

Aula passada

- Algoritmo de busca em largura.

2

Busca em profundidade

3

Busca em profundidade

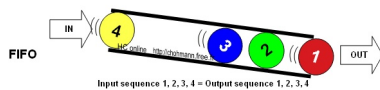
- A estratégia seguida pela busca em profundidade é, como seu nome implica, **procurar cada vez mais “fundo” no grafo**.
- Nessa busca as arestas são **exploradas a partir do vértice mais recentemente descoberto** que ainda possui arestas inexploradas saindo dele.
- Baseado em **Pilha**.



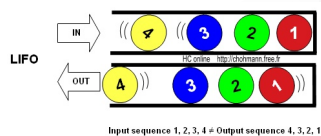
4

Busca em profundidade

Fila

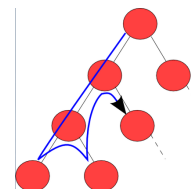


Pilha



5

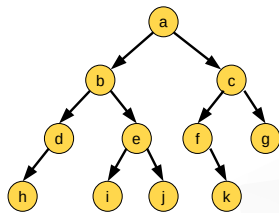
Busca em profundidade



- Inicialmente **{s}**.
- Os vértices são explorados a partir do vértice mais recentemente descoberto.

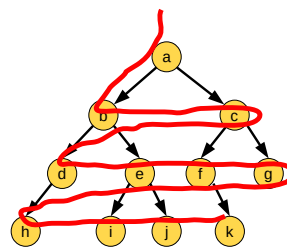
6

Busca em grafos

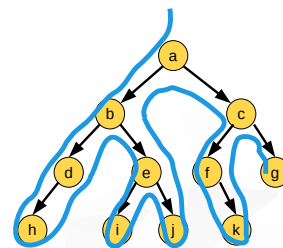


7

Busca em grafos



Busca em largura
(Breadth First Search - **BFS**)



Busca em profundidade
(Depth First Search - **DFS**)

8

Busca em profundidade

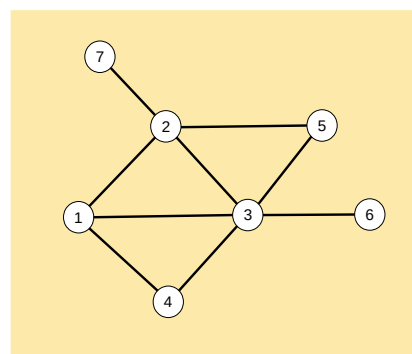
O algoritmo de busca em profundidade também **atribuirá cores** a cada vértice

- Cor **branca** = "não visitado". Inicialmente todos os vértices são brancos.
- Cor **cinza** = "visitado pela primeira vez".
- Cor **preta** = "teve seus vizinhos visitados".

9

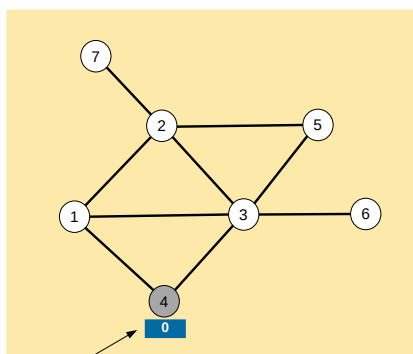
Busca em profundidade

Grafo inicial



10

Busca em profundidade (origem s=4)

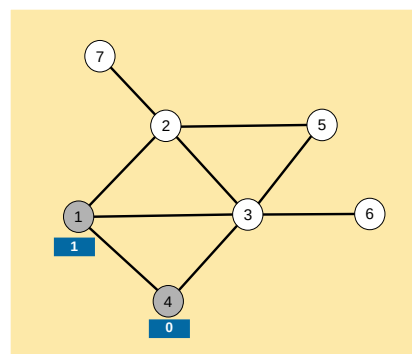


Pilha={4}

Indica tempo de visita

11

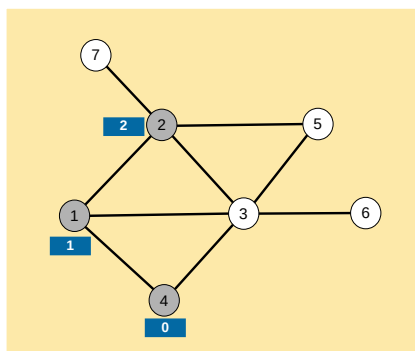
Busca em profundidade (origem s=4)



Pilha={4,1}

12

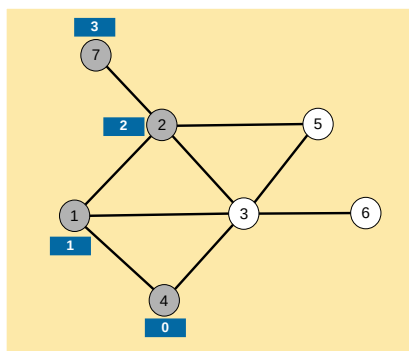
Busca em profundidade (origem s=4)



Pilha={4,1,2}

13

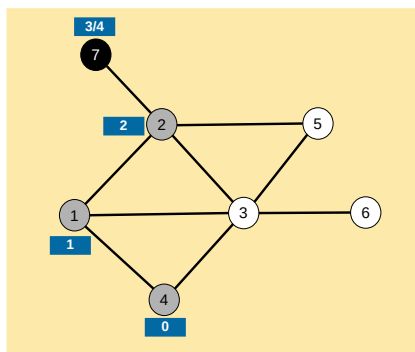
Busca em profundidade (origem s=4)



Pilha={4,1,2,7}

14

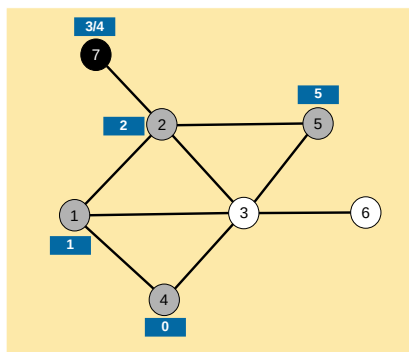
Busca em profundidade (origem s=4)



