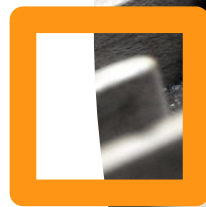
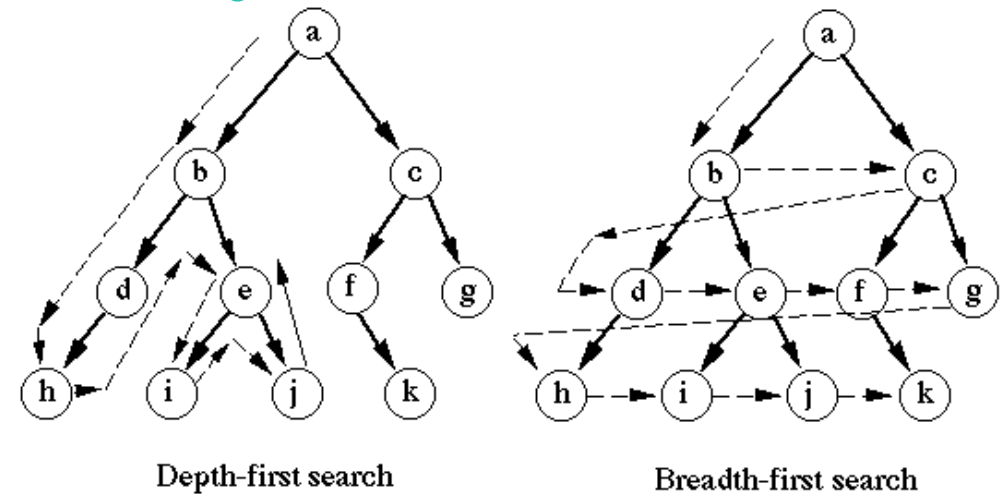




Busca em Profundidade



- 



Algoritmo DFS

- Inicia em um vértice inicial v
- Percorre as arestas a partir do vértice v que ainda possui arestas não exploradas saindo dele
- Quando todas arestas adjacentes a v tiverem sido exploradas, a busca “anda para trás” (do inglês *backtrack*) para explorar vértices do qual v foi descoberto;
- O processo continua até que sejam descobertos todos os vértices que são alcançáveis a partir do vértice original;
- Se todos os vértices já foram descobertos, então é o fim.
- Caso contrário o processo continua a partir de um novo vértice de origem ainda não descoberto (grafos desconexos).
 - Diferente de uma busca em árvore
 - ao final de uma busca em uma árvore, pode haver vértices que não foram alcançados.

Algoritmo DFS

- Toda vez que se descobre um vértice **v** durante a varredura da lista de adjacências de um vértice já descoberto **u**, registra-se esse evento definindo o predecessor de v: **$\Pi[v] = u$**
- Diferente da busca em largura, cujo subgrafo dos predecessores forma uma árvore, o subgrafo dos predecessores da busca em profundidade pode ser formada por várias árvores
- O subgrafo dos predecessores de uma busca em profundidade forma uma **Floresta de busca em profundidade**
 - Abrange várias arvores de busca em profundidade



Algoritmo DFS

- Esse algoritmo (proposto) também identifica cada vértice com dois carimbos de tempo (*timestamp*)
 - O primeiro carimbo de tempo **v.d** registra quando v é descoberto a primeira vez
 - O segundo carimbo de tempo **v.f** registra quando a busca termina de examinar a lista de adjacências de v
- Esses carimbos de tempo dão informações importantes sobre a estrutura do grafo e, em geral, são úteis para deduzir o comportamento da busca em profundidade



Características do Algoritmo

- Também utiliza o artifício de cores
 - Branco (não visitado)
 - Cinza (visitado, porém os seus adjacentes ainda não foram visitados)
 - Preto (visitado, e seus adjacentes já foram visitados)
- 1 variável global de tempo
- 1 vetor de predecessores (Π)
- Recursivo (ou pilha)



Algoritmo DFS

DFS (Grafo G)

Para cada vértice u de G

Cor[u] = BRANCO

Predecessor[u] = NIL

Tempo = 0

Para cada vértice u de G

Se Cor[u] == BRANCO então

DFS-Visit(G, u)

DFS-Visit(Grafo G, Vértice u)

Tempo = Tempo + 1

D[u] = tempo

Cor[u] = CINZA

Para cada v no conjunto em Adj[u]

SE Cor[v] == BRANCO

Predecessor[v] = u

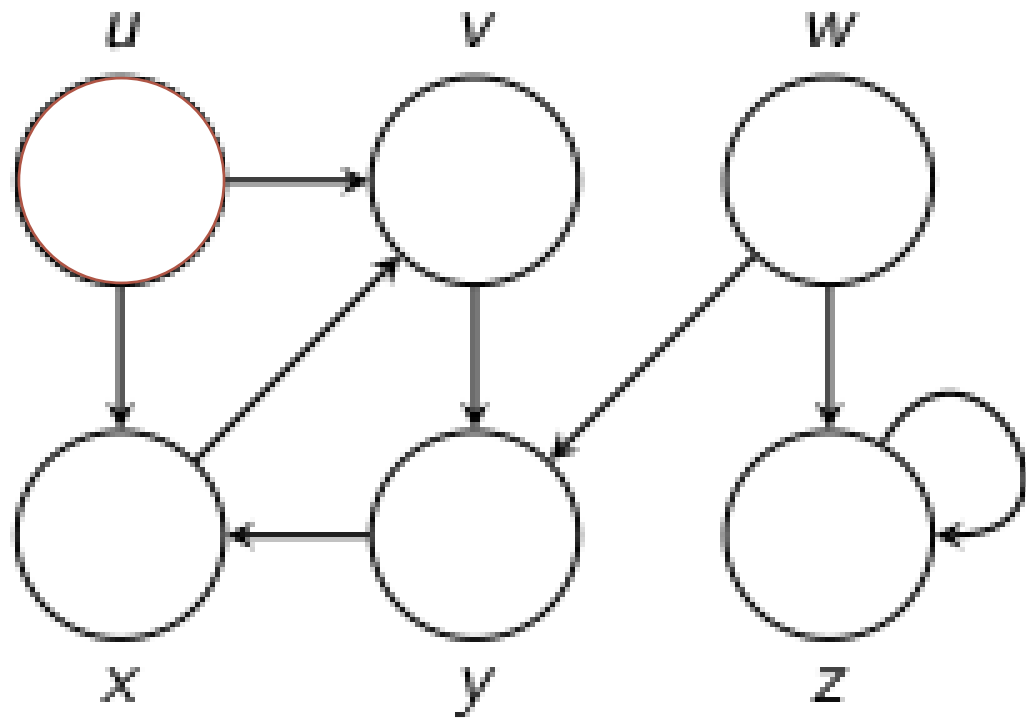
DFS-Visit(G, v)

