

# Grafos – Busca em Profundidade - DFS

Estrutura de Dados Avançada — QXD0015



UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS QUIXADÁ

Prof. Atílio Gomes Luiz  
gomes.atilio@ufc.br

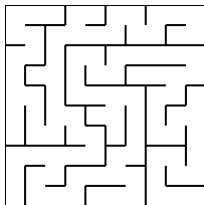
Universidade Federal do Ceará

1º semestre/2024

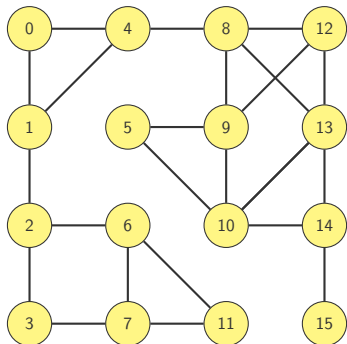


# Busca em Profundidade

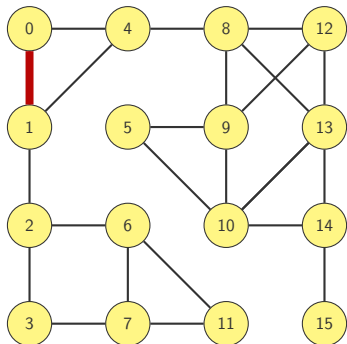
- Ordem de exploração: explorar vértices descobertos mais recentes primeiro.
- **Interpretação:**
  - Estamos tentando procurar a saída de um labirinto.
  - Vamos fundo até a saída, tomando decisões a cada encruzilhada.
  - Voltamos à última encruzilhada quando encontramos um beco sem saída (ou um lugar já visitado)



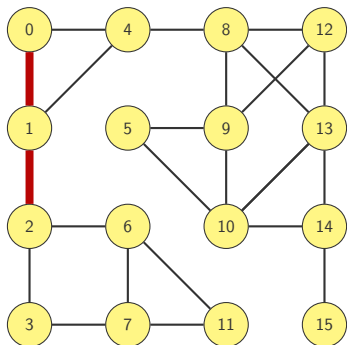
Exemplo - Existe caminho de 0 até 15?



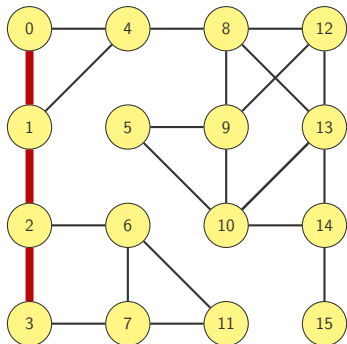
Exemplo - Existe caminho de 0 até 15?



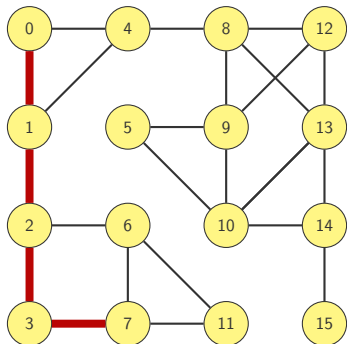
Exemplo - Existe caminho de 0 até 15?



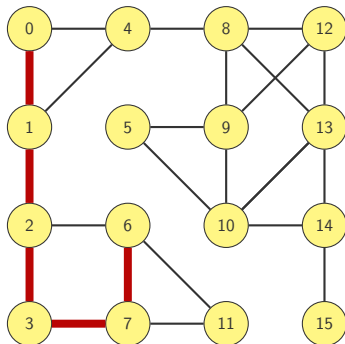
Exemplo - Existe caminho de 0 até 15?



Exemplo - Existe caminho de 0 até 15?

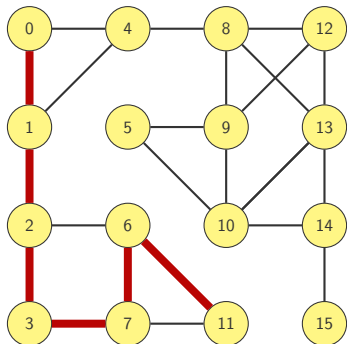


Exemplo - Existe caminho de 0 até 15?

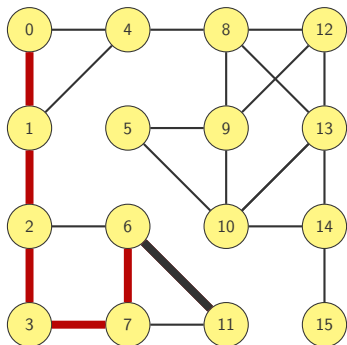




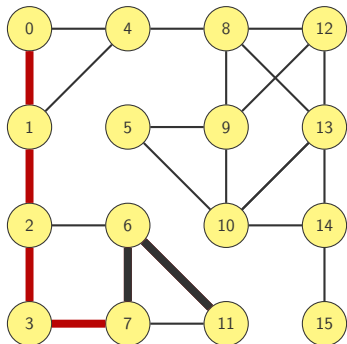
Exemplo - Existe caminho de 0 até 15?



Exemplo - Existe caminho de 0 até 15?

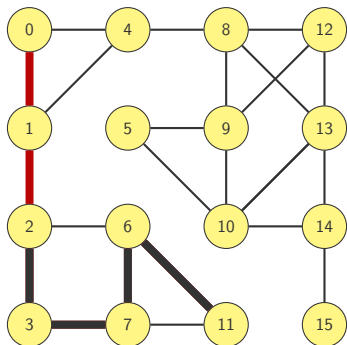


Exemplo - Existe caminho de 0 até 15?

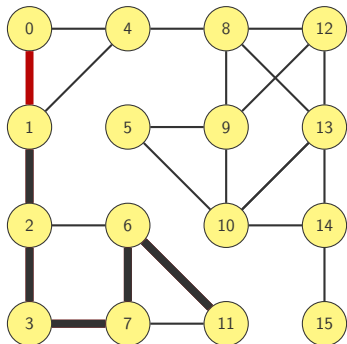




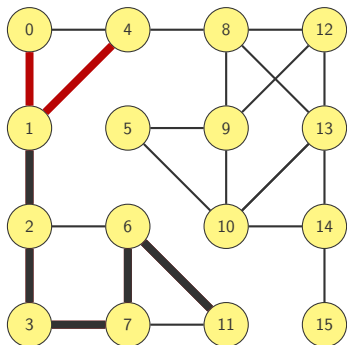
Exemplo - Existe caminho de 0 até 15?



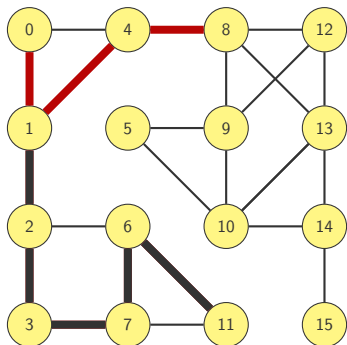
Exemplo - Existe caminho de 0 até 15?



Exemplo - Existe caminho de 0 até 15?

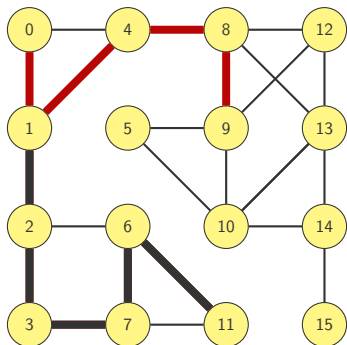


Exemplo - Existe caminho de 0 até 15?

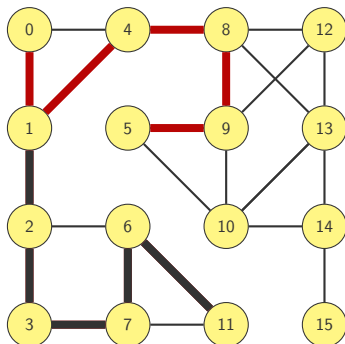




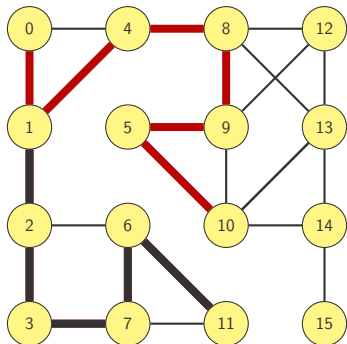
Exemplo - Existe caminho de 0 até 15?



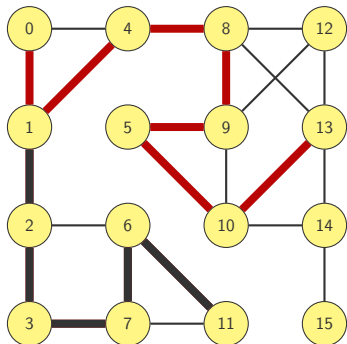
Exemplo - Existe caminho de 0 até 15?



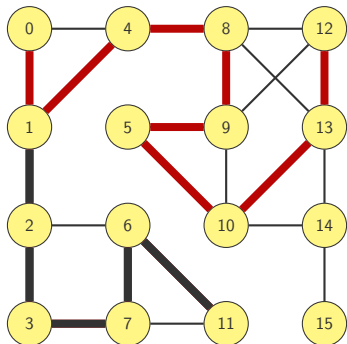
Exemplo - Existe caminho de 0 até 15?



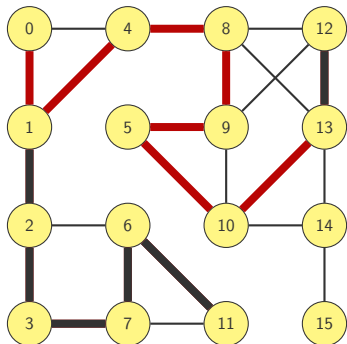
Exemplo - Existe caminho de 0 até 15?



Exemplo - Existe caminho de 0 até 15?



Exemplo - Existe caminho de 0 até 15?



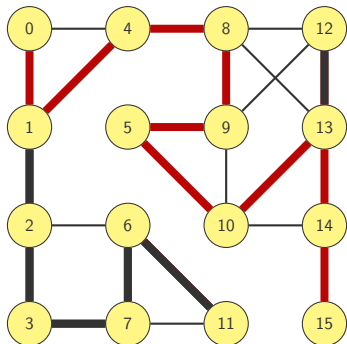








## Exemplo - Existe caminho de 0 até 15?

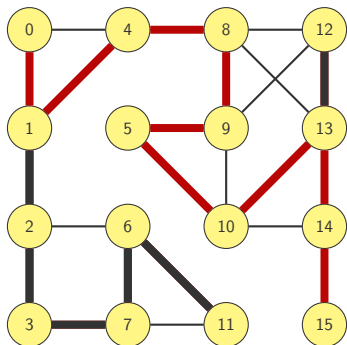


Essa é uma **busca em profundidade**:

- Vá o máximo possível em uma direção



### Exemplo - Existe caminho de 0 até 15?

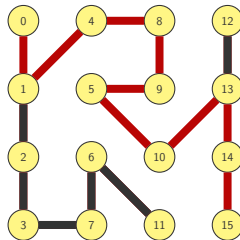
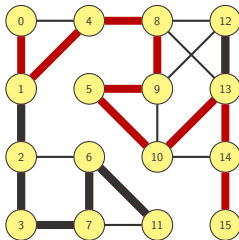


Essa é uma busca em profundidade:

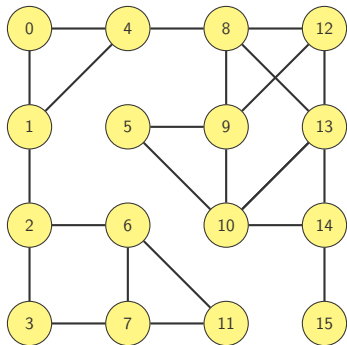
- Vá o máximo possível em uma direção
- Se não encontrar o vértice, volte o mínimo possível e pegue um novo caminho por um vértice não visitado
- Note que as arestas que foram visitadas formam uma árvore.  
Elas poderiam ter formado uma floresta?