Pilha={4,1,2}

15

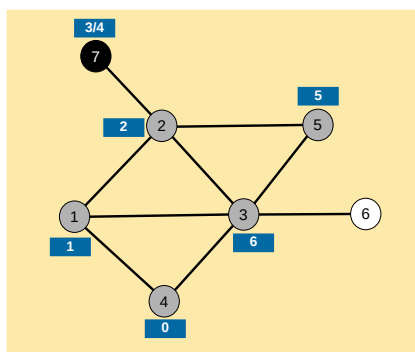
Busca em profundidade (origem s=4)



Pilha={4,1,2,5}

16

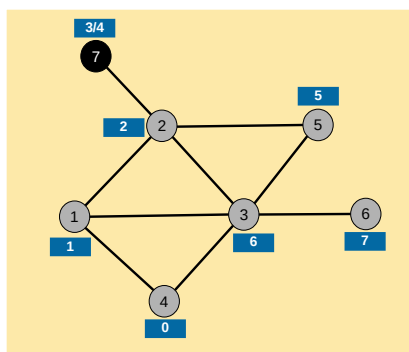
Busca em profundidade (origem s=4)



Pilha={4,1,2,5,3}

17

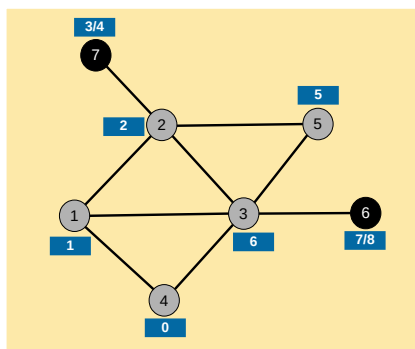
Busca em profundidade (origem s=4)



Pilha={4,1,2,5,3,6}

18

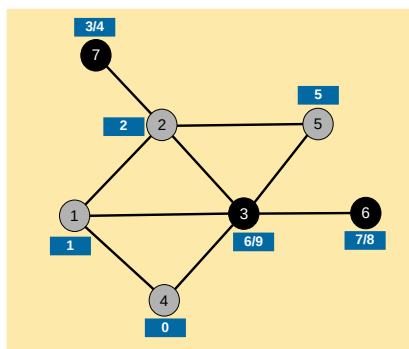
Busca em profundidade (origem s=4)



Pilha={4,1,2,5,3}

19

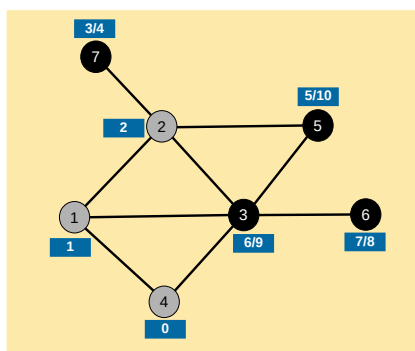
Busca em profundidade (origem s=4)



Pilha={4,1,2,5}

20

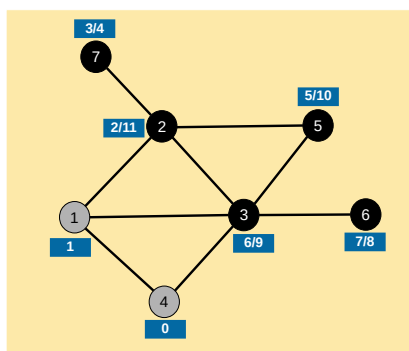
Busca em profundidade (origem s=4)



Pilha={4,1,2}

21

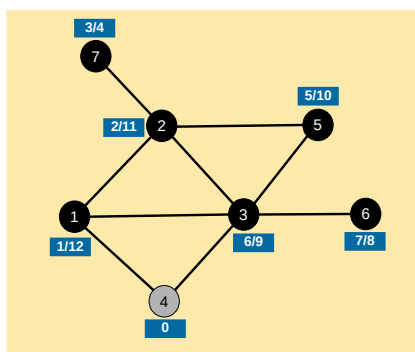
Busca em profundidade (origem s=4)



Pilha={4,1}

22

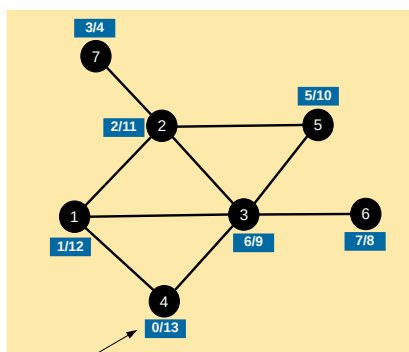
Busca em profundidade (origem s=4)



Pilha={4}

23

Busca em profundidade (origem s=4)



Pilha={}

Indica tempo de visita

24

Busca em profundidade (Algoritmo)

Constantes:

- BRANCO, CINZA, PRETO, INFINITO

Variáveis:

- P (Pilha) , s (vértice de origem)

Propriedades do vértice v:

- v.cor (cor do vértice)
- v.t1 (tempo de visita inicial do vértice v)
- v.t2 (tempo de visita final do vértice v)

Funções :

- Inserir(P,v), permite inserir o vértice v na pilha P.
- Remove(P), permite remover um vértice da pilha P.
- Consulta(P), permite consultar o último vértice na pilha P.

