Teoria dos Grafos

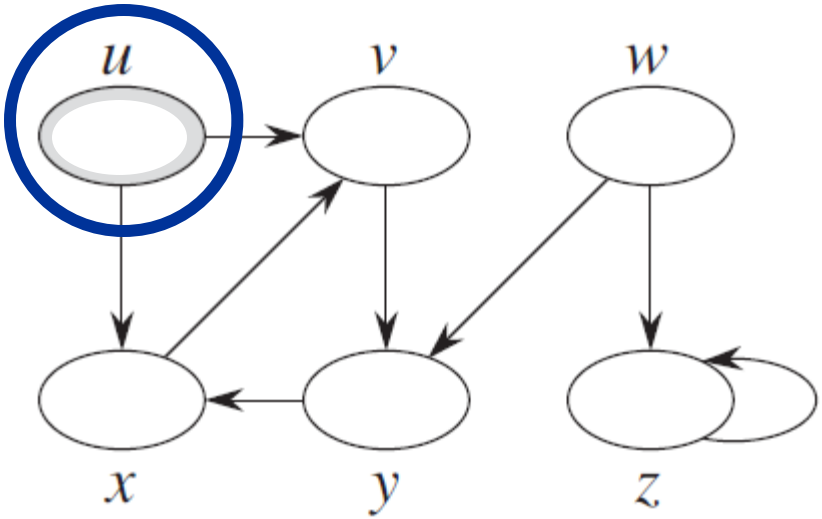
Universidade Federal do ABC

Introdução

- Algoritmos de busca
  - Maneiras sistemáticas de visitar vértices

- Principais algoritmos
  - BFS: largura
  - DFS: profundidade

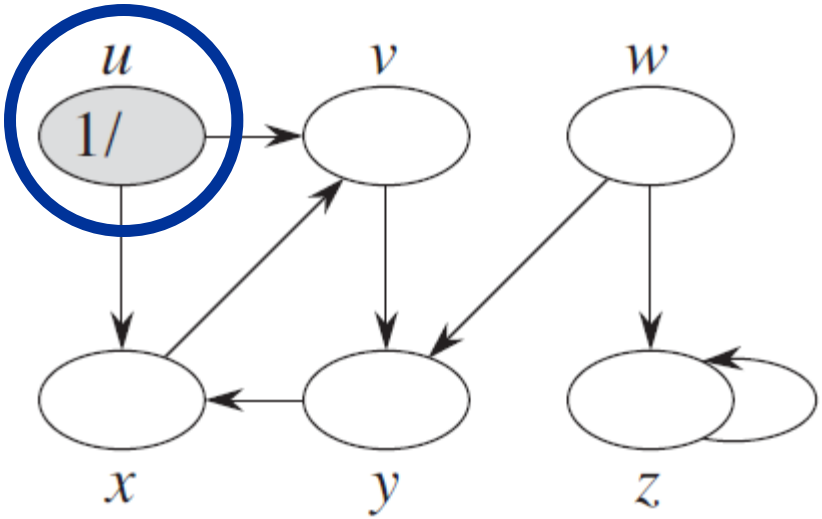
- Ideia: "descoberta" e "finalização"



(Pilha)

S =

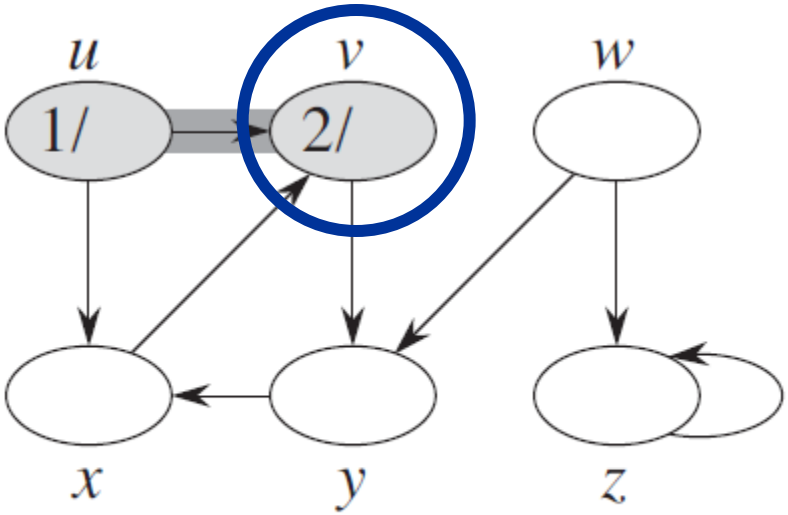
- Ideia: "descoberta" e "finalização"



(Pilha)

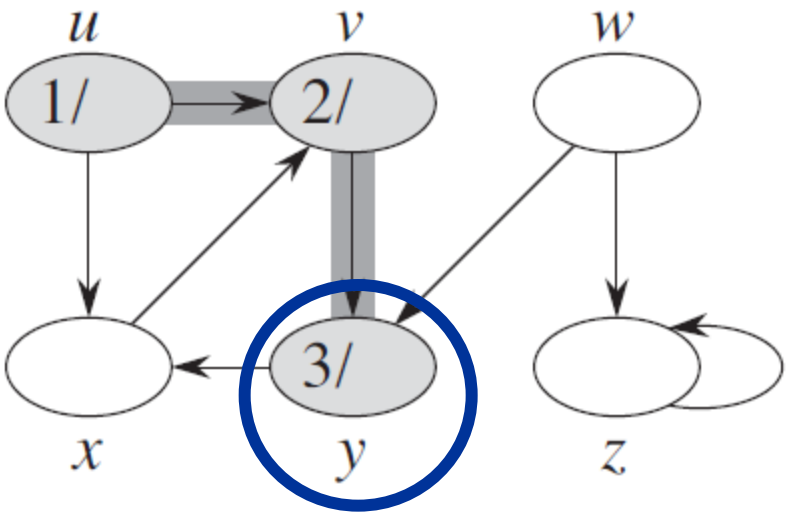
S = u

- Ideia: "descoberta" e "finalização"



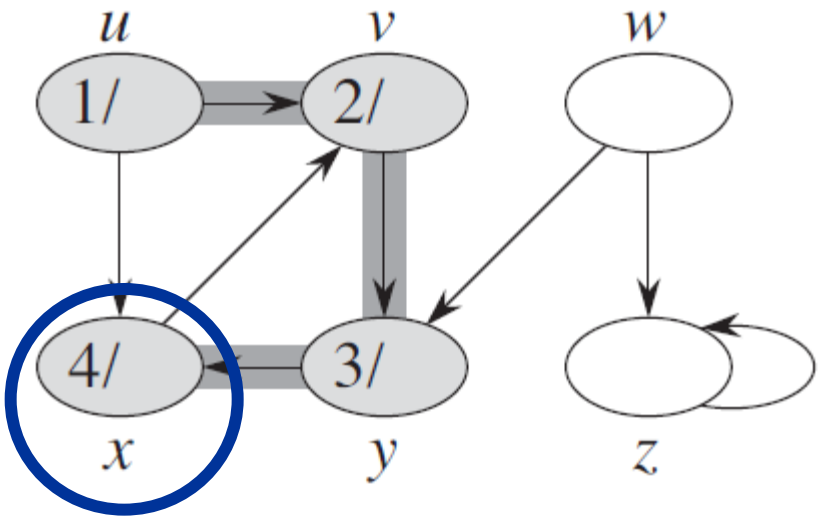
S = u, v

- Ideia: "descoberta" e "finalização"



S = u, v, y

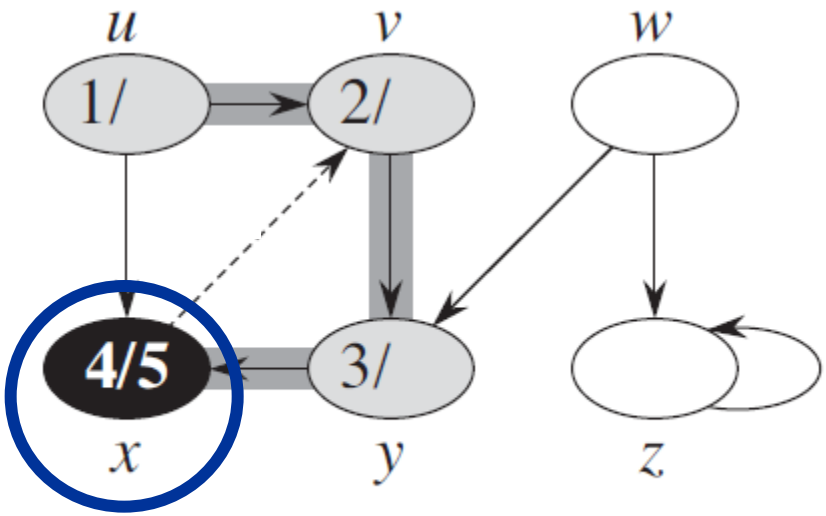
- Ideia: "descoberta" e "finalização"



Topo

S = u, v, y, x

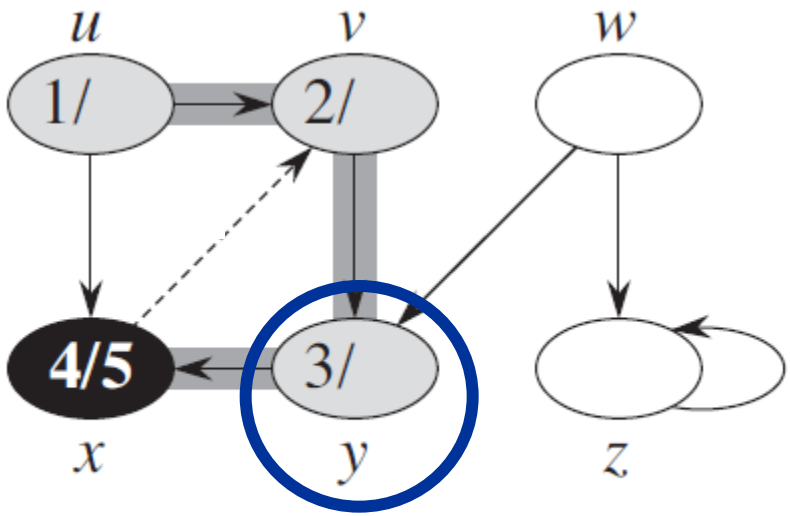
- Ideia: "descoberta" e "finalização"



Topo

S = u, v, y, ~~x~~

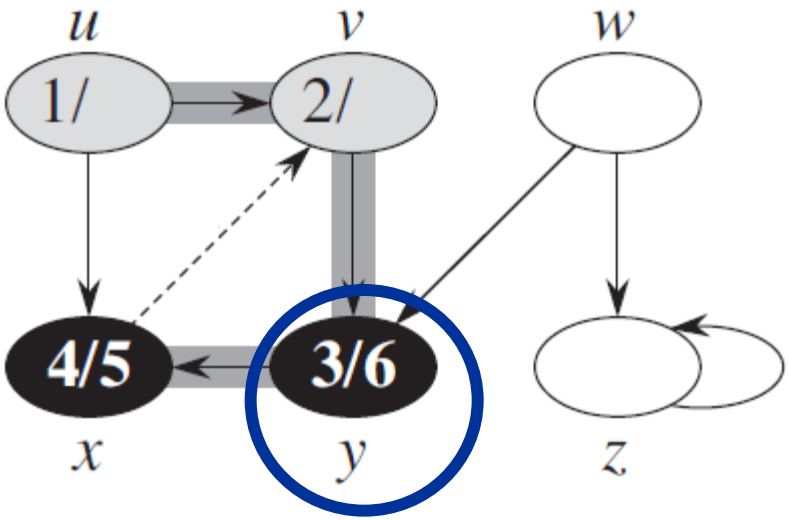
- Ideia: "descoberta" e "finalização"



Topo

S = u, v, y

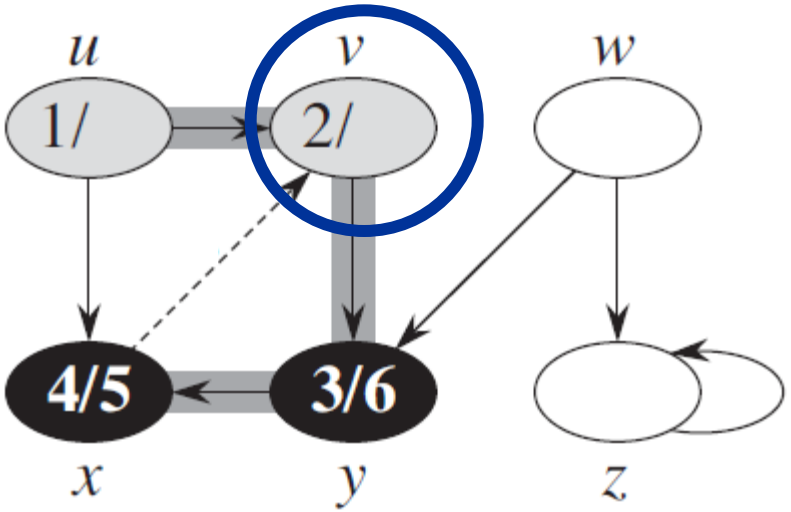
- Ideia: "descoberta" e "finalização"



Topo

S = u, v, ~~y~~

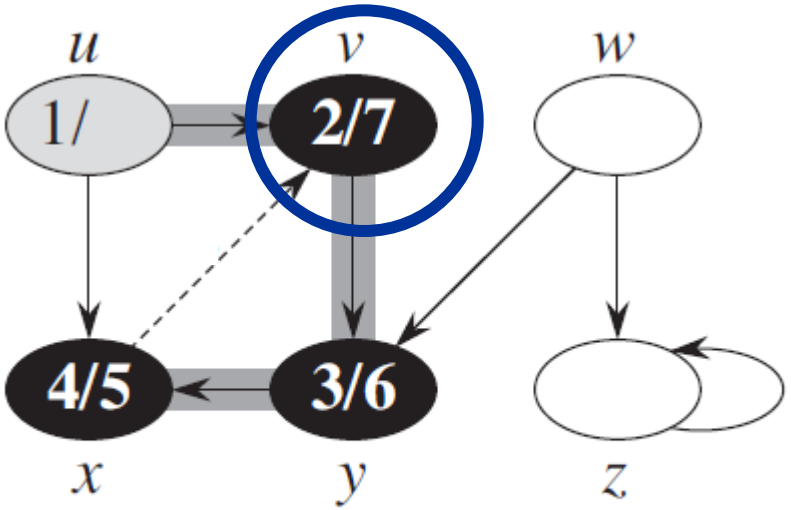
- Ideia: "descoberta" e "finalização"



