Busca em Grafos

Prof. Fábio Dias

26 de novembro de 2024

Busca em Grafos

Como percorrer os vértices de um grafo?

- Mais complicado que lista, vetor, árvore binária
- Podem ser direcionados ou não direcionados
- Queremos descobrir informações sobre sua estrutura
- Podemos pensar em cada componente separadamente
- Gera uma árvore de busca

Dois algoritmos

- Busca em largura (diremos apenas BFS)
- Busca em profundidade (diremos apenas DFS)

Busca em Largura

Busca em Largura

- Seja um grafo $G = (V, E)$ e um vértice fonte/origem s de onde a busca irá iniciar.
- A busca em largura irá percorrer as arestas de G descobrindo todos os vértices que podem ser alcançados a partir de s.

Vértices alcançáveis v:

- Alcançamos v a partir de s se há caminho de s a v
- Pode haver diversos caminhos entre s a v
- Queremos algum com o menor comprimento

A distância de s a v é o comprimento de um caminho mais curto de s a v:

- Denotamos este valor por $dist(s, v)$
- Se v não for alcançável, definimos $dist(s, v) = \infty$

Busca em Largura

Ideia principal:

- Começa por algum vértice origem s e o **descobre**;
- Descobre (alcança) todos os vértices de distância 1, ou seja, os vizinhos do vértice origem;
- Depois, descobre todos os vértices de distância 2, ou seja, os vizinhos dos vizinhos do vértice origem que ainda não foram descobertos;
- Depois, descobre todos os vértices de distância 3, ...;

  $\vdots$

BFS corresponde a encontrar (calcular) o *menor caminho* (em relação a quantidade de arestas) de s para todos os demais vértices alcançáveis.

Busca em Largura

Ideia do algoritmo:

- Percorremos os vértices usando uma fila Q
- Começamos com o vértice de origem s
- Para cada vizinho v do vértice atual u
  - Adicionamos uma aresta (u, v) à árvore de busca
  - Inserimos v na fila de processamento
- Repetimos com o primeiro vértice da fila

Cores dos vértices

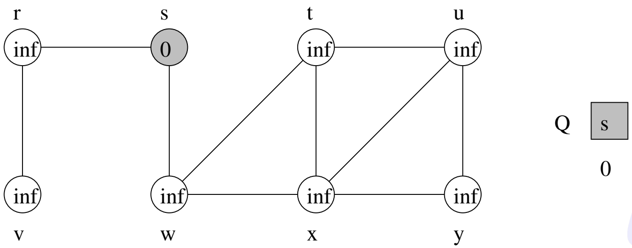
Vamos pintar o grafo durante a busca:

- $cor[v] = branco$ se não descobrimos v ainda
- $cor[v] = cinza$ se já descobrimos, mas não finalizamos v
- $cor[v] = preto$ se já descobrimos e já finalizamos v

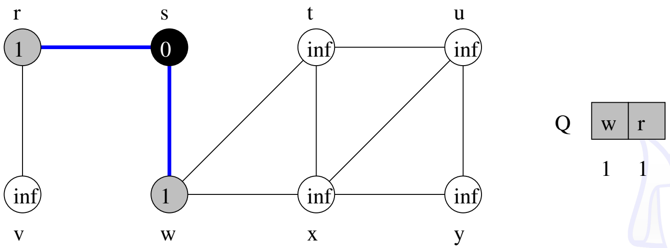
Observações:

- Não é necessário em uma implementação
- Mas facilita o entendimento do algoritmo