25

Busca em profundidade (Depth First Search)

DFS(G,s):

Para cada vértice v em G.V-{s} faça

v.cor = BRANCO
v.t1 = INFINITO
v.t2 = INFINITO

tempo = 0

s.cor = CINZA

s.t1 = tempo

P = VAZIO

Inserir(P,s)

Enquanto P ≠ VAZIO faça

u = Topo(P)

Se u tem pelo menos um vértice adjacente BRANCO

v = escolhe um dos vértices adjacentes com v.cor=BRANCO

v.cor = CINZA

tempo = tempo+1

v.t1 = tempo

Inserir(P,v)

Caso-contrário

u.cor = PRETO

tempo = tempo+1

v.t2 = tempo

Remove(P)

26

Busca em profundidade (Depth First Search)

DFS(G,s):

Para cada vértice v em G.V-{s} faça

Inicialização

v.cor = BRANCO
v.t1 = INFINITO
v.t2 = INFINITO

tempo = 0

s.cor = CINZA

s.t1 = tempo

P = VAZIO

Inserir(P,s)

Enquanto P ≠ VAZIO faça

Percorre o grafo

u = Topo(P)

Se u tem pelo menos um vértice adjacente BRANCO

v = escolhe um dos vértices adjacentes com v.cor=BRANCO

v.cor = CINZA

tempo = tempo+1

v.t1 = tempo

Inserir(P,v)

Caso-contrário

u.cor = PRETO

tempo = tempo+1

v.t2 = tempo

Remove(P)

27

Busca em profundidade (Depth First Search)

Para cada vértice v em G.V-{s} faça

Inicialização

v.cor = BRANCO
v.t1 = INFINITO
v.t2 = INFINITO

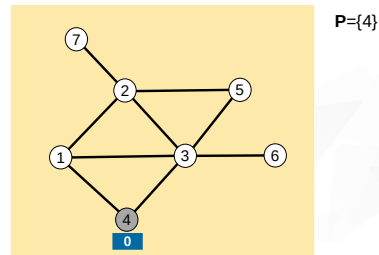
tempo = 0

s.cor = CINZA

s.t1 = tempo

P = VAZIO

Inserir(P,s)



28

Busca em profundidade (Depth First Search)

Enquanto P ≠ VAZIO faça

Percorre o grafo

u = Topo(P)

Se u tem pelo menos um vértice adjacente BRANCO

v = escolhe um dos vértices adjacentes com v.cor=BRANCO

v.cor = CINZA

tempo = tempo+1

v.t1 = tempo

Inserir(P,v)

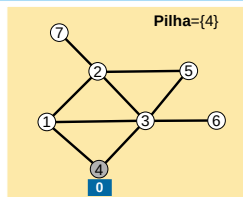
Caso-contrário

u.cor = PRETO

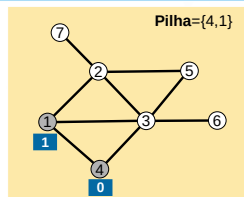
tempo = tempo+1

v.t2 = tempo

Remove(P)



Inicialização



Iteração 1

29

Busca em profundidade (Depth First Search)

Enquanto P ≠ VAZIO faça

Percorre o grafo

u = Topo(P)

Se u tem pelo menos um vértice adjacente BRANCO

v = escolhe um dos vértices adjacentes com v.cor=BRANCO

v.cor = CINZA

tempo = tempo+1

v.t1 = tempo

Inserir(P,v)

Caso-contrário

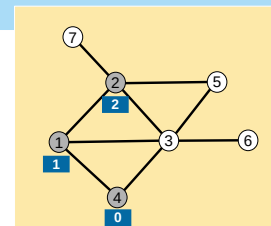
u.cor = PRETO

tempo = tempo+1

v.t2 = tempo

Remove(P)

Iteração 2



30

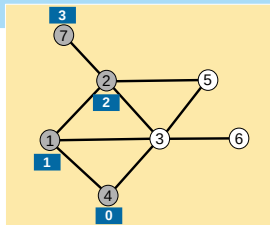
Busca em profundidade (Depth First Search)

```

Enquanto P ≠ VAZIO faça
  u = Topo(P)
  Se u tem pelo menos um vértice adjacente BRANCO
    v = escolhe um dos vértices adjacentes com v.cor=BRANCO
    v.cor = CINZA
    tempo = tempo+1
    v.t1 = tempo
    Insere(P,v)
  Caso-contrário
    u.cor = PRETO
    tempo = tempo+1
    v.t2 = tempo
    Remove(P)
    
```

Percorre o grafo

Iteração 3



Pilha={4,1,2,7}

31

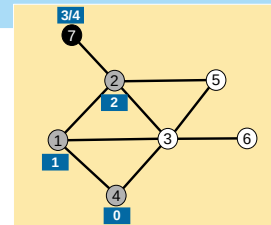
Busca em profundidade (Depth First Search)

```

Enquanto P ≠ VAZIO faça
  u = Topo(P)
  Se u tem pelo menos um vértice adjacente BRANCO
    v = escolhe um dos vértices adjacentes com v.cor=BRANCO
    v.cor = CINZA
    tempo = tempo+1
    v.t1 = tempo
    Insere(P,v)
  Caso-contrário
    u.cor = PRETO
    tempo = tempo+1
    v.t2 = tempo
    Remove(P)
    
```

Percorre o grafo

Iteração 4



Pilha={4,1,2}

32

Busca em profundidade

- O algoritmo anterior pode ser transformado a uma versão **RECURSIVA**.
Dado um grafo **G** e um vértice **v**, seja percorrido todo o grafo **G** usando a **busca em profundidade**.
- O algoritmo **recursivo**, é uma variante que **simplifica o uso da estrutura de pilha (P)**.

33

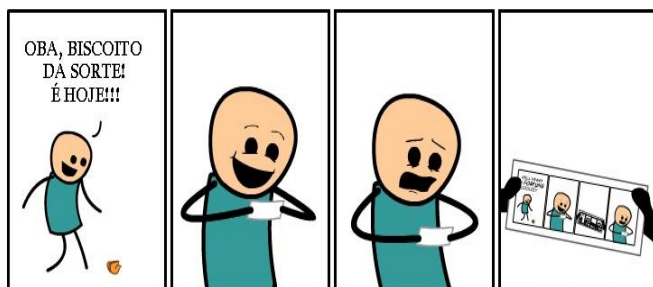
Recursividade

Uma função (programa) recursivo é uma função que se "chama a si mesma".