Cor[u] = PRETO

tempo = tempo + 1

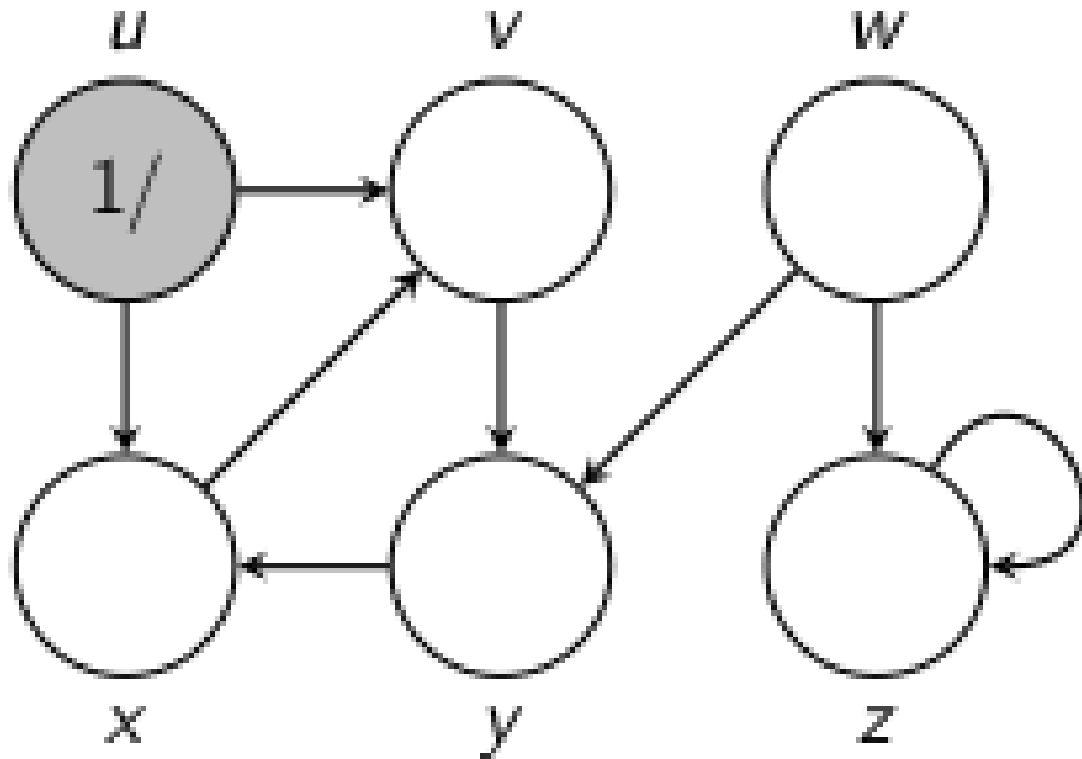
F[u] = tempo

Algoritmo



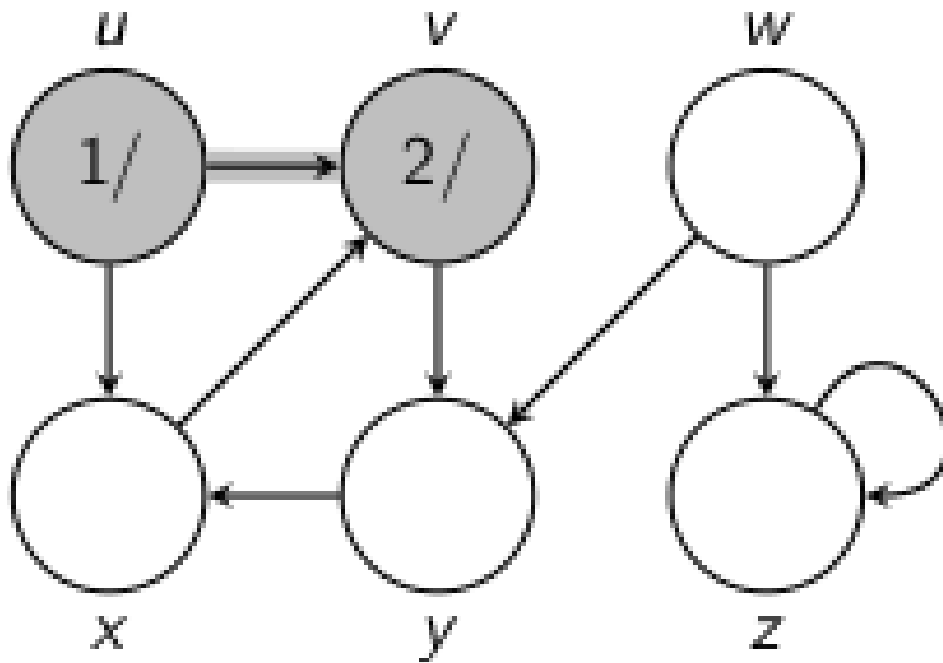
- Grafo G
- $V = \{u, v, w, x, y, z\}$

Algoritmo



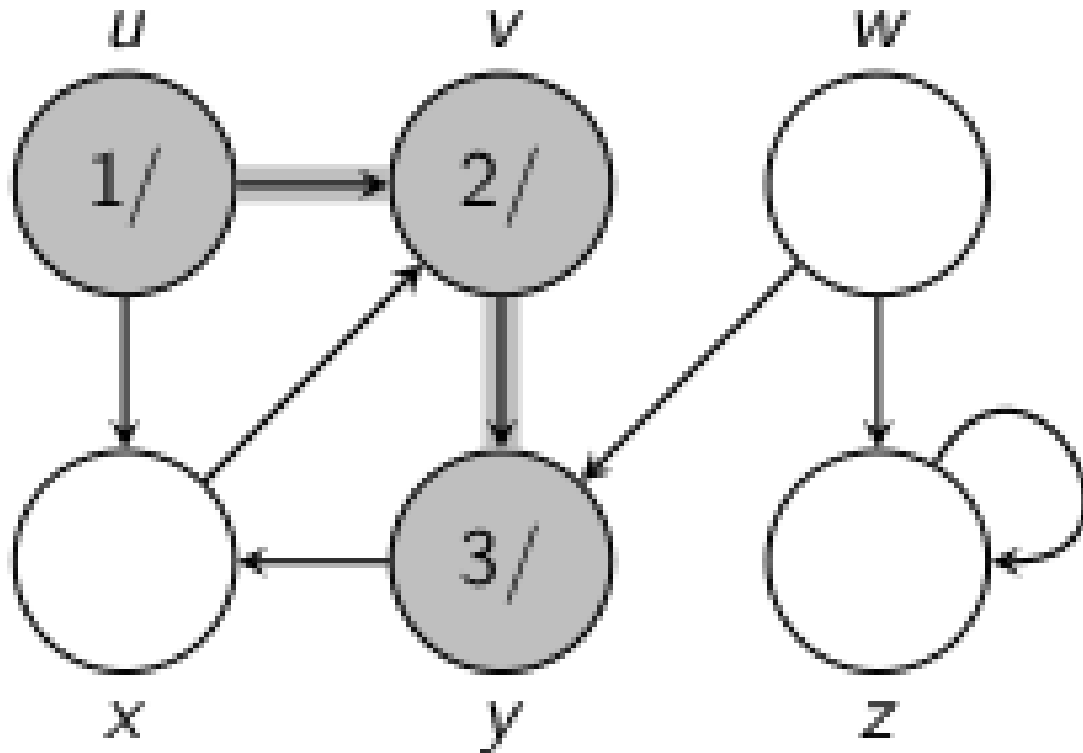
- Grafo G
- Vértice u
- Tempo = 1
- $u.d = 1$

