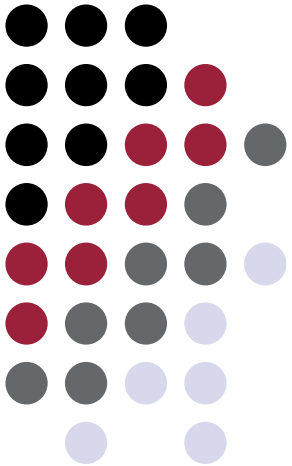
A06 Busca em Largura e em Profundidade

CSI466 – Teoria dos Grafos

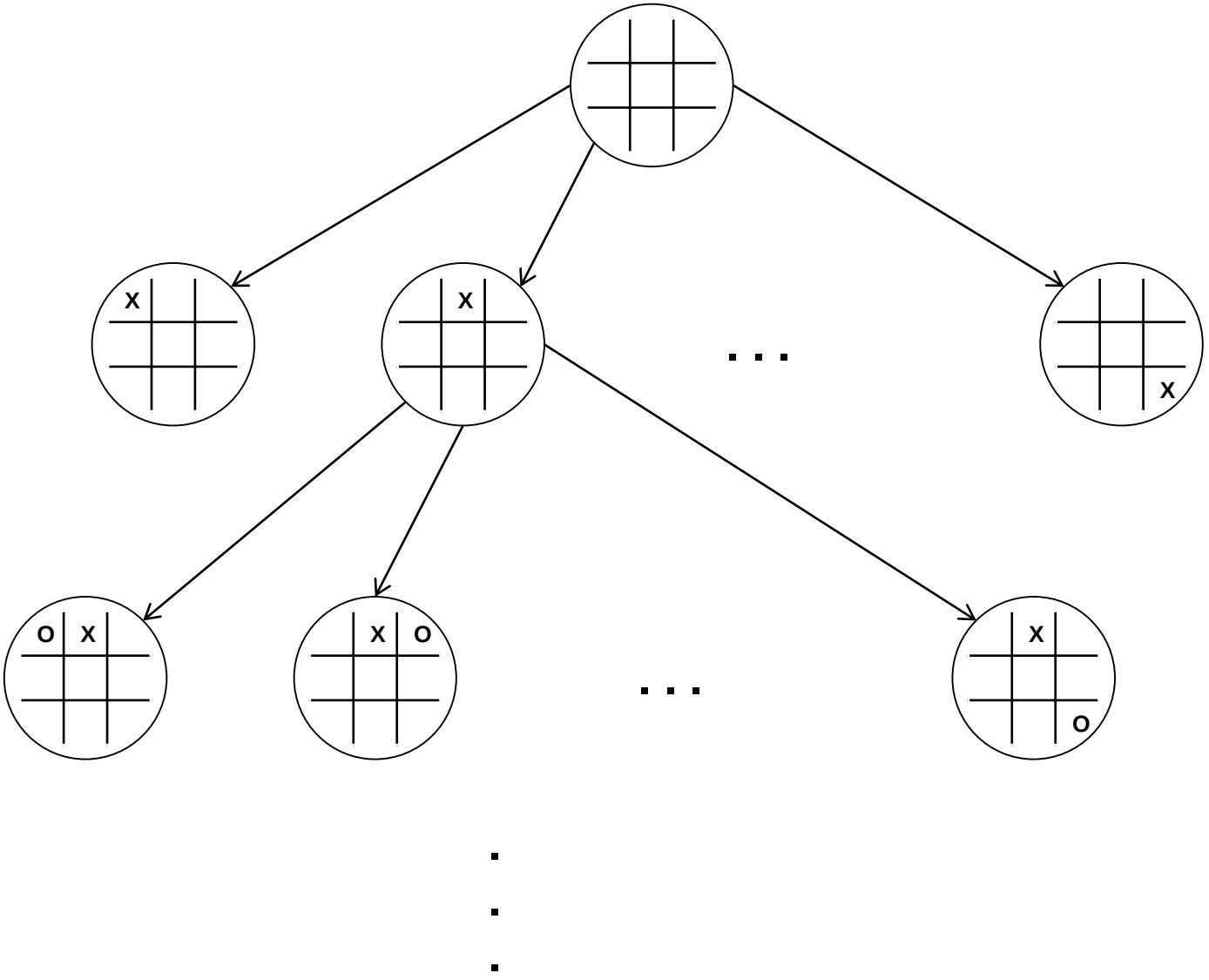
Prof. Dr. George H. G. Fonseca
Universidade Federal de Ouro Preto



Introdução

- Boa parte dos problemas da Inteligência Artificial podem ser representados como busca em grafo
  - Representa-se um estado como um vértice e uma alteração de estado como aresta levando a um novo vértice (estado)
  - Imagine procurar no jogo da velha uma solução em que o jogador ‘X’ ganha...
    - Para *n* posições, grafo com *n*! + 1 vértices
    - Supondo 9 posições, 362.880 vértices
  - Agora, imagine para um jogo de xadrez, onde há diversos movimentos possíveis! ... 10^{120} estados possíveis

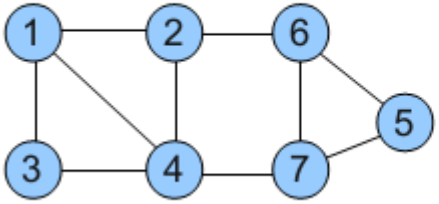
Introdução

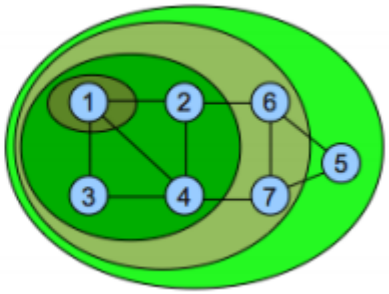


Busca em Largura

- Ideia: processar os vértices por níveis, começando por aqueles vértices mais próximos à origem *s* e deixando os mais distantes para depois

- O algoritmo descobre todos os vértices com distância k antes de descobrir os vértices com distância k + 1