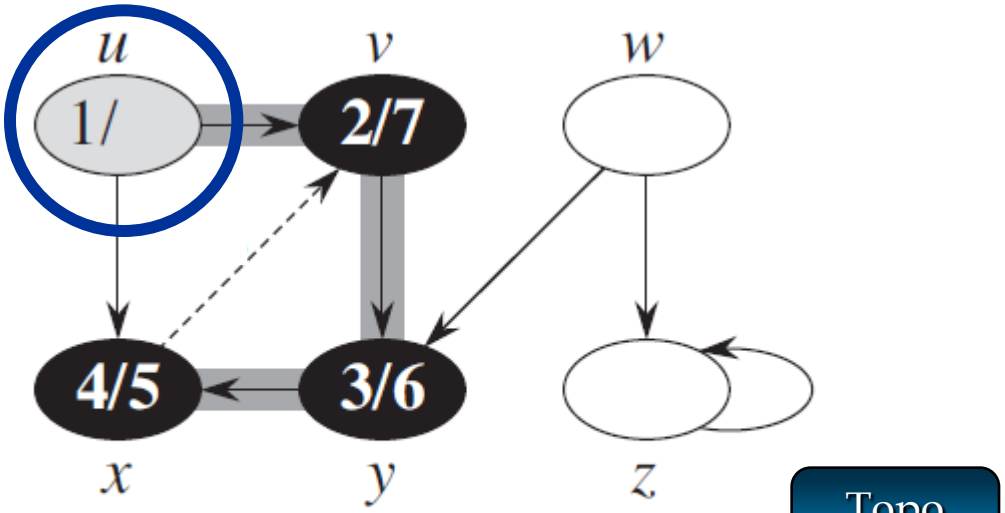
Topo

S = u, v

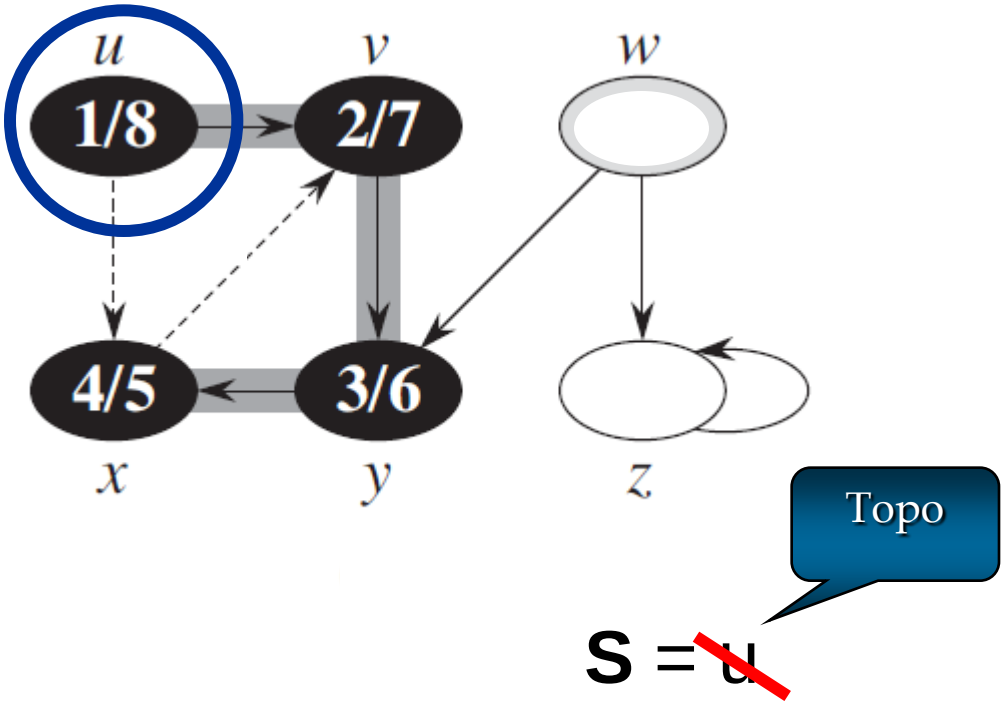
- Ideia: "descoberta" e "finalização"



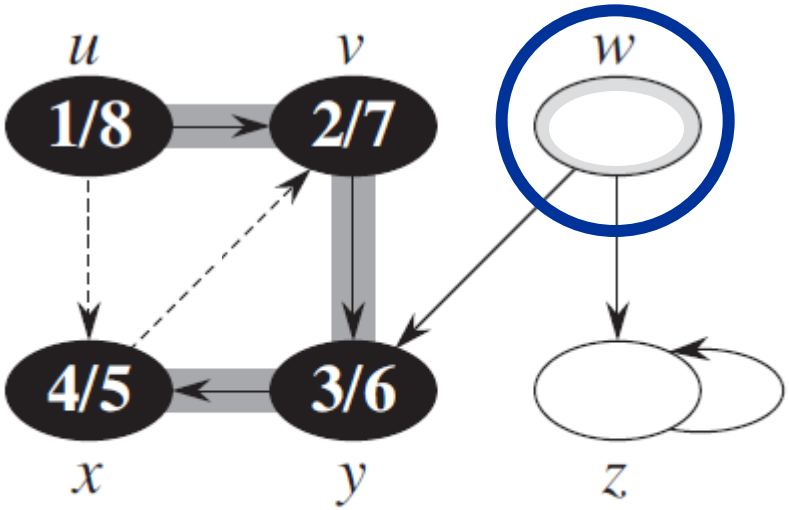
- Ideia: "descoberta" e "finalização"



- Ideia: "descoberta" e "finalização"

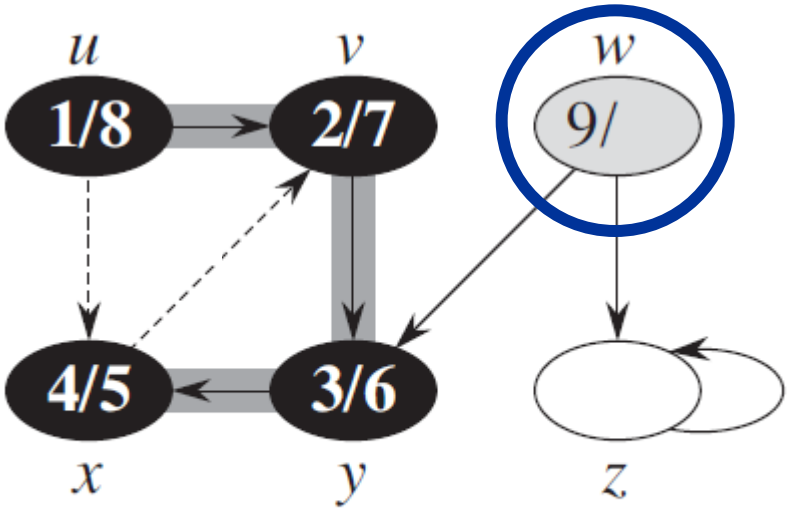


- Ideia: "descoberta" e "finalização"



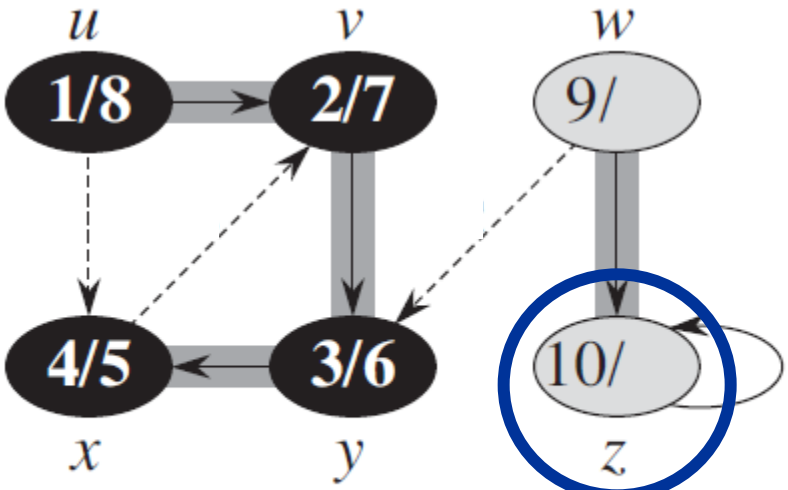
S =

- Ideia: "descoberta" e "finalização"



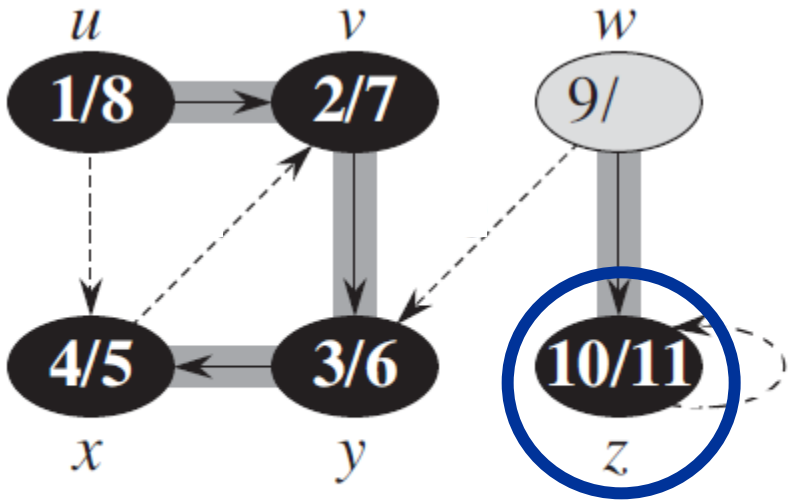
S = w

- Ideia: "descoberta" e "finalização"

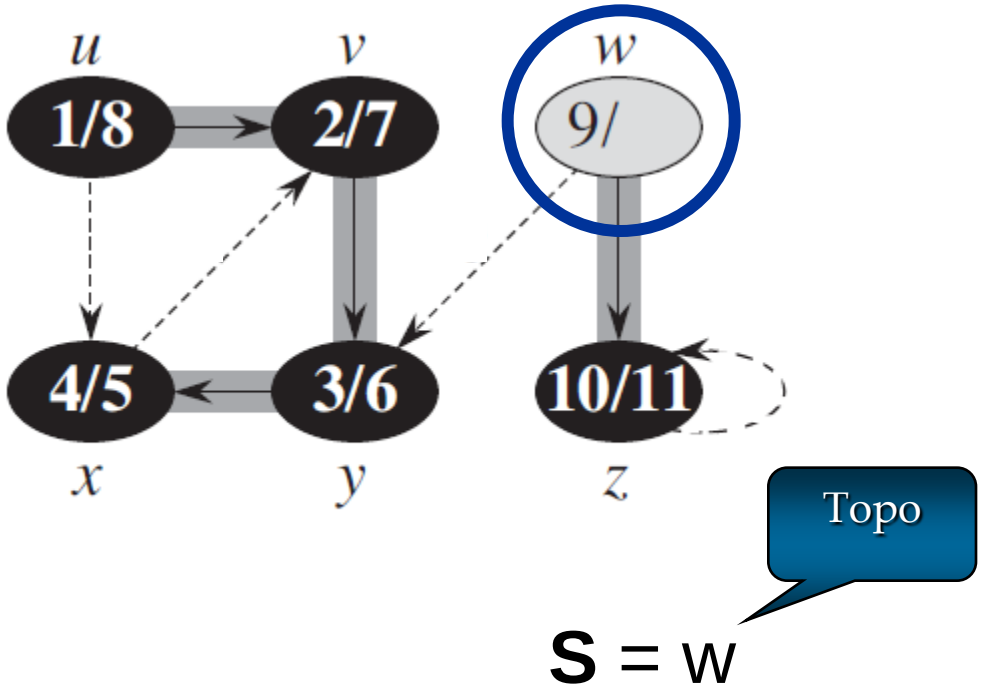


S = w, z

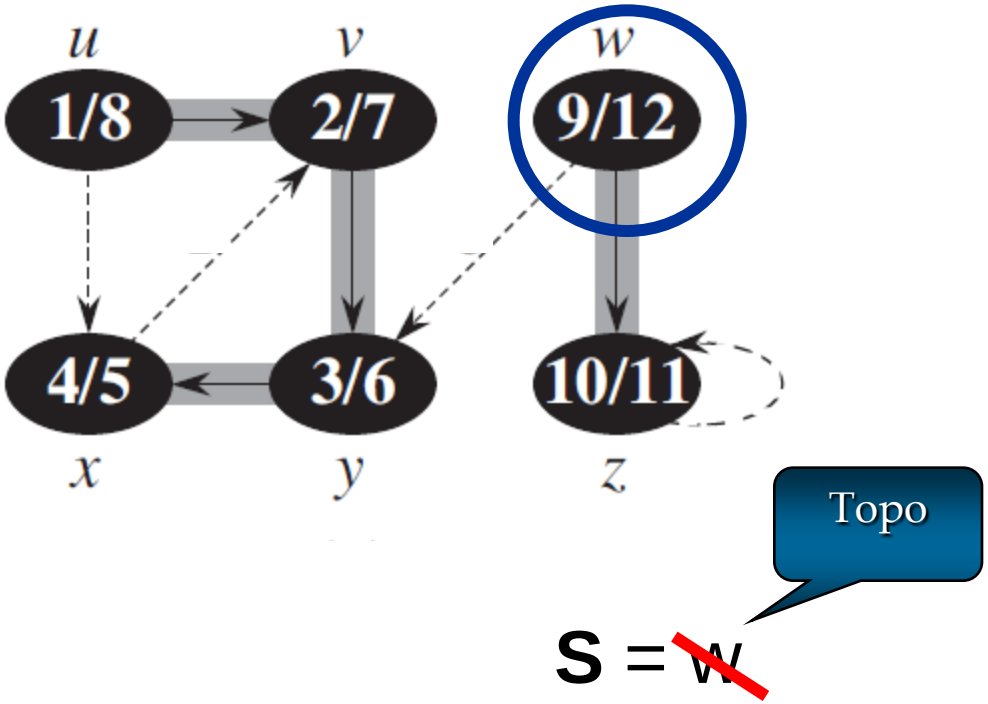
- Ideia: "descoberta" e "finalização"