Algoritmo



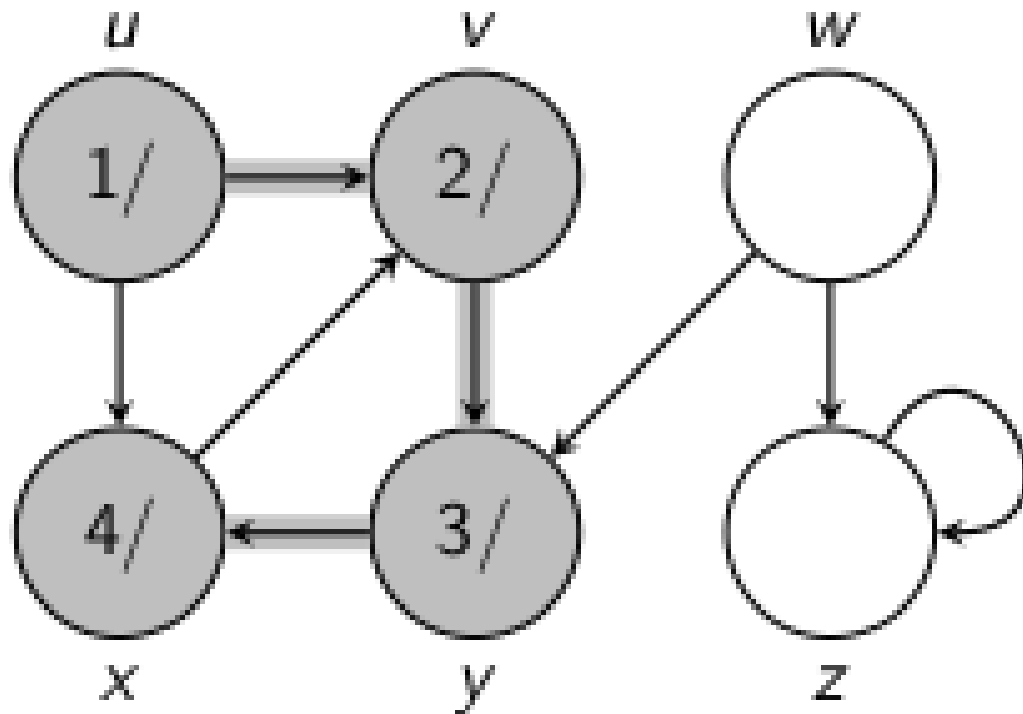
- Grafo G
- Vértice u
- Tempo = 2
- $u.d = 1$
- $v.d = 2$

Algoritmo



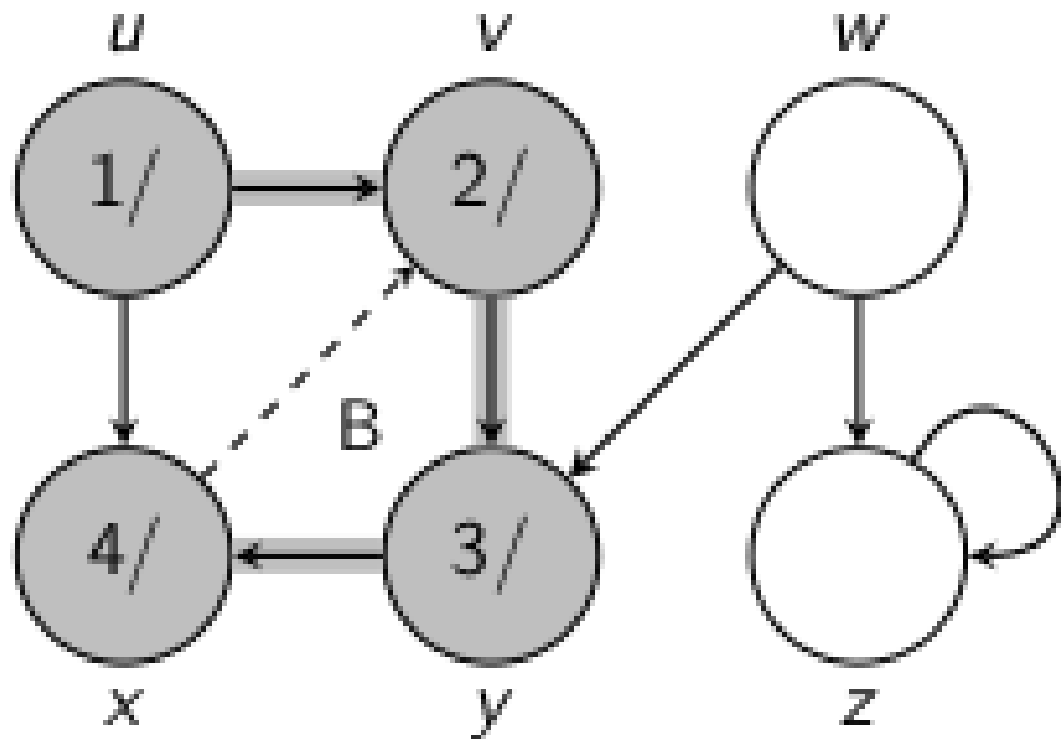
- Grafo G
- Vértice u
- Tempo = 3
- $u.d = 1$
- $v.d = 2$
- $y.d = 3$

Algoritmo



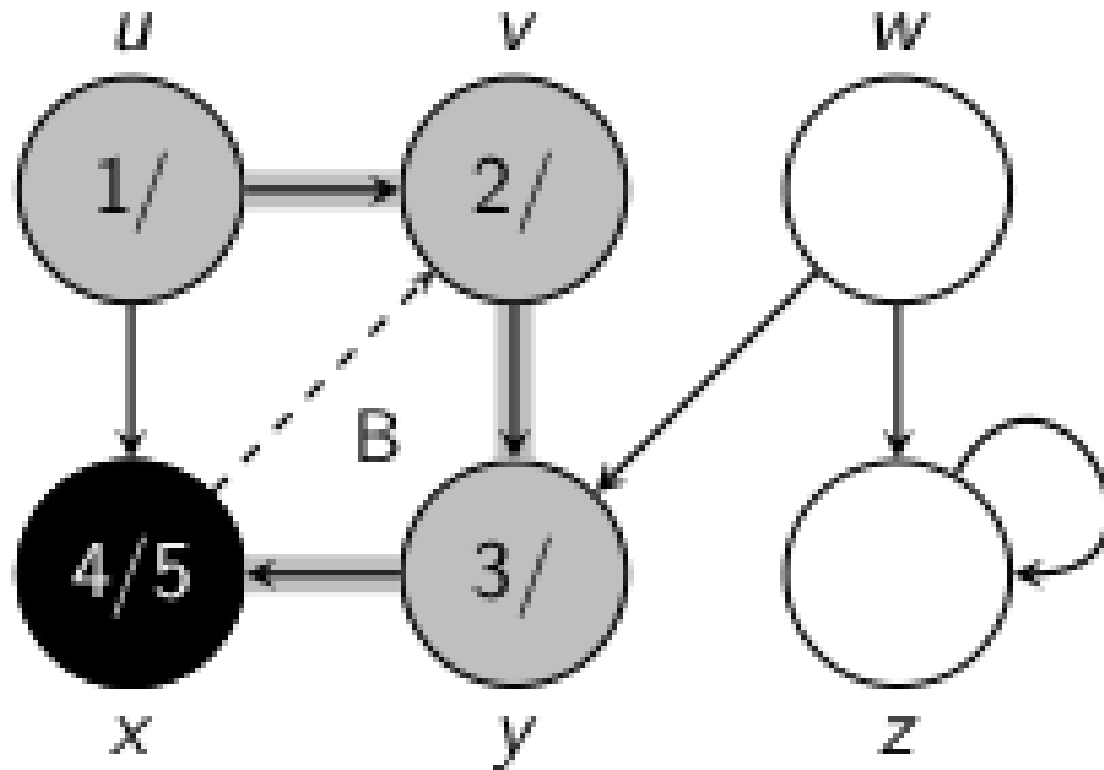
- Grafo G
- Vértice u
- Tempo = 4
- $u.d = 1$
- $v.d = 2$
- $y.d = 3$
- $x.d = 4$