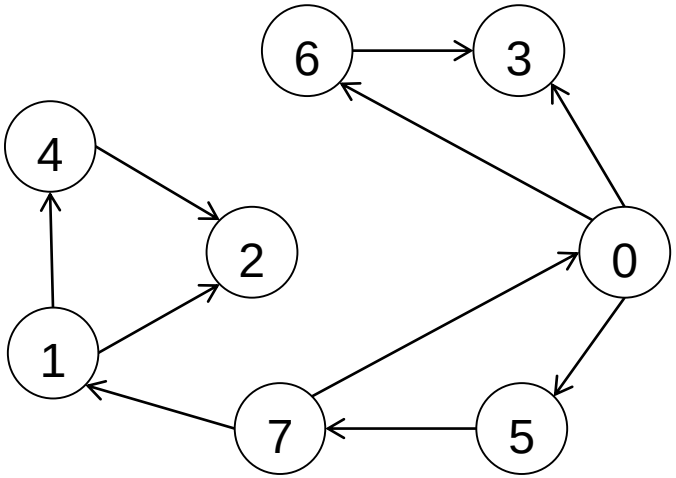
Busca em Largura

```
BUSCA-LARGURA(G(V, E), s)
1.  visitado[ ]
2.  para cada vértice v em V faça
3.      visitado[v] ← 0
4.  fim-para
5.  visitado[s] ←  1
6.  imprime s
7.  coloque s em uma fila f
8.  enquanto fila f não vazia faça
9.      u ← retirar o elemento da frente da fila
10.     para cada vértice v adjacente a u faça
11.         se visitado[v] = 0 então
12.             visitado[v] ← 1 //encontrou v
13.             imprime v
14.             colocar v no final da fila f
15.         fim-se
16.     fim-para
17. fim-enquanto
FIM
```

O comando **imprime** pode ser substituído por qualquer operação em s, dependendo da aplicação. Ex.: se os vétices representassem estados de um jogo essa etapa poderia verificar se o jogador ganha nesse estado.

Complexidade de tempo O(|V|+|E|)

Busca em Largura



s = 0

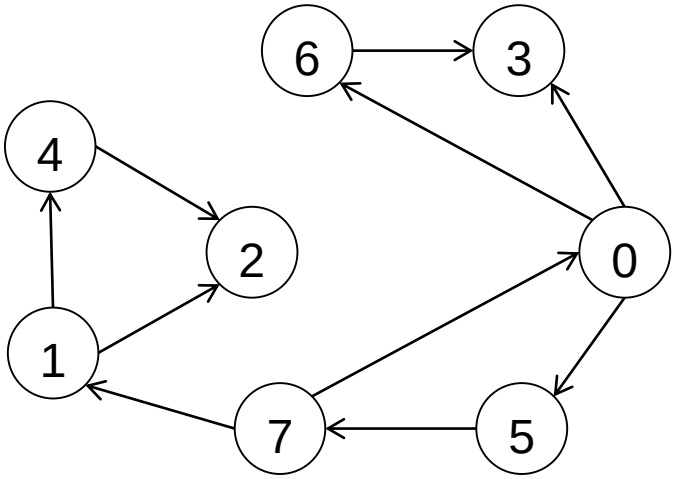
visitado[ ]	
0	~~0~~1
1	0
2	0
3	0
4	0
5	0
6	0
7	0

Saída: 0

F = {0}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	~~0~~1
6	~~0~~1
7	0

u = 0

Saída: 0 3 5 6

F = {~~0~~}

F = {}

F = {3}

F = {3, 5}

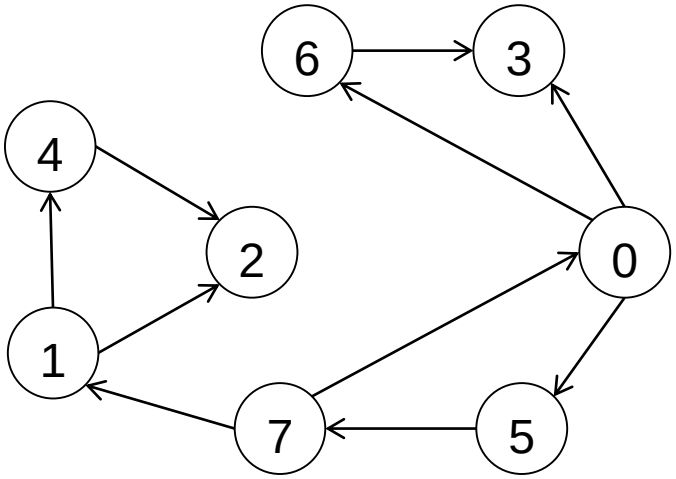
F = {3, 5, 6}

BUSCA-LARGURA(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. coloque s em uma fila F
8. **enquanto** fila F não vazia faça
9. u ← retirar o elemento da frente da fila F
10. **para** cada vértice v adjacente a u **faça**
11. **se** visitado[v] = 0 **então**
12. visitado[v] ← 1 //encontrou v
13. imprime v
14. colocar v no final da fila F
15. **fim-se**
16. **fim-para**
17. **fim-enquanto**