- Ideia: "descoberta" e "finalização"

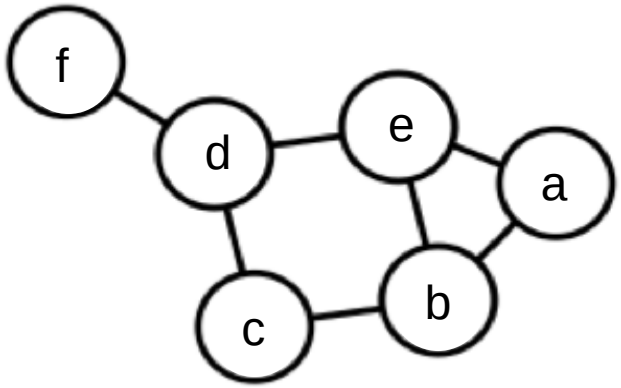


- Ideia: "descoberta" e "finalização"

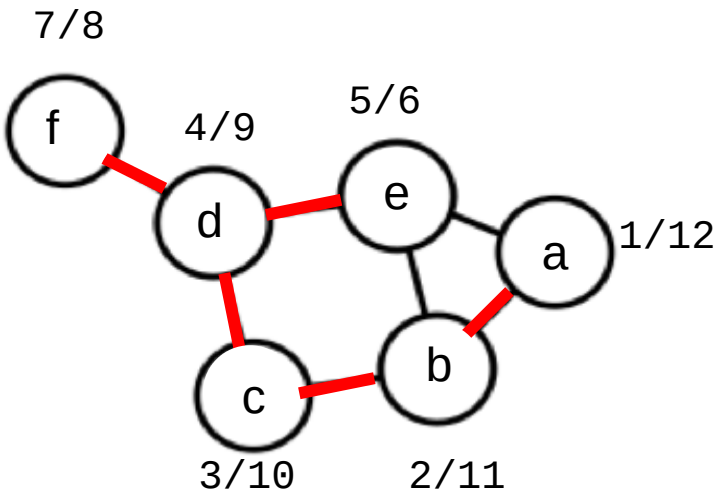


Exercício

- Executar a busca em profundidade, dando preferência para vértices de “menor“ letra.
  - "descoberta" e "finalização"



- Expressão de parênteses

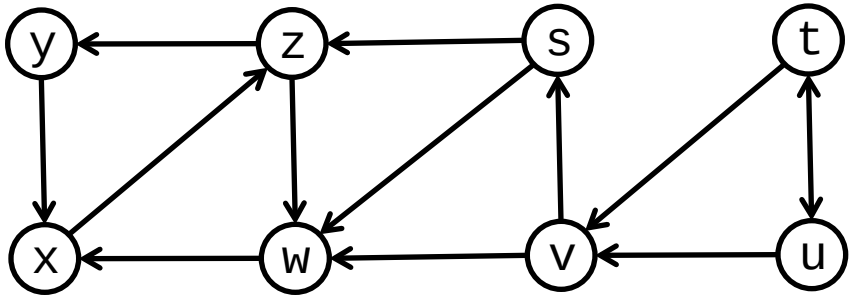


instantes 1 2 3 4 5 6 7 8 9 10 11 12

- Ex: (a (b (c (d (e e) (f f) d) c) b) a)

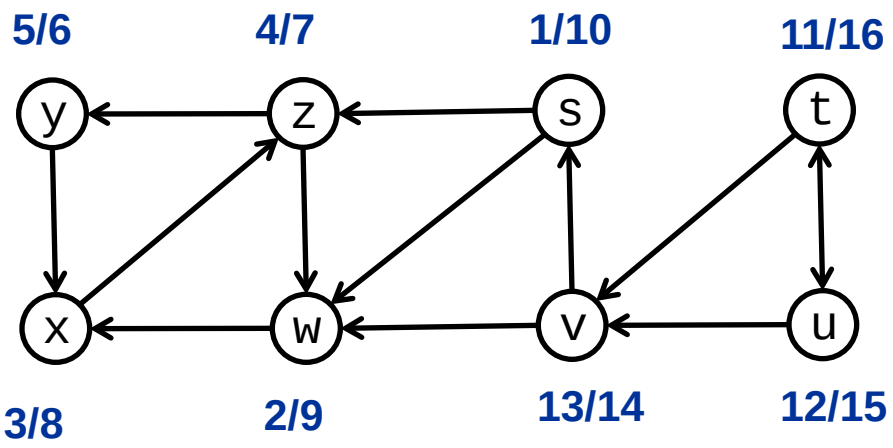
Exercício

- Calcular expressão de parênteses
  - dar preferência para vértices de “menor“ letra



Solução

- Calcular expressão de parênteses
  - dar preferência para vértices de “menor“ letra



Expressão:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(s	(w	(x	(z	(y	y)	z)	x)	w)	s)	(t	(u	(v	v)	u)	t)

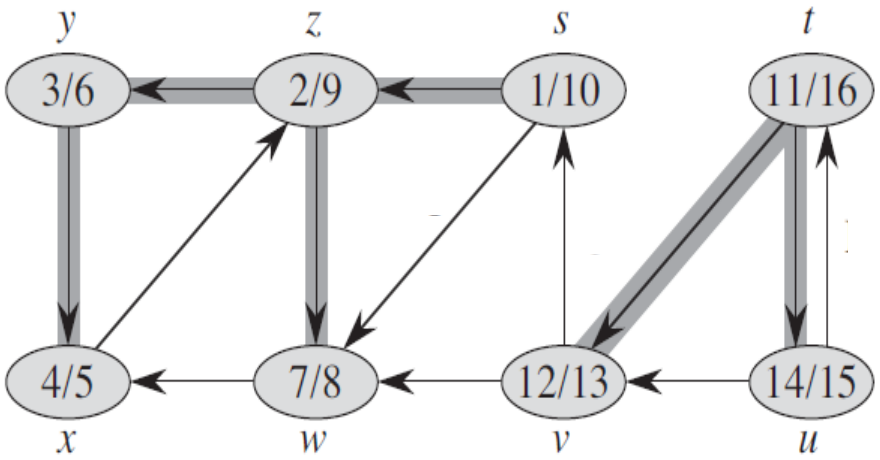
Tarefa

- Exercícios:
  - Lista 3

(Calcular os instantes de descoberta e finalização,
dando preferência para vértices de menor índice ou letra.
Calcular as expressões de parênteses.)

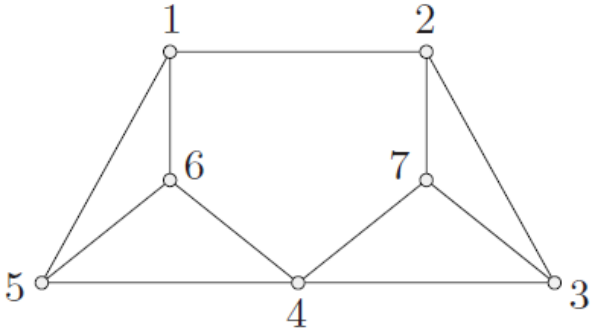
- Obs:
  A escolha dos vértices (não-visitados) para a visita DFS é arbitrária.

- Ex.

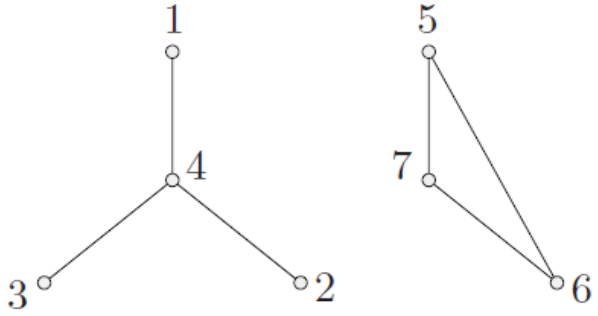


Expressão: 1 (s 2 (z 3 (y 4 (x 5 x) 6 y) 7 (w 8 w) 9 z) 10 s) 11 (t 12 (v 13 v) 14 (u 15 u) 16 t)

- Conexo vs Desconexo

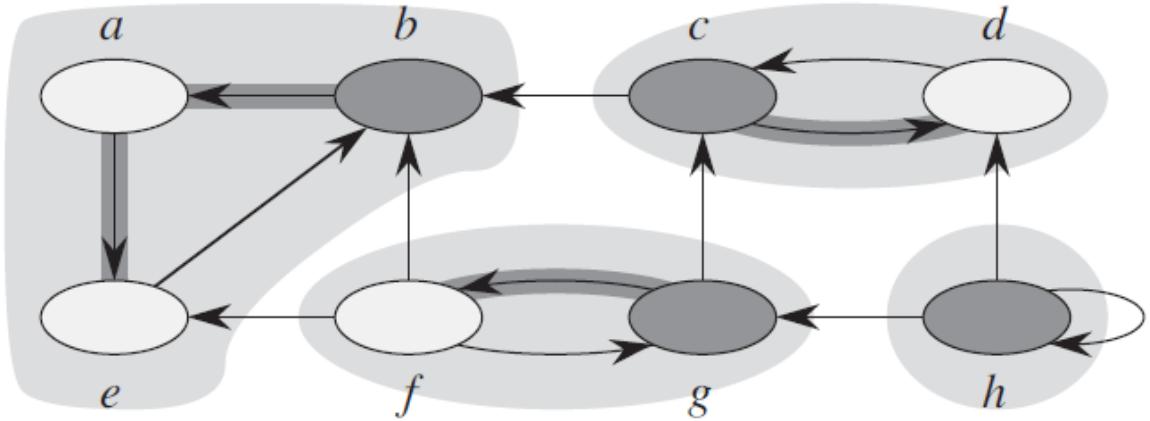


(a)



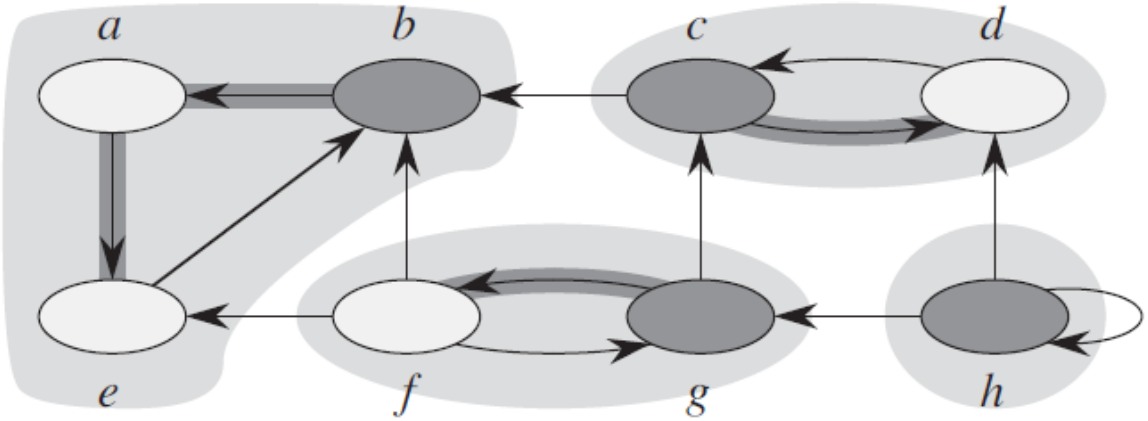
(b)

