$. Portanto:

$$f[w] < f[u] < d[z],$$

$$f[w] < d[z].$$

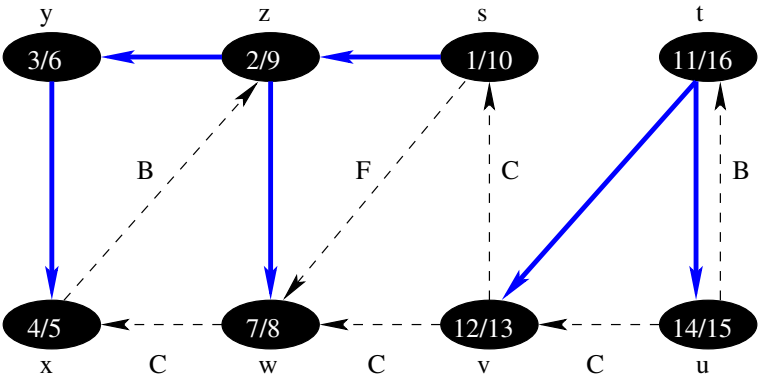
Ou seja, z é um vizinho branco de w no instante $f[w]$, uma contradição.

Busca em profundidade

Dada a floresta de busca, podemos classificar os arcos (ou arestas, em grafos não direcionados):

- arestas de árvore (*tree edges*, **T**), as arestas da floresta de busca em profundidade;
- arestas de retorno (*backward edges*, **B**) ligam um vértice a um ancestral;
- arestas de avanço (*forward edges*, **F**) ligam um vértice a um descendente;
- arestas de cruzamento (*cross edges*, **C**) são todas as outras arestas do grafo.

Busca em profundidade



Busca em profundidade: Exercício

Exercício: Modifique o algoritmo **DFS** para que ele classifique as arestas do grafo em arestas de árvore, de retorno, de avanço e de cruzamento.

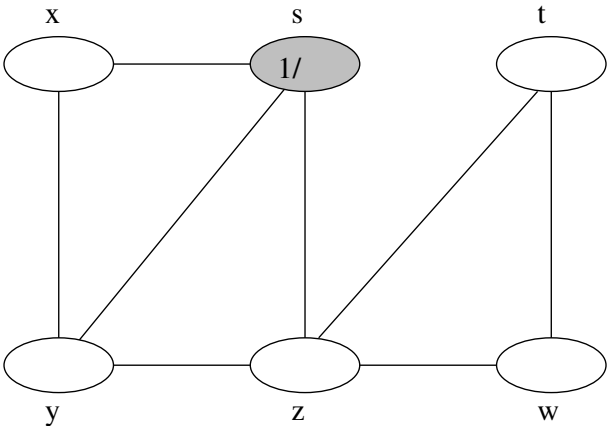
Busca em profundidade em um grafo não direcionado

Em grafos não direcionados:

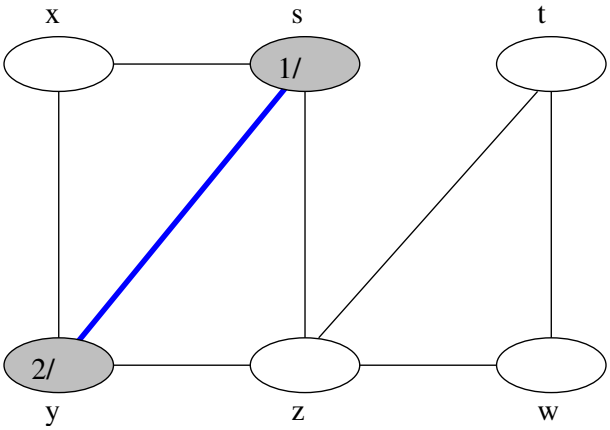
- não pode haver aresta de cruzamento (por quê?),
- nem arestas de avanço.

Depois de uma execução do **DFS** em um grafo não direcionado, cada aresta pode ser classificada como aresta de árvore ou aresta de retorno.