# Árvore de Profundidade

- **Árvore de Profundidade:** é árvore induzida pela busca em profundidade em uma componente conexa de um grafo  $G$ .
  - **Raiz da árvore:** vértice inicial da busca.
  - **Pai de  $v \in V(G)$ :** nó que levou à descoberta de  $v$ .
- Vértice inicial e ordem dos vizinhos variam e podem levar a árvores distintas.



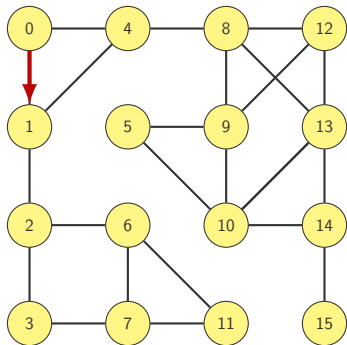
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL															

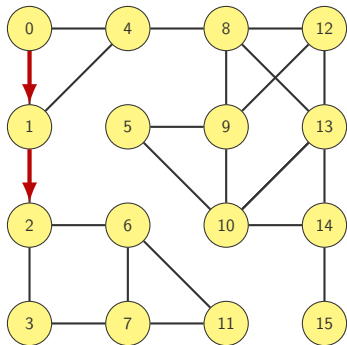
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0														

## Determinando o pai de um nó na árvore DFS

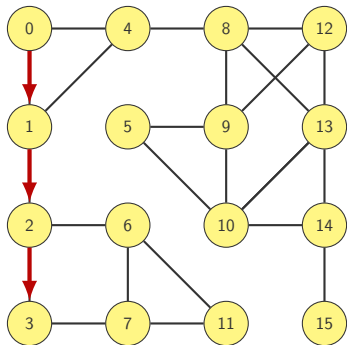


Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1													



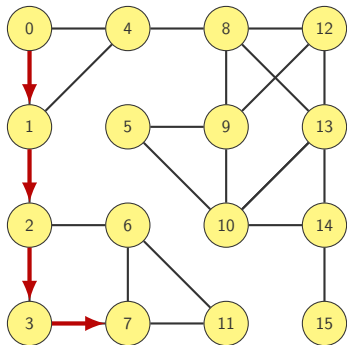
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2												

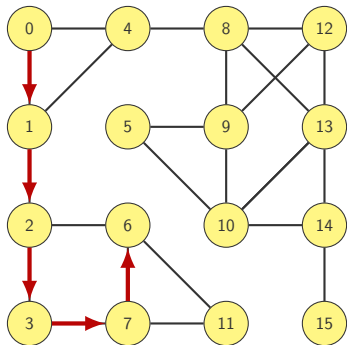
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2				3								

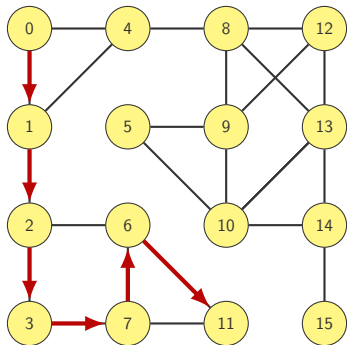
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3								

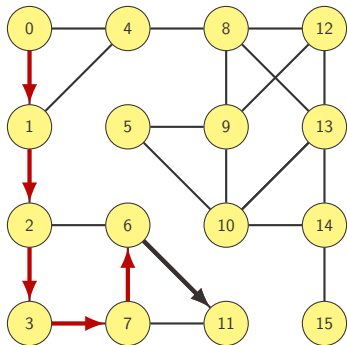
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3				6				

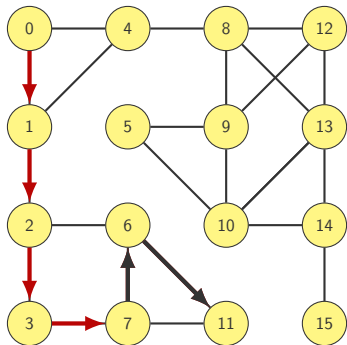
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3				6				

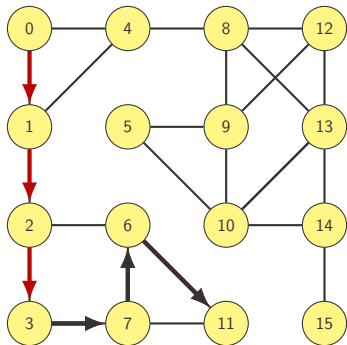
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3				6				

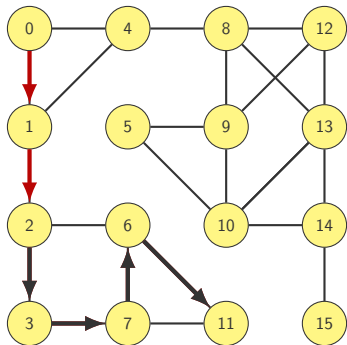
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3				6				

## Determinando o pai de um nó na árvore DFS

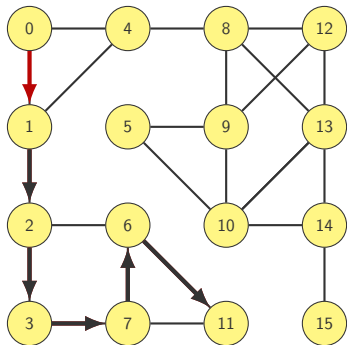


Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3				6				



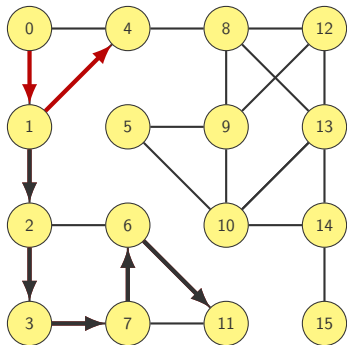
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2			7	3				6				

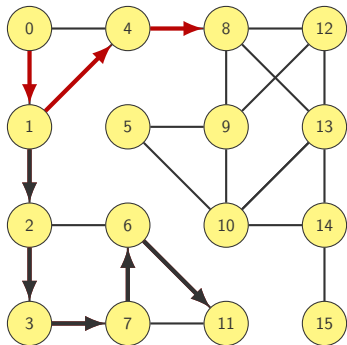
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1		7	3				6				

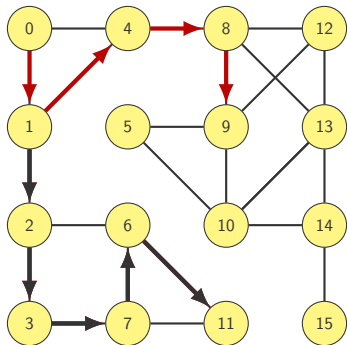
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1		7	3	4			6				

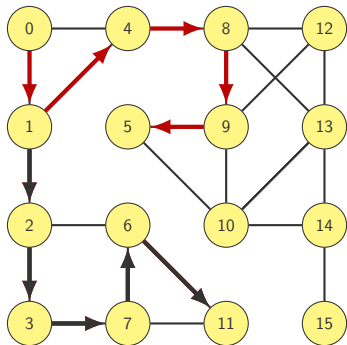
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1		7	3	4	8		6				

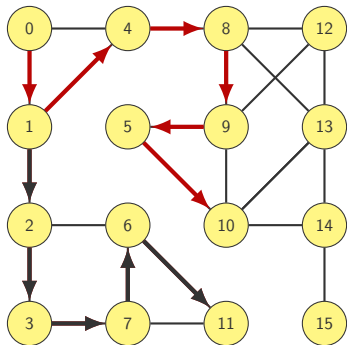
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8		6				

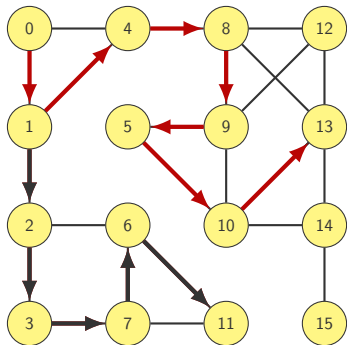
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6				

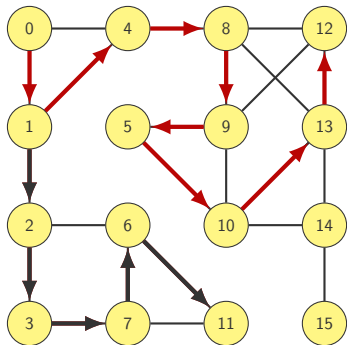
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6		10		

## Determinando o pai de um nó na árvore DFS

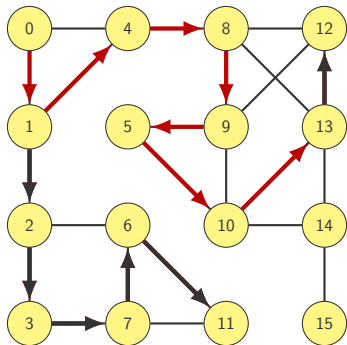


Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6	13	10		



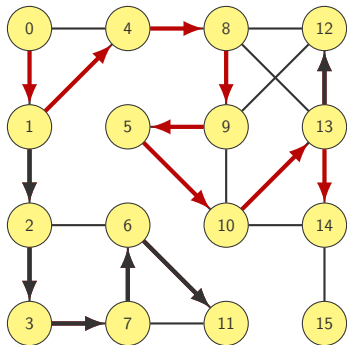
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6	13	10		

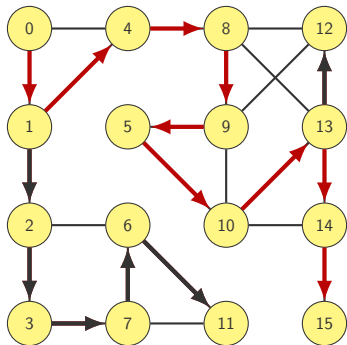
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6	13	10	13	

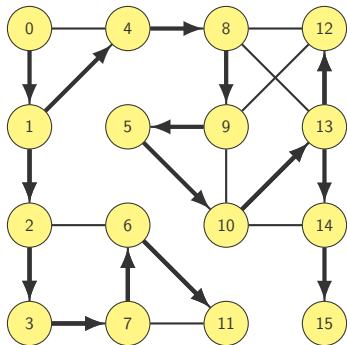
## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6	13	10	13	14

## Determinando o pai de um nó na árvore DFS



Guardamos o pai de cada nó na árvore no vetor de predecessores  $\pi$ .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NIL	0	1	2	1	9	7	3	4	8	5	6	13	10	13	14

## Como implementar a busca em profundidade?

- Note que o DFS volta atrás (backtracks) quando encontra um “beco sem saída”, ou seja, quando encontra um vértice sem arestas a serem exploradas.
  - Ele volta até encontrar o primeiro vértice descoberto que tem vizinhos ainda não descobertos.