34

Recursividade



Recursividade é uma das coisas mágicas e interessantes em Programação.

35

Recursividade

Anuncio de cacao com uma imagem recursiva.



Recursividade

```
def contagem_regressiva(n):
    if n==0:
        print "Fogo!"
    else:
        print n
        contagem_regressiva(n-1)
```

37

Recursividade

Porque não usar Iteração ao invés de Recursividade?

Depende muito do estilo de programação. Entretanto, algumas vezes é mais apropriado usar Recursividade para resolver um problema.

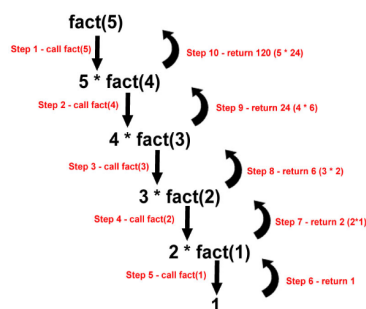
```
def contagem_regressiva(n):
    if n==0:
        print "Fogo!"
    else:
        print n
        contagem_regressiva(n-1)
```

```
def contagem_regressiva2(n):
    while n>0:
        print n
        n = n-1
    print "Fogo!"
```

38

Recursividade

```
def fact(n):
    if n==1:
        return 1
    else:
        return n*fact(n-1)
```



39

Busca em profundidade (versão recursiva)

```
DFS(G,s):
    Para cada vértice v em G.V-{s} faça
        v.cor = BRANCO
        v.t1 = INFINITO
        v.t2 = INFINITO
    tempo = 0
    VisitaDFS(G,s)
```

Inicialização

```
VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
```

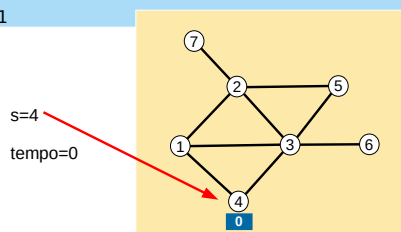
Percorre o grafo

40

Busca em profundidade (versão recursiva)

```
VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
```

Percorre o grafo

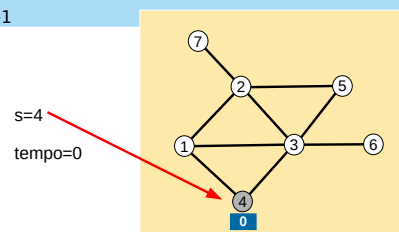


41

Busca em profundidade (versão recursiva)

```
VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
```

Percorre o grafo



42

Busca em profundidade (versão recursiva)

VisitaDFS(G,s):

s.t1 = tempo

s.cor = CINZA

tempo = tempo+1

→ Para cada v em G.Adj[s] faça

Se v.cor == BRANCO

VisitaDFS(G,v)

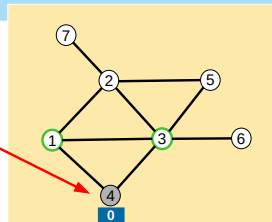
s.cor = PRETO

s.t2 = tempo

tempo = tempo+1

Percorre o grafo

s=4
tempo=1



43

Busca em profundidade (versão recursiva)

VisitaDFS(G,s):

s.t1 = tempo

s.cor = CINZA

tempo = tempo+1

→ Para cada v em G.Adj[s] faça

Se v.cor == BRANCO

VisitaDFS(G,v)

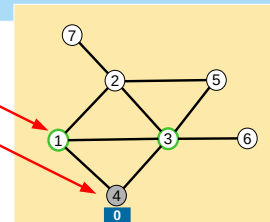
s.cor = PRETO

s.t2 = tempo

tempo = tempo+1

Percorre o grafo

v=1
s=4
tempo=2



44

Busca em profundidade (versão recursiva)

VisitaDFS(G,s):

s.t1 = tempo

s.cor = CINZA

tempo = tempo+1

→ Para cada v em G.Adj[s] faça

Se v.cor == BRANCO

VisitaDFS(G,v)

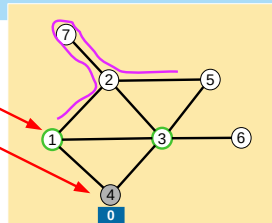
s.cor = PRETO

s.t2 = tempo

tempo = tempo+1

Percorre o grafo

v=1
s=4
tempo=3



45

Busca em profundidade (versão recursiva)

VisitaDFS(G,s):

s.t1 = tempo

s.cor = CINZA

tempo = tempo+1

→ Para cada v em G.Adj[s] faça

Se v.cor == BRANCO

VisitaDFS(G,v)

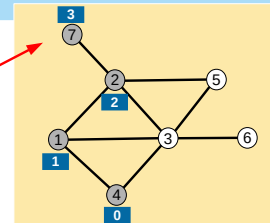
s.cor = PRETO

s.t2 = tempo

tempo = tempo+1

Percorre o grafo

s=7
tempo=4



Após alguns chamados
recursivos...

46

Busca em profundidade (versão recursiva)

VisitaDFS(G,s):

s.t1 = tempo

s.cor = CINZA

tempo = tempo+1

Para cada v em G.Adj[s] faça

Se v.cor == BRANCO

VisitaDFS(G,v)

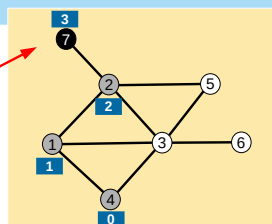
→ s.cor = PRETO

s.t2 = tempo

tempo = tempo+1

Percorre o grafo

s=7
tempo=4



Após alguns chamados
recursivos...