- Componentes fortemente conectados
  (SCC: Strongly Connected Components**)**

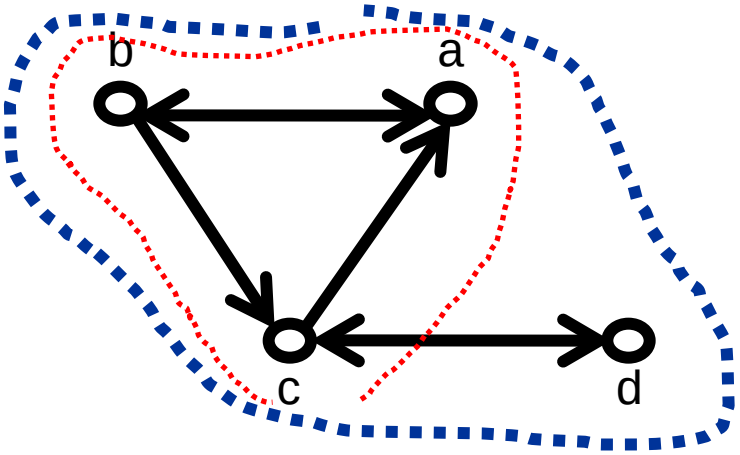


DFS: busca em profundidade

- Componentes fortemente conectados

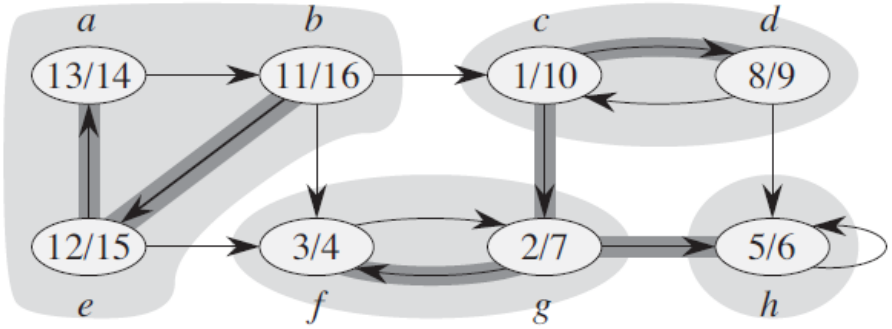


- Maximal

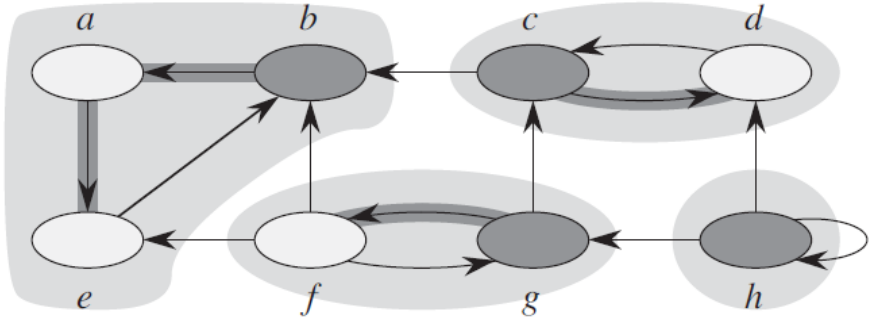


DFS: busca em profundidade

- G



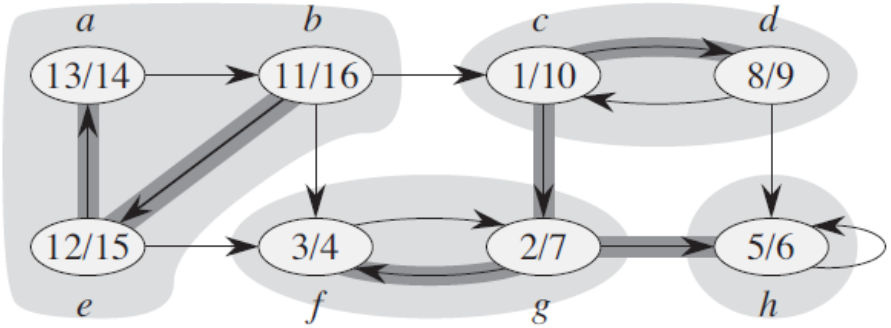
- G^T: grafo transposto



DFS: busca em profundidade

Strongly-Connected-Components(G)
1 executar **DFS**(G) para calcular instantes v.f
2 executar **DFS**(G^T), considerando **ordem decrescente** de v.f do passo 1

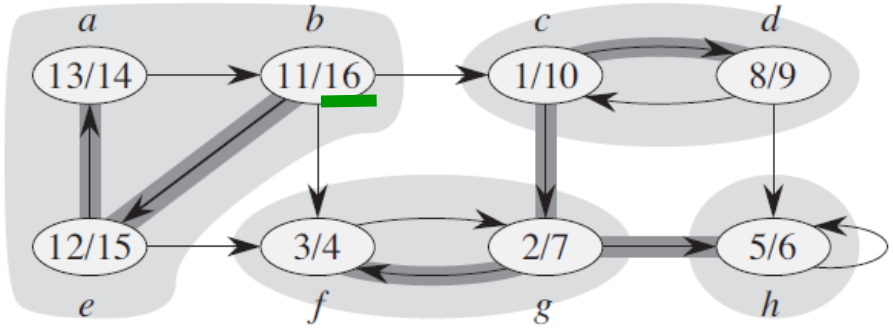
G



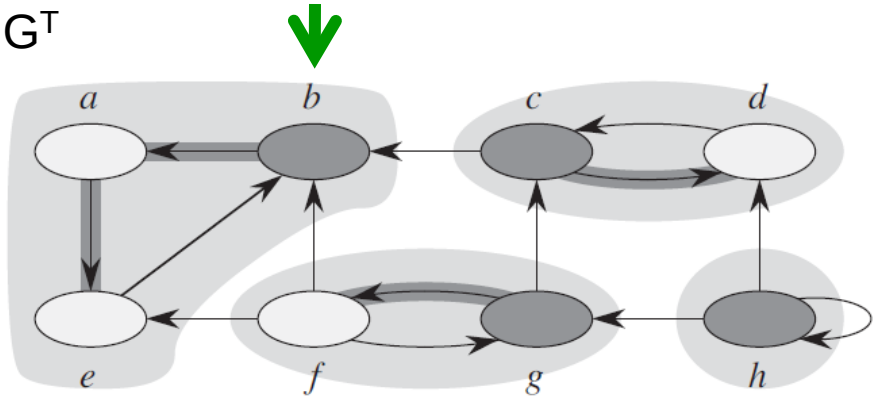
DFS: busca em profundidade

```
Strongly-Connected-Components(G)
 1 executar DFS(G) para calcular instantes v.f
 2 executar DFS(G^T), considerando ordem
decrescente de v.f do passo 1
```

G



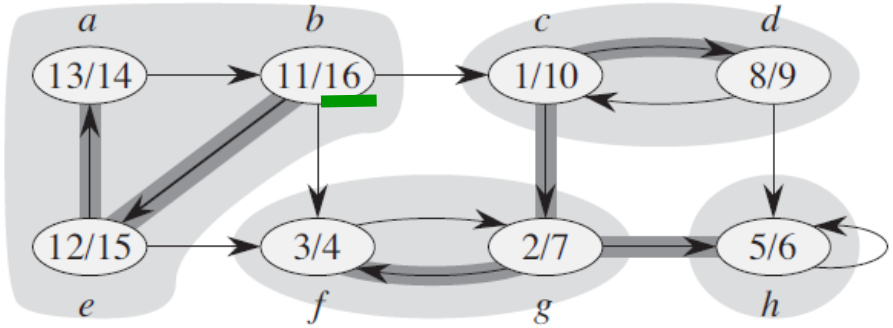
G^T



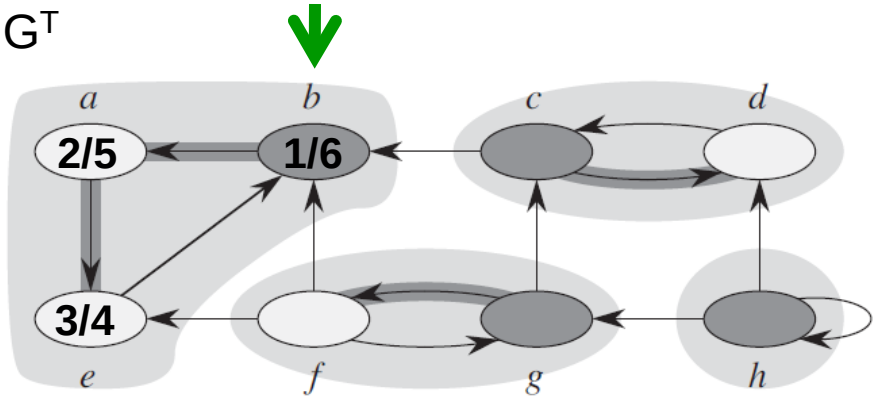
DFS: busca em profundidade

```
Strongly-Connected-Components(G)
 1 executar DFS(G) para calcular instantes v.f
 2 executar DFS(G^T), considerando ordem
decrescente de v.f do passo 1
```

G



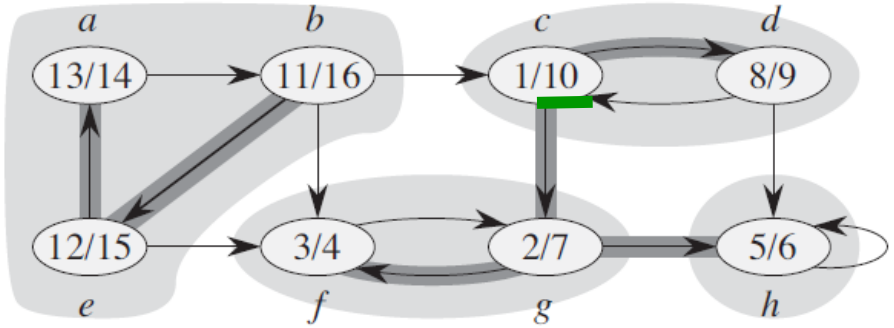
G^T



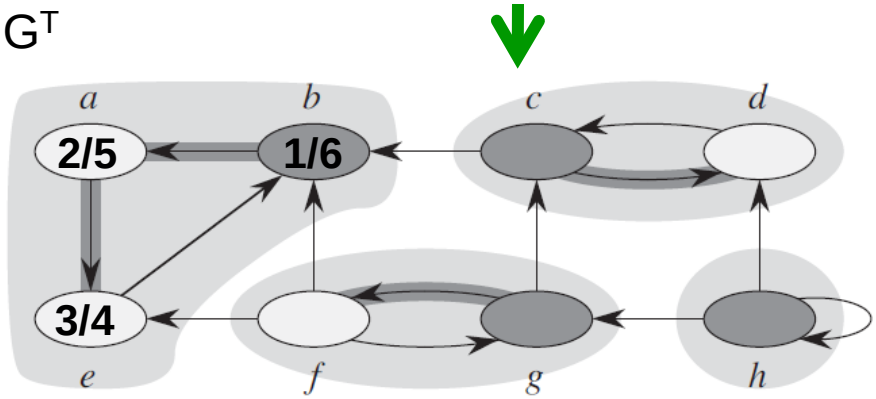
DFS: busca em profundidade

Strongly-Connected-Components(G)
1 executar **DFS**(G) para calcular instantes v.f
2 executar **DFS**(G^T), considerando **ordem decrescente** de v.f do passo 1

G

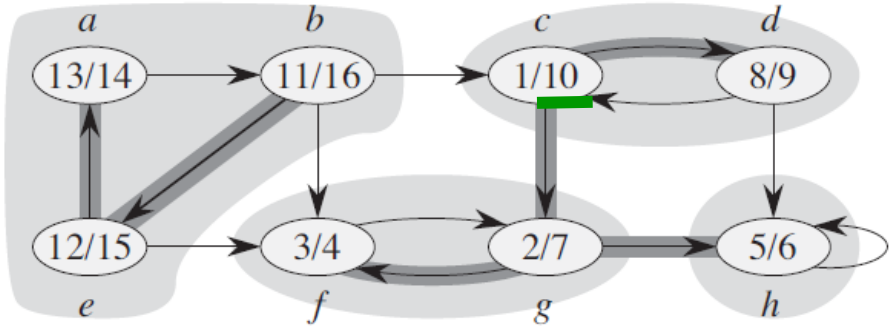


G^T

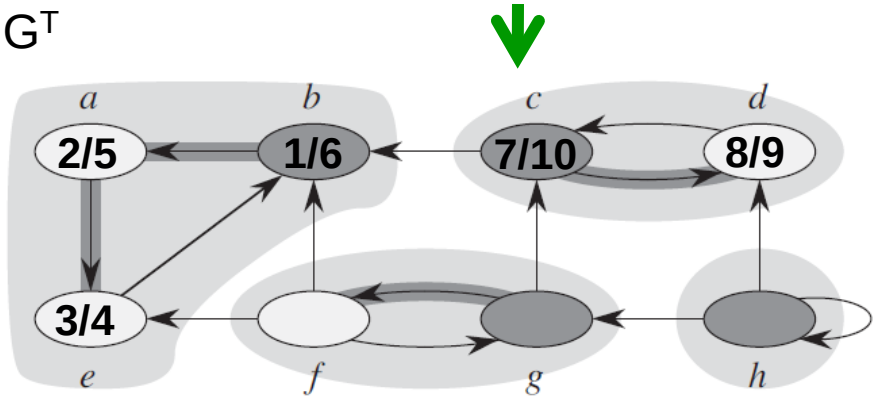


```
Strongly-Connected-Components(G)
 1 executar DFS(G) para calcular instantes v.f
 2 executar DFS(G^T), considerando ordem
decrescente de v.f do passo 1
```

G

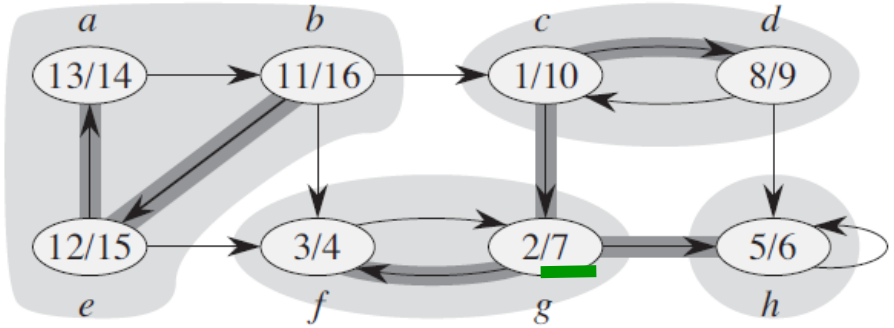


G^T

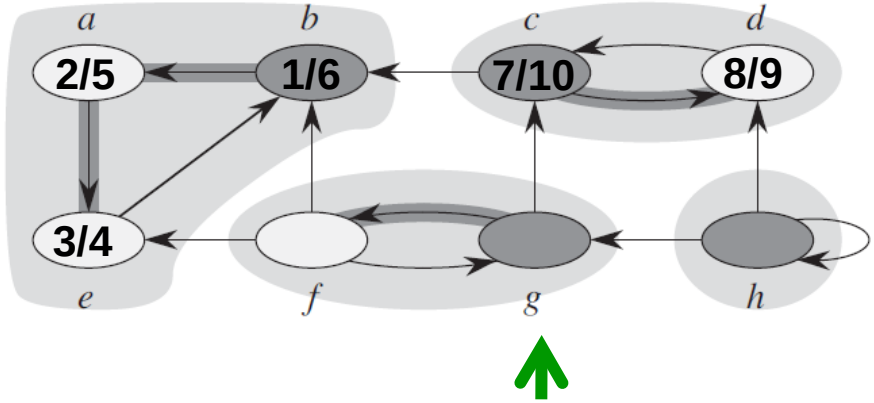


```
Strongly-Connected-Components(G)
 1 executar DFS(G) para calcular instantes v.f
 2 executar DFS(G^T), considerando ordem
decrescente de v.f do passo 1
```