FIM

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	~~0~~1
6	~~0~~1
7	0

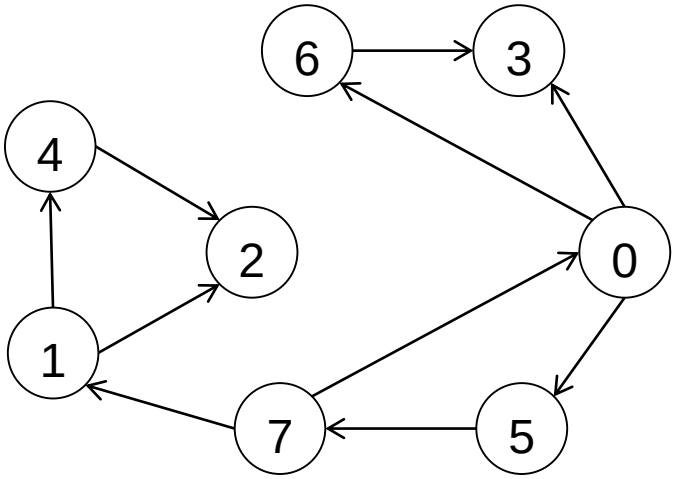
u = 3

Saída: 0 3 5 6

F = {~~3~~, 5, 6}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 5

Saída: 0 3 5 6 7

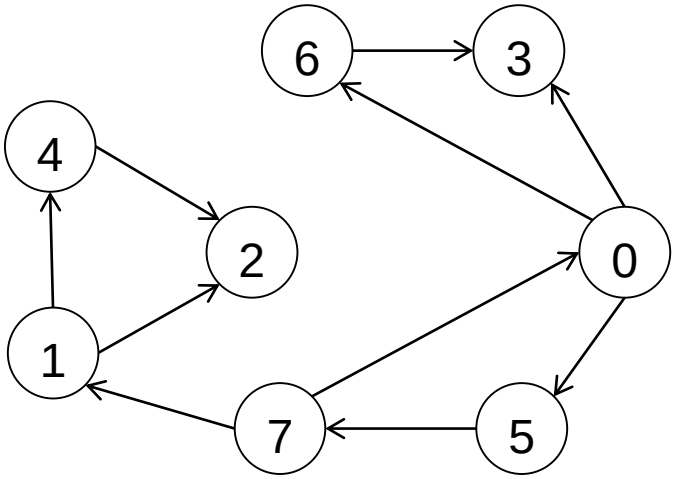
F = {~~5~~, 6}

F = {6}

F = {6, 7}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 6

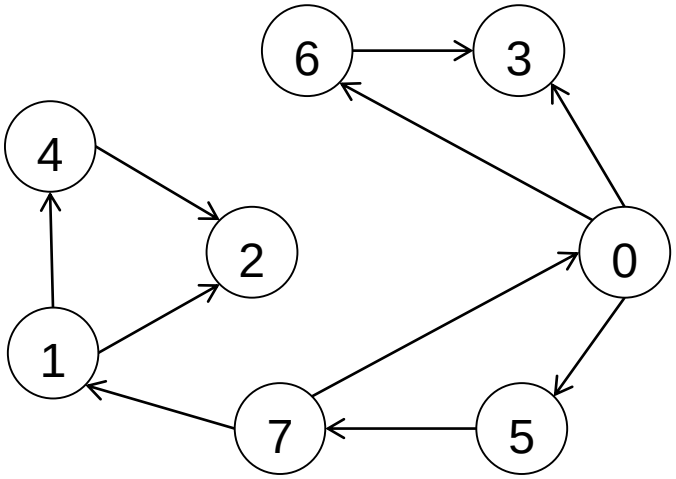
Saída: 0 3 5 6 7

F = {~~6~~, 7}

F = {7}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	0
3	~~0~~1
4	0
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 7

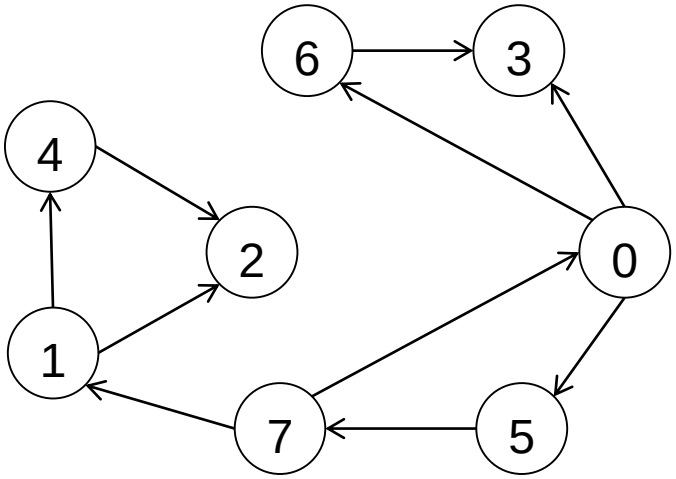
Saída: 0 3 5 6 7 1

F = {~~7~~, 1}

F = {1}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 1

Saída: 0 3 5 6 7 1 2 4

F = {~~1~~}

F = {}

F = {2}

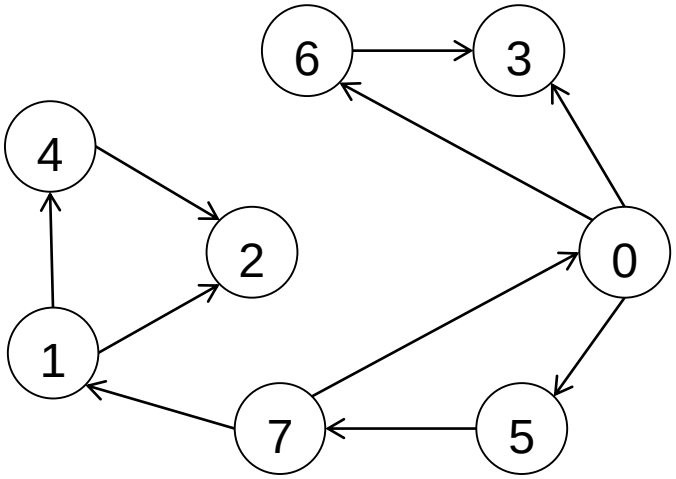
F = {2, 4}

BUSCA-LARGURA(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. coloque s em uma fila F
8. **enquanto** fila F não vazia faça
9. u ← retirar o elemento da frente da fila F
10. **para** cada vértice v adjacente a u **faça**
11. **se** visitado[v] = 0 **então**
12. visitado[v] ← 1 //encontrou v
13. imprime v
14. colocar v no final da fila F
15. **fim-se**
16. **fim-para**
17. **fim-enquanto**