47

Busca em profundidade (versão recursiva)

VisitaDFS(G,s):

s.t1 = tempo

s.cor = CINZA

tempo = tempo+1

Para cada v em G.Adj[s] faça

Se v.cor == BRANCO

VisitaDFS(G,v)

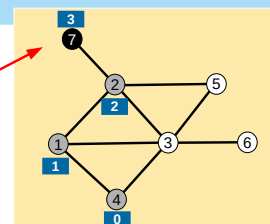
s.cor = PRETO

s.t2 = tempo

tempo = tempo+1

Percorre o grafo

s=7
tempo=4



Após alguns chamados
recursivos...

48

Busca em profundidade (versão recursiva)

```
VisitaDFS(G,s):  
    s.t1 = tempo  
    s.cor = CINZA  
    tempo = tempo+1  
    Para cada v em G.Adj[s] faça  
        Se v.cor == BRANCO  
            VisitaDFS(G,v)  
    s.cor = PRETO  
    s.t2 = tempo  
    tempo = tempo+1
```

Percorre o grafo

The diagram shows a graph with 7 nodes (1-7) and a search tree structure. Node 7 is the root of the tree, with a blue box containing '3/4' above it. Node 7 is connected to nodes 2 and 5. Node 2 is connected to nodes 1 and 3. Node 3 is connected to nodes 1, 4, and 6. Node 1 is connected to node 4. Node 4 is connected to node 0. A red arrow points from the text 's=7' to node 7. The text 'tempo=5' is also present.

$s=7$

tempo=5

Após alguns chamados recursivos...

49

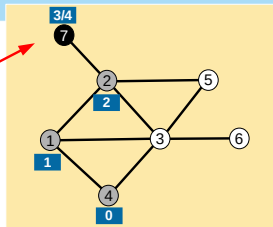
```
s.t1 = tempo
s.cor = CINZA
tempo = tempo+1
Para cada v em G.Adj[s] faça
    Se v.cor == BRANCO
        VisitaDFS(G,v)
s.cor = PRETO
s.t2 = tempo
tempo = tempo+1
```

s=7

tempo=5

Após alguns chamados
recursivos...

49



Busca em profundidade (versão recursiva)

```

VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
            
```

Percorre o grafo

→

s=7

tempo=5

Após alguns chamados recursivos...

50

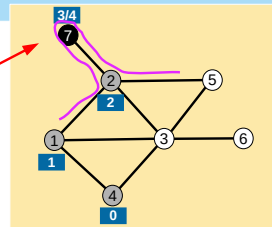
```
s.t1 = tempo
s.cor = CINZA
tempo = tempo+1
Para cada v em G.Adj[s] faça
    Se v.cor == BRANCO
        VisitaDFS(G,v)
s.cor = PRETO
s.t2 = tempo
tempo = tempo+1
```

s=7

tempo=5

Após alguns chamados
recursivos...

57



Busca em profundidade (versão recursiva)

```

VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            → VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
            
```

Percorre o grafo

s=5

tempo=5

Após alguns chamados recursivos...

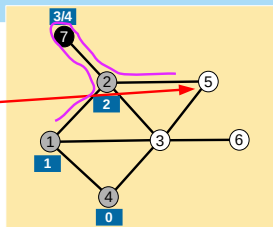
```
s.t1 = tempo
s.cor = CINZA
tempo = tempo+1
Para cada v em G.Adj[s] faça
    Se v.cor = BRANCO
        VisitaDFS(G,v)
s.cor = PRETO
s.t2 = tempo
tempo = tempo+1
```

s=5

tempo=5

Após alguns chamados
recursivos...

51



Busca em profundidade (versão recursiva)

```

VisitaDFS(G,s):
→ s.t1 = tempo
  s.cor = CINZA
  tempo = tempo+1
  Para cada v em G.Adj[s] faça
    Se v.cor == BRANCO
      VisitaDFS(G,v)
  s.cor = PRETO
  s.t2 = tempo
  tempo = tempo+1
        
```

Percorre o grafo

s=5

tempo=5

Após alguns chamados recursivos...

52

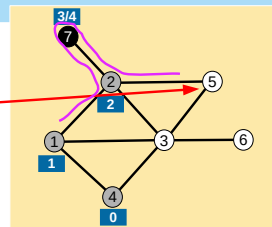
```
s.t1 = tempo
s.cor = CINZA
tempo = tempo+1
Para cada v em G.Adj[s] faça
    Se v.cor == BRANCO
        VisitaDFS(G,v)
s.cor = PRETO
s.t2 = tempo
tempo = tempo+1
```

s=5

tempo=5

Após alguns chamados
recursivos...

52



Busca em profundidade (versão recursiva)

```

VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
        
```

Percorre o grafo

s=5

tempo=5

Após alguns chamados recursivos...

53

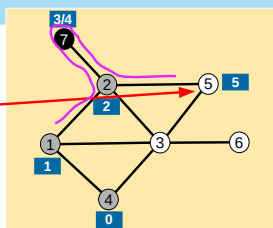
```
s.t1 = tempo
s.cor = CINZA
tempo = tempo+1
Para cada v em G.Adj[s] faça
    Se v.cor = BRANCO
        VisitaDFS(G,v)
s.cor = PRETO
s.t2 = tempo
tempo = tempo+1
```

s=5

tempo=5

Após alguns chamados
recursivos...

53



Busca em profundidade (versão recursiva)

```

VisitaDFS(G,s):
    s.t1 = tempo
    s.cor = CINZA
    tempo = tempo+1
    Para cada v em G.Adj[s] faça
        Se v.cor == BRANCO
            VisitaDFS(G,v)
    s.cor = PRETO
    s.t2 = tempo
    tempo = tempo+1
        
```

Percorre o grafo

→ s=5

tempo=6

Após alguns chamados recursivos...