## Como implementar a busca em profundidade?

- Note que o DFS volta atrás (backtracks) quando encontra um “beco sem saída”, ou seja, quando encontra um vértice sem arestas a serem exploradas.
  - Ele volta até encontrar o primeiro vértice descoberto que tem vizinhos ainda não descobertos.
- Logo, o DFS pode ser projetado como um **algoritmo recursivo**.
  - ou como um algoritmo iterativo usando pilha.

## Cores dos vértices

De novo, vamos pintar o grafo durante a busca:

- $cor[v] = \text{branco}$  se não descobrimos  $v$  ainda.

# Cores dos vértices

**De novo, vamos pintar o grafo durante a busca:**

- $\text{cor}[v] = \text{branco}$  se não descobrimos  $v$  ainda.
- $\text{cor}[v] = \text{cinza}$  se já descobrimos, mas não finalizamos  $v$ .



# Cores dos vértices

**De novo, vamos pintar o grafo durante a busca:**

- $\text{cor}[v] = \text{branco}$  se não descobrimos  $v$  ainda.
- $\text{cor}[v] = \text{cinza}$  se já descobrimos, mas não finalizamos  $v$ .
- $\text{cor}[v] = \text{preto}$  se já descobrimos e já finalizamos  $v$ .

# Cores dos vértices

**De novo, vamos pintar o grafo durante a busca:**

- $\text{cor}[v] = \text{branco}$  se não descobrimos  $v$  ainda.
- $\text{cor}[v] = \text{cinza}$  se já descobrimos, mas não finalizamos  $v$ .
- $\text{cor}[v] = \text{preto}$  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

**A busca em profundidade associa dois rótulos aos vértices:**

- $d[u]$ : instante da **descoberta** de  $u$ .
- $f[u]$ : instante da **finalização** de  $u$  (completamente explorado.)

# Tempo de descoberta e finalização

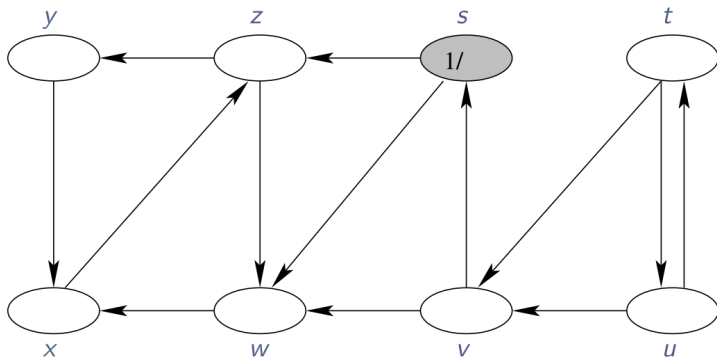
**A busca em profundidade associa dois rótulos aos vértices:**

- $d[u]$ : instante da **descoberta** de  $u$ .
- $f[u]$ : instante da **finalização** de  $u$  (completamente explorado.)