FIM

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 2

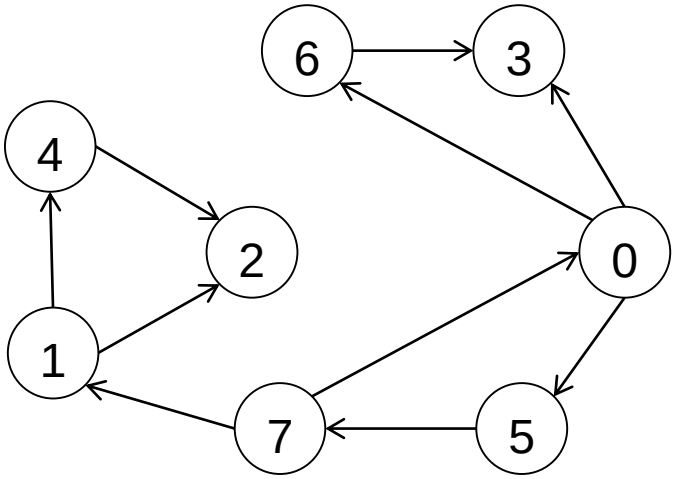
Saída: 0 3 5 6 7 1 2 4

F = {~~2~~, 4}

F = {4}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 4

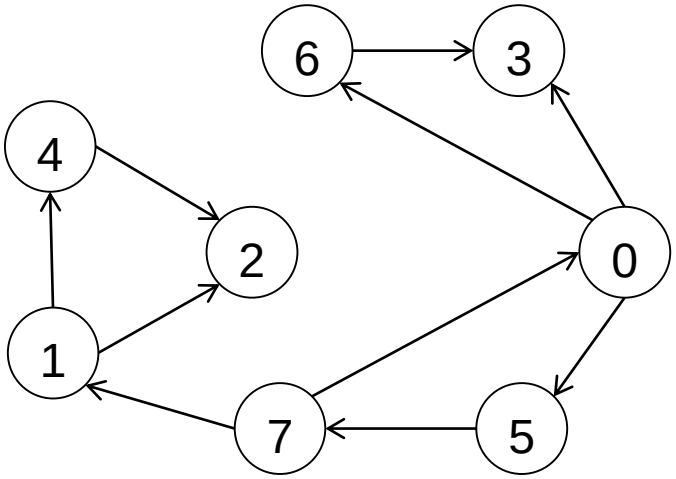
Saída: 0 3 5 6 7 1 2 4

F = {4}

F = {}

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 4

Saída: 0 3 5 6 7 1 2 4

F = {}

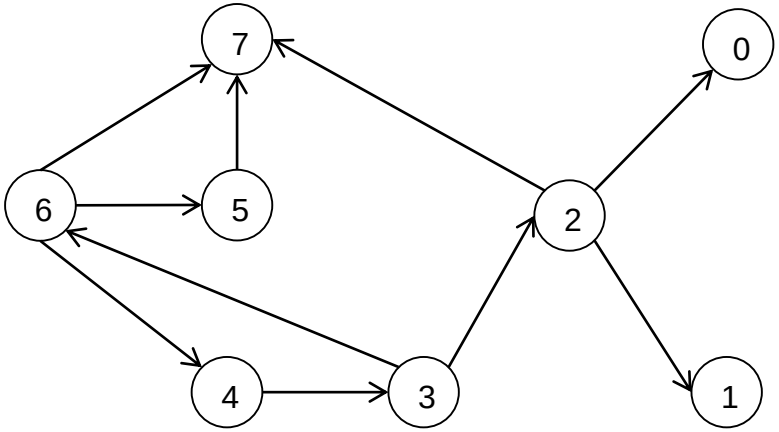
Encerra o algoritmo

```
BUSCA-LARGURA(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    coloque s em uma fila F
8.    enquanto fila F não vazia faça
9.        u ← retirar o elemento da frente da fila F
10.       para cada vértice v adjacente a u faça
11.           se visitado[v] = 0 então
12.               visitado[v] ← 1 //encontrou v
13.               imprime v
14.               colocar v no final da fila F
15.           fim-se
16.       fim-para
17.   fim-enquanto
FIM
```

Busca em Largura

Exercício

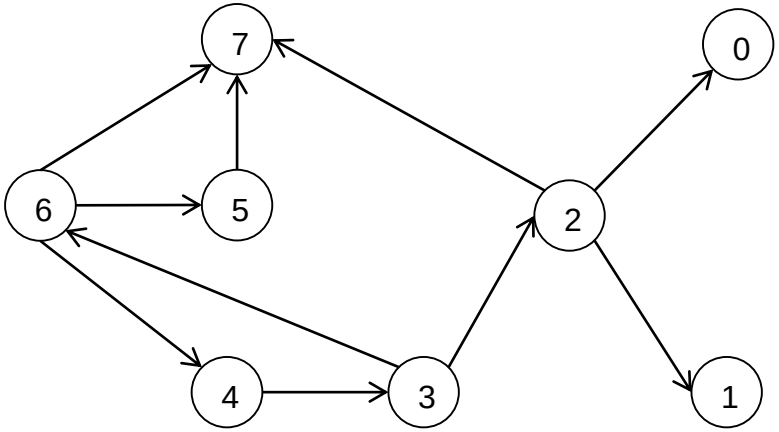
- Apresente a ordem de visitação dos vértices, partindo do vértice G, considerando a busca em largura para o grafo abaixo



Busca em Largura

Solução do Exercício

- Apresente a ordem de visitação dos vértices, partindo do vértice 6, considerando a busca em largura para o grafo abaixo



6 4 5 7 3 2 0 1

Busca em Profundidade

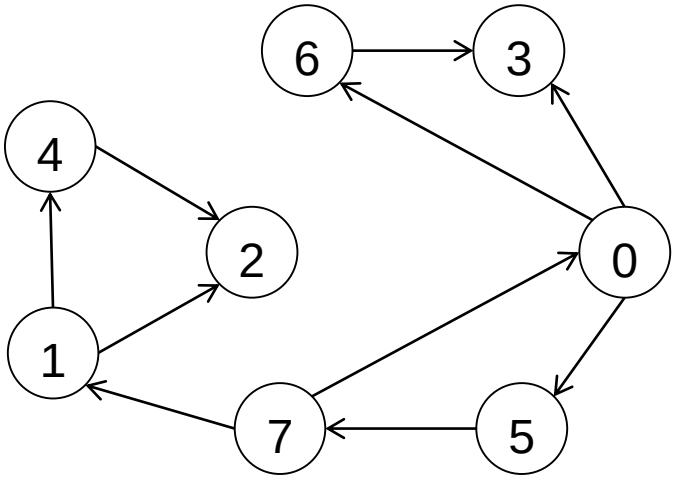
- Ideia: Pesquisar o grafo sempre o mais profundo possível
- Sempre que um novo vértice é descoberto, ele deve ser explorado completamente
- Quando v é totalmente explorado, retornar ao seu predecessor (*backtracking*)

Busca em Profundidade

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Complexidade de tempo O(|V|+|E|)

Busca em Profundidade



s = 0

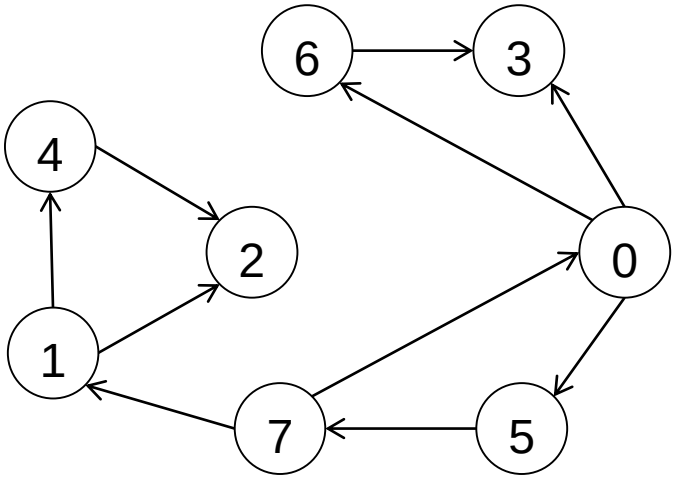
visitado[ ]	
0	~~0~~1
1	0
2	0
3	0
4	0
5	0
6	0
7	0