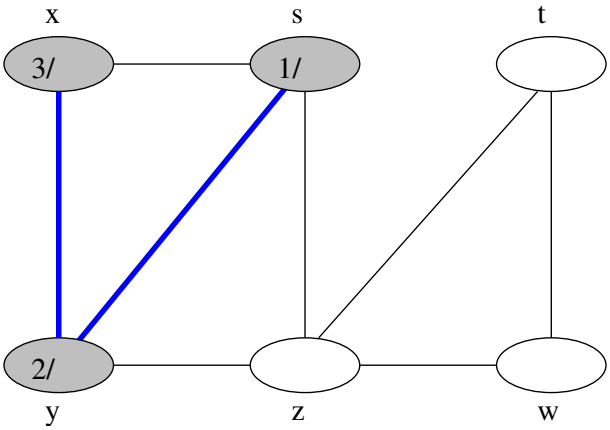
Busca em profundidade em um grafo não direcionado



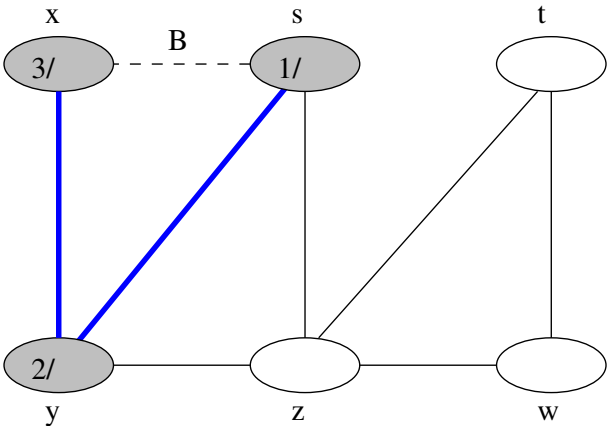
Busca em profundidade em um grafo não direcionado



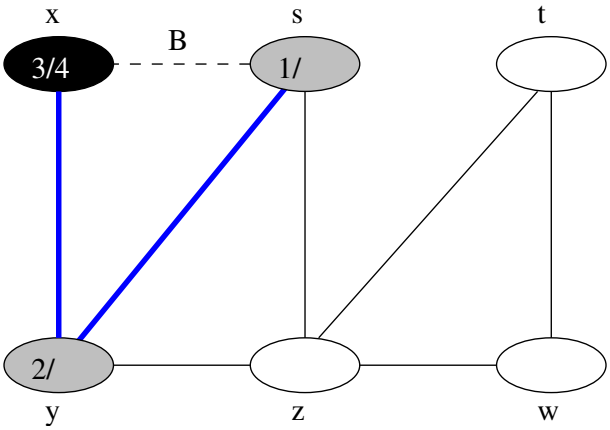
Busca em profundidade em um grafo não direcionado



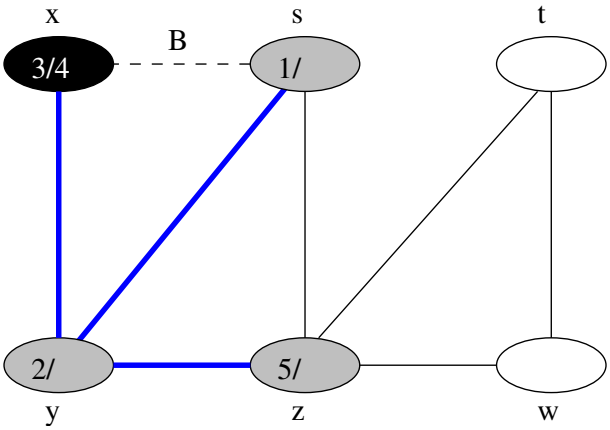
Busca em profundidade em um grafo não direcionado



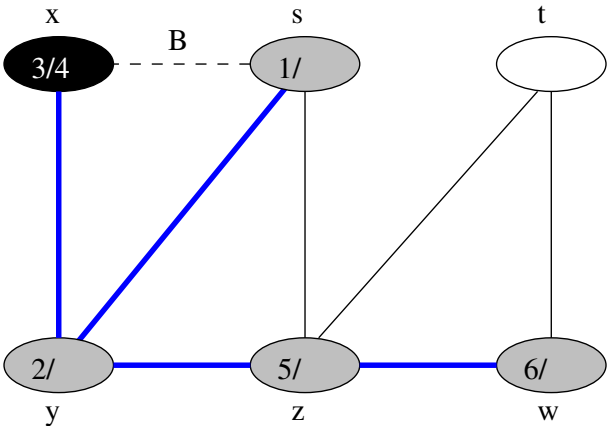
Busca em profundidade em um grafo não direcionado