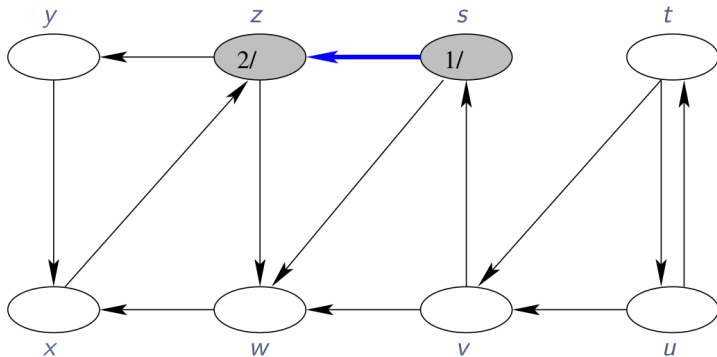
Observações

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

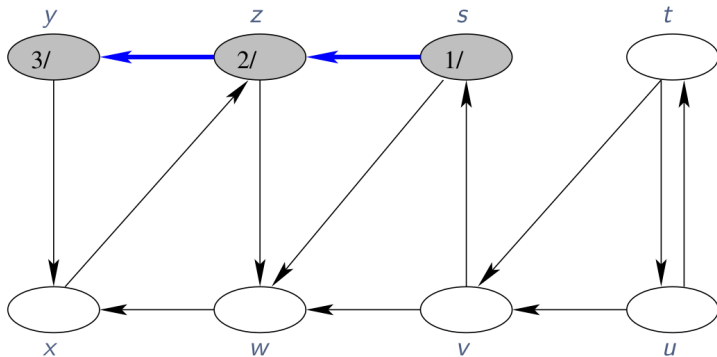
## Exemplo de busca em profundidade



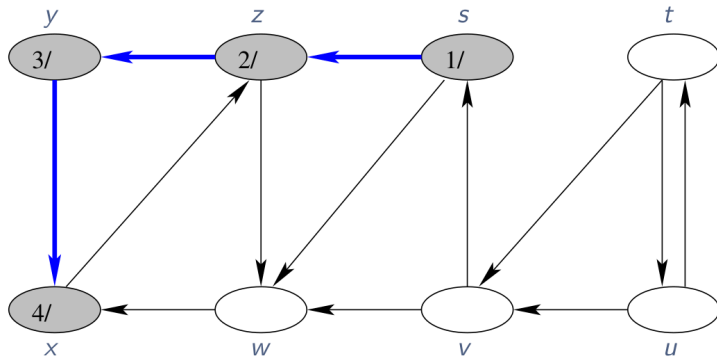
## Exemplo de busca em profundidade



## Exemplo de busca em profundidade

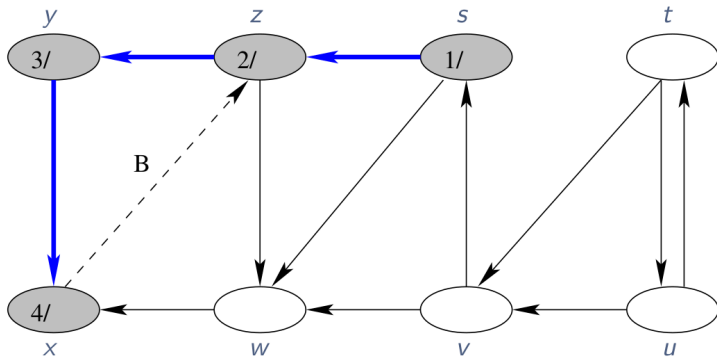


## Exemplo de busca em profundidade

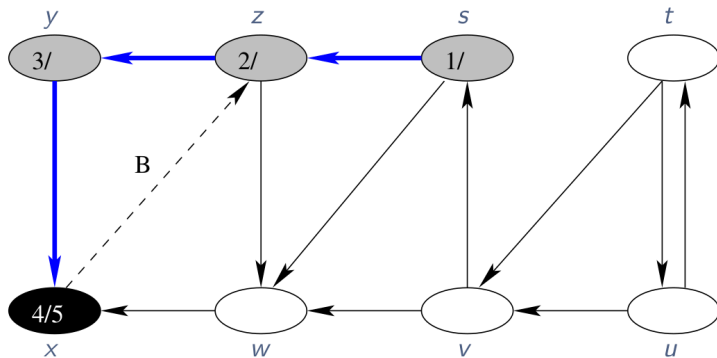




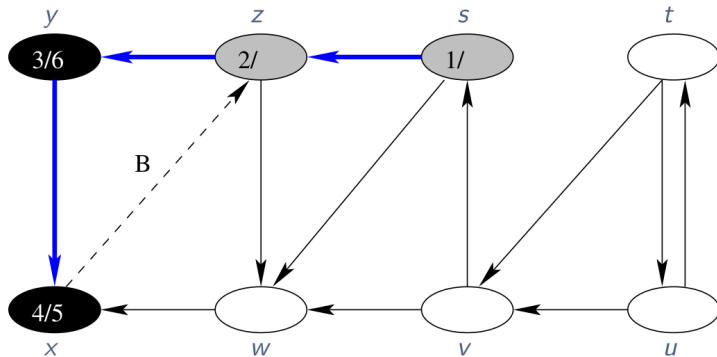
## Exemplo de busca em profundidade



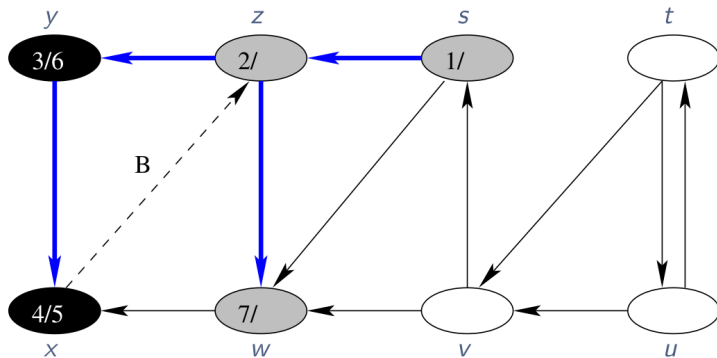
## Exemplo de busca em profundidade



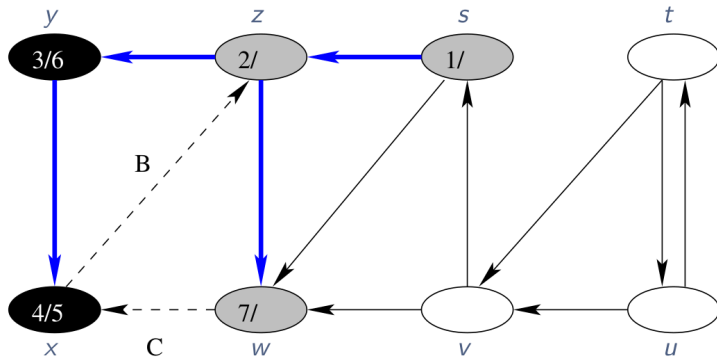
## Exemplo de busca em profundidade



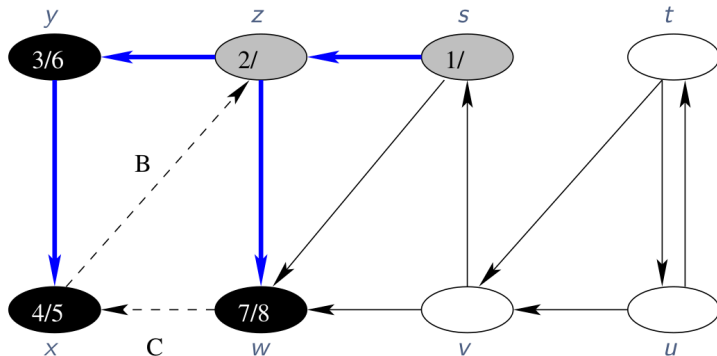
## Exemplo de busca em profundidade



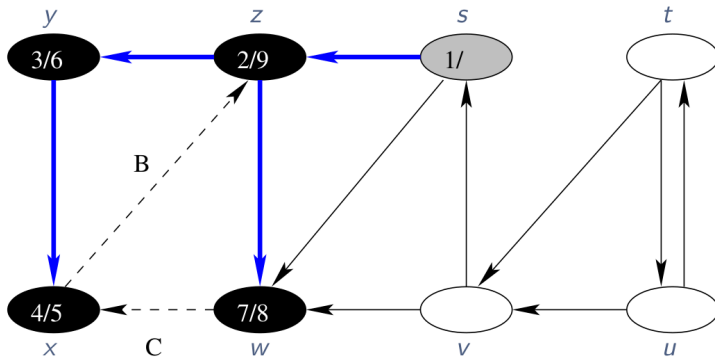
## Exemplo de busca em profundidade



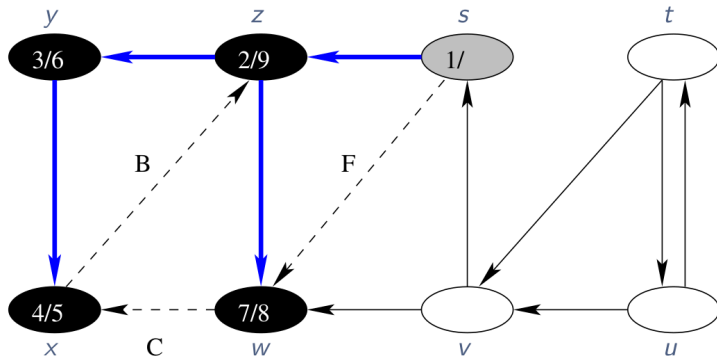
## Exemplo de busca em profundidade



## Exemplo de busca em profundidade

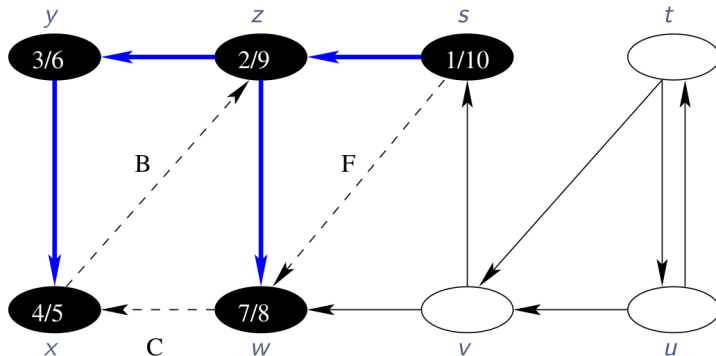


## Exemplo de busca em profundidade

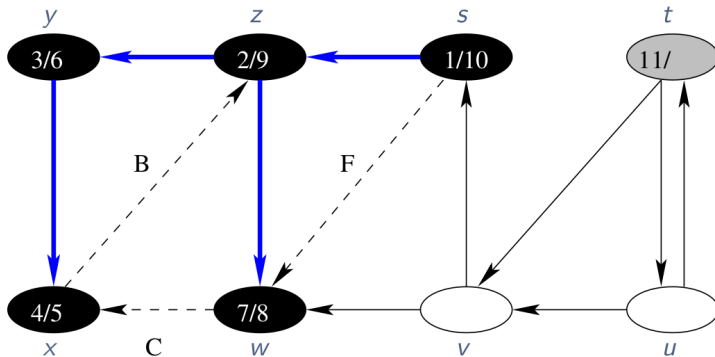




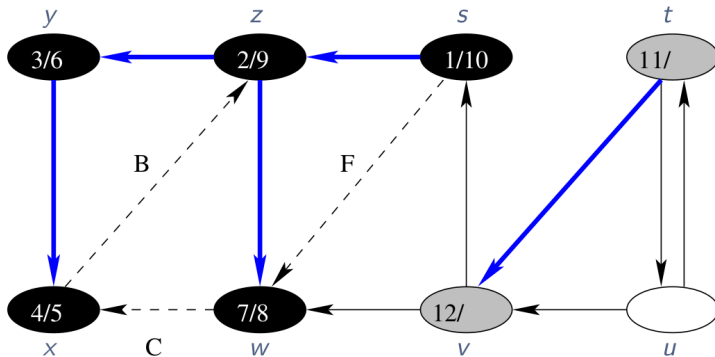
## Exemplo de busca em profundidade



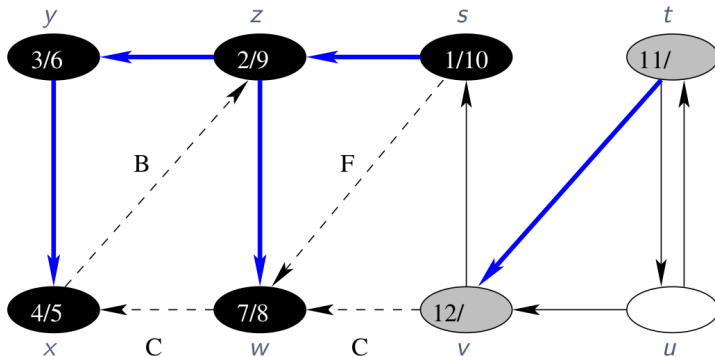
## Exemplo de busca em profundidade



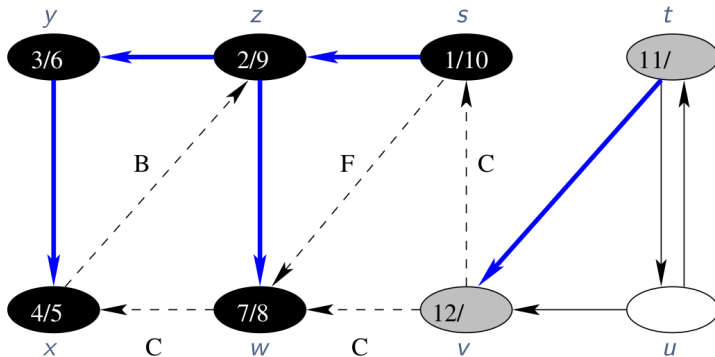
## Exemplo de busca em profundidade



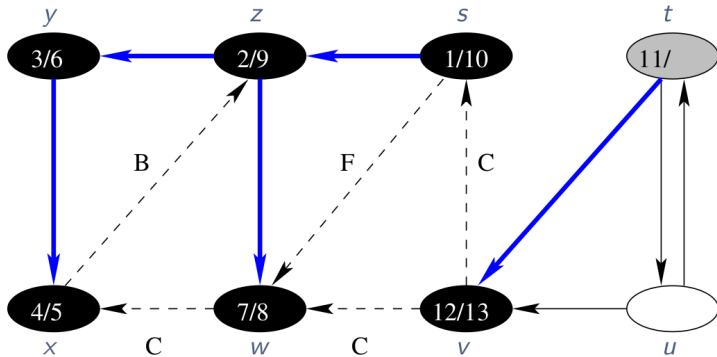
## Exemplo de busca em profundidade



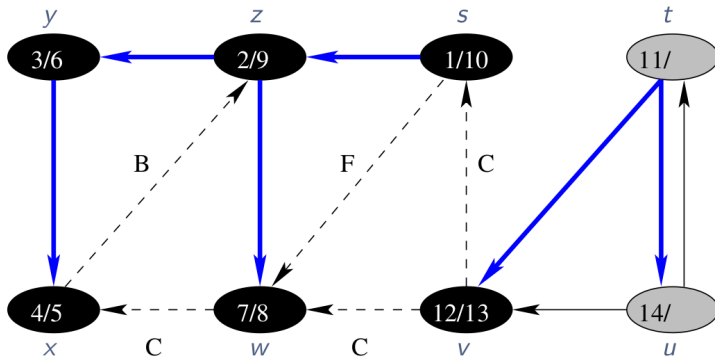
## Exemplo de busca em profundidade



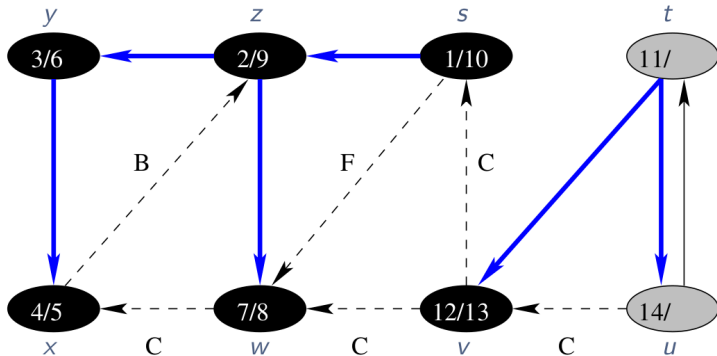
## Exemplo de busca em profundidade



## Exemplo de busca em profundidade

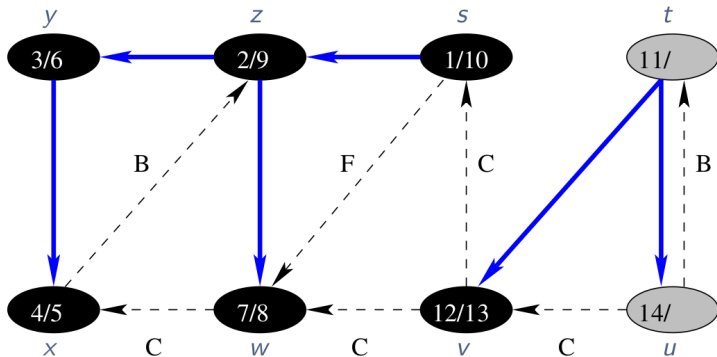


## Exemplo de busca em profundidade





## Exemplo de busca em profundidade



# Rótulos versus cores

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

# Busca em Profundidade

## DFS(G)

1. **para cada**  $u \in V(G)$  **faça**
2.      $cor[u] \leftarrow \text{BRANCO}$
3.      $\pi[u] \leftarrow \text{NIL}$
4.  $\text{tempo} \leftarrow 0$
5. **para cada**  $u \in V(G)$  **faça**
6.     **se**  $cor[u] == \text{BRANCO}$
7.         DFS-VISIT( $G, u$ )