Saída: 0

P = {0}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	0
6	0
7	0

u = 0

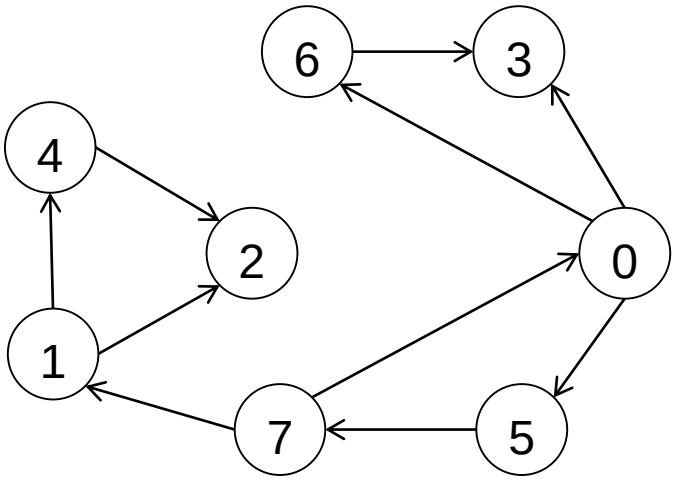
Saída: 0 3

P = {0}

P = {0, 3}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	0
6	0
7	0

u = 3

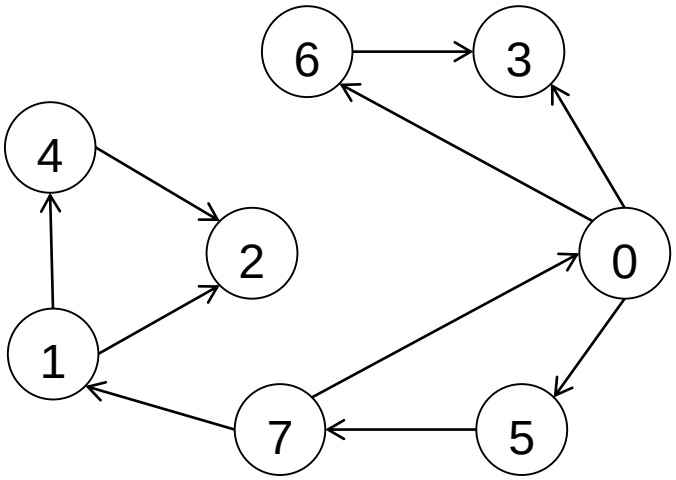
Saída: 0 3

P = {0, ~~3~~}

P = {0}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	~~0~~1
6	0
7	0

u = 0

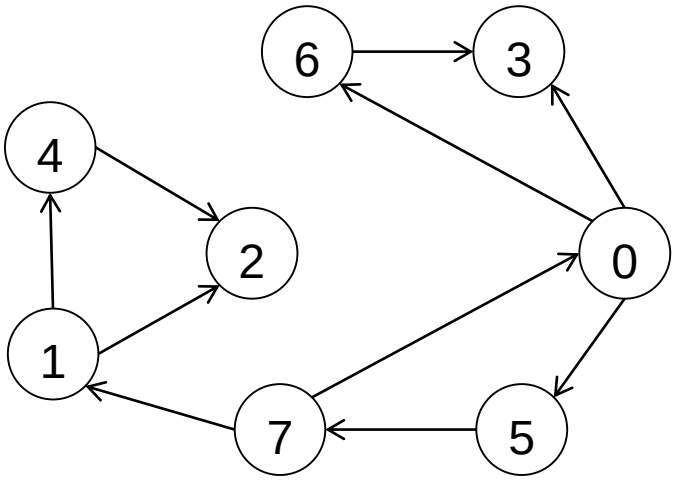
Saída: 0 3 5

P = {0}

P = {0, 5}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	0
2	0
3	~~0~~1
4	0
5	~~0~~1
6	0
7	~~0~~1

u = 5

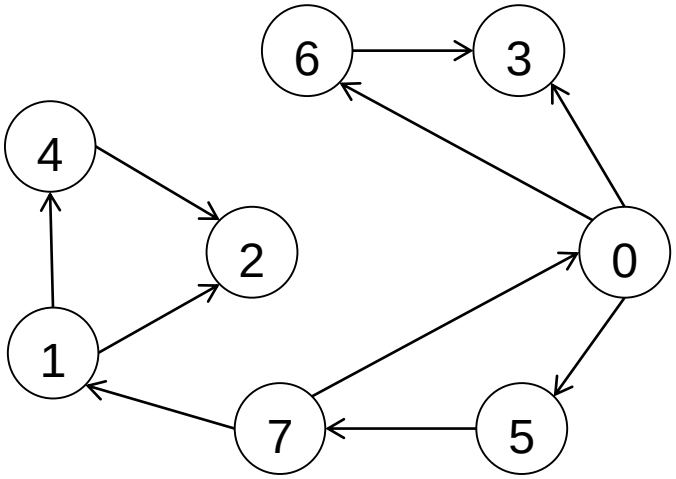
Saída: 0 3 5 7

P = {0, 5}

P = {0, 5, 7}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	0
3	~~0~~1
4	0
5	~~0~~1
6	0
7	~~0~~1

u = 7

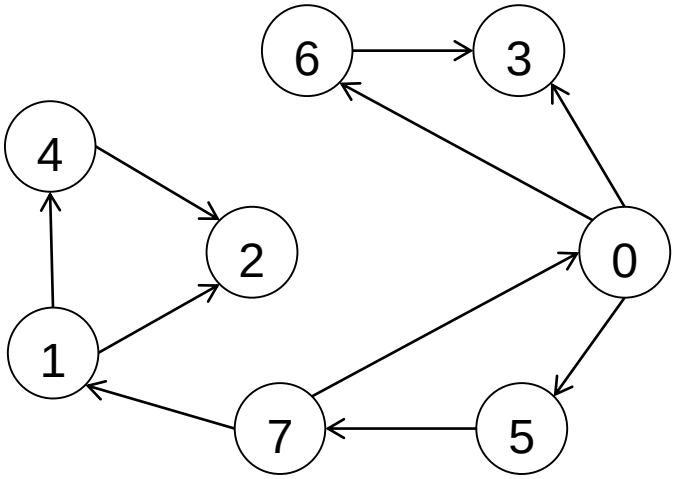
Saída: 0 3 5 7 1

P = {0, 5, 7}

P = {0, 5, 7, 1}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	0
5	~~0~~1
6	0
7	~~0~~1