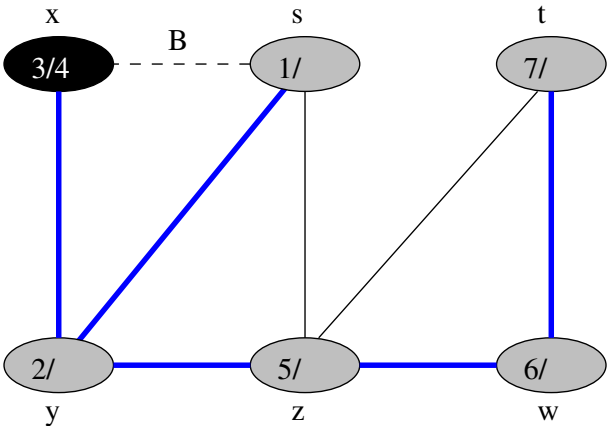
Busca em profundidade em um grafo não direcionado



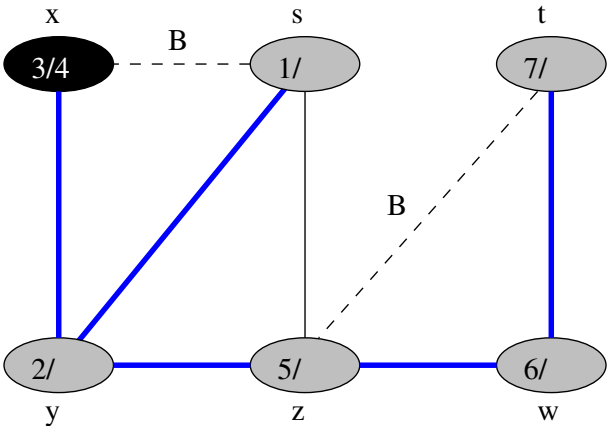
Busca em profundidade em um grafo não direcionado



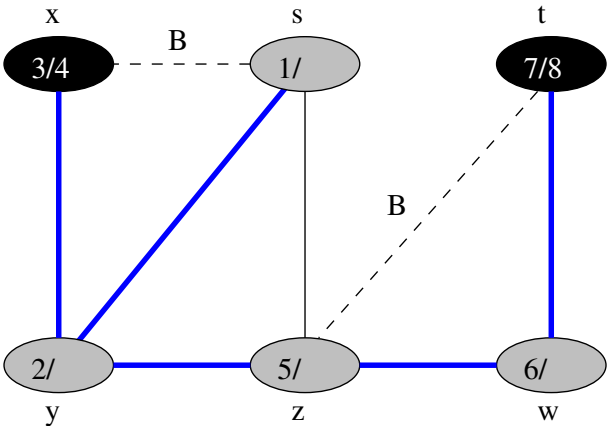
Busca em profundidade em um grafo não direcionado



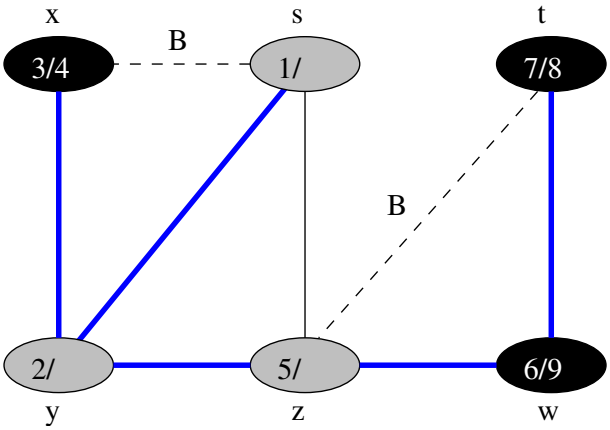
Busca em profundidade em um grafo não direcionado