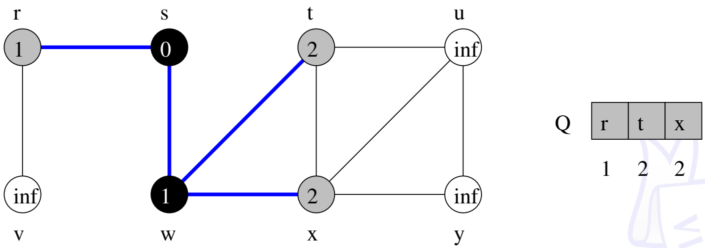
Busca em Largura - Exemplo



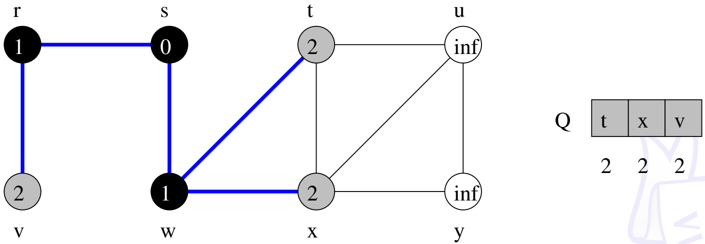
Busca em Largura - Exemplo



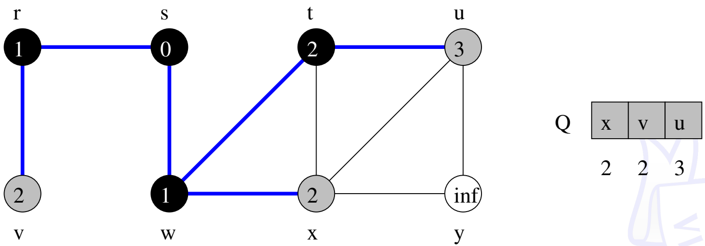
Busca em Largura - Exemplo



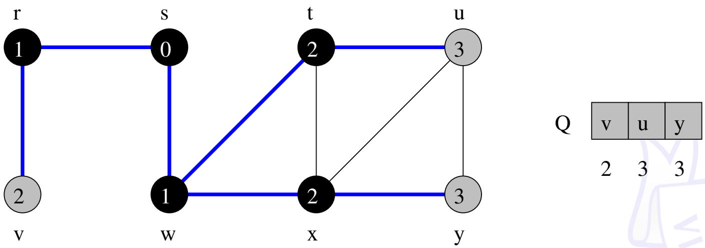
Busca em Largura - Exemplo



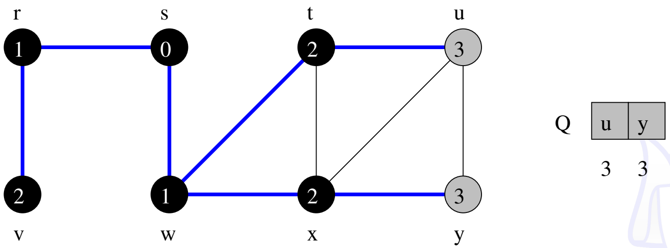
Busca em Largura - Exemplo



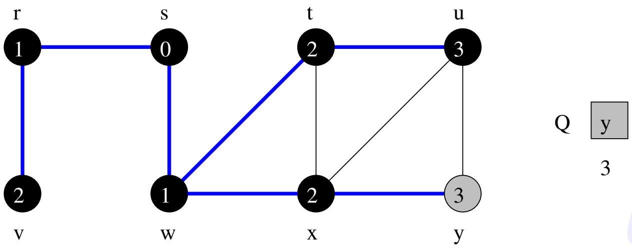
Busca em Largura - Exemplo



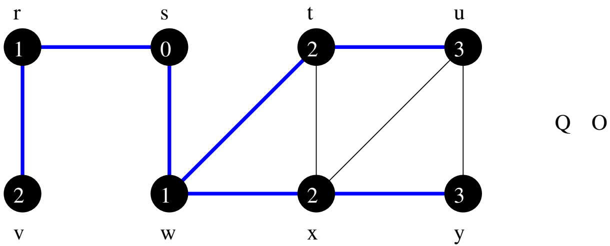
Busca em Largura - Exemplo



Busca em Largura - Exemplo



Busca em Largura - Exemplo



Algoritmo Busca em Largura

BFS(G, s)

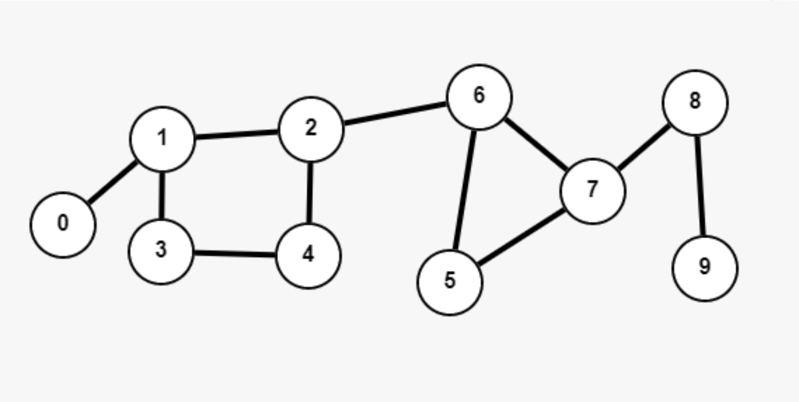
    para cada $u \in V[G] \setminus \{s\}$ **faça**
        $cor[u] \leftarrow \text{branco}$
        $d[u] \leftarrow \infty$
        $\pi[u] \leftarrow \text{NIL}$
    $cor[s] \leftarrow \text{cinza}$
    $d[s] \leftarrow 0$
    $\pi[s] \leftarrow \text{NIL}$

- $\pi[u]$ representa o pai de u na árvore de busca
- $d[u]$ é a distância do vértice origem s a u.
- Representamos G com listas de adjacências
- A árvore de busca em largura é representada por π.