# Busca em Profundidade

## DFS(G)

1. **para cada**  $u \in V(G)$  **faça**
2.      $cor[u] \leftarrow \text{BRANCO}$
3.      $\pi[u] \leftarrow \text{NIL}$
4.  $tempo \leftarrow 0$
5. **para cada**  $u \in V(G)$  **faça**
6.     **se**  $cor[u] == \text{BRANCO}$
7.         DFS-VISIT(G,u)

## DFS-VISIT(G,u)

1.  $cor[u] \leftarrow \text{CINZA}$
2.  $tempo \leftarrow tempo + 1$
3.  $d[u] \leftarrow tempo$
4. **para cada**  $v \in Adj[u]$  **faça**
5.     **se**  $cor[v] == \text{BRANCO}$
6.          $\pi[v] \leftarrow u$
7.         DFS-VISIT(G,v)
8.  $cor[u] \leftarrow \text{PRETO}$
9.  $tempo \leftarrow tempo + 1$
10.  $f[u] \leftarrow tempo$

# Busca em Profundidade

## DFS(G)

1. **para cada**  $u \in V(G)$  **faça**
2.      $cor[u] \leftarrow \text{BRANCO}$
3.      $\pi[u] \leftarrow \text{NIL}$
4.  $tempo \leftarrow 0$
5. **para cada**  $u \in V(G)$  **faça**
6.     **se**  $cor[u] == \text{BRANCO}$
7.         DFS-VISIT(G,u)

## DFS-VISIT(G,u)

1.  $cor[u] \leftarrow \text{CINZA}$
2.  $tempo \leftarrow tempo + 1$
3.  $d[u] \leftarrow tempo$
4. **para cada**  $v \in Adj[u]$  **faça**
5.     **se**  $cor[v] == \text{BRANCO}$
6.          $\pi[v] \leftarrow u$
7.         DFS-VISIT(G,v)
8.  $cor[u] \leftarrow \text{PRETO}$
9.  $tempo \leftarrow tempo + 1$
10.  $f[u] \leftarrow tempo$

Qual a complexidade de tempo deste algoritmo?

# Análise de complexidade

Tempo do algoritmo principal DFS

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT



# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT

Tempo da sub-rotina DFS-VISIT

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT

Tempo da sub-rotina DFS-VISIT

- DFS-VISIT é chamado cada vértice exatamente uma vez

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT

Tempo da sub-rotina DFS-VISIT

- DFS-VISIT é chamado cada vértice exatamente uma vez
- cada chamada percorre sua lista de adjacências

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT

Tempo da sub-rotina DFS-VISIT

- DFS-VISIT é chamado cada vértice exatamente uma vez
- cada chamada percorre sua lista de adjacências
- o tempo gasto percorrendo adjacências é  $O(E)$

# Análise de complexidade

Tempo do algoritmo principal DFS

- a inicialização consome tempo  $O(V)$
- realizamos  $|V|$  chamadas a DFS-VISIT

Tempo da sub-rotina DFS-VISIT

- DFS-VISIT é chamado 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  $O(V + E)$ .

# Propriedades da DFS



# Estrutura de parênteses balanceados

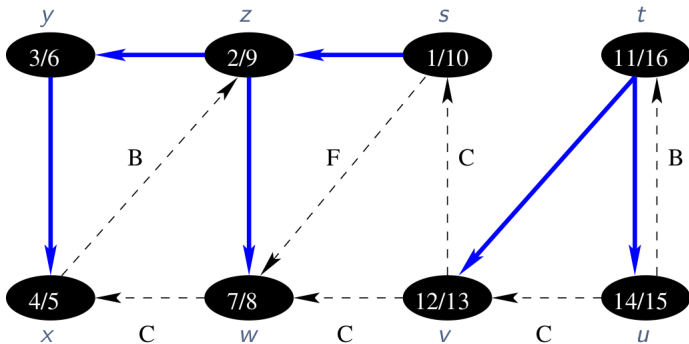
## Teorema dos parênteses:

Se  $u$  e  $v$  são vértices de uma árvore de busca em profundidade, então ocorre exatamente um entre os três casos abaixo:

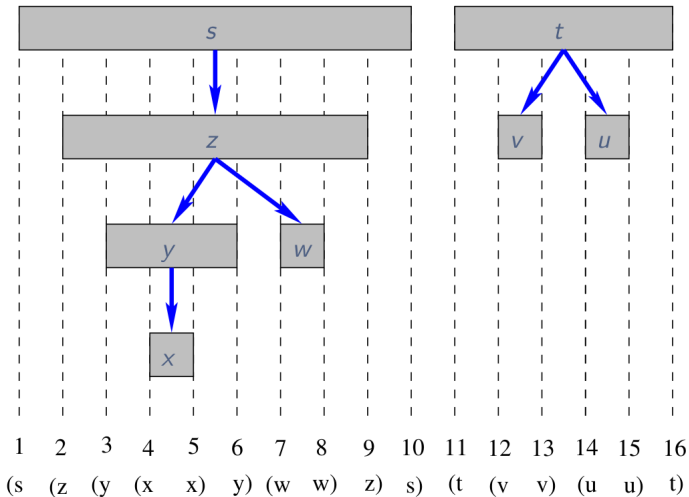
- (a) Os intervalos  $[d[u], f[u]]$  e  $[d[v], f[v]]$  são disjuntos, e nem  $u$  é descendente de  $v$ , nem  $v$  é descendente de  $u$ .
- (b) O intervalo  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e  $v$  é descendente de  $u$ .
- (c) O intervalo  $[d[u], f[u]]$  está contido em  $[d[v], f[v]]$ , e  $u$  é descendente de  $v$ .



## Exemplo de floresta de busca



## Exemplo de estrutura de parênteses



## Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

## Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$

## Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$



# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$
- neste caso,  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e o resultado segue.

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$
- neste caso,  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e o resultado segue.

**Caso 2:** suponha que  $f[u] < d[v]$

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$
- neste caso,  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e o resultado segue.

**Caso 2:** suponha que  $f[u] < d[v]$

- então  $u$  foi finalizado enquanto  $v$  era branco

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$
- neste caso,  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e o resultado segue.

**Caso 2:** suponha que  $f[u] < d[v]$

- então  $u$  foi finalizado enquanto  $v$  era branco
- e a chamada de  $u$  termina antes que a de  $v$  comece

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$
- neste caso,  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e o resultado segue.

**Caso 2:** suponha que  $f[u] < d[v]$

- então  $u$  foi finalizado enquanto  $v$  era branco
- e a chamada de  $u$  termina antes que a de  $v$  comece
- portanto  $u$  e  $v$  não são descendentes um do outro

# Demonstração do teorema dos parênteses

Sejam  $u$  e  $v$  dois vértices de uma árvore de busca em profundidade.

- podemos supor que  $d[u] < d[v]$
- analisamos dois casos

**Caso 1:** suponha que  $d[v] < f[u]$

- então  $v$  foi descoberto enquanto  $u$  era cinza
- e a chamada recursiva para  $v$  termina antes da de  $u$
- portanto  $v$  é descendente de  $u$
- neste caso,  $[d[v], f[v]]$  está contido em  $[d[u], f[u]]$ , e o resultado segue.

**Caso 2:** suponha que  $f[u] < d[v]$

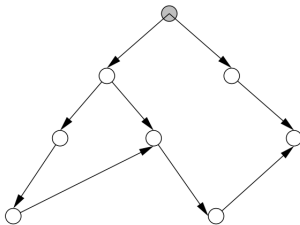
- então  $u$  foi finalizado enquanto  $v$  era branco
- e a chamada de  $u$  termina antes que a de  $v$  comece
- portanto  $u$  e  $v$  não são descendentes um do outro
- neste caso,  $[d[v], f[v]]$  e  $[d[u], f[u]]$  são disjuntos, e o resultado segue.

# Vértices alcançáveis

## Teorema do caminho branco

Considere dois vértices  $u$  e  $v$  de um grafo  $G$  sobre o qual DFS foi executado. As afirmações a seguir são equivalentes:

- (1)  $v$  é descendente de  $u$  na floresta de busca
- (2) quando  $u$  foi descoberto, existia um caminho de  $u$  a  $v$  formado apenas por vértices brancos





## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$

## Prova do teorema do caminho branco

- (1)  $\Rightarrow$  (2)
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$ 
  - considere um caminho branco de  $u$  a  $v$  no instante  $d[u]$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$ 
  - considere um caminho branco de  $u$  a  $v$  no instante  $d[u]$
  - suponha que todo vértice no caminho virou descendente de  $u$ , com exceção do vértice  $v$



## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$ 
  - considere um caminho branco de  $u$  a  $v$  no instante  $d[u]$
  - suponha que todo vértice no caminho virou descendente de  $u$ , com exceção do vértice  $v$
  - seja  $w$  o vértice antecessor de  $v$  nesse caminho ( $w$  pode ser o próprio  $u$ )

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$ 
  - considere um caminho branco de  $u$  a  $v$  no instante  $d[u]$
  - suponha que todo vértice no caminho virou descendente de  $u$ , com exceção do vértice  $v$
  - seja  $w$  o vértice antecessor de  $v$  nesse caminho ( $w$  pode ser o próprio  $u$ )
  - como  $w$  é descendente de  $u$ , temos  $f[w] \leq f[u]$  (T.P.)

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$ 
  - considere um caminho branco de  $u$  a  $v$  no instante  $d[u]$
  - suponha que todo vértice no caminho virou descendente de  $u$ , com exceção do vértice  $v$
  - seja  $w$  o vértice antecessor de  $v$  nesse caminho ( $w$  pode ser o próprio  $u$ )
  - como  $w$  é descendente de  $u$ , temos  $f[w] \leq f[u]$  (T.P.)
  - como  $v$  é descoberto depois de  $u$  ser descoberto e antes de  $w$  ser finalizado, temos  $d[u] < d[v] < f[w] \leq f[u]$

## Prova do teorema do caminho branco

- $(1) \Rightarrow (2)$ 
  - suponha que  $v$  é um descendente de  $u$
  - pelo Teorema dos Parênteses (T.P.),  $d[u] < d[v]$
  - portanto,  $v$  é branco no tempo  $d[u]$
  - Como  $v$  pode ser qualquer descendente de  $u$ , todos os vértices no caminho de  $u$  a  $v$  na árvore de busca em profundidade eram brancos no tempo  $d[u]$
- $(2) \Rightarrow (1)$ 
  - considere um caminho branco de  $u$  a  $v$  no instante  $d[u]$
  - suponha que todo vértice no caminho virou descendente de  $u$ , com exceção do vértice  $v$
  - seja  $w$  o vértice antecessor de  $v$  nesse caminho ( $w$  pode ser o próprio  $u$ )
  - como  $w$  é descendente de  $u$ , temos  $f[w] \leq f[u]$  (T.P.)
  - como  $v$  é descoberto depois de  $u$  ser descoberto e antes de  $w$  ser finalizado, temos  $d[u] < d[v] < f[w] \leq f[u]$
  - Pelo T.P. temos que  $[d[v], f[v]]$  está inteiramente contido no intervalo  $[d[u], f[u]]$ , e  $v$  é descendente de  $u$ , contradição.