G

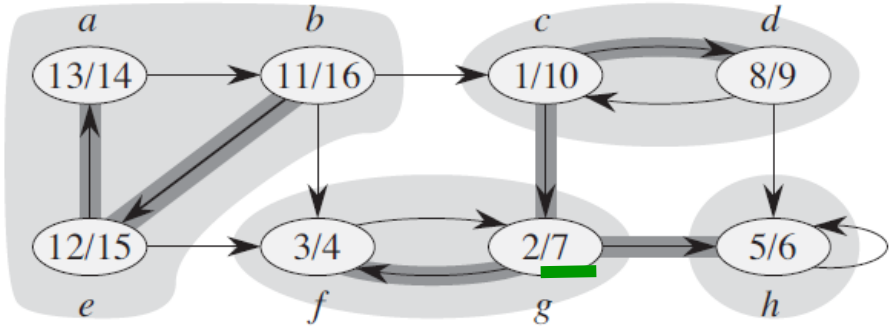


G^T

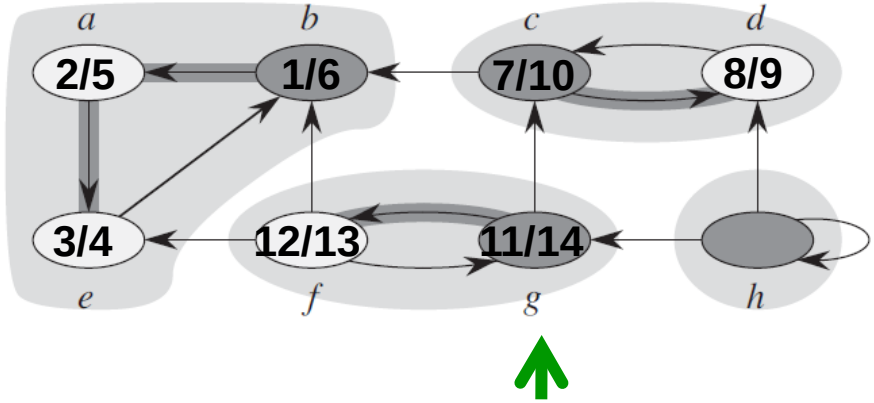


```
Strongly-Connected-Components(G)
 1 executar DFS(G) para calcular instantes v.f
 2 executar DFS(G^T), considerando ordem
decrescente de v.f do passo 1
```

G



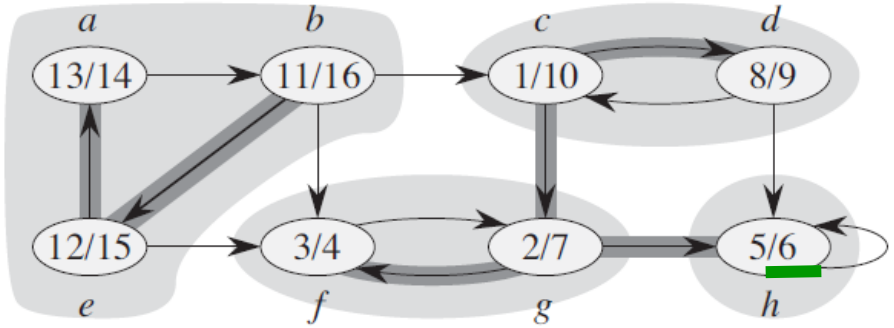
G^T



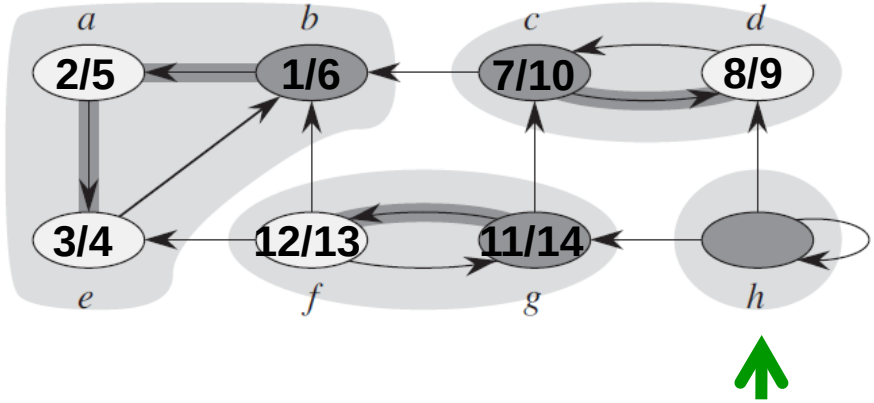
DFS: busca em profundidade

Strongly-Connected-Components(G)
1 executar **DFS**(G) para calcular instantes v.f
2 executar **DFS**(G^T), considerando **ordem decrescente** de v.f do passo 1

G

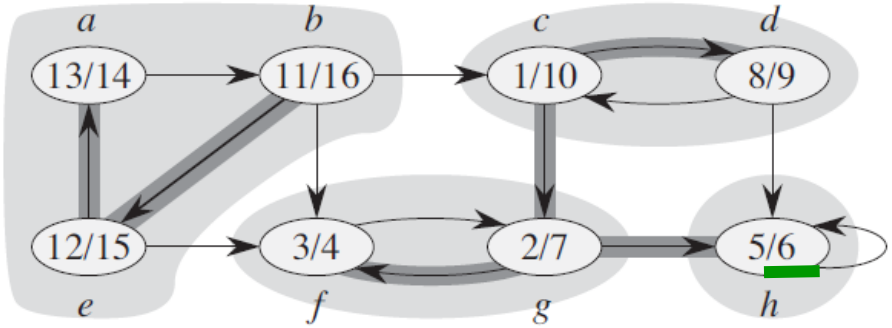


G^T

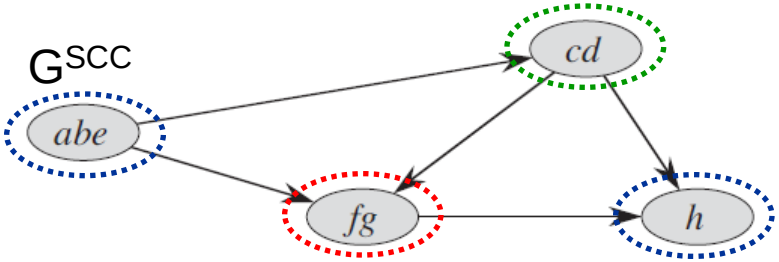


```
Strongly-Connected-Components(G)
 1 executar DFS(G) para calcular instantes v.f
 2 executar DFS(G^T), considerando ordem
decrescente de v.f do passo 1
```

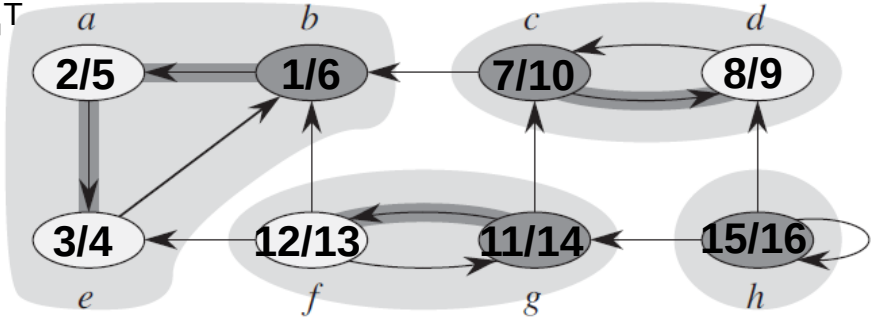
G



G^{SCC}



G^T

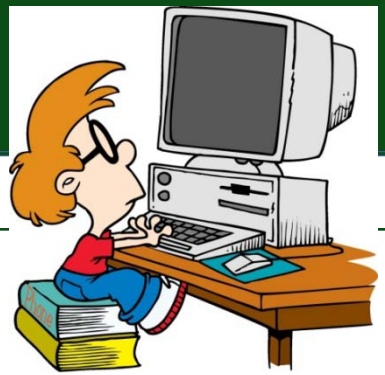


Expressão:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(b	(a	(e	e)	a)	b)	(c	(d	d)	c)	(g	(f	f)	g)	(h	h)

Tarefa

- Exercícios:
  - Lista 3



DFS: busca em profundidade

- DFS(G)
  - Entrada: um grafo G
  - Saída: instantes descoberta e finalização
  - Baseado em pilha
- DFSVisita(G,u)
  - Visita recursiva

- Atributos de vértices
  - v.d: instante de descoberta
  - v.f: instante de finalização
  - v.cor
    - branca (inicial)
    - cinza (descoberto)
    - preta (finalizado)

Pilha: operações

- PilhaVazia(S)
  - Devolve verdadeiro se a pilha S está vazia, ou falso caso contrário
- Empilha(S, x)
  - insere elemento x no topo da pilha S
- Desempilha(S)
  - remove e devolve o elemento no topo da pilha S

(Recursão: Não usaremos estas operações explicitamente...)

Pilha: acesso pelo topo

DFS: busca em profundidade

```
DFS(G)
 1 para cada vértice u_i em G.V faça
 2     u_i.cor = BRANCO
 3 tempo = 0
 4 para cada vértice u_i em G.V faça
 5     se u_i.cor == BRANCO
 6       entao VisitaDFS(G, u_i)
```

"Inicialização"

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5     se v_i.cor == BRANCO
 6        VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

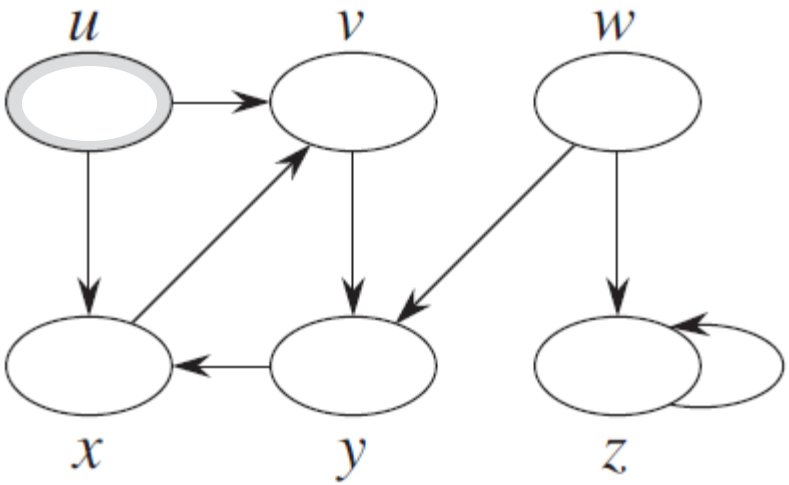
"Execução"

(Recursão)

DFS: busca em profundidade

```
DFS(G)
 1 para cada vértice u_i em G.V faça
 2     u_i.cor = BRANCO
 3 tempo = 0
 4 para cada vértice u_i em G.V faça
 5     se u_i.cor == BRANCO
 6       entao VisitaDFS(G, u_i)
```

"Inicialização"

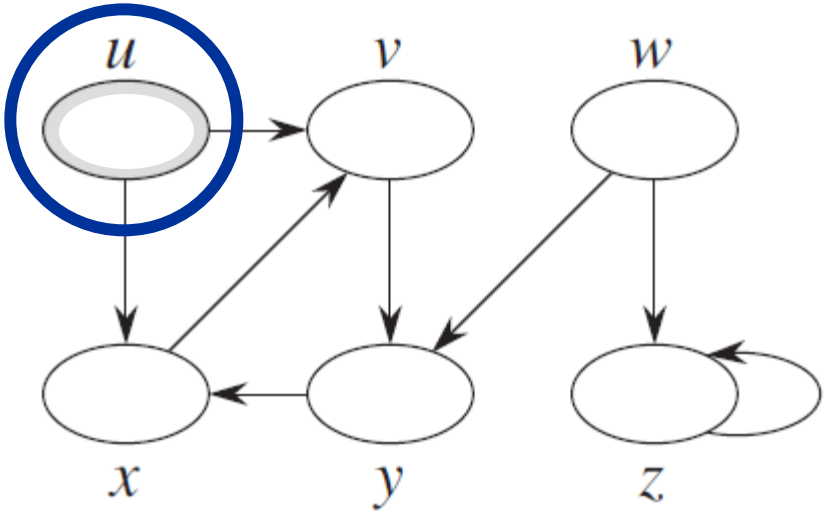



```
tempo = 0
```

DFS: busca em profundidade

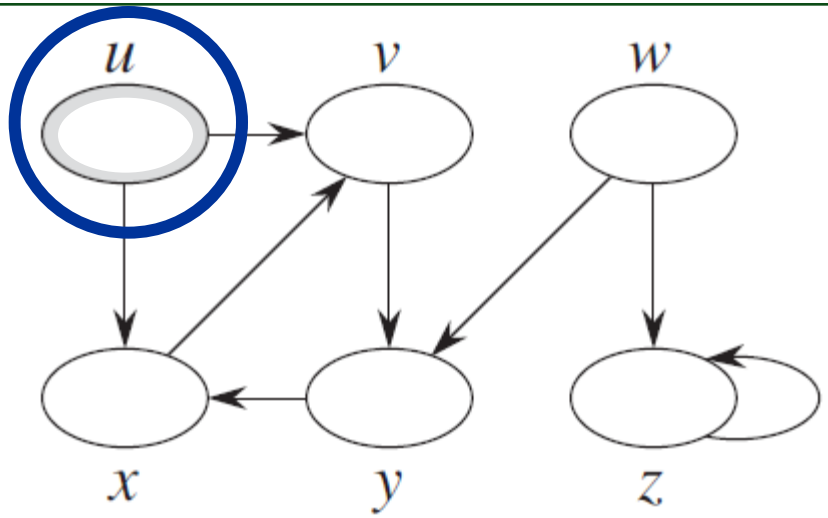
```
DFS(G)
 1 para cada vértice u_i em G.V faça
 2     u_i.cor = BRANCO
 3 tempo = 0
 4 para cada vértice u_i em G.V faça
 5     se u_i.cor == BRANCO
 6       entao VisitaDFS(G, u_i)
```