Algoritmo Busca em Largura

```
Q ← ∅
Enqueue(Q, s)
enquanto Q ≠ ∅ faça
    u ← Dequeue(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            Enqueue(Q, v)
    cor[u] ← preto
```

Algoritmo Busca em Largura



Busca em Largura - Analise de Complexidade

- O tempo de inicialização é $O(V)$
- Um vértice não volta a ser branco
  - Enfileiramos cada vértice no máximo uma vez
  - Desenfileiramos cada vértice no máximo uma vez
  - Cada operação na fila leva tempo $O(1)$
  - O tempo gasto com a fila é $O(V)$
- Processamos cada vértice uma vez
  - Cada lista de adjacências é percorrida uma vez
  - No pior caso, percorremos todas as listas
  - O tempo gasto percorrendo adjacências é $O(E)$

A complexidade da busca em largura é $O(V + E)$

Busca em Largura

- A busca tem esse nome porque expande a fronteira entre vértices descobertos e não descobertos uniformemente.
- Explora arestas partindo do vértice descoberto mais antigo.
- O algoritmo descobre todos os vértices à distância k de s, antes de descobrir quaisquer vértices à distância $k+1$.

Busca em Profundidade

Busca em Profundidade

Se nós usássemos uma pilha ao invés da fila Q, nós teríamos uma outra busca ordenada, chamada de busca em profundidade:

- Nós vamos "mais fundo quanto o possível";
- Volte até encontrar um vértice adjacente inexplorado;
- Vá mais fundo quanto possível;

  $\vdots$

Busca em Profundidade

- Como o próprio nome indica, busca mais fundo no grafo, sempre que possível.
- A busca em profundidade explora arestas partindo do vértice mais recentemente descoberto.
- Oposto do BFS que explora o vértice mais antigo descoberto.
- Cria uma floresta ao invés de uma única árvore.
- Uma floresta é um grafo onde cada componente é uma árvore.

Busca em Profundidade

Ideia do algoritmo:

- começamos com o vértice de origem s
- Removemos um vértice, digamos u da pilha
- Para cada vizinho não visitado v do vértice u
  1. Adicionamos uma aresta (u, v) à árvore de busca
  2. Visitamos recursivamente a partir de v

Cores dos vértices

De novo, vamos pintar o grafo durante a busca

- $cor[v] = branco$ se não descobrimos v ainda
- $cor[v] = cinza$ se já descobrimos, mas não finalizamos v
- $cor[v] = preto$ se já descobrimos e já finalizamos v

Observações

- Os vértices cinza têm suas chamadas recursivas ativas
- A pilha de chamadas induz um caminho na floresta

Tempo de descoberta e finalização

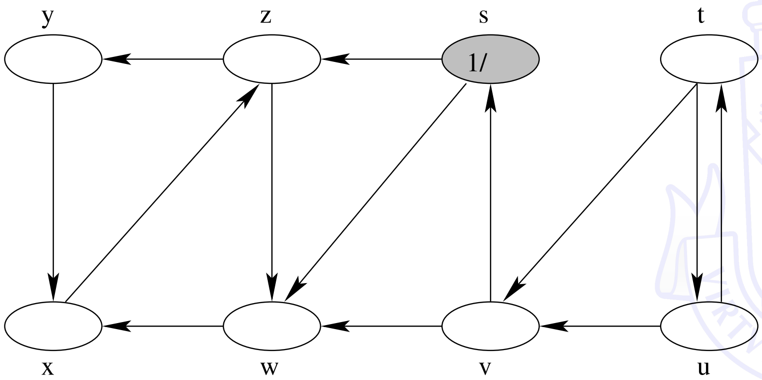
Vamos associar rótulos aos vértices:

- $d[v]$ é instante de descoberta de v
- $f[v]$ é instante de finalização de v

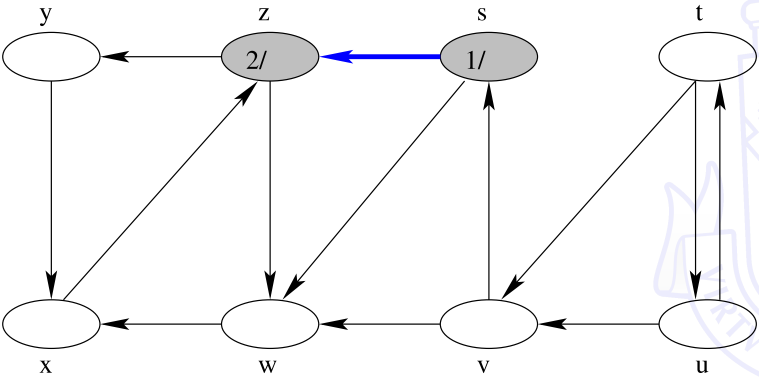
Observações:

- Os rótulos são inteiros distintos entre 1 e $2|V|$
- Refletem os instantes em que v muda de cor

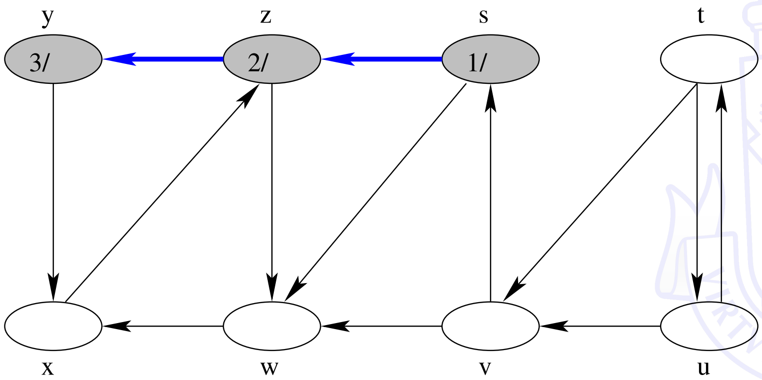
Exemplo de busca em profundidade



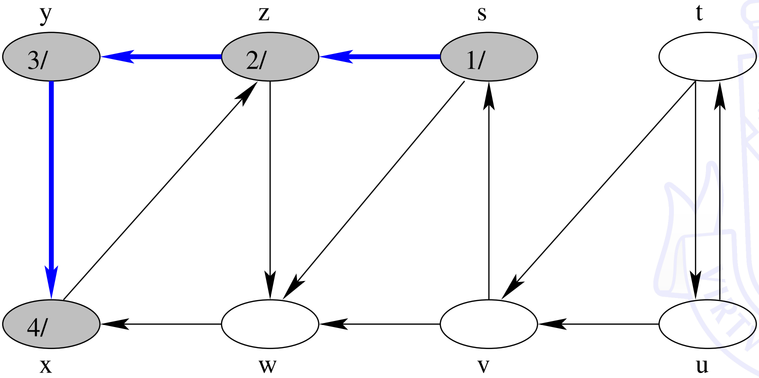
Exemplo de busca em profundidade



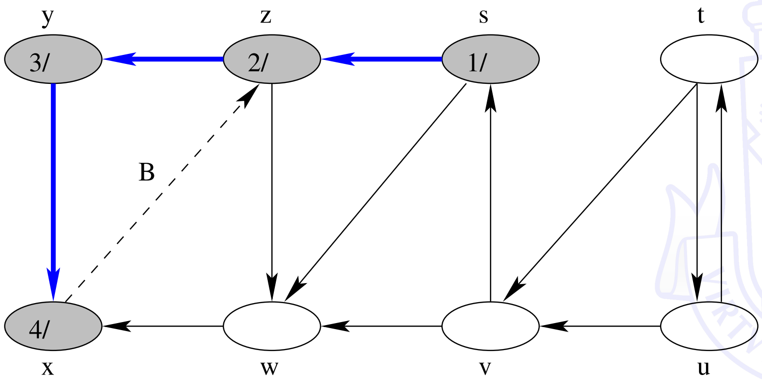
Exemplo de busca em profundidade



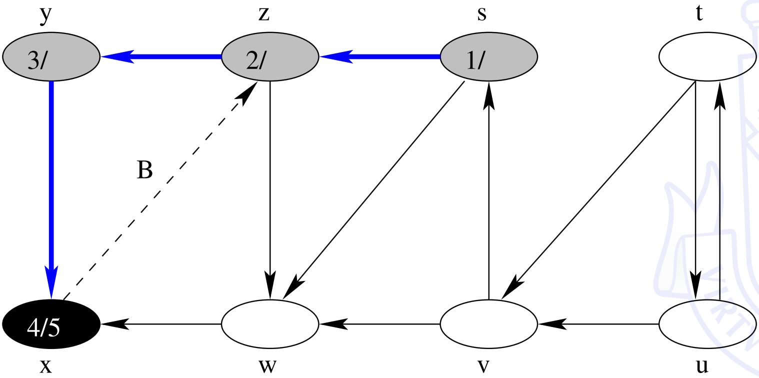
Exemplo de busca em profundidade



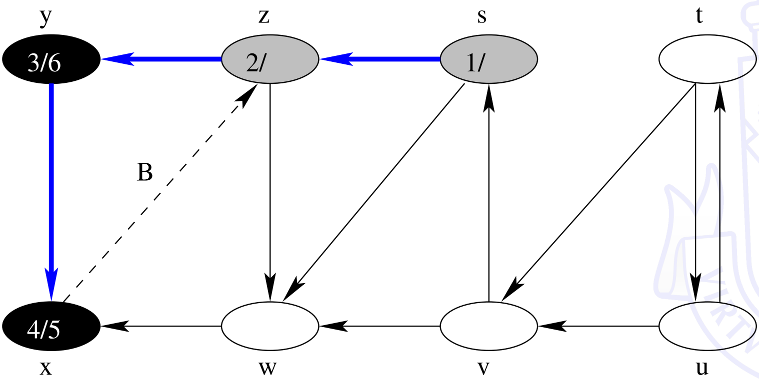
Exemplo de busca em profundidade



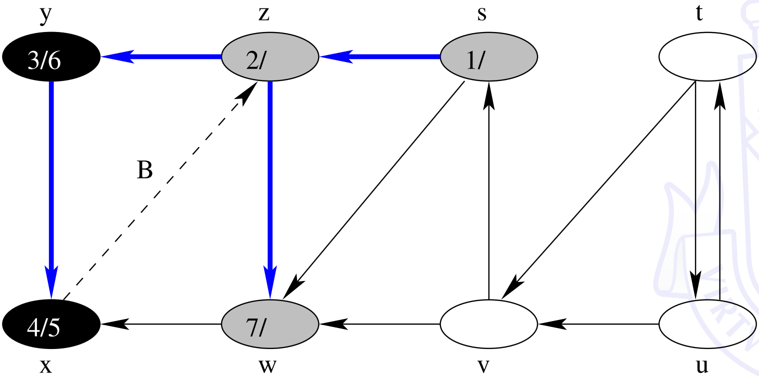
Exemplo de busca em profundidade



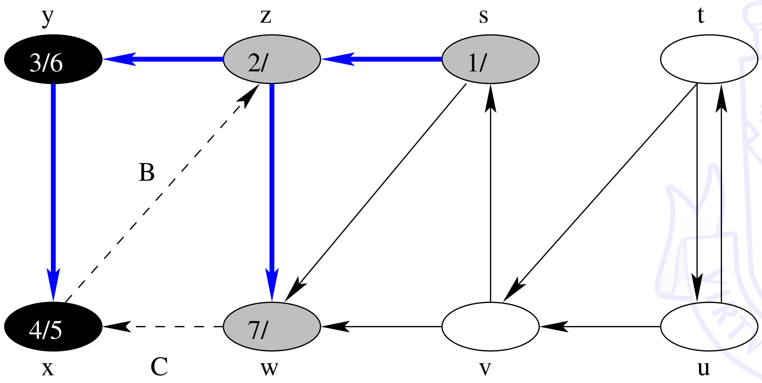
Exemplo de busca em profundidade



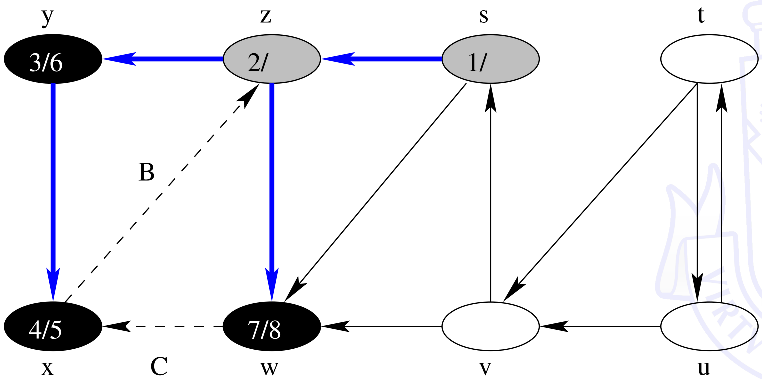