Algoritmo



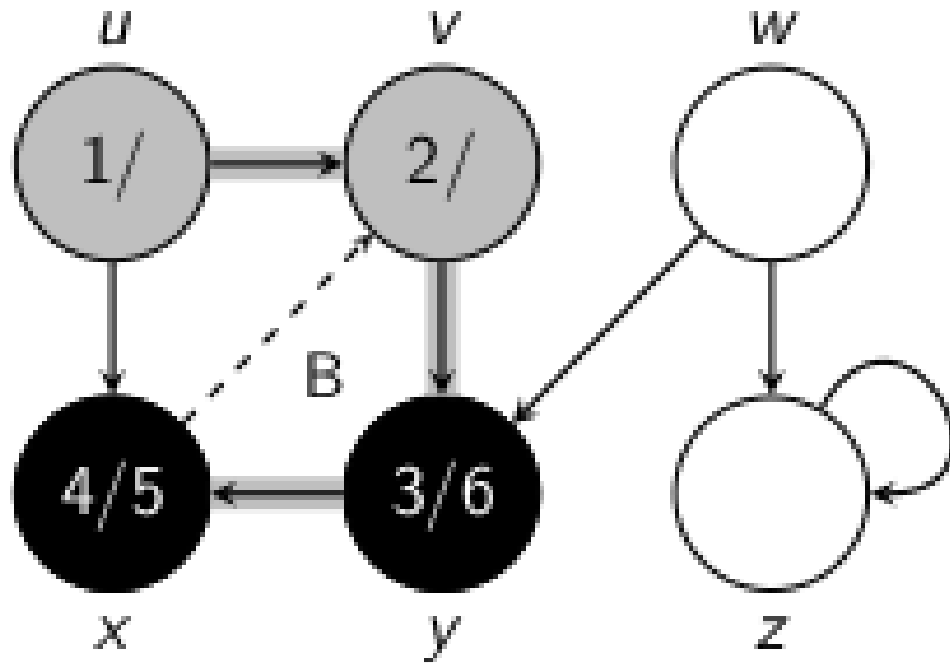
- Grafo G
- Vértice u
- Tempo = 4
- $u.d = 1$
- $v.d = 2$
- $y.d = 3$
- $x.d = 4$

Algoritmo



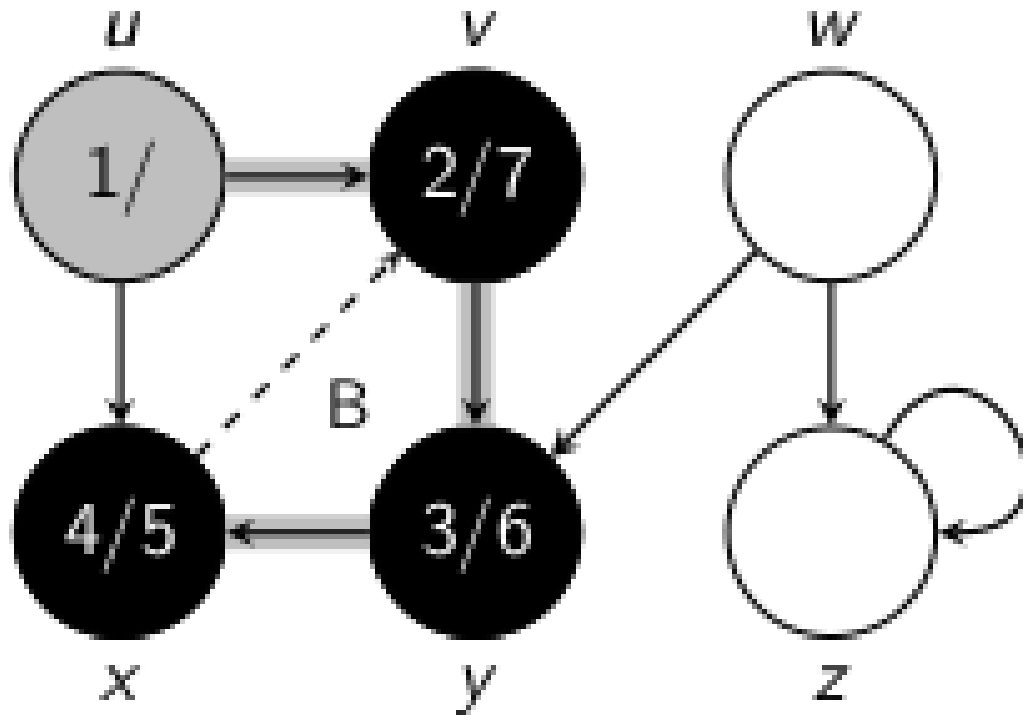
- Grafo G
- Vértice u
- Tempo = 5
- $u.d = 1$
- $v.d = 2$
- $y.d = 3$
- $x.d = 4; x.f = 5$

Algoritmo



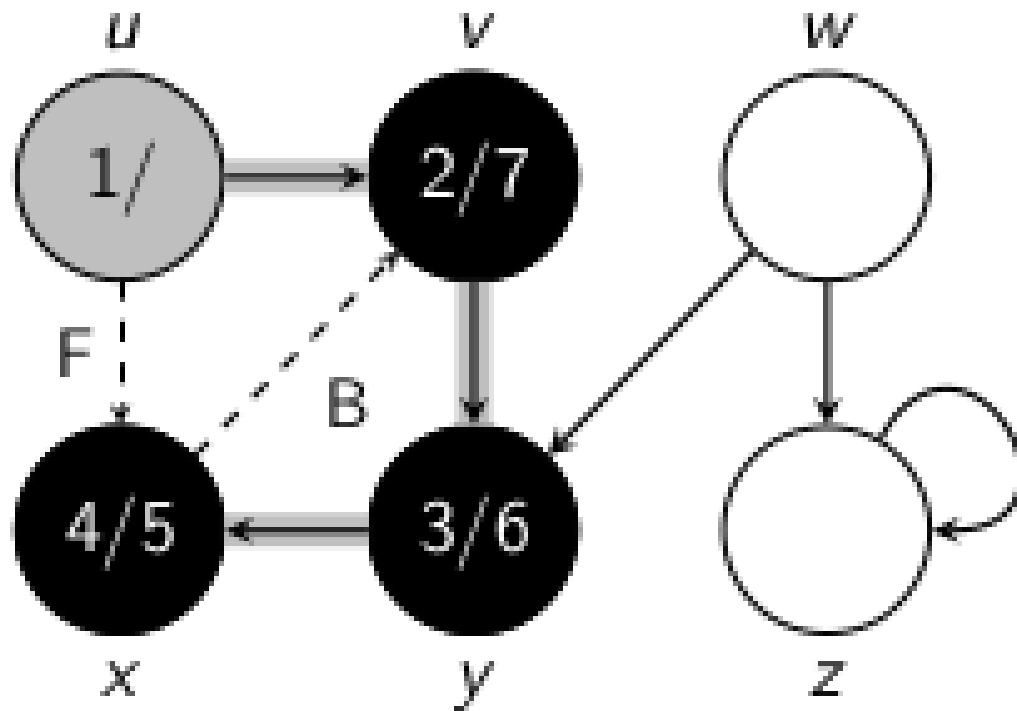
- Grafo G
- Vértice u
- Tempo = 6
- $u.d = 1$
- $v.d = 2$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$

Algoritmo



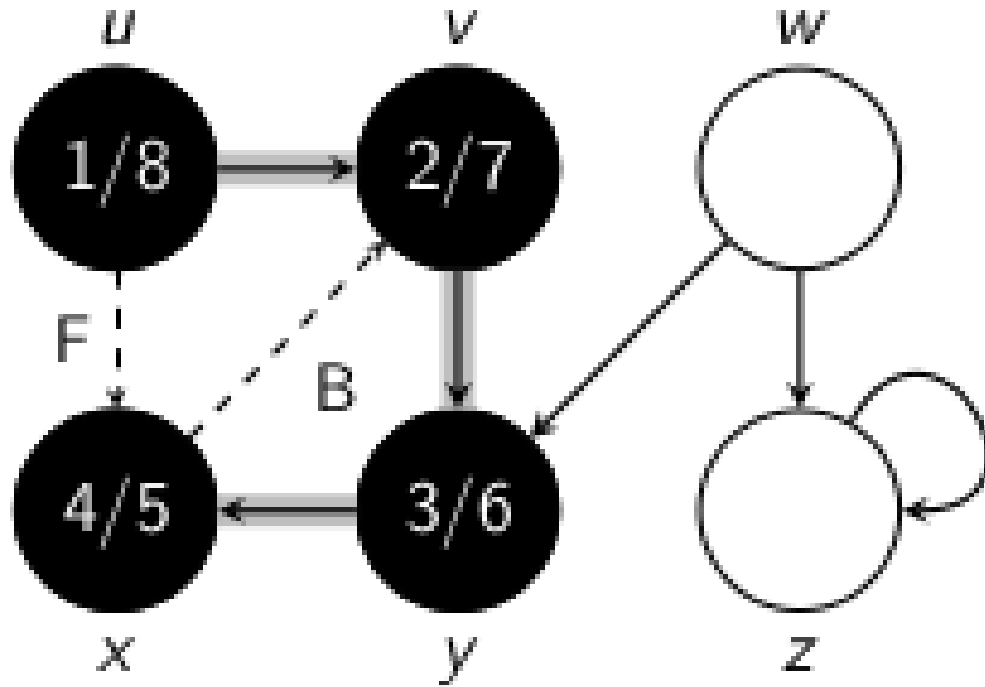
- Grafo G
- Vértice u
- Tempo = 7
- $u.d = 1$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$