# Classificação das arestas em digrafos

Quatro tipos de arestas derivadas de uma DFS:

1. **Aresta de árvore:** aresta da floresta DFS

# Classificação das arestas em digrafos

Quatro tipos de arestas derivadas de uma DFS:

1. **Aresta de árvore:** aresta da floresta DFS
2. **Aresta de retorno:** de um vértice para um ancestral na floresta DFS.

# Classificação das arestas em digrafos

Quatro tipos de arestas derivadas de uma DFS:

1. **Aresta de árvore:** aresta da floresta DFS
2. **Aresta de retorno:** de um vértice para um ancestral na floresta DFS.
3. **Aresta de avanço:** de um vértice para um descendente na floresta DFS.

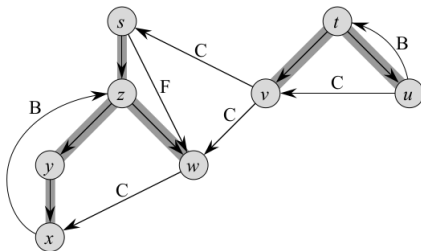
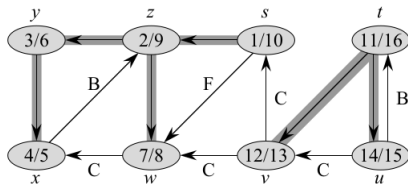
# Classificação das arestas em digrafos

Quatro tipos de arestas derivadas de uma DFS:

1. **Aresta de árvore:** aresta da floresta DFS
2. **Aresta de retorno:** de um vértice para um ancestral na floresta DFS.
3. **Aresta de avanço:** de um vértice para um descendente na floresta DFS.
4. **Arestas cruzadas:** todas as outras arestas.



## Classificação das arestas em digrafos



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

## DFS em grafos não direcionados

Classificando arestas em grafos não direcionados

# DFS em grafos não direcionados

Classificando arestas em grafos não direcionados

- não pode haver aresta de avanço (por quê?)

# DFS em grafos não direcionados

Classificando arestas em grafos não direcionados

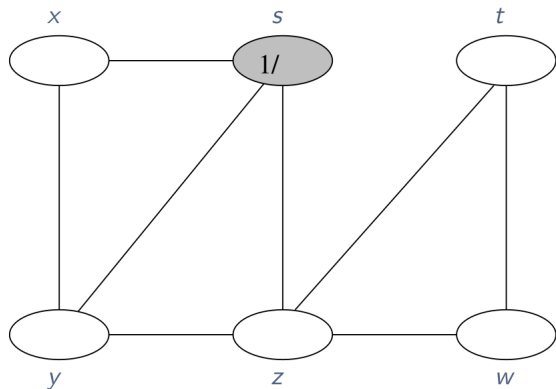
- não pode haver aresta de avanço (por quê?)
- tampouco aresta de cruzamento (por quê?)

# DFS em grafos não direcionados

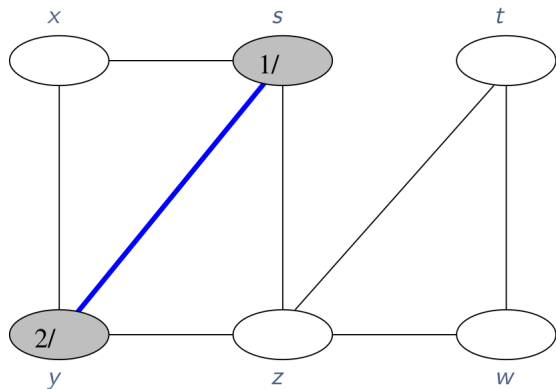
Classificando arestas em grafos não direcionados

- não pode haver aresta de avanço (por quê?)
- tampouco aresta de cruzamento (por quê?)
- daí cada aresta é **aresta de árvore** ou **aresta de retorno**