94

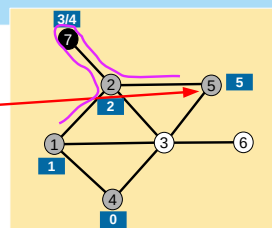
```
s.t1 = tempo
s.cor = CINZA
tempo = tempo+1
Para cada v em G.Adj[s] faça
    Se v.cor == BRANCO
        VisitaDFS(G,v)
s.cor = PRETO
s.t2 = tempo
tempo = tempo+1
```

s=5

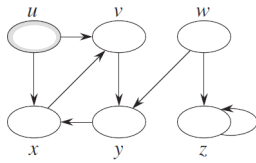
tempo=6

Após alguns chamados
recursivos...

54

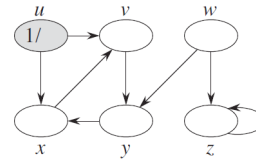


Para grafos direccionados



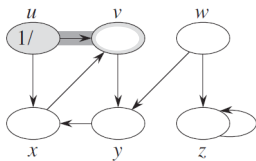
55

Para grafos direccionados



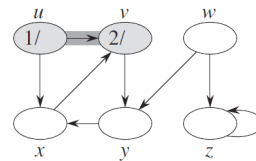
56

Para grafos direccionados



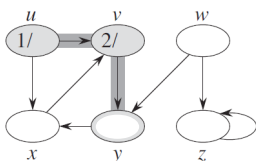
57

Para grafos direccionados



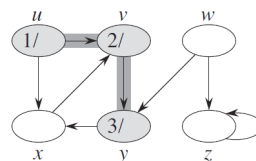
58

Para grafos direccionados



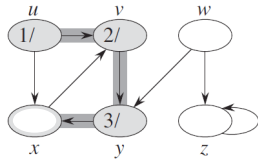
59

Para grafos direccionados



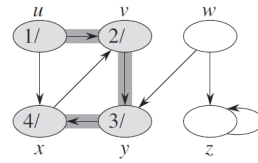
60

Para grafos direccionados



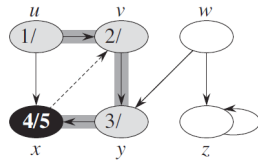
61

Para grafos direccionados



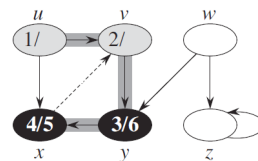
62

Para grafos direccionados



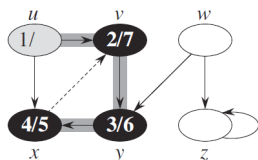
63

Para grafos direccionados



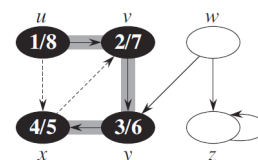
64

Para grafos direccionados



65

Para grafos direccionados



66

Busca em profundidade

// Esta função permite percorrer os elementos da componente conexa contendo s.

DFS(G, s):

Para cada vértice v em G.V-{s} faça

Inicialização

v.cor = BRANCO

v.t1 = INFINITO

v.t2 = INFINITO

tempo = 0

VisitaDFS(G, s)

// Esta nova função permite percorrer todos os elementos do grafo

DFS(G):

Para cada vértice v em G.V faça

Inicialização

v.cor = BRANCO

v.t1 = INFINITO

v.t2 = INFINITO

tempo = 0

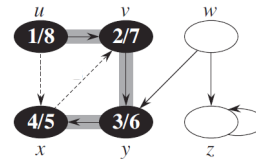
Para cada vértice u em G.V faça

se u.cor==BRANCO

VisitaDFS(G, u)

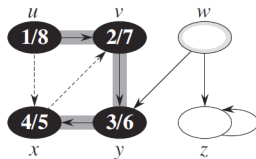
67

Para grafos direcionados



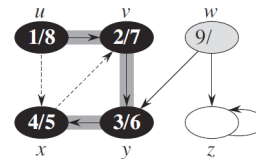
68

Para grafos direcionados



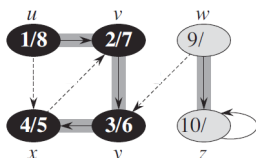
69

Para grafos direcionados



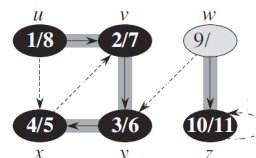
70

Para grafos direcionados



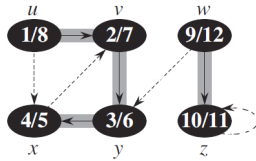
71

Para grafos direcionados



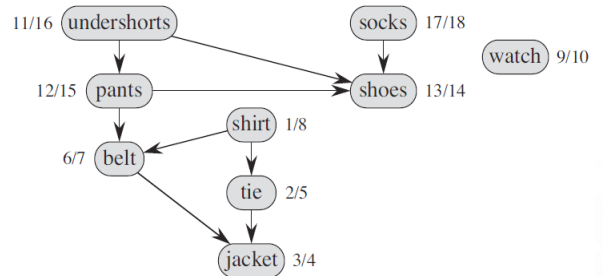
72

Para grafos direcionados



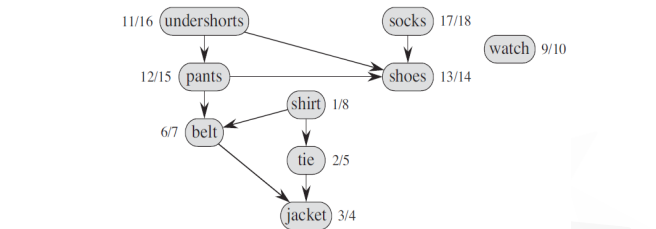
73

Ordenação 'topológica'



74

Ordenação 'topológica'



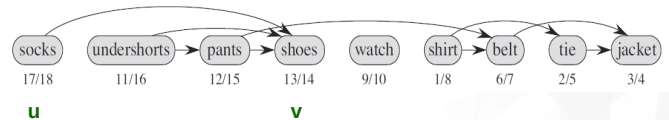
75

Ordenação 'topológica'

A **ordenação topológica** (de um grafo direcionado) é uma **ordem linear** de seus vértices em que:

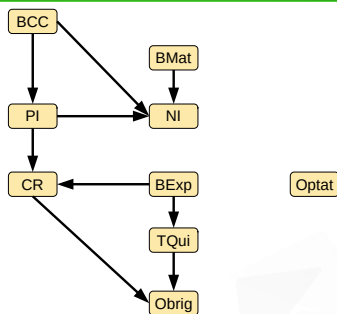
- Cada aresta direcionada uv (do vértice u ao vértice v), o vértice u vem antes do vértice v na ordenação.