Algoritmo



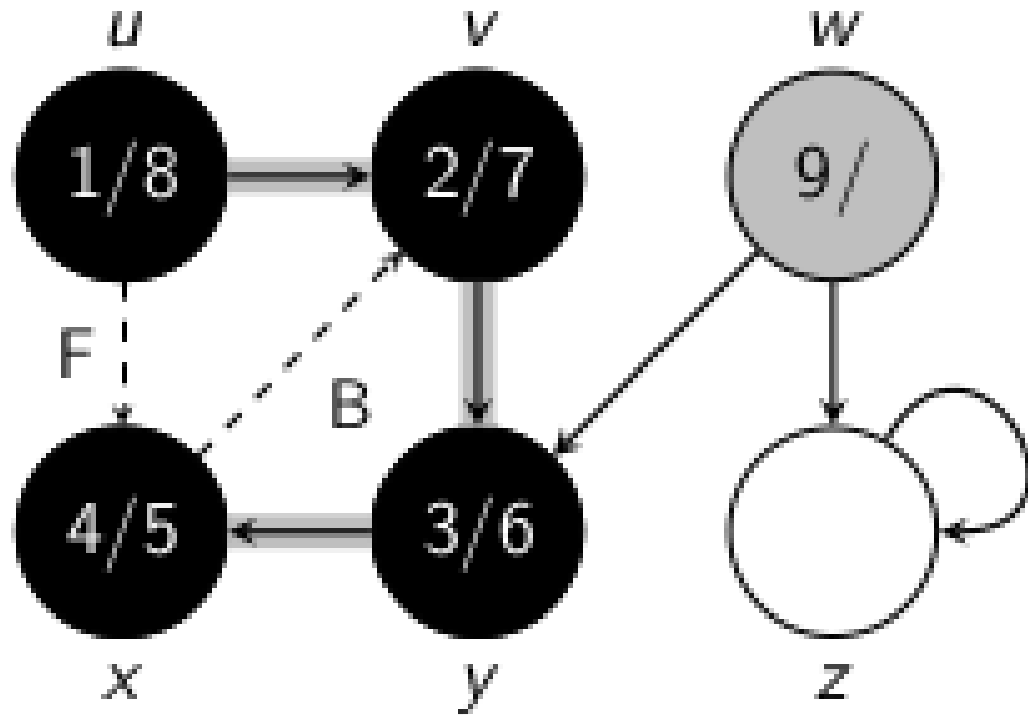
- Grafo G
- Vértice u
- Tempo = 7
- $u.d = 1$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$

Algoritmo



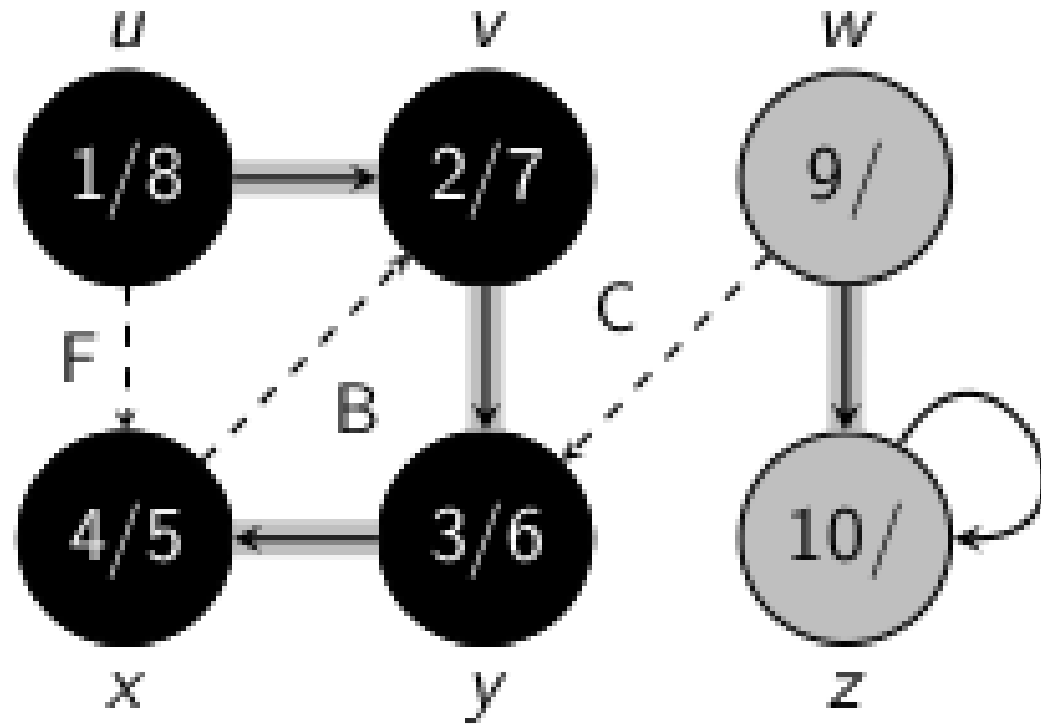
- Grafo G
- Vértice u
- Tempo = 8
- $u.d = 1$; $u.f = 8$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$

Algoritmo



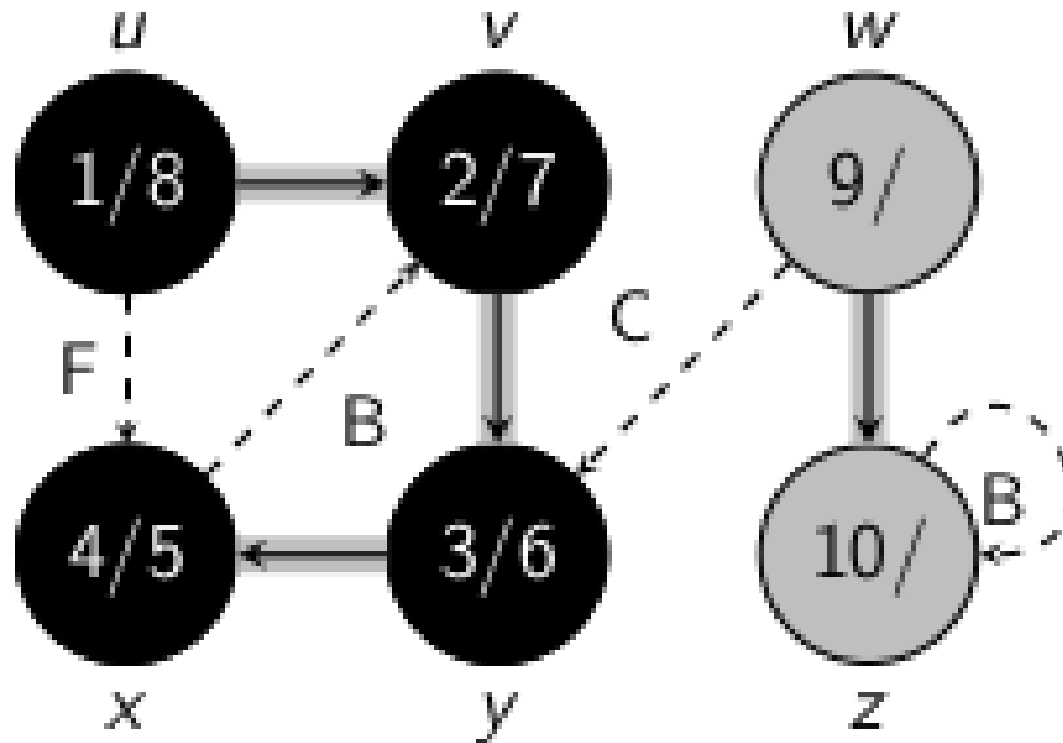
- Grafo G
- Vértice u
- Tempo = 9
- $u.d = 1$; $u.f = 8$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$
- $w.d = 9$