u = 1

Saída: 0 3 5 7 1 2

P = {0, 5, 7, 1}

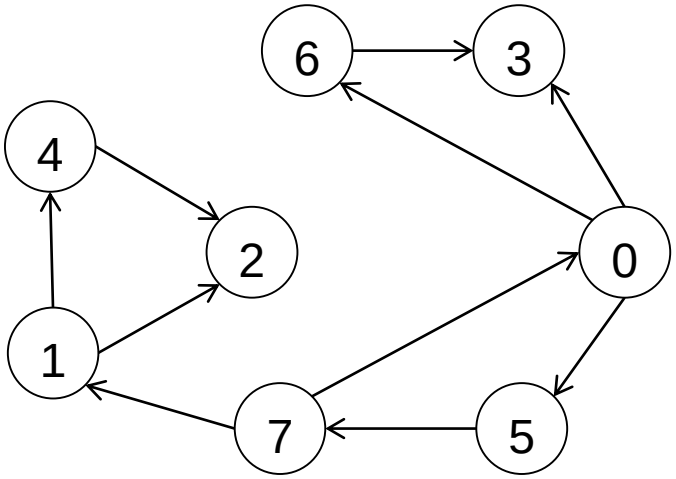
P = {0, 5, 7, 1, 2}

BUSCA-PROF(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. empilhar s na pilha P
8. **enquanto** pilha P não vazia **faça**
9. u ← elemento topo da pilha P
10. **se** existe vértice v adjacente a u e visitado[v] = 0 **faça**
11. visitado[v] = 1
12. imprime v
13. empilhar v em P
14. **senão**
15. desempilhar u de P
16. **fim-se**
17. **fim-enquanto**

FIM

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	0
5	~~0~~1
6	0
7	~~0~~1

u = 2

Saída: 0 3 5 7 1 2

P = {0, 5, 7, 1, ~~2~~}

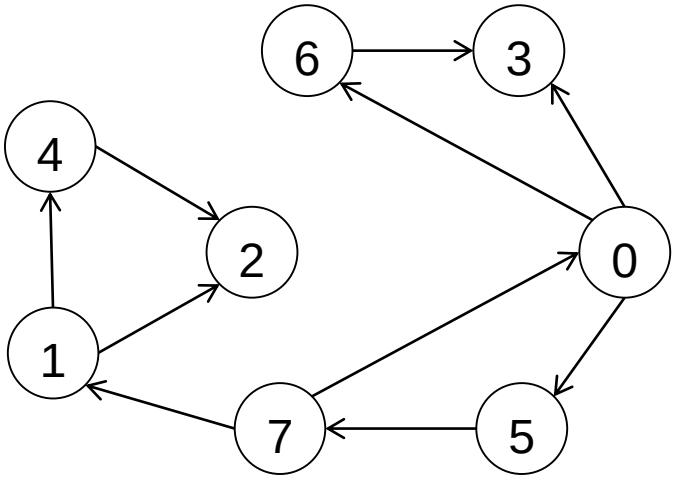
P = {0, 5, 7, 1}

BUSCA-PROF(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. empilhar s na pilha P
8. **enquanto** pilha P não vazia **faça**
9. u ← elemento topo da pilha P
10. **se** existe vértice v adjacente a u e visitado[v] = 0 **faça**
11. visitado[v] = 1
12. imprime v
13. empilhar v em P
14. **senão**
15. desempilhar u de P
16. **fim-se**
17. **fim-enquanto**

FIM

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	0
7	~~0~~1

u = 1

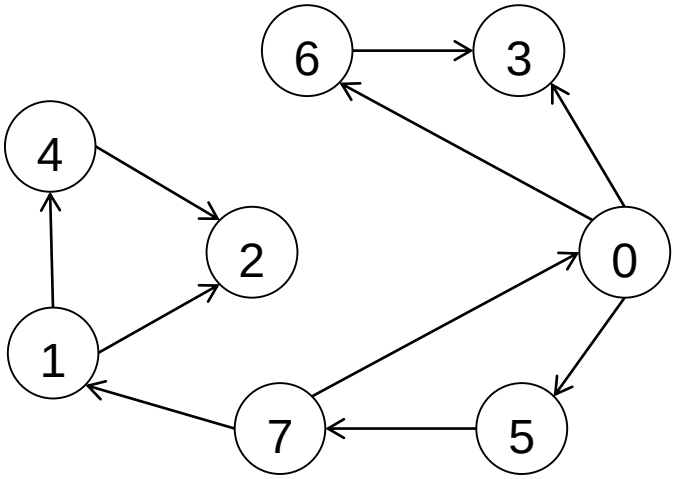
Saída: 0 3 5 7 1 2 4

P = {0, 5, 7, 1}

P = {0, 5, 7, 1, 4}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	0
7	~~0~~1