Exemplo de busca em profundidade



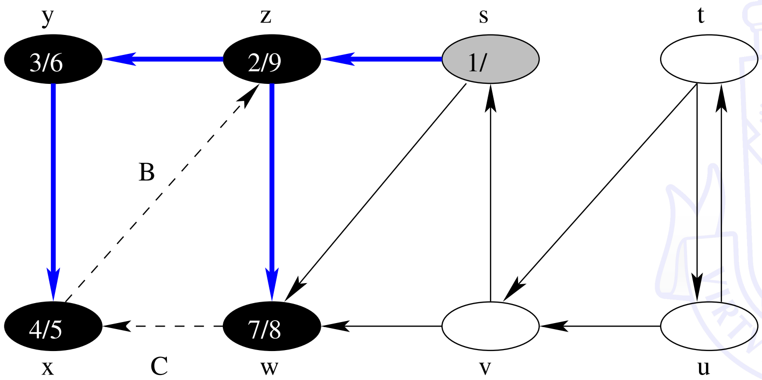
Exemplo de busca em profundidade



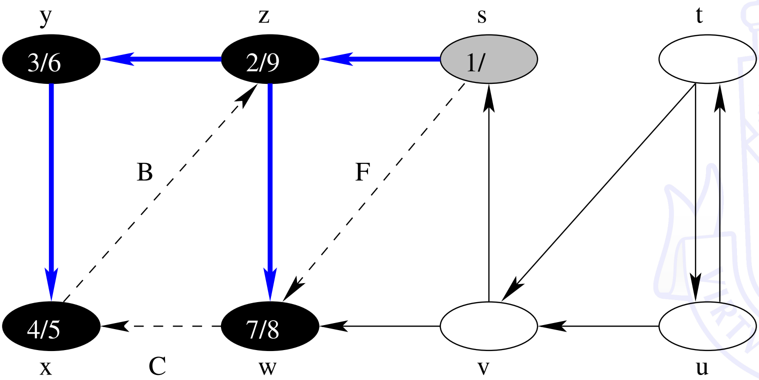
Exemplo de busca em profundidade



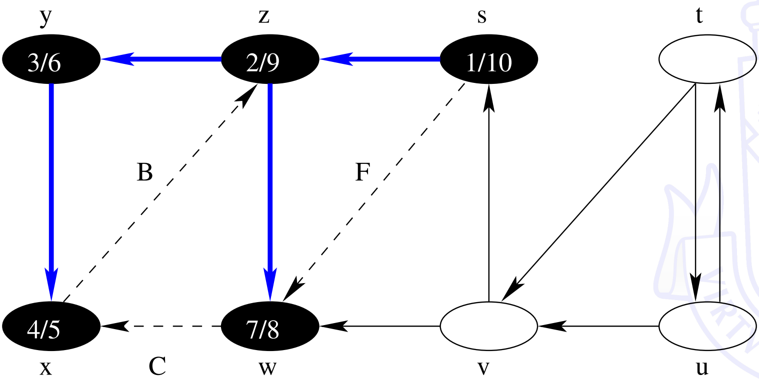
Exemplo de busca em profundidade



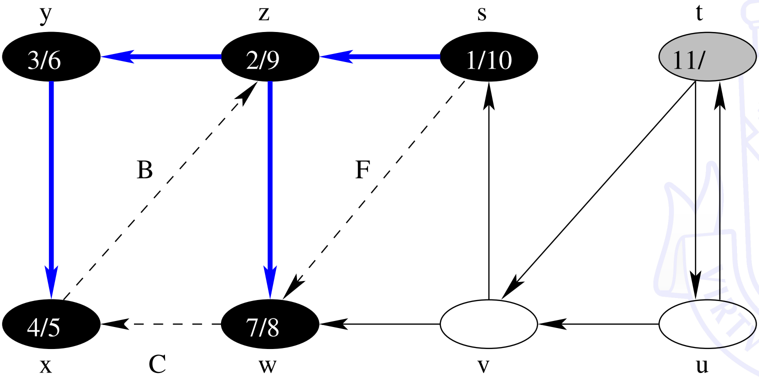
Exemplo de busca em profundidade



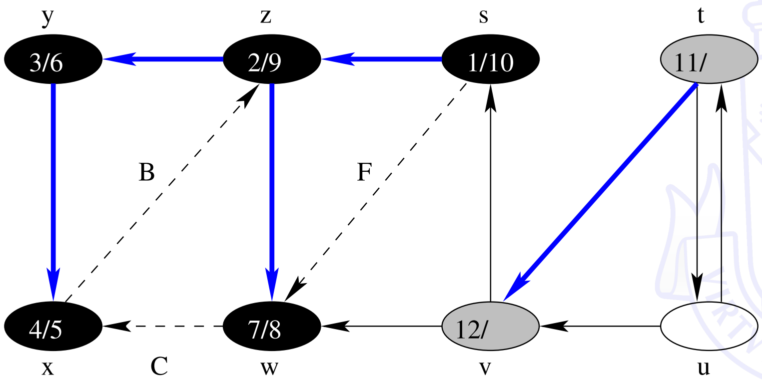
Exemplo de busca em profundidade



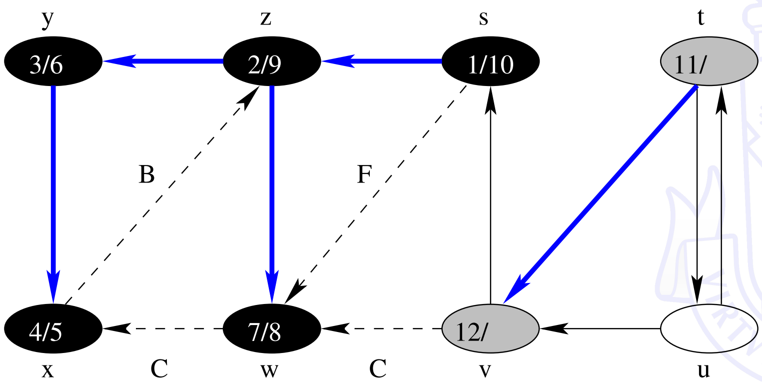
Exemplo de busca em profundidade



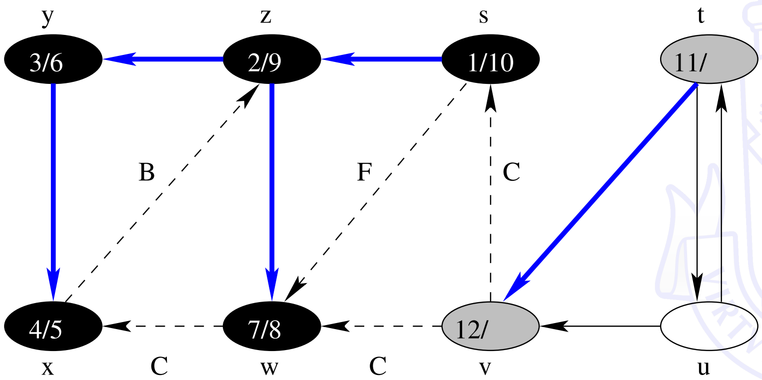
Exemplo de busca em profundidade



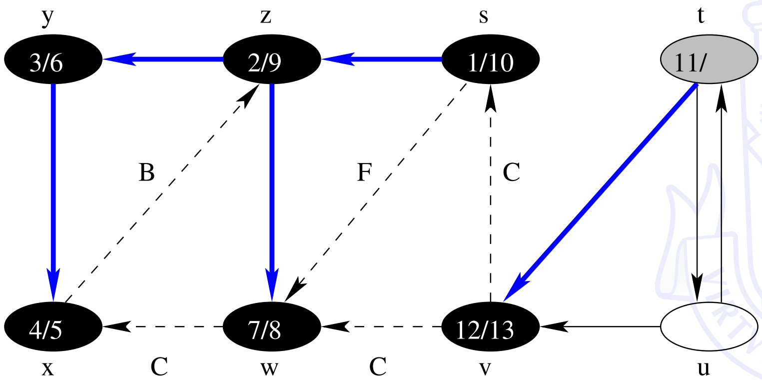
Exemplo de busca em profundidade



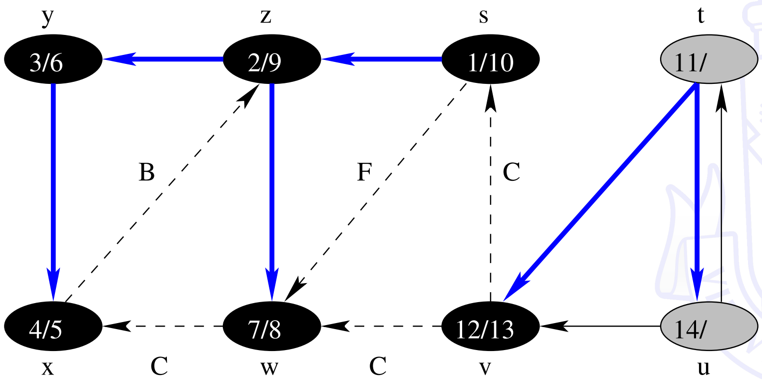