Podem existir uma ou mais ordenações topológicas.



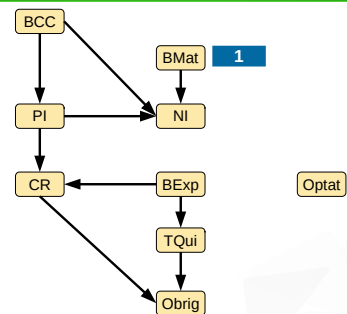
76

Ordenação 'topológica'



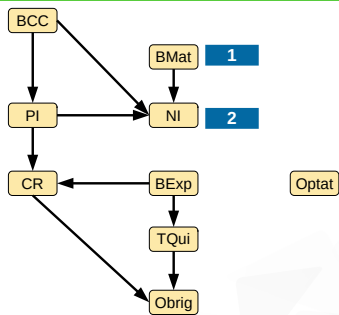
77

Ordenação 'topológica'



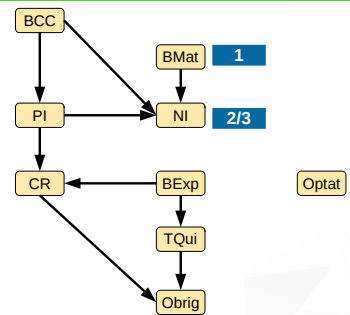
78

Ordenação 'topologica'



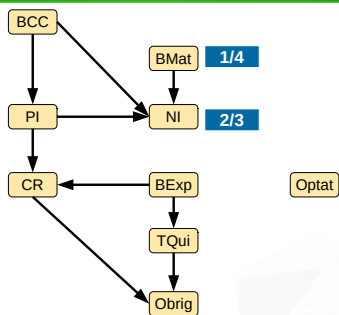
79

Ordenação 'topologica'



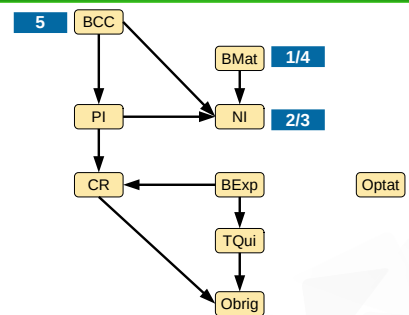
80

Ordenação 'topologica'



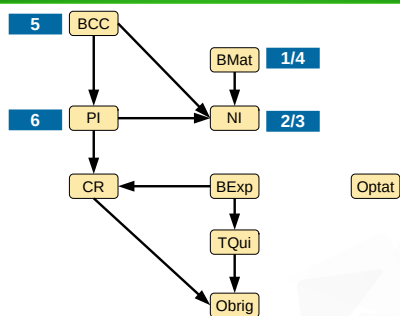
81

Ordenação 'topologica'



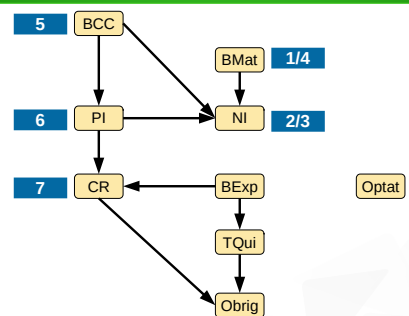
82

Ordenação 'topologica'



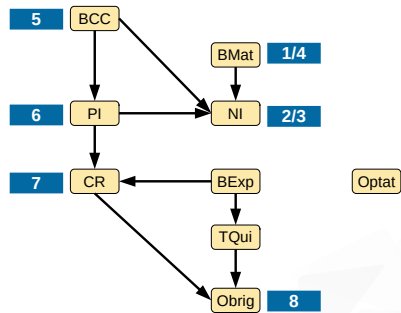
83

Ordenação 'topologica'



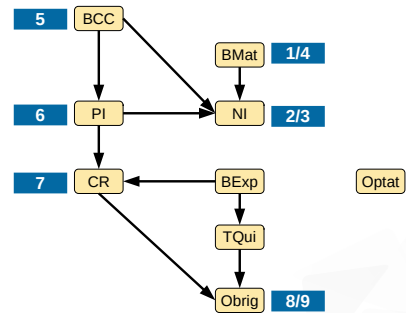
84

Ordenação 'topologica'



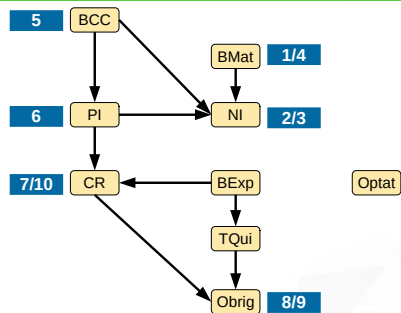
85

Ordenação 'topologica'



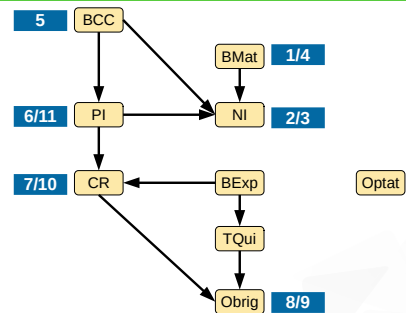
86

Ordenação 'topologica'