u = 4

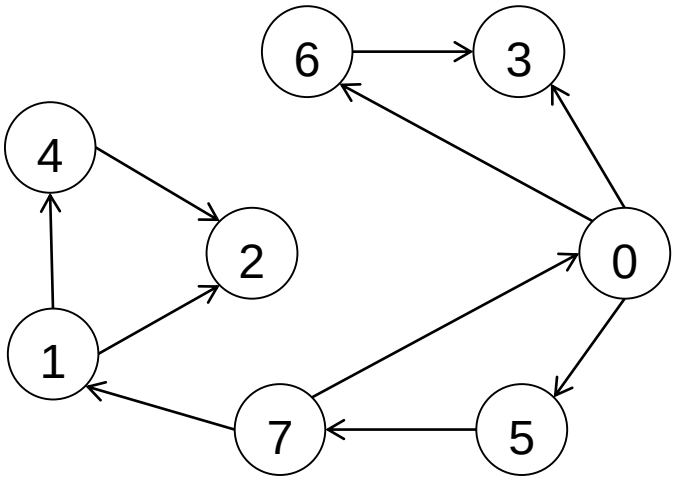
Saída: 0 3 5 7 1 2 4

P = {0, 5, 7, 1, 4}

P = {0, 5, 7, 1}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	0
7	~~0~~1

u = 1

Saída: 0 3 5 7 1 2 4

P = {0, 5, 7, ~~1~~}

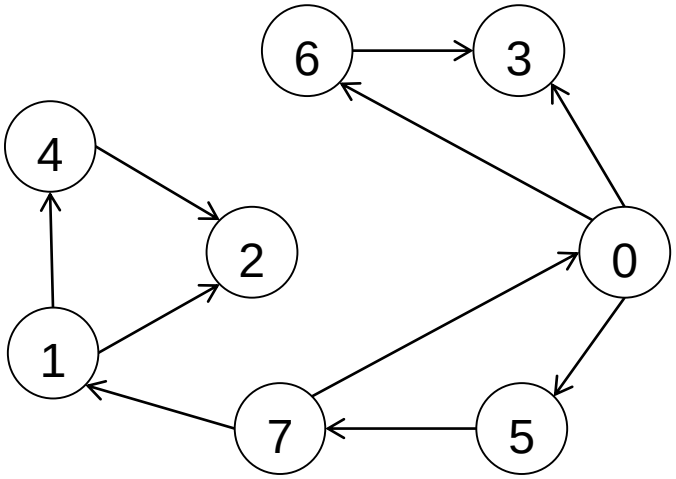
P = {0, 5, 7}

BUSCA-PROF(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. empilhar s na pilha P
8. **enquanto** pilha P não vazia **faça**
9. u ← elemento topo da pilha P
10. **se** existe vértice v adjacente a u e visitado[v] = 0 **faça**
11. visitado[v] = 1
12. imprime v
13. empilhar v em P
14. **senão**
15. desempilhar u de P
16. **fim-se**
17. **fim-enquanto**

FIM

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	0
7	~~0~~1

u = 7

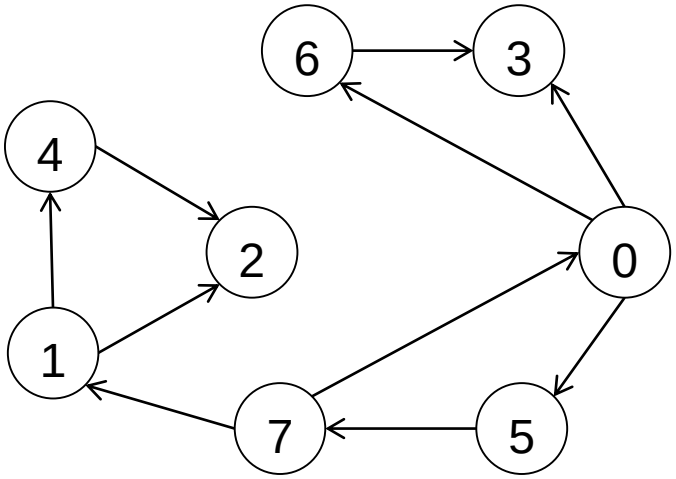
Saída: 0 3 5 7 1 2 4

P = {0, 5, ~~7~~}

P = {0, 5}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	0
7	~~0~~1

u = 5

Saída: 0 3 5 7 1 2 4

P = {0, ~~5~~}

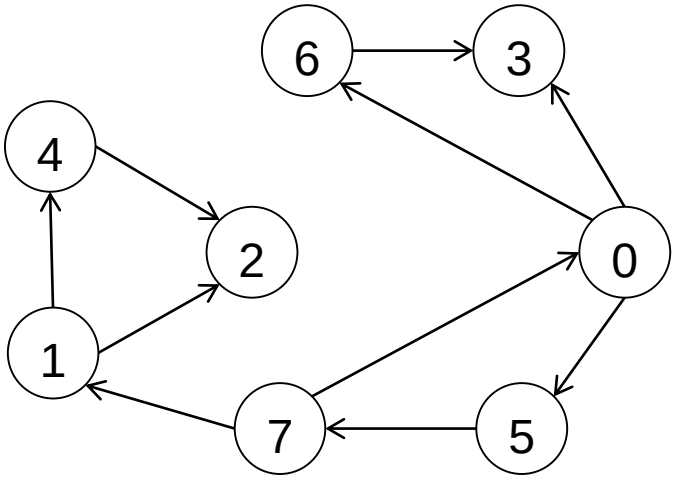
P = {0}

BUSCA-PROF(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. empilhar s na pilha P
8. **enquanto** pilha P não vazia **faça**
9. u ← elemento topo da pilha P
10. **se** existe vértice v adjacente a u e visitado[v] = 0 **faça**
11. visitado[v] = 1
12. imprime v
13. empilhar v em P
14. **senão**
15. desempilhar u de P
16. **fim-se**
17. **fim-enquanto**

FIM

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 0

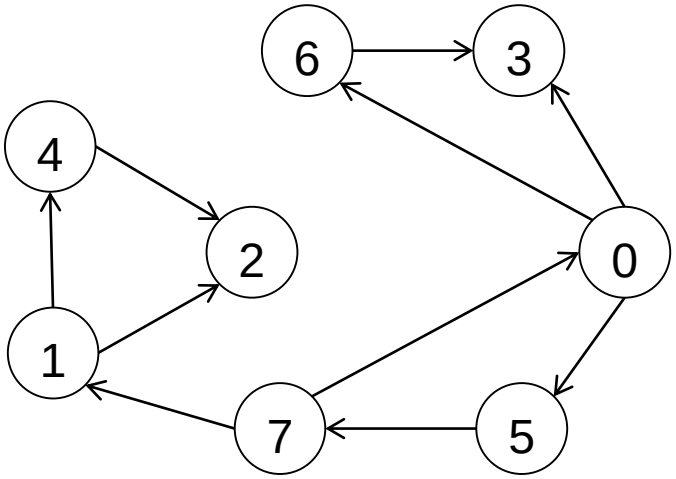
Saída: 0 3 5 7 1 2 4 6

P = {0}

P = {0, 6}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 6

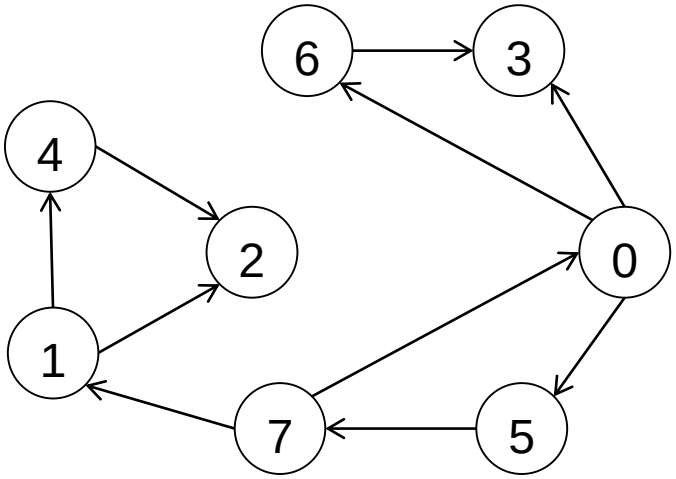
Saída: 0 3 5 7 1 2 4 6

P = {0, ~~6~~}

P = {0}

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ← 1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 0