"Inicialização"



tempo = 0

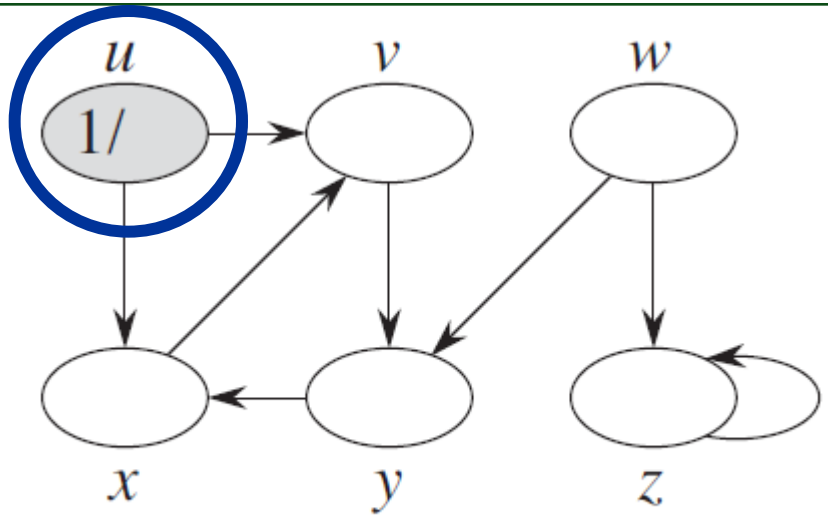
DFS: busca em profundidade



tempo = 0

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

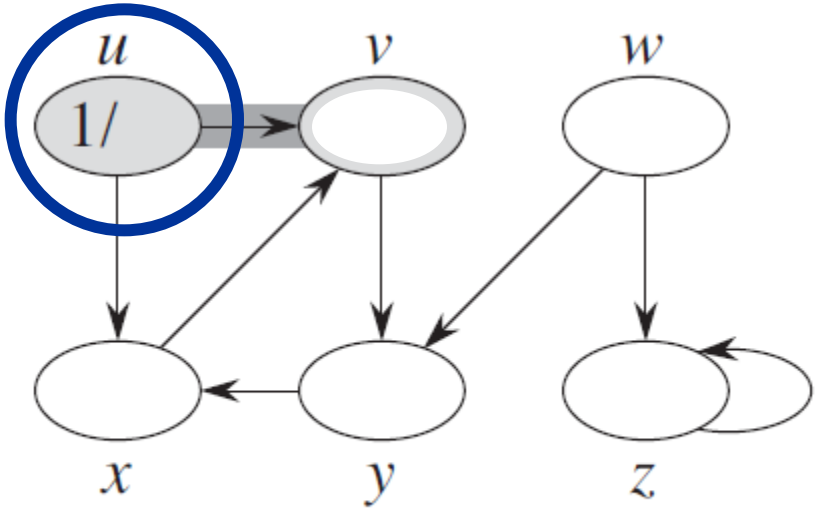
DFS: busca em profundidade



tempo = ~~0~~ 1

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

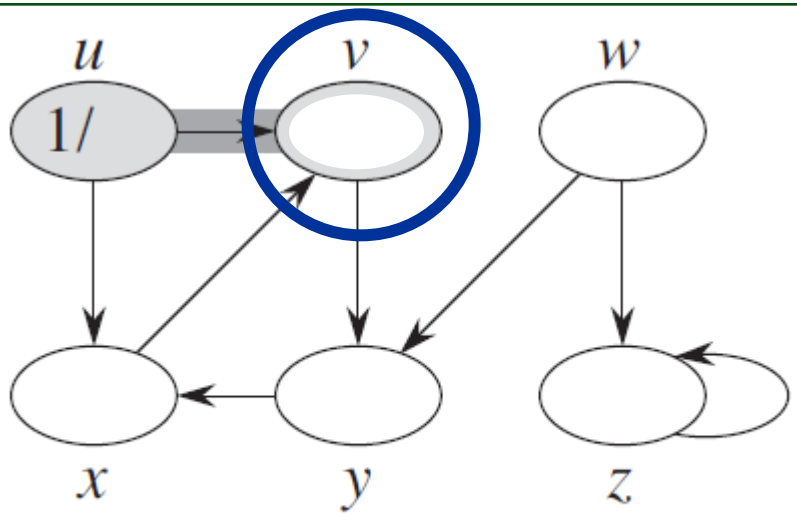
DFS: busca em profundidade



tempo = 1

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

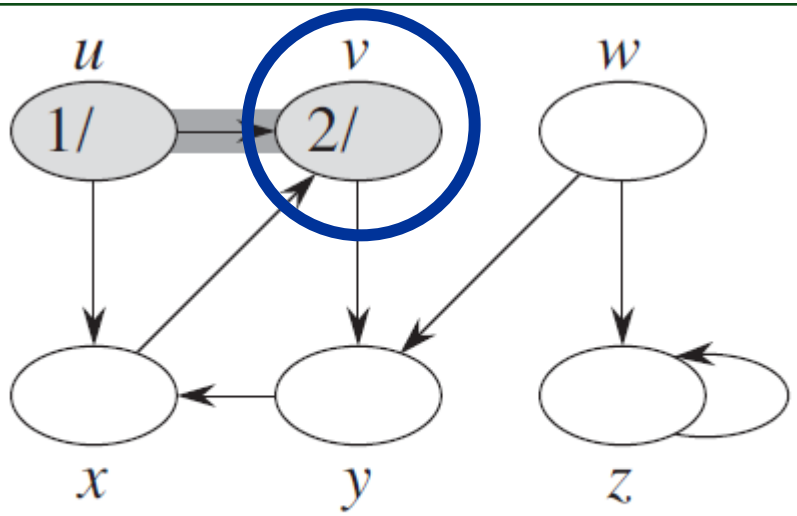
DFS: busca em profundidade



tempo = 1

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

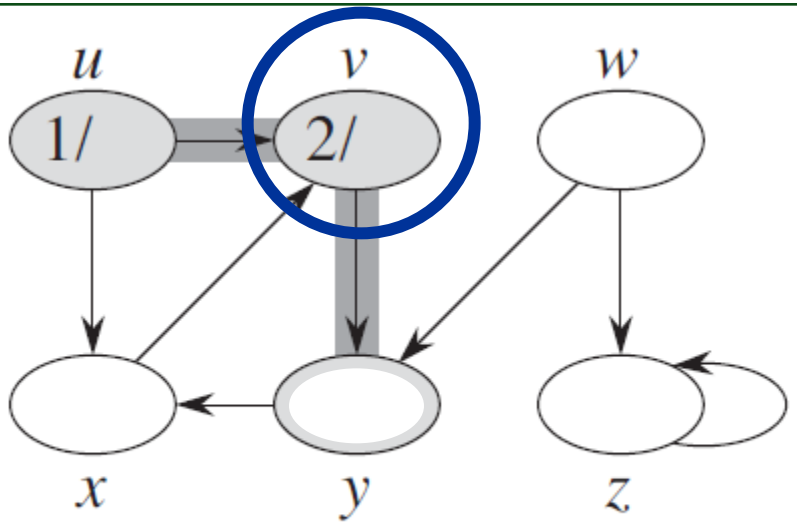
DFS: busca em profundidade



tempo = ~~1~~ 2

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

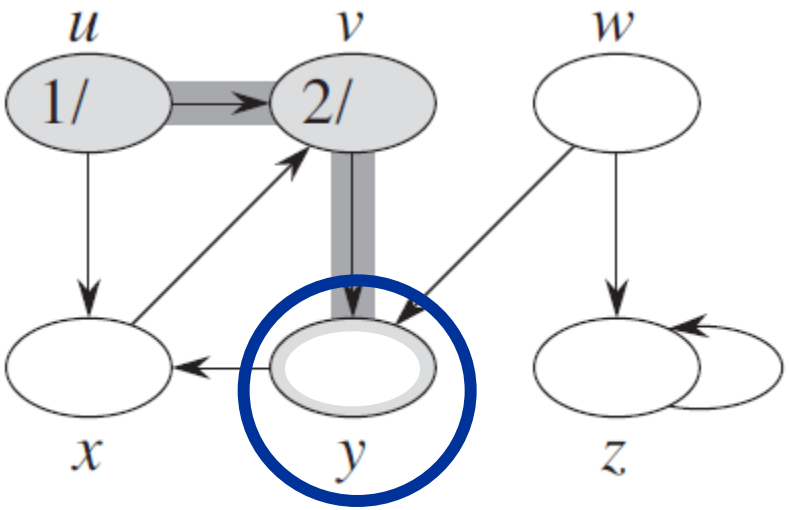


tempo = 2

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

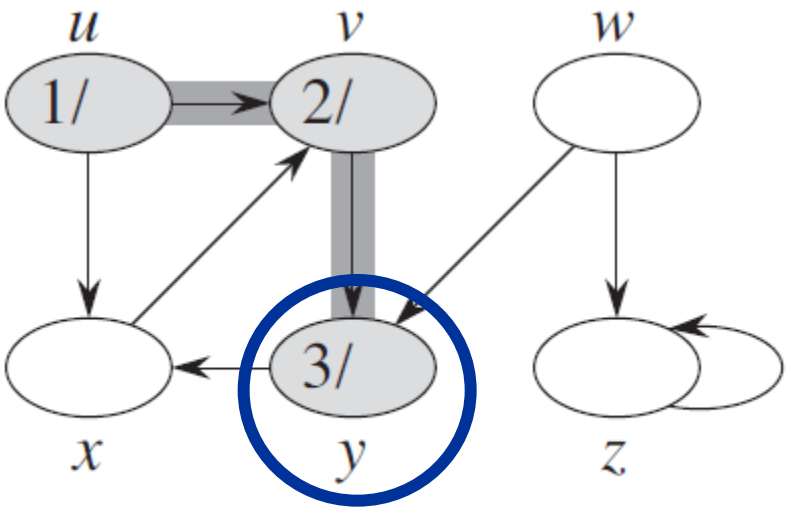
tempo = 2




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

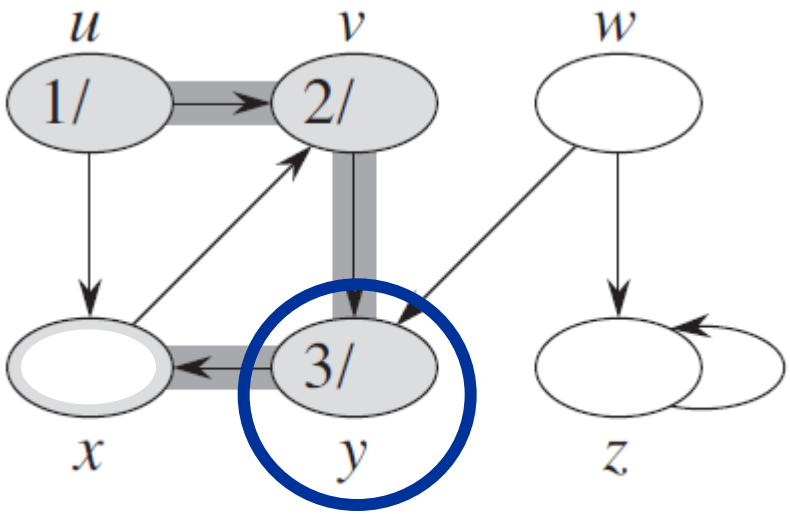
tempo = ~~2~~ 3




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

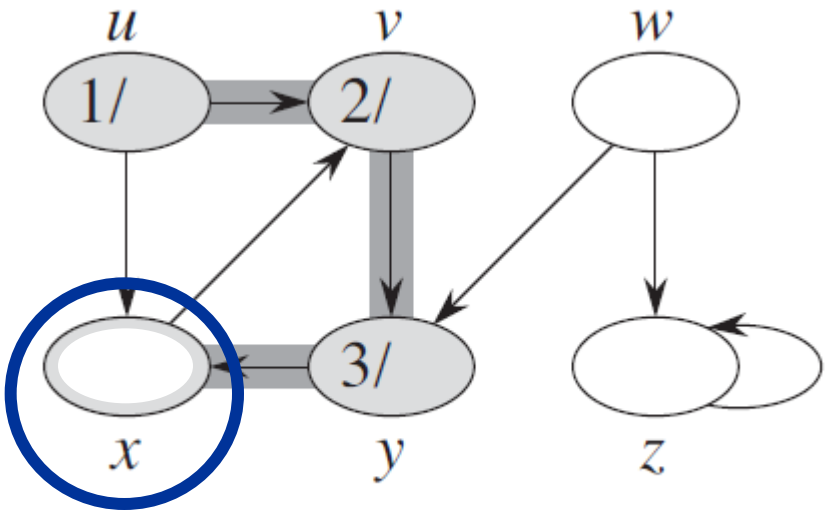
tempo = 3




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

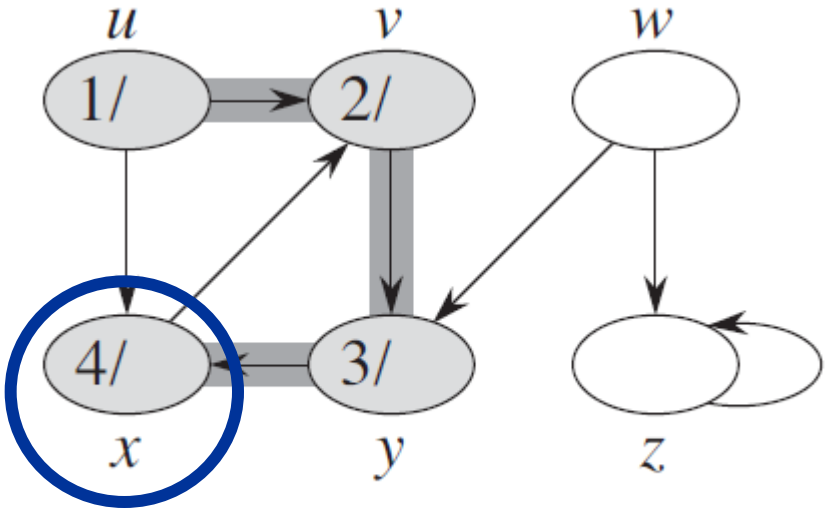
tempo = 3




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

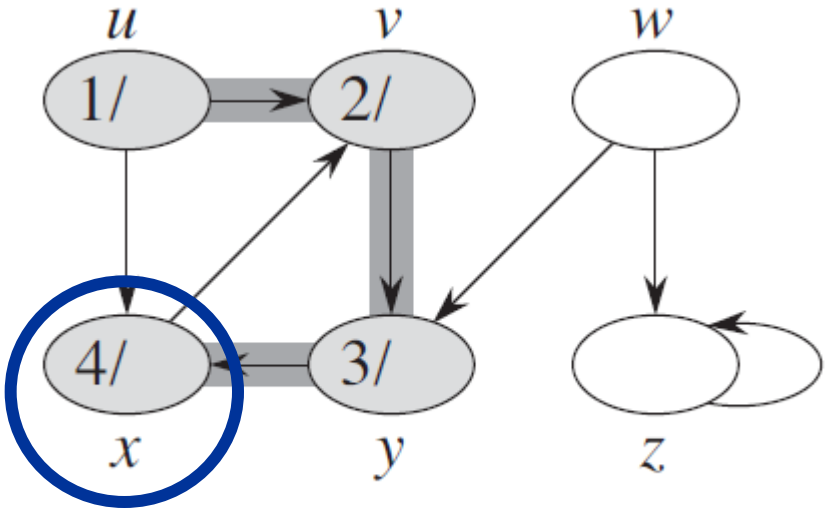
tempo = ~~3~~ 4




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