Exemplo de busca em profundidade



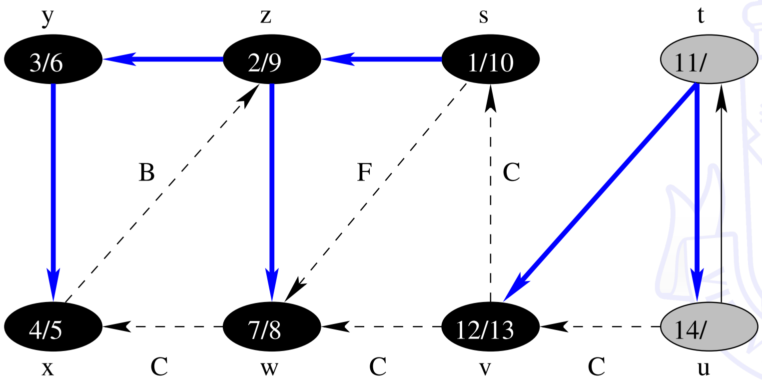
Exemplo de busca em profundidade



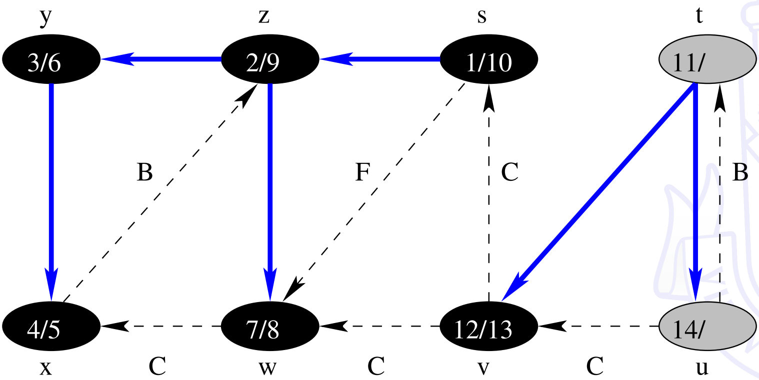
Exemplo de busca em profundidade



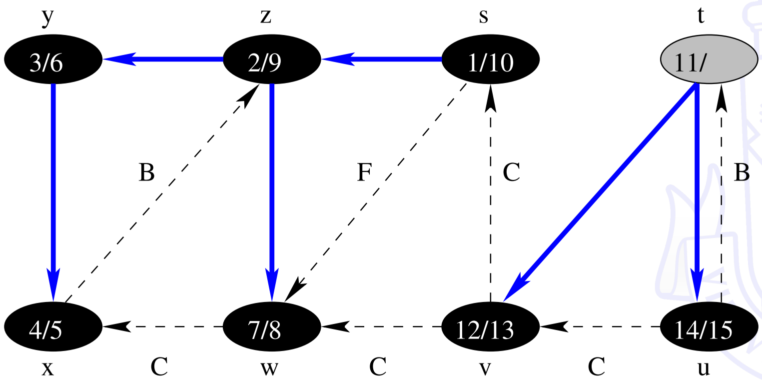
Exemplo de busca em profundidade



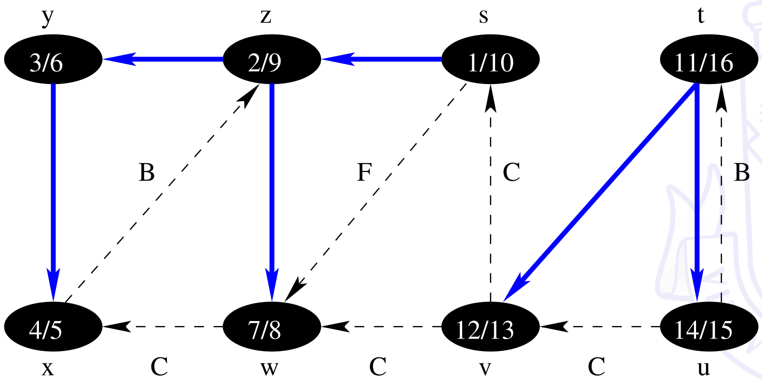
Exemplo de busca em profundidade



Exemplo de busca em profundidade



Exemplo de busca em profundidade



Rótulos versus cores

Observe que para todo vértice v:

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

Algoritmo Busca em Profundidade

```
DFS(G)
  para cada u ∈ V[G] faça
      cor[u] ← branco
      π[u] ← NIL
  tempo ← 0
  para cada u ∈ V[G] faça
      se cor[u] = branco então