Algoritmo



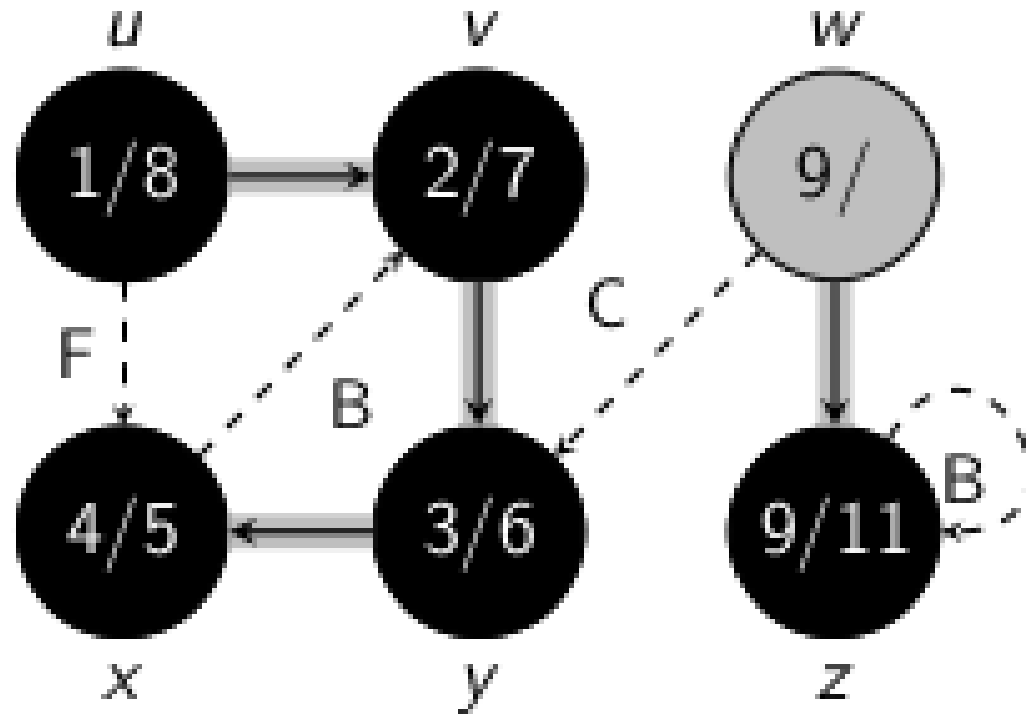
- Grafo G
- Vértice u
- Tempo = 10
- $u.d = 1$; $u.f = 8$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$
- $w.d = 9$
- $z.d = 10$

Algoritmo



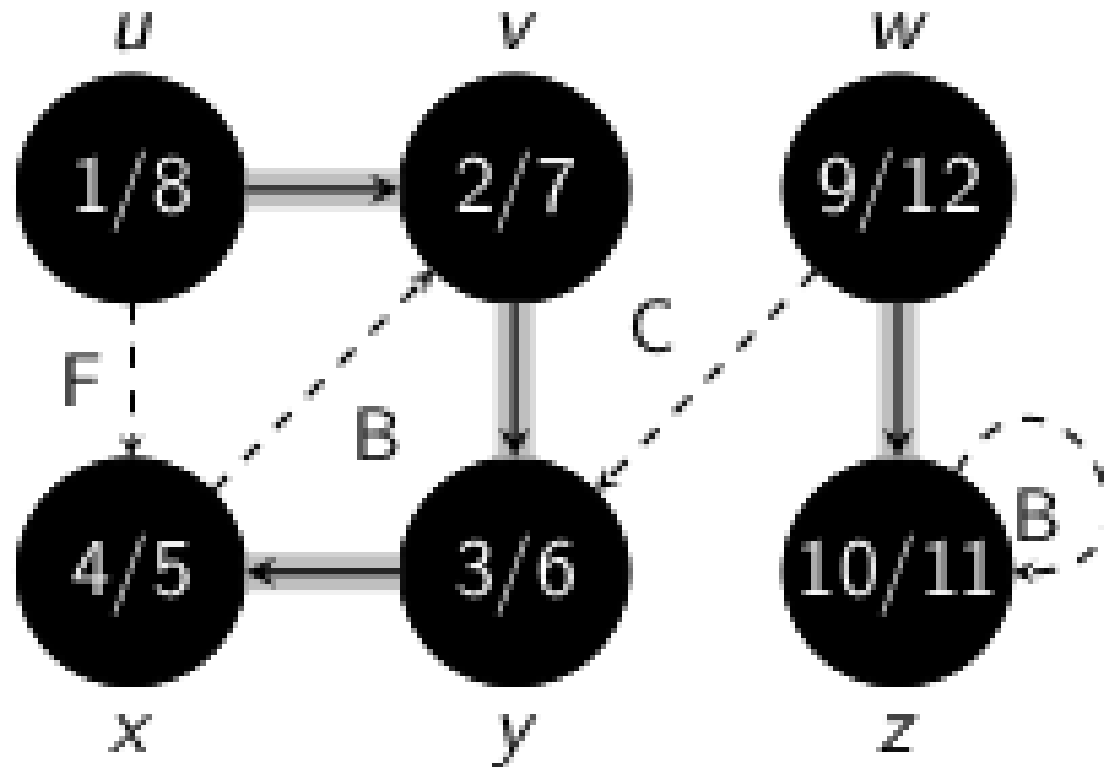
- Grafo G
- Vértice u
- Tempo = 10
- $u.d = 1$; $u.f = 8$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$
- $w.d = 9$
- $z.d = 10$

Algoritmo



- Grafo G
- Vértice u
- Tempo = 11
- $u.d = 1$; $u.f = 8$
- $v.d = 2$; $v.f = 7$
- $y.d = 3$; $y.f = 6$
- $x.d = 4$; $x.f = 5$
- $w.d = 9$
- $z.d = 10$; $z.f = 11$