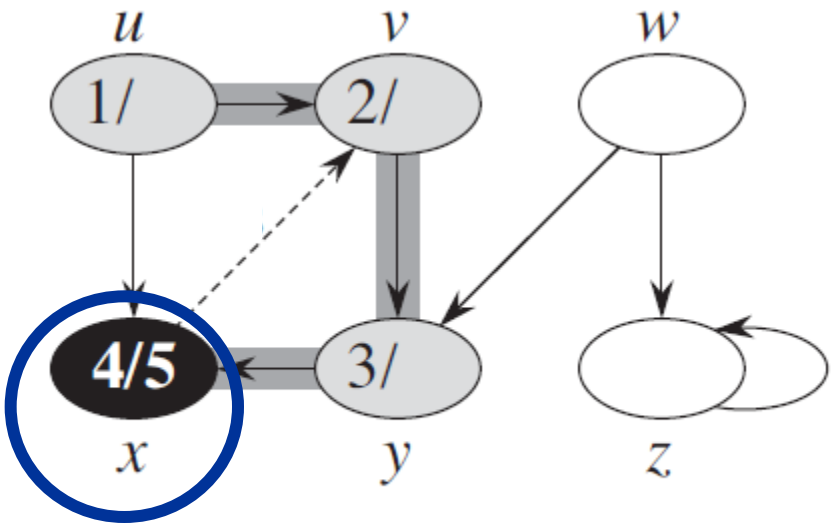
tempo = 4




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

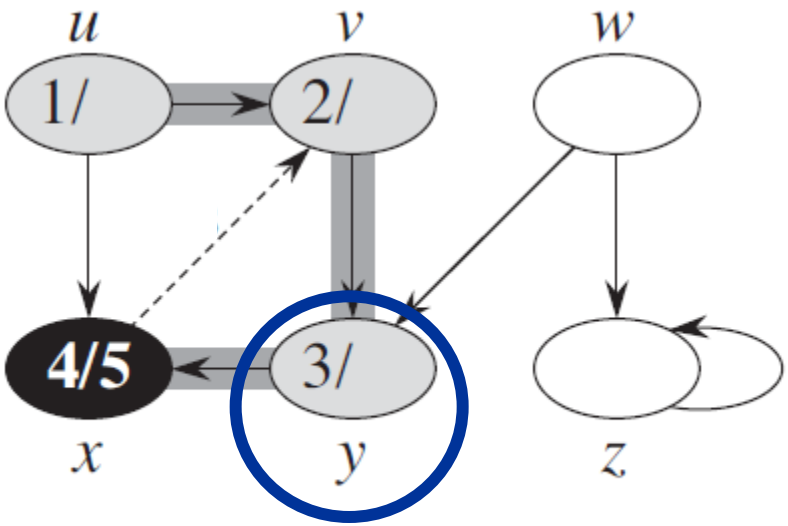
DFS: busca em profundidade

tempo = ~~4~~ 5




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

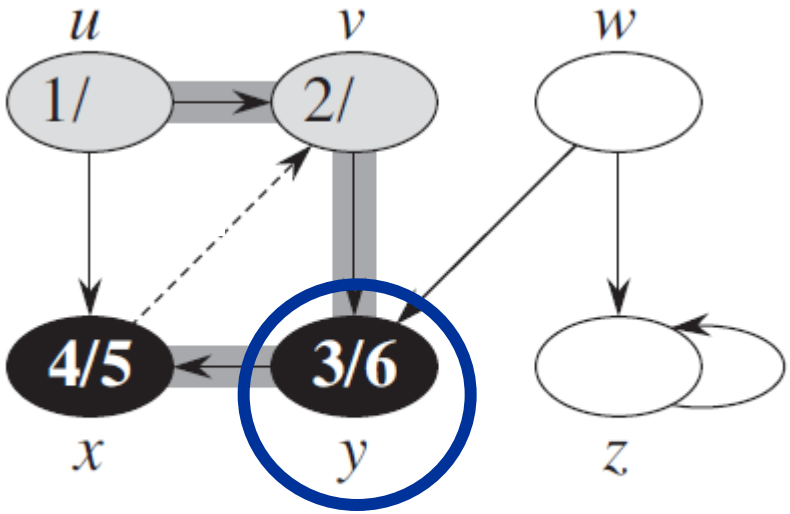


tempo = 5

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

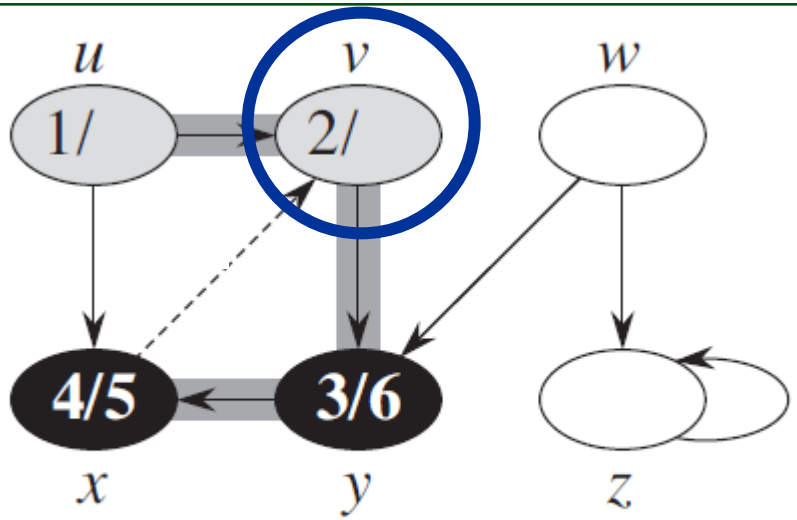
DFS: busca em profundidade

tempo = ~~5~~ 6




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

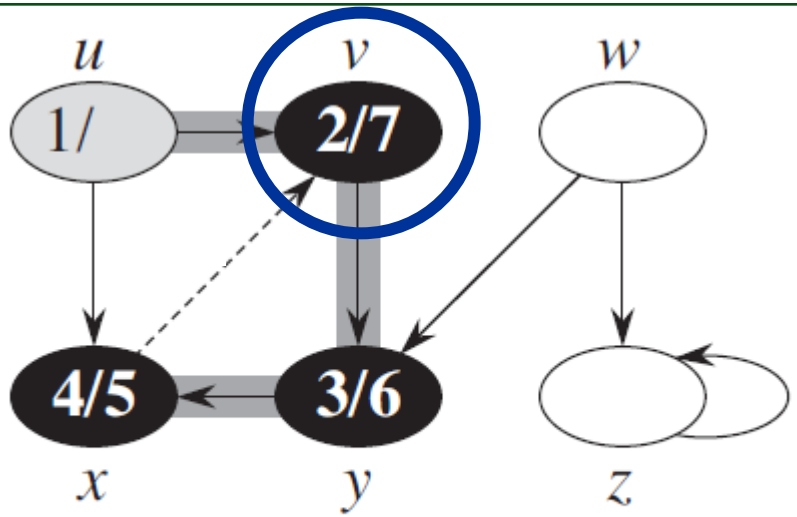


tempo = 6

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

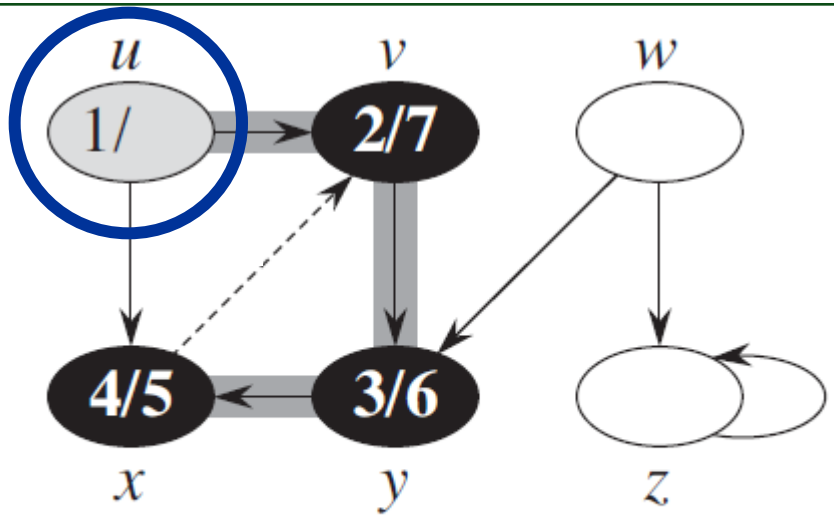
DFS: busca em profundidade

tempo = ~~6~~ 7




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

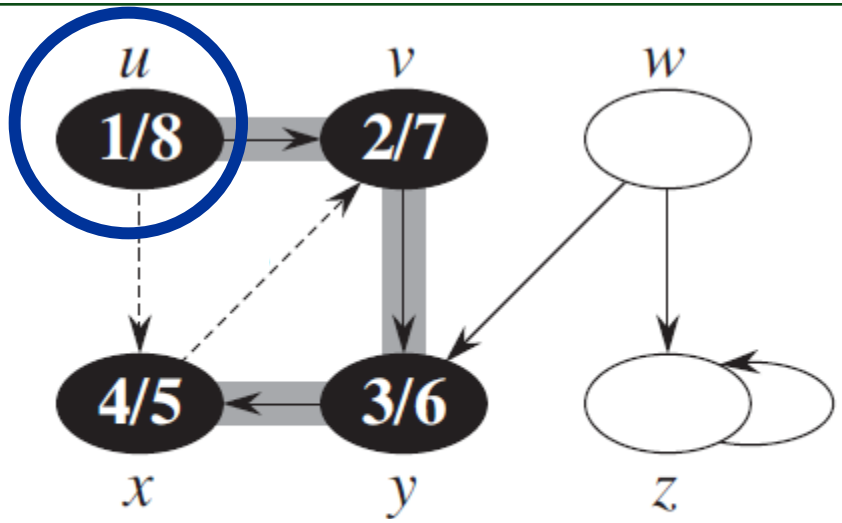
DFS: busca em profundidade



tempo = 7

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade



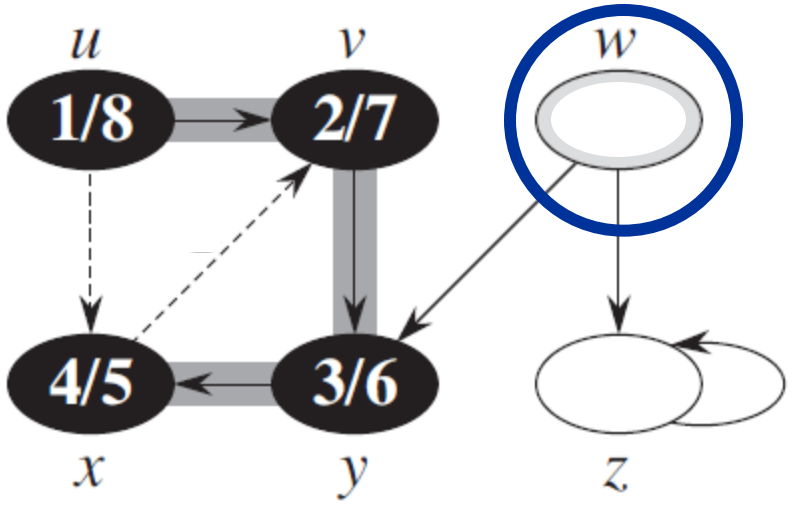
tempo = ~~7~~ 8

```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

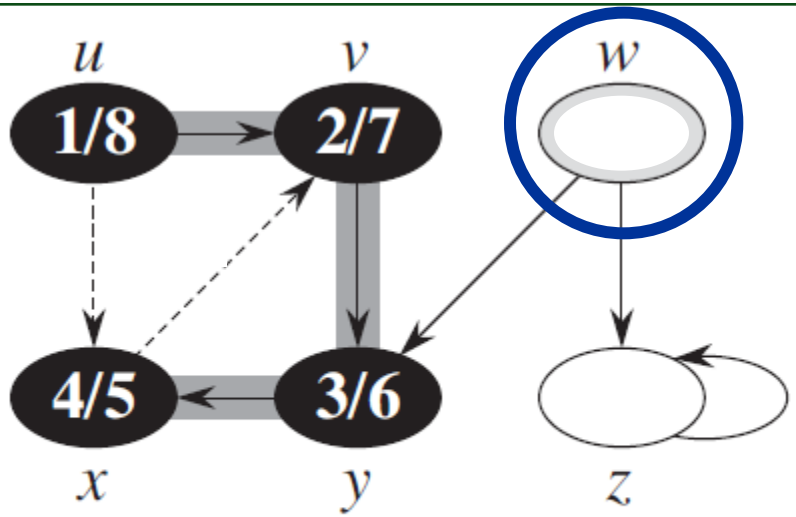
```
DFS(G)
 1 para cada vértice u_i em G.V faça
 2      u_i.cor = BRANCO
 3 tempo = 0
 4 para cada vértice u_i em G.V faça
 5      se u_i.cor == BRANCO
 6        entao VisitaDFS(G, u_i)
```

tempo = 8



DFS: busca em profundidade

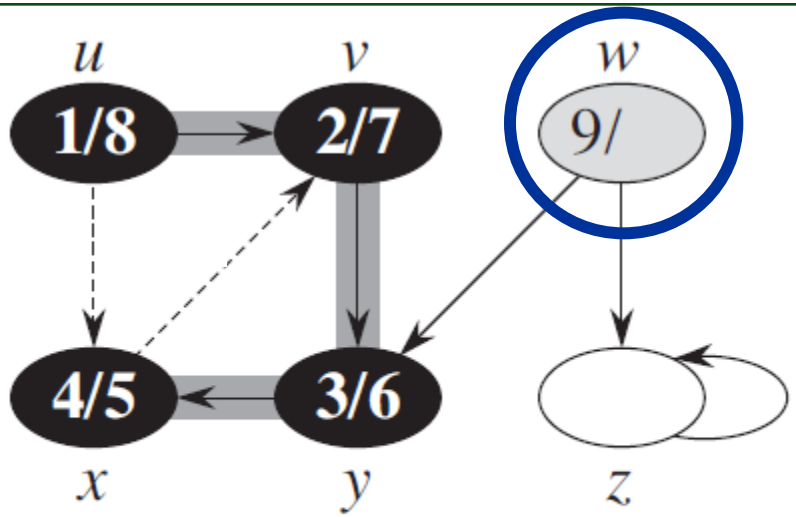
tempo = 8




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