          DFS-VISIT(u)
```

- Representamos G com listas de adjacências
- A floresta de busca em profundidade é representada por π
- Calcula os instantes $d[v]$ e $f[v]$

Algoritmo Busca em Profundidade

```
DFS-VISIT(u)
    cor[u] ← cinza
    tempo ← tempo + 1
    d[u] ← tempo
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            π[v] ← u
            DFS-VISIT(v)
    cor[u] ← preto
    tempo ← tempo + 1
    f[u] ← tempo
```

Busca em Profundidade - Análise de complexidade

Analisamos o tempo do algoritmo principal DFS()

- A inicialização consome tempo $O(V)$
- Realizamos $|V|$ chamadas a DFS-visit()

E o tempo da sub-rotina DFS-visit()

- Processamos cada vértice exatamente uma vez
- Cada chamada percorre sua lista de adjacências
- O tempo gasto percorrendo adjacências é $O(E)$

A complexidade da busca em profundidade é $O(V + E)$

Busca em Profundidade

- Usamos esse algoritmo para encontrar ciclos no tempo $O(|V| + |E|)$;
- Também pode ser usado em grafos não conexo, repetindo a chamada da DFS para todo vértice branco.

Aplicações das Buscas

- A Busca em Largura é usada para encontrar as componentes conexas em um grafo não direcionado no tempo $O(|V| + |E|)$, além de calcular o caminho mínimo em número de arestas;
- A Busca em Profundidade é usada para encontrar ciclos ou determinar se um grafo contém ciclos no tempo $O(|V| + |E|)$.