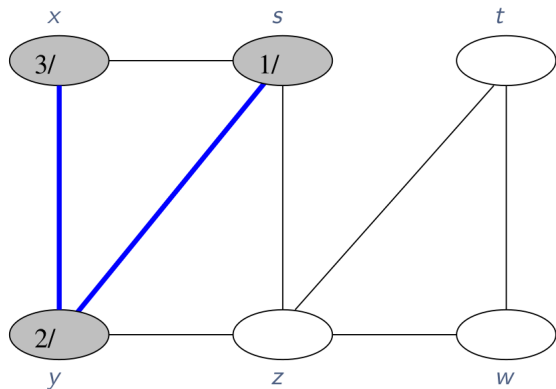
## DFS em grafos não direcionados



## DFS em grafos não direcionados

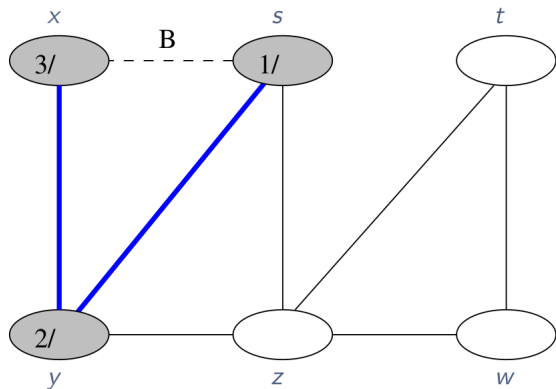


## DFS em grafos não direcionados

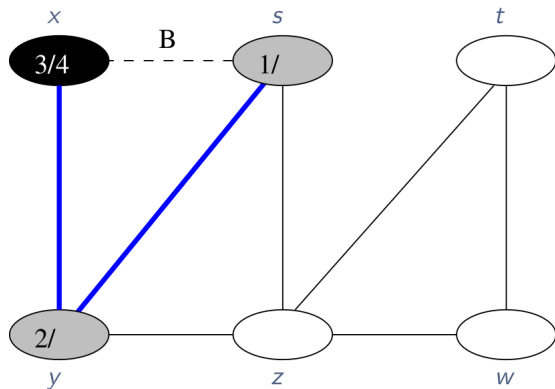




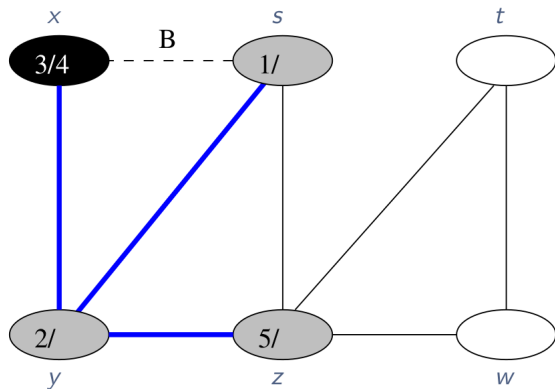
## DFS em grafos não direcionados



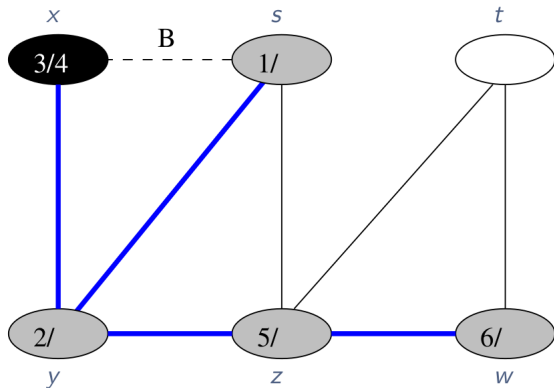
## DFS em grafos não direcionados



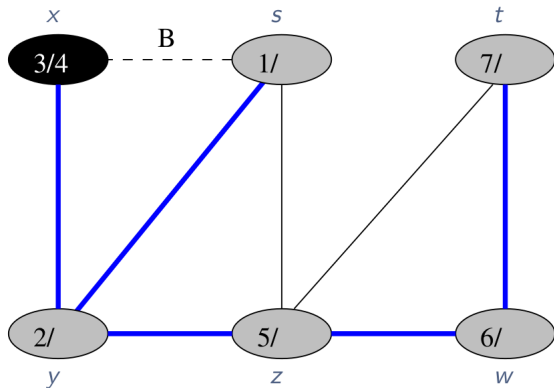
## DFS em grafos não direcionados



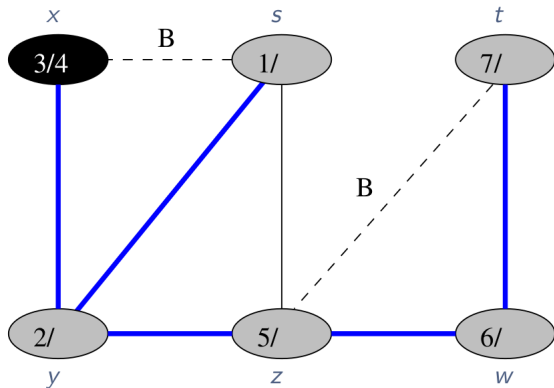
## DFS em grafos não direcionados



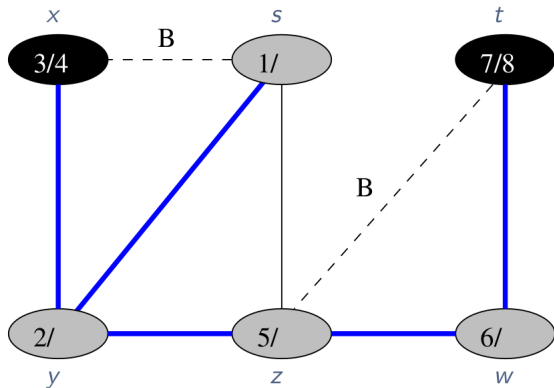
## DFS em grafos não direcionados



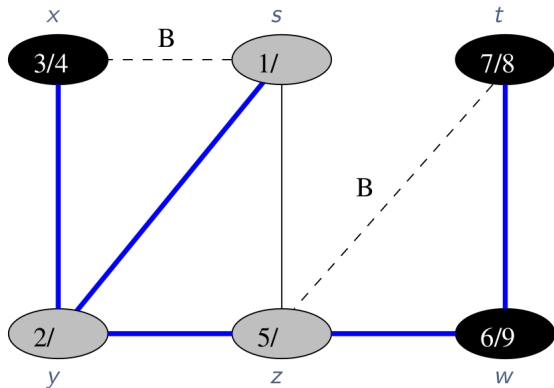
## DFS em grafos não direcionados



## DFS em grafos não direcionados

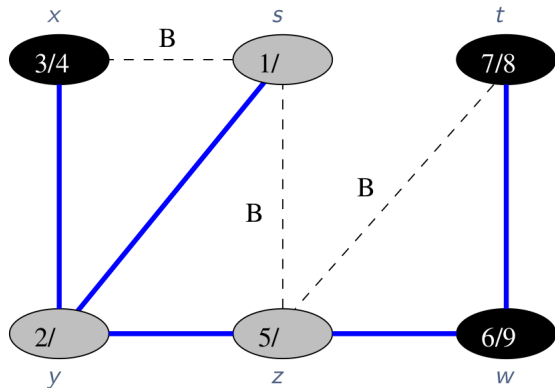


## DFS em grafos não direcionados

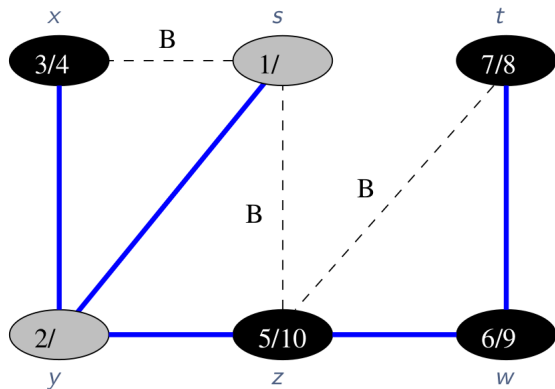




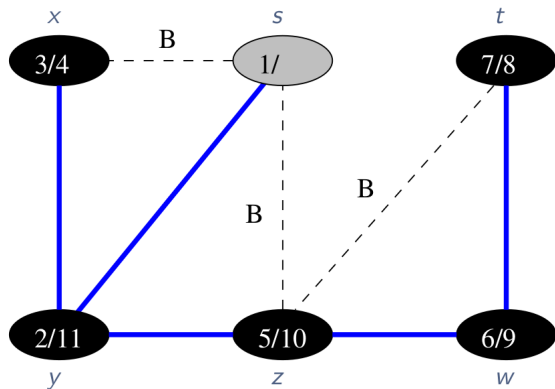
## DFS em grafos não direcionados



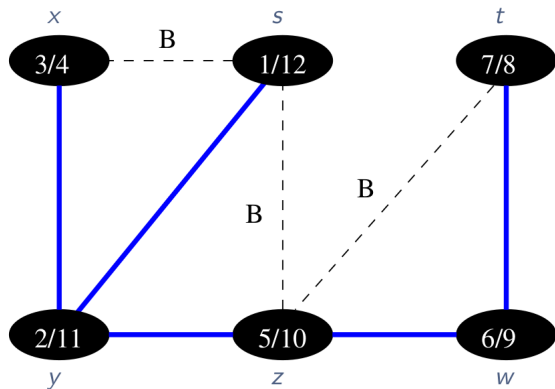
## DFS em grafos não direcionados



## DFS em grafos não direcionados



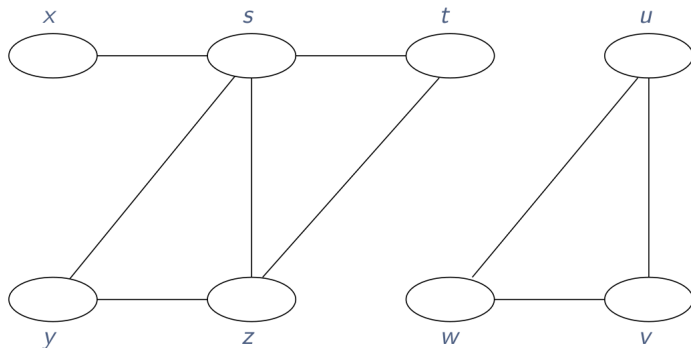
## DFS em grafos não direcionados



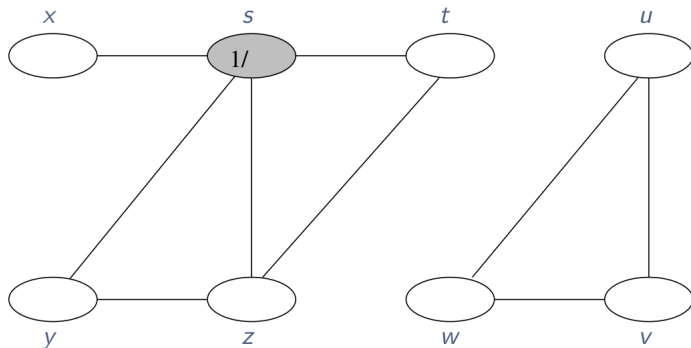
## Componentes conexas



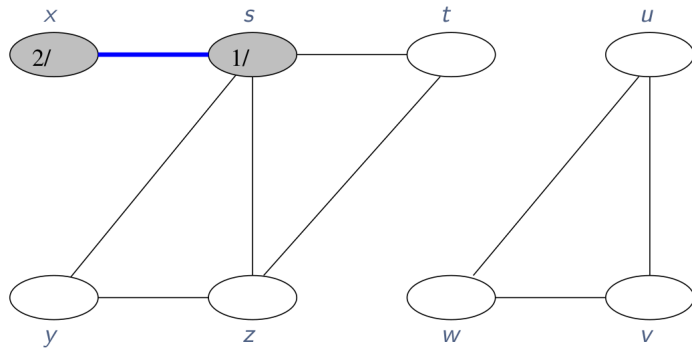
## Componentes conexas de grafos não direcionados



## Componentes conexos de grafos não direcionados

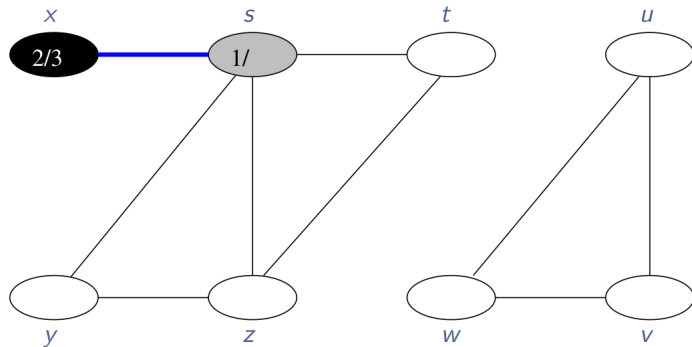


## Componentes conexos de grafos não direcionados

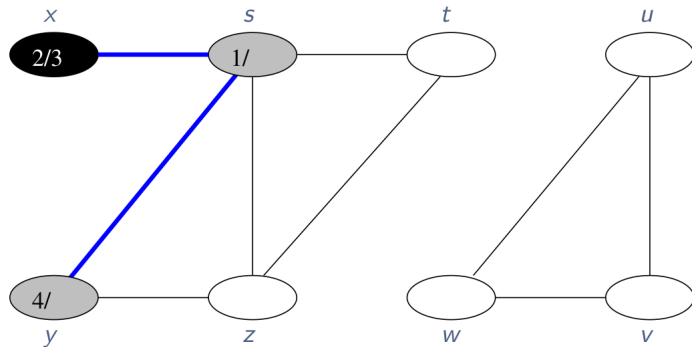




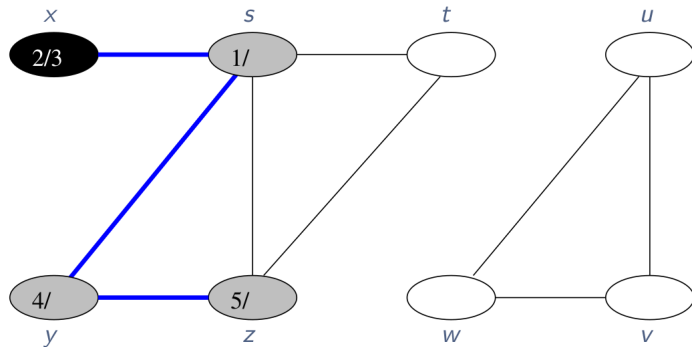
## Componentes conexos de grafos não direcionados



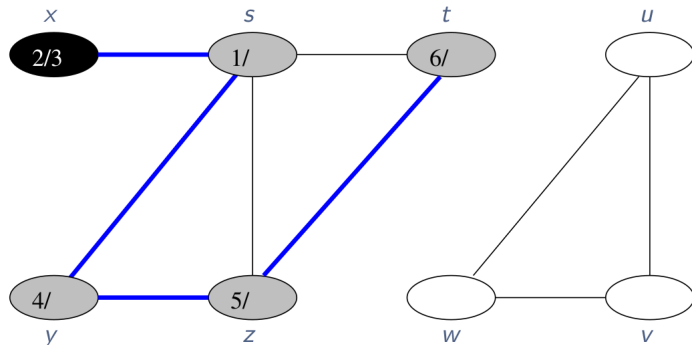
## Componentes conexos de grafos não direcionados



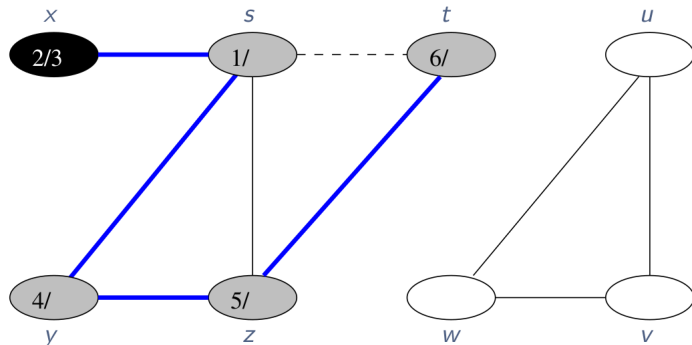
## Componentes conexos de grafos não direcionados



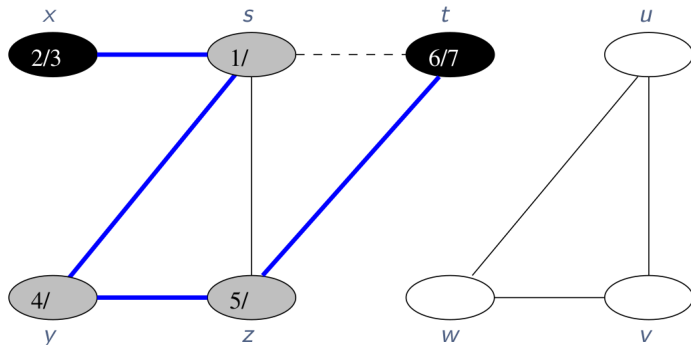
## Componentes conexos de grafos não direcionados



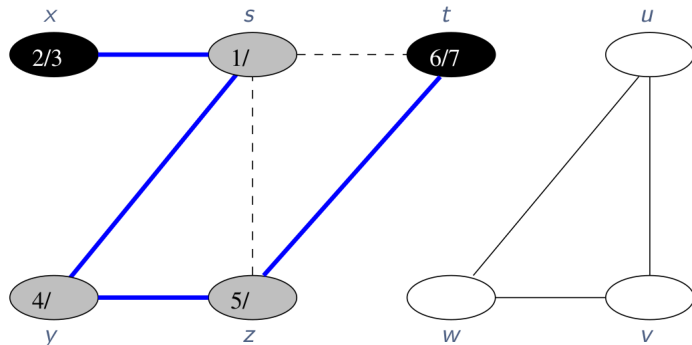
## Componentes conexos de grafos não direcionados