Algoritmo

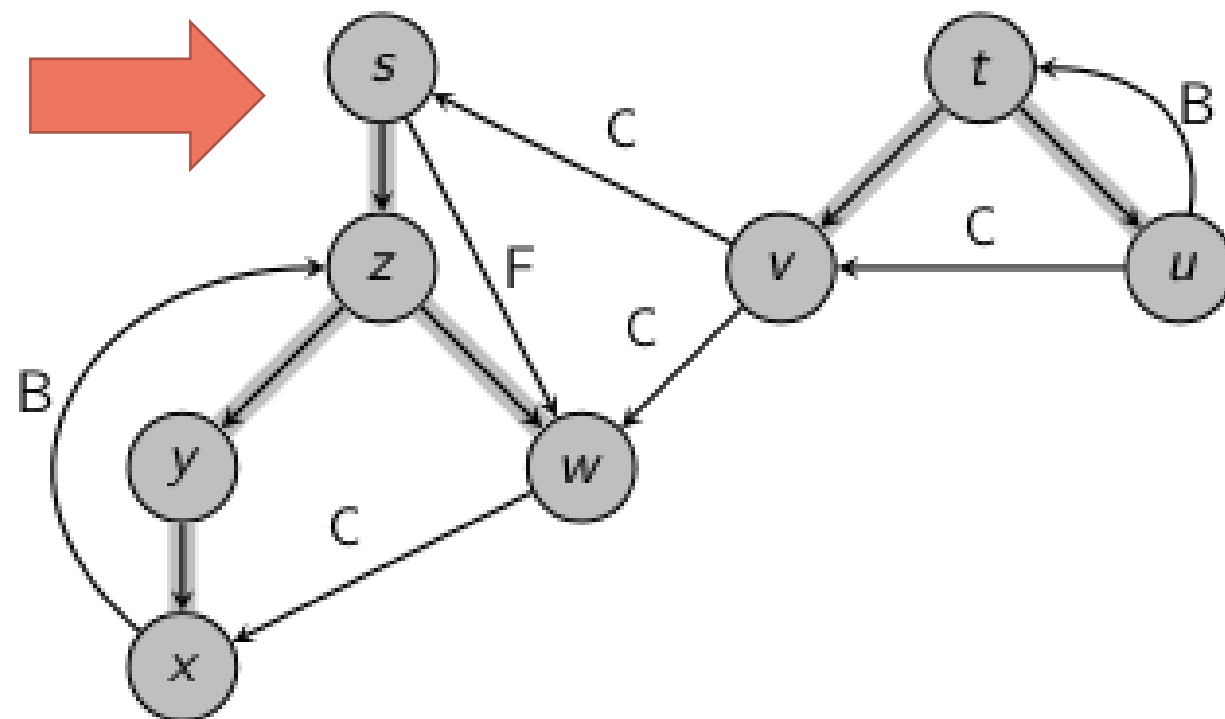
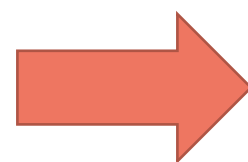
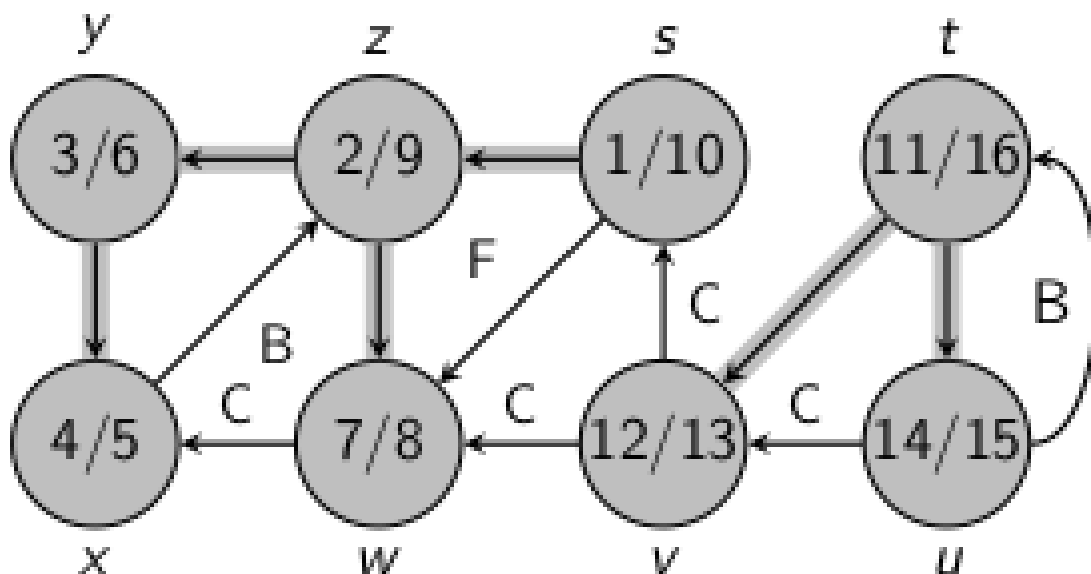


- Grafo G
- Vértice u
- Tempo = 12
- $u.d = 1; u.f = 8$
- $v.d = 2; v.f = 7$
- $y.d = 3; y.f = 6$
- $x.d = 4; x.f = 5$
- $w.d = 9; w.f = 12$
- $z.d = 10; z.f = 11$

Tipos de Aresta em uma DFS

- Podemos definir quatro tipos de arestas em termos da floresta em profundidade G produzida por uma busca em profundidade em G :
 1. **Arestas de árvores** são arestas que compõem a floresta em profundidade
 - A aresta $(u; v)$ é uma aresta de árvore se foi descoberta primeiro pela exploração da aresta $(u; v)$,
 2. **Arestas de retorno** (BACK) são as arestas $(u; v)$ que conectam um vértice u a um ancestral v em um árvore em profundidade.
 - Considera-se laços de um grafo dirigido como aresta de retorno
 3. **Arestas diretas** (FORWARD) são as arestas $(u; v)$ não pertencentes à árvore, que conectam um vértice u a um descendente v em uma árvore em profundidade
 4. **Arestas cruzadas** (CROSS) são todas as outras arestas. Elas podem estar entre vértices na mesma árvore, desde que um vértice não seja um ancestral do outro, ou podem estar entre vértices em diferentes árvores de busca

Tipos de Arestas



- Todas as arestas de **árvore** e **diretas** dirijam-se para baixo em uma árvore de profundidade
- Todas as arestas de retorno dirijam-se para cima

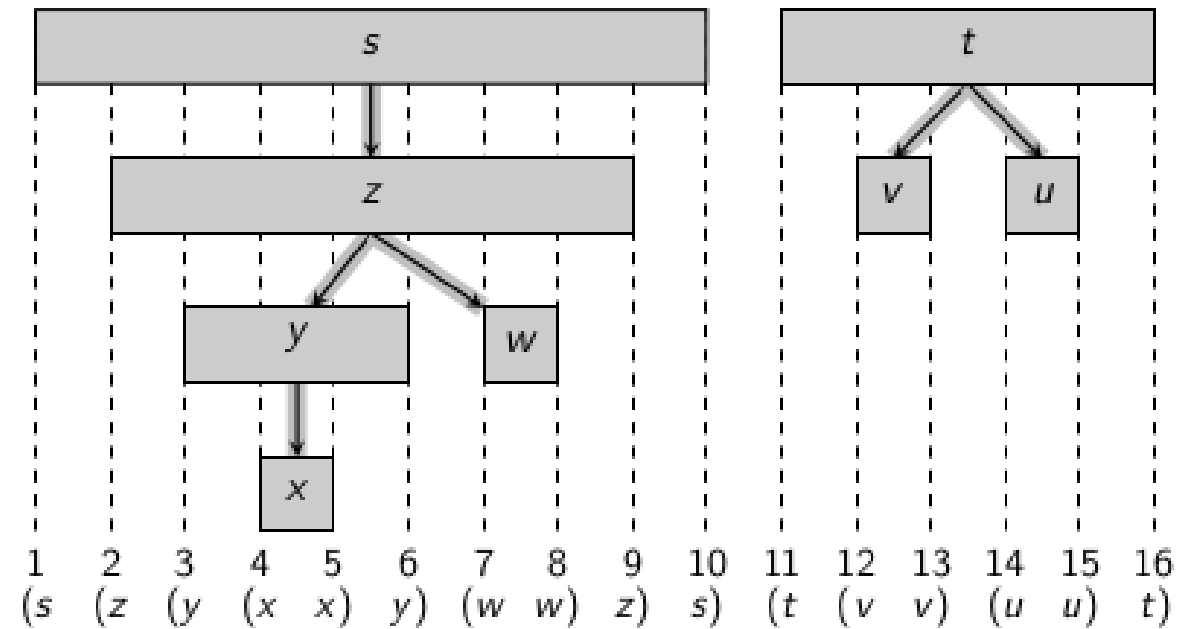
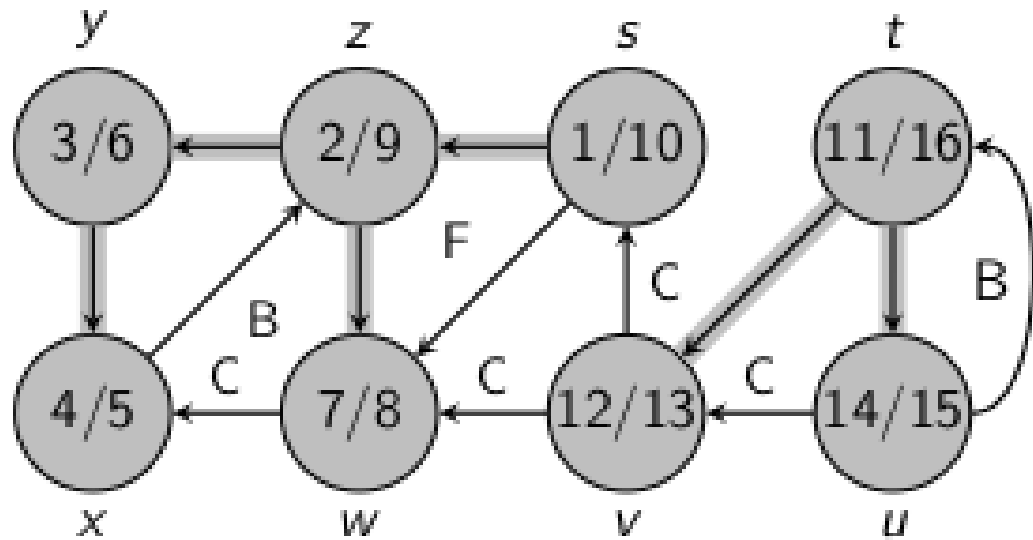
Tipos de Arestas e Algoritmo DFS