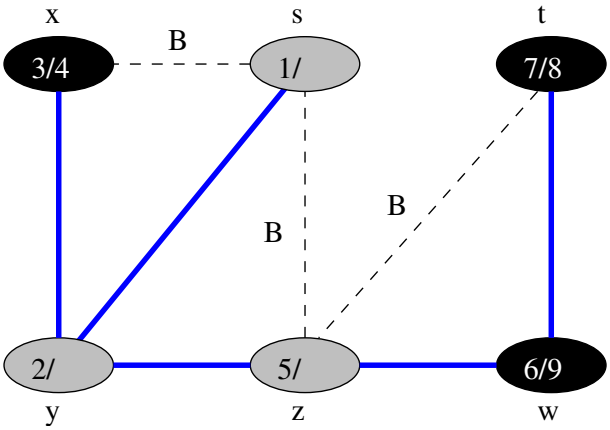
Busca em profundidade em um grafo não direcionado



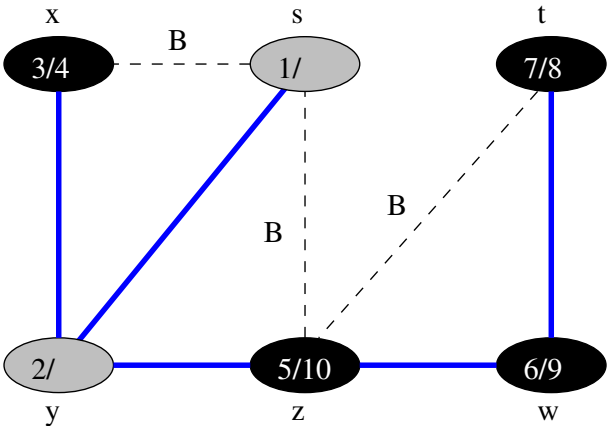
Busca em profundidade em um grafo não direcionado



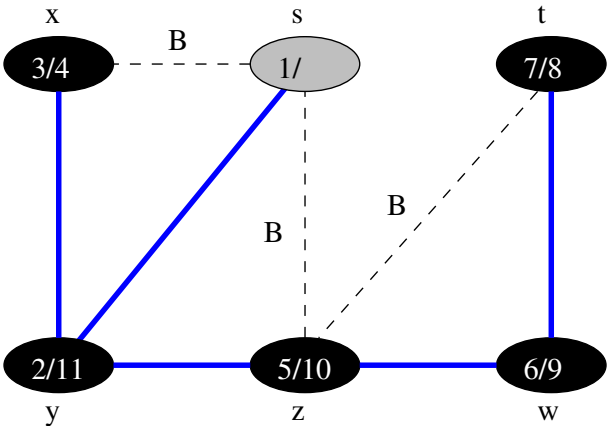
Busca em profundidade em um grafo não direcionado



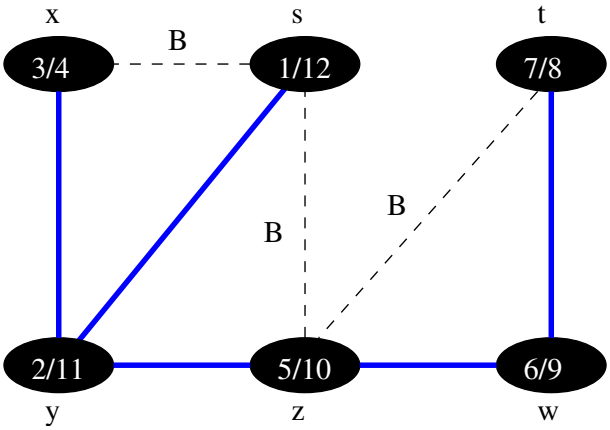
Busca em profundidade em um grafo não direcionado



Busca em profundidade em um grafo não direcionado



Busca em profundidade em um grafo não direcionado



Resumo

Síntese

- **Busca em profundidade** (**DFS**) é um algoritmo de busca em grafos recursivo que gera uma floresta de busca.

Material bibliográfico e exercícios

T. Cormen et al. Algoritmos - Teoria e Prática (3a ed.). – **Cap. 22** (22.3)

Exercícios: ver exercícios no final do capítulo 22.3.

Dúvidas

Dúvidas?