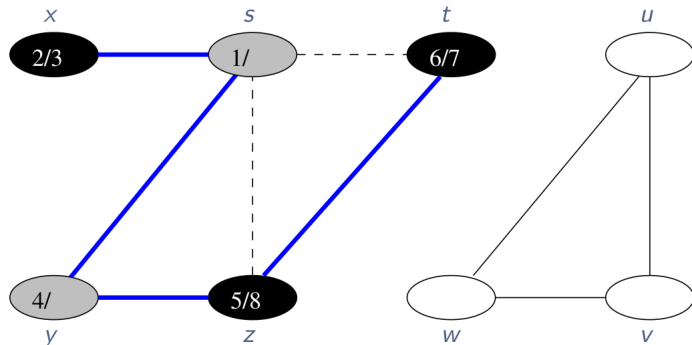
## Componentes conexos de grafos não direcionados



## Componentes conexos de grafos não direcionados

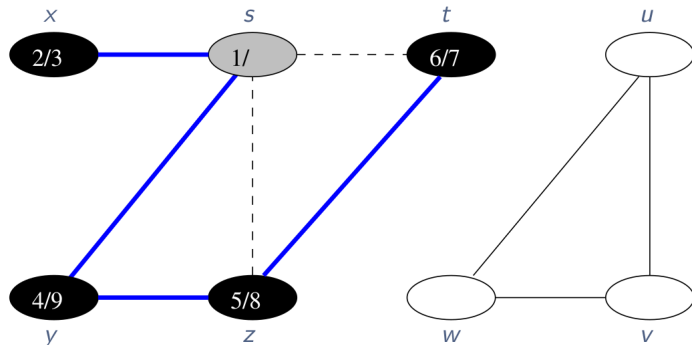


## Componentes conexos de grafos não direcionados

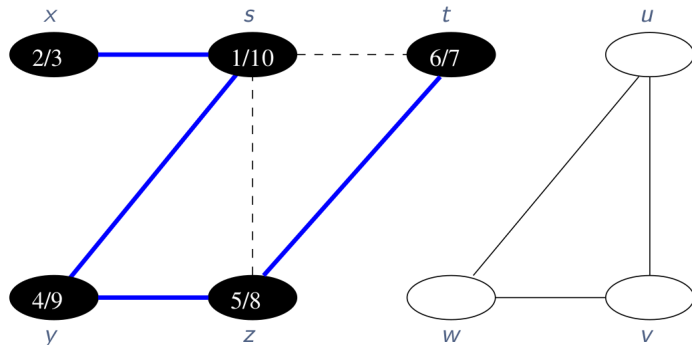




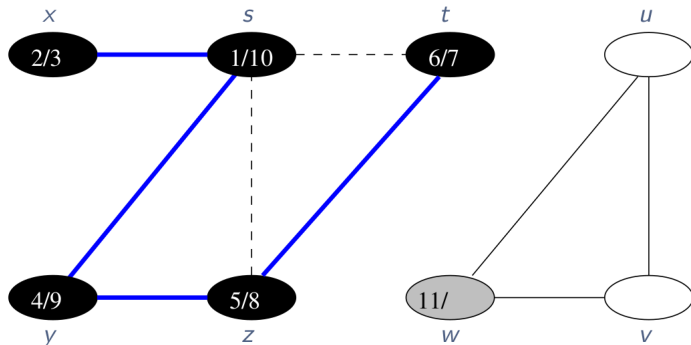
## Componentes conexos de grafos não direcionados



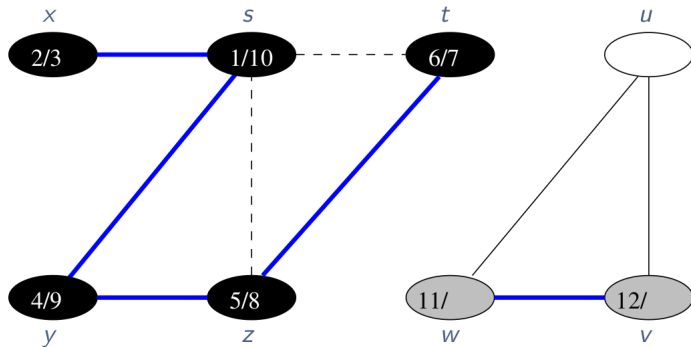
## Componentes conexos de grafos não direcionados



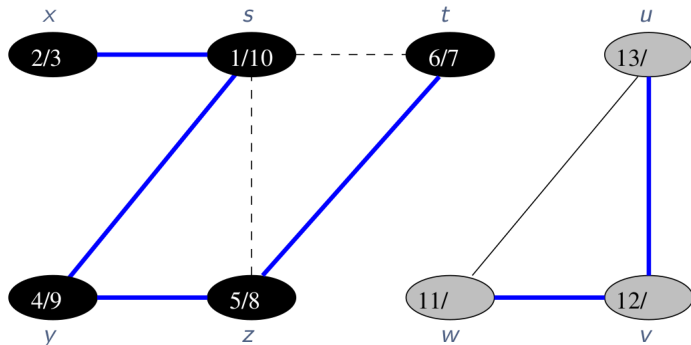
## Componentes conexos de grafos não direcionados



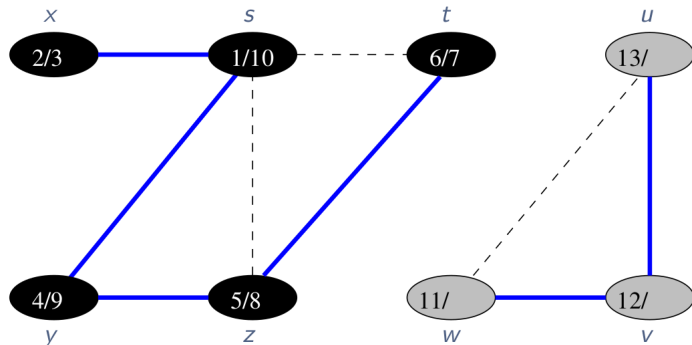
## Componentes conexos de grafos não direcionados



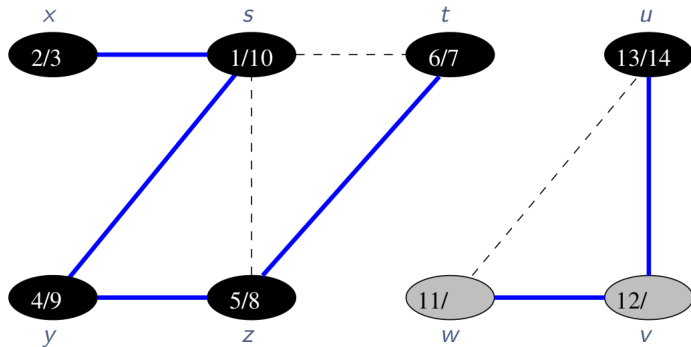
## Componentes conexos de grafos não direcionados



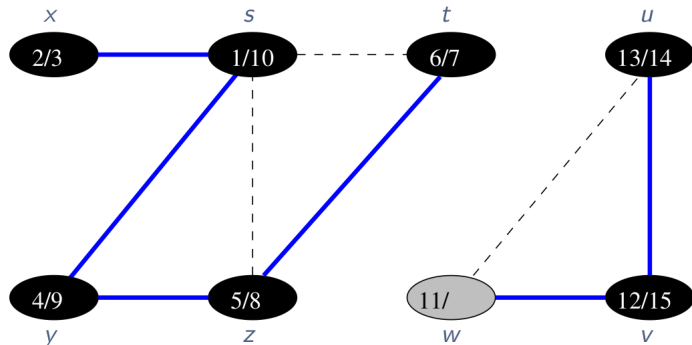
## Componentes conexos de grafos não direcionados



## Componentes conexos de grafos não direcionados

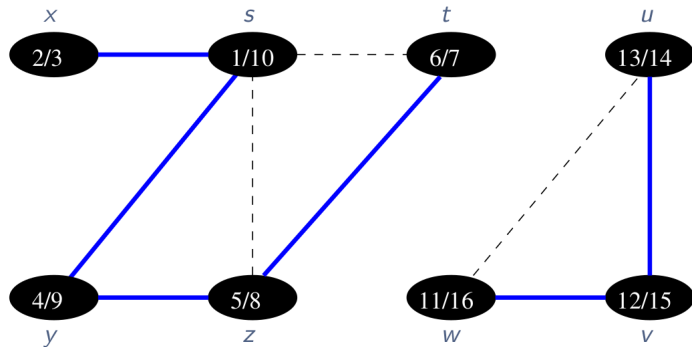


## Componentes conexos de grafos não direcionados





## Componentes conexos de grafos não direcionados



# Componentes conexas de grafos não direcionados

Contando o número de componentes

# Componentes conexas de grafos não direcionados

Contando o número de componentes

- cada componente corresponde a uma árvore de busca

# Componentes conexas de grafos não direcionados

Contando o número de componentes

- cada componente corresponde a uma árvore de busca
- é o **número de chamadas** a **DFS-VISIT** a partir de **DFS**

# Componentes conexas de grafos não direcionados

Contando o número de componentes

- cada componente corresponde a uma árvore de busca
- é o **número de chamadas** a **DFS-VISIT** a partir de **DFS**

Vamos modificar **DFS**

# Componentes conexas de grafos não direcionados

Contando o número de componentes

- cada componente corresponde a uma árvore de busca
- é o **número de chamadas** a **DFS-VISIT** a partir de **DFS**

Vamos modificar **DFS**

- identificamos cada componente por um número

# Componentes conexas de grafos não direcionados

Contando o número de componentes

- cada componente corresponde a uma árvore de busca
- é o **número de chamadas** a **DFS-VISIT** a partir de **DFS**

Vamos modificar **DFS**

- identificamos cada componente por um número
- denotaremos por  $comp[v]$  a componente de  $v$

## Algoritmo DFS modificado

**DFS**( $G$ )

```
1  para cada  $u \in V[G]$  faça
2       $cor[u] = \text{branco}$ 
3   $\ell = 0$ 
4  para cada  $u \in V[G]$  faça
5      se  $cor[u] == \text{branco}$  então
6           $\ell = \ell + 1$ 
7          DFS-VISIT( $u$ )
```

- $\ell$  é o número de chamadas a **DFS-VISIT** a partir de **DFS**



## Algoritmo DFS-VISIT modificado

### **DFS-visit**( $u$ )

```
1   $cor[u] = \text{cinza}$   
2  para cada  $v \in Adj[u]$  faça  
3      se  $cor[v] == \text{branco}$  então  
4          DFS-VISIT( $v$ )  
5   $cor[u] = \text{preto}$   
6   $comp[u] = \ell$ 
```

FIM