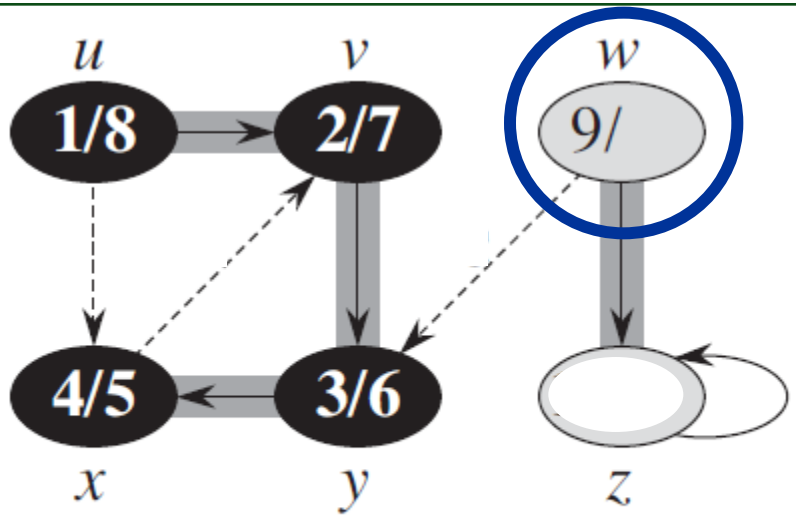
tempo = ~~8~~ 9




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

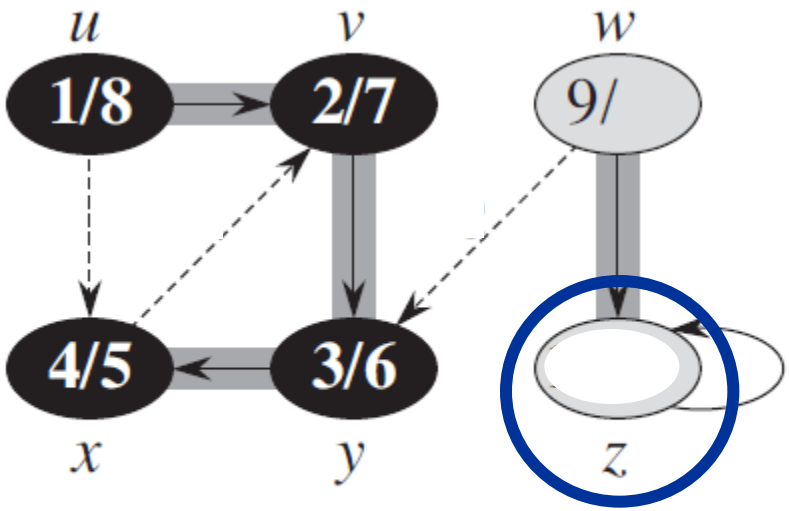
tempo = 9




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

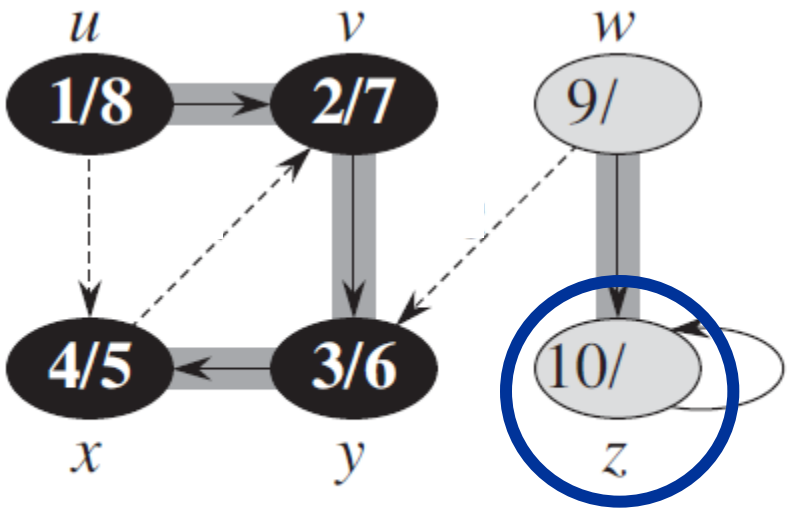
tempo = 9




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

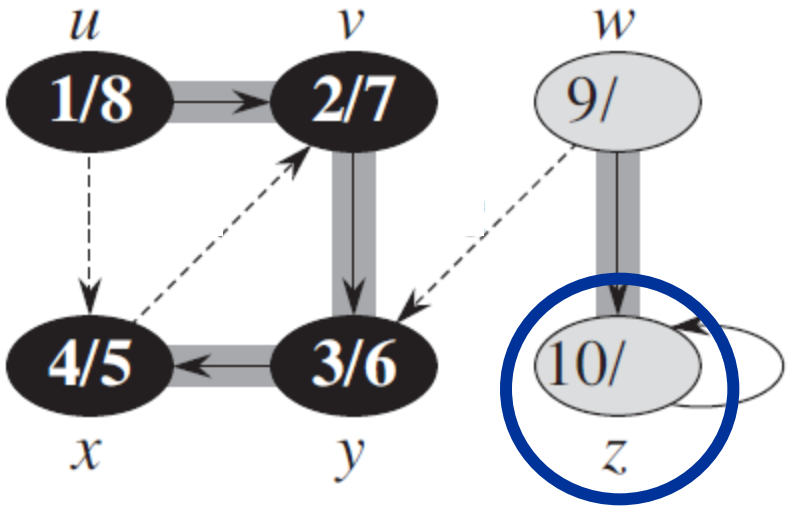
tempo = ~~9~~ 10




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

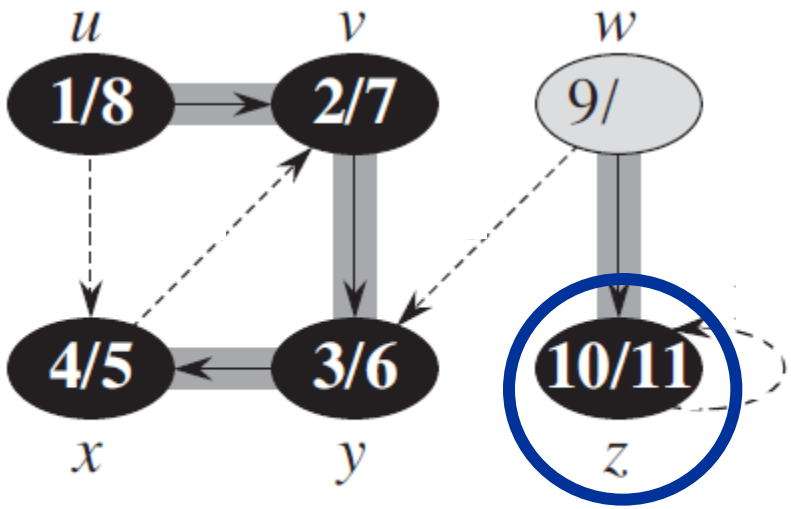
tempo = 10




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

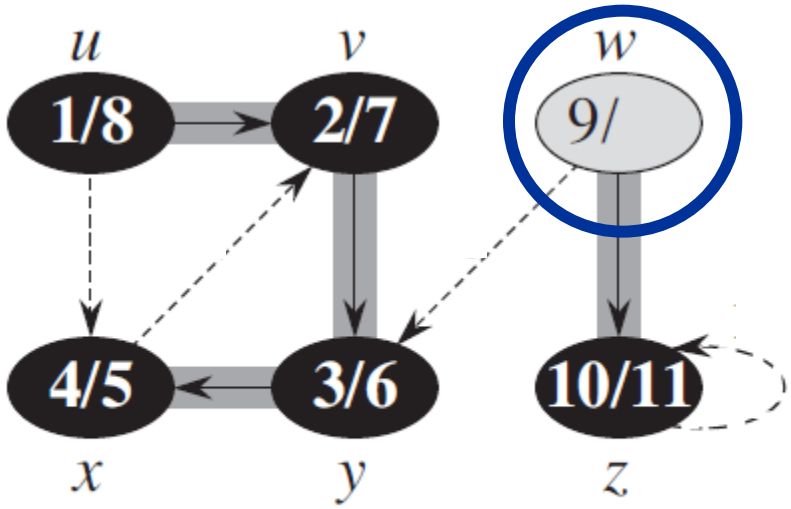
tempo = 1~~0~~ 11




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

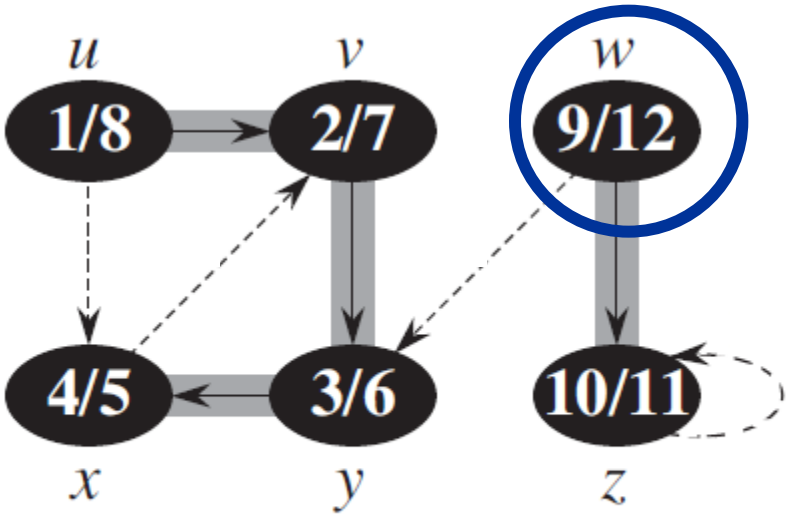
tempo = 11




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

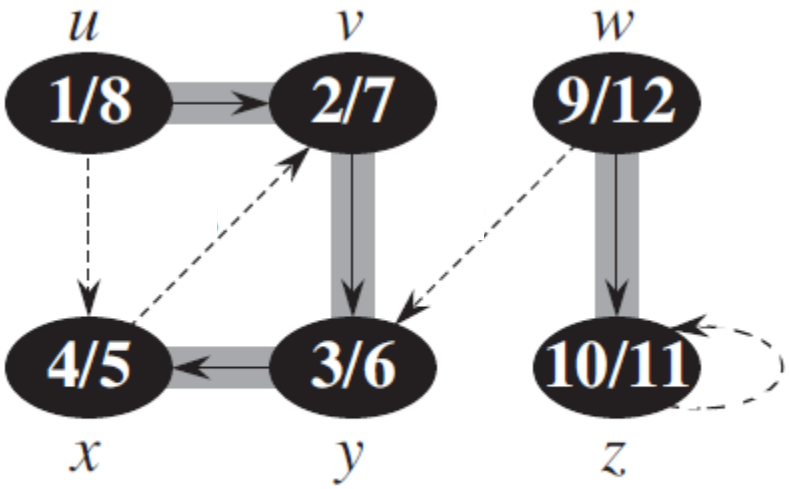
tempo = 1 12




```
VisitaDFS(G, u_i)
 1 tempo = tempo + 1
 2 u_i.d = tempo
 3 u_i.cor = CINZA
 4 para cada v_i em G.Adj[u_i]
 5      se v_i.cor == BRANCO
 6          VisitaDFS(G, v_i)
 7 tempo=tempo+1; u_i.f=tempo; u_i.cor=PRETO
```

DFS: busca em profundidade

```
DFS(G)
 1 para cada vértice u_i em G.V faça
 2      u_i.cor = BRANCO
 3 tempo = 0
 4 para cada vértice u_i em G.V faça
 5      se u_i.cor == BRANCO
 6        entao VisitaDFS(G, u_i)
```



Tarefa

- EP 3

- Página da disciplina:
  - https://sites.google.com/site/alexnoma/home/grafos