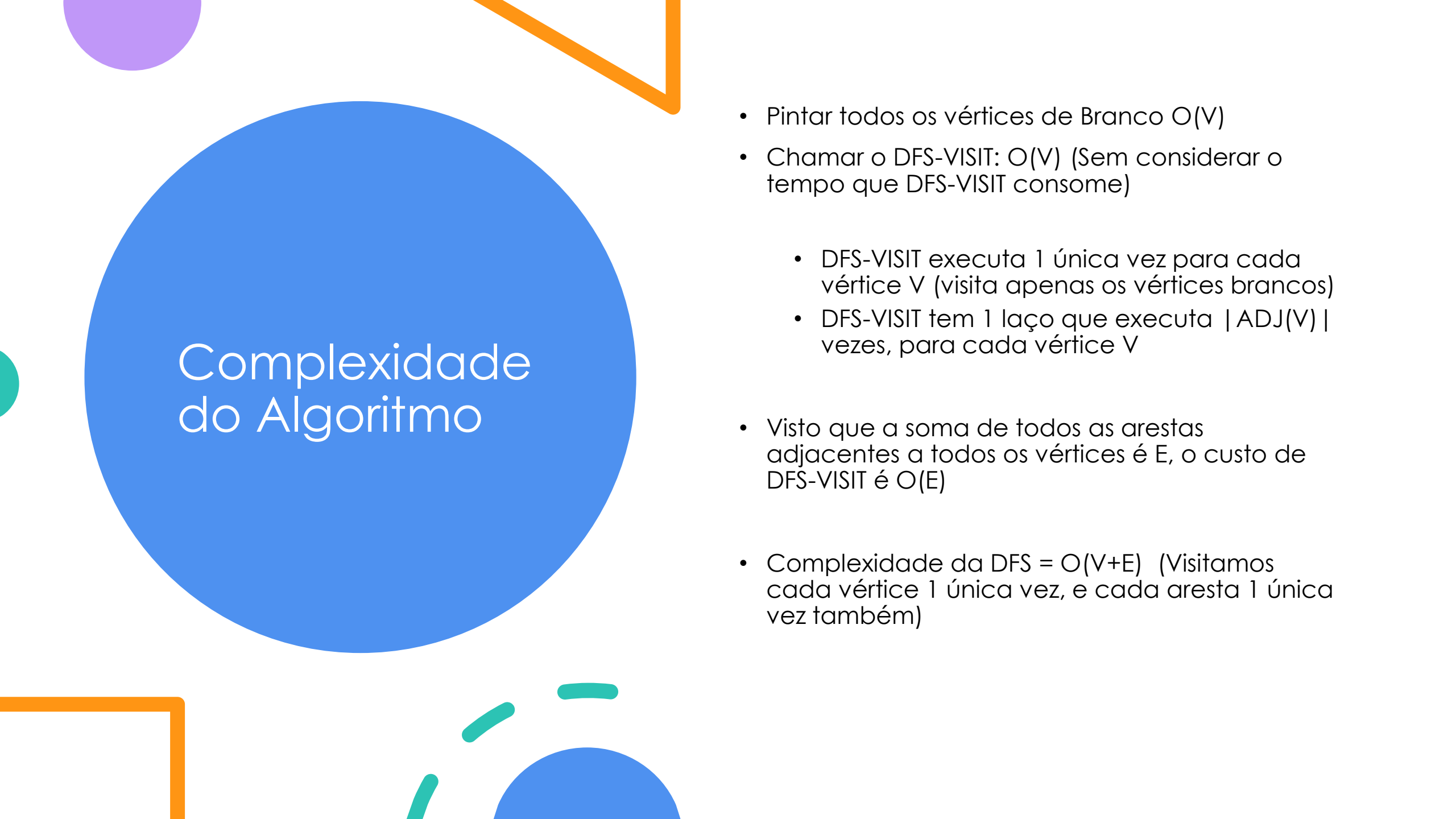
- O algoritmo DFS tem informações suficientes para classificar algumas arestas à medida que as encontra
- A ideia fundamental é que, quando explora-se uma aresta $(u; v)$ pela primeira vez, a cor do vértice de origem v diga algo sobre a aresta:
 - 1. BRANCO indica uma aresta de **árvore**
 - 2. CINZA indica uma aresta de **retorno**
 - 3. PRETO indica uma aresta **direta** ou **cruzada**

Timestamps e o Algoritmo DFS

- Os tempos de descoberta e finalização apresentam estrutura parentizada
- Se representarmos a descoberta do vértice u com um parêntese à esquerda " $(u$ " e representarmos seu término por um parênteses à direita " $u)$ ", o histórico de descobertas gera uma expressão bem formada
- Isso acontece porque a visitação nos vértices obedece uma PILHA (FILO)

Timestamps e o Algoritmo DFS





Complexidade do Algoritmo

- Pintar todos os vértices de Branco $O(V)$
- Chamar o DFS-VISIT: $O(V)$ (Sem considerar o tempo que DFS-VISIT consome)
 - DFS-VISIT executa 1 única vez para cada vértice V (visita apenas os vértices brancos)
 - DFS-VISIT tem 1 laço que executa $| \text{ADJ}(V) |$ vezes, para cada vértice V
- Visto que a soma de todas as arestas adjacentes a todos os vértices é E , o custo de DFS-VISIT é $O(E)$
- Complexidade da DFS = $O(V+E)$ (Visitamos cada vértice 1 única vez, e cada aresta 1 única vez também)



Aplicações

- Encontrar caminhos entre 2 vértices
- Sinalizar a presença de ciclos*
- Em um grafo não-valorado, encontra uma **árvore geradora mínima**
- **Ordenação Topológica**
- **Componentes Fortemente Conexas**