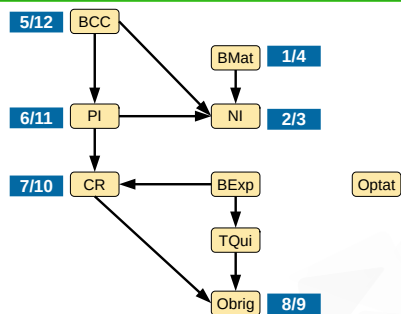
87

Ordenação 'topologica'



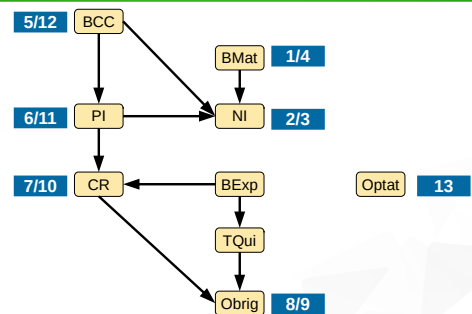
88

Ordenação 'topologica'



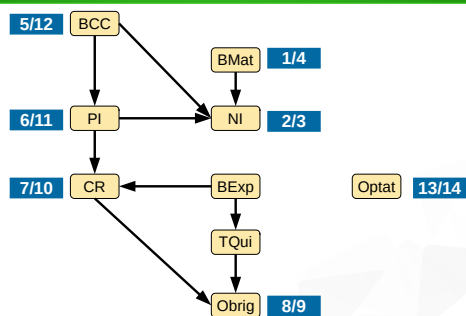
89

Ordenação 'topologica'



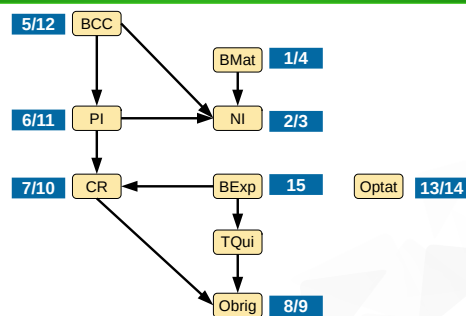
90

Ordenação 'topologica'



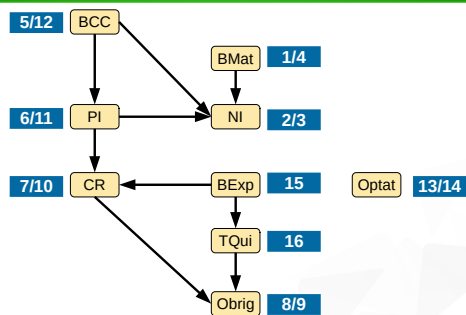
91

Ordenação 'topologica'



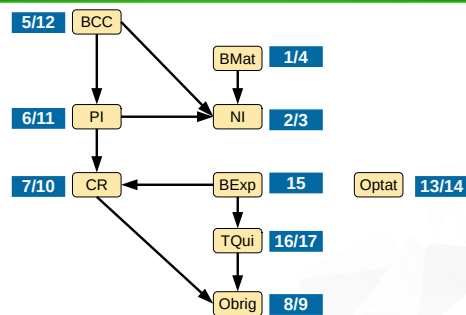
92

Ordenação 'topologica'



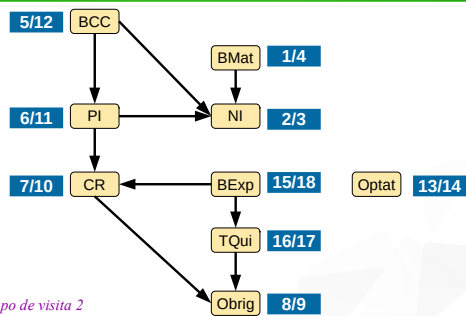
93

Ordenação 'topologica'



94

Ordenação 'topologica'



Ordenado pelo tempo de visita 2
(ordem decrescente)



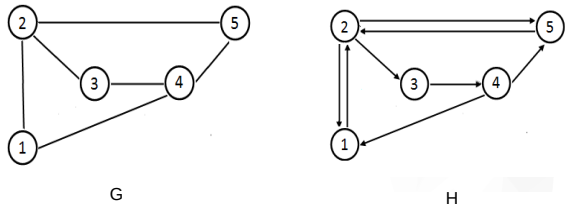
III. Atividade Prática

96

Atividade Prática

1. Para os grafos G e H abaixo, execute a **busca em profundidade** a partir do **vértice 1**:
- (a) Dando preferência para vértices de **menor** índice.
 - (b) Dando preferência para vértices de **maior** índice.

Para cada exercício indique a sequência de vértices visitados.



97

Atividade Prática

Grafo G:

- (a) Dando preferência para vértices de **menor** índice.
<1,2,3,4,5>
- (b) Dando preferência para vértices de **maior** índice.
<1,4,5,2,3>

Grafo H:

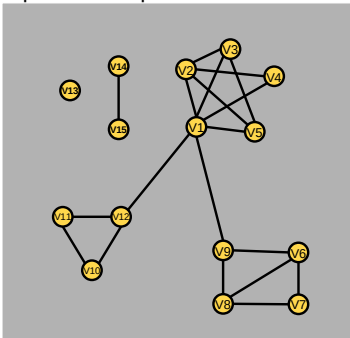
- (a) Dando preferência para vértices de **menor** índice.
<1,2,3,4,5>
- (b) Dando preferência para vértices de **maior** índice.
<1,2,5,3,4>

98

Atividade Prática

2. Execute o algoritmo de Busca em Profundidade a partir do vértice 1 do grafo. Indique a sequência de vértices visitados, considerando na busca a preferência para vértices de:

- a) menor índice
- b) maior índice



99

Atividade Prática

2. Execute o algoritmo de Busca em Profundidade a partir do vértice 1 do grafo. Indique a sequência de vértices visitados, considerando na busca a preferência para vértices de:

(a) menor índice	1,2,3,5,4,9,6,7,8,12,10,11
(b) maior índice	1,12,11,10,9,8,7,6,5,3,2,4

100