Saída: 0 3 5 7 1 2 4 6

P = {~~0~~}

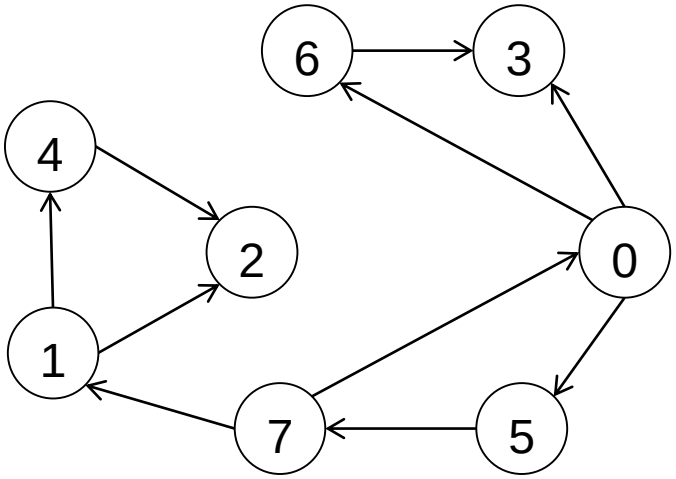
P = {}

BUSCA-PROF(G(V, E), s)

1. visitado[ ]
2. **para cada** vértice v em V **faça**
3. visitado[v] ← 0
4. **fim-para**
5. visitado[s] ← 1
6. imprime s
7. empilhar s na pilha P
8. **enquanto** pilha P não vazia **faça**
9. u ← elemento topo da pilha P
10. **se** existe vértice v adjacente a u e visitado[v] = 0 **faça**
11. visitado[v] = 1
12. imprime v
13. empilhar v em P
14. **senão**
15. desempilhar u de P
16. **fim-se**
17. **fim-enquanto**

FIM

Busca em Profundidade



s = 0

visitado[ ]	
0	~~0~~1
1	~~0~~1
2	~~0~~1
3	~~0~~1
4	~~0~~1
5	~~0~~1
6	~~0~~1
7	~~0~~1

u = 0

Saída: 0 3 5 7 1 2 4 6

P = {}

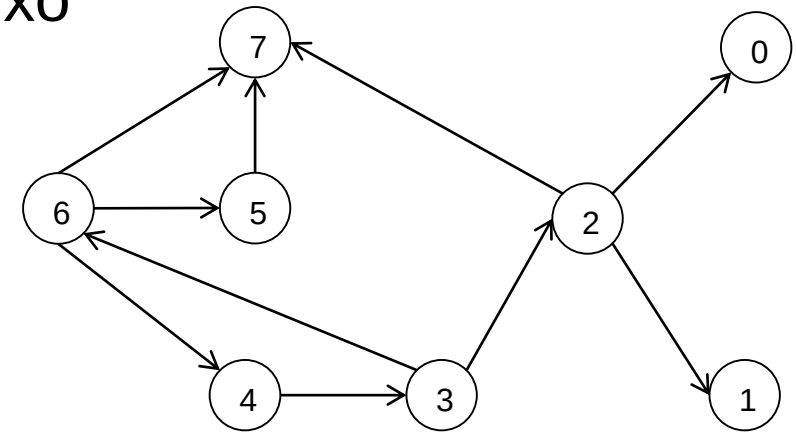
Encerra o algoritmo

```
BUSCA-PROF(G(V, E), s)
1.    visitado[ ]
2.    para cada vértice v em V faça
3.        visitado[v] ← 0
4.    fim-para
5.    visitado[s] ←  1
6.    imprime s
7.    empilhar s na pilha P
8.    enquanto pilha P não vazia faça
9.        u ← elemento topo da pilha P
10.       se existe vértice v adjacente a u e visitado[v] = 0 faça
11.           visitado[v] = 1
12.           imprime v
13.           empilhar v em P
14.       senão
15.           desempilhar u de P
16.       fim-se
17.   fim-enquanto
FIM
```

Busca em Profundidade

Exercício

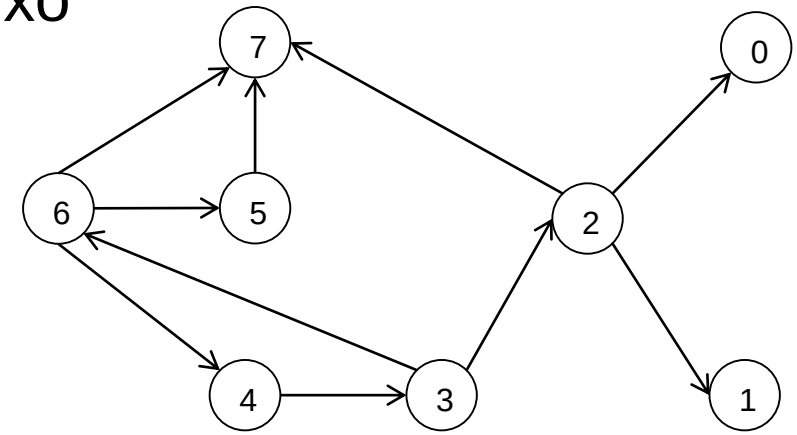
- Apresente a ordem de visitação dos vértices, partindo do vértice 6, considerando a busca em profundidade para o grafo abaixo



Busca em Profundidade

Solução do Exercício

- Apresente a ordem de visitação dos vértices, partindo do vértice 6, considerando a busca em profundidade para o grafo abaixo



6 4 3 2 0 1 7 5

Busca em Profundidade
Versão Recursiva

Variável global
visitado[ ]

PROF(G(V, E), u)

1. visitado[u] ← 1
2. imprime u
3. **para cada** nó v adjacente a u **faça**
4. **se** visitado[v] = 0 **então**
5. **PROF**(G(V, E), v)
6. **fim-se**
7. **fim-para**

FIM

Complexidade de tempo |V| chamadas a PROF( )

Busca em Profundidade
Aplicações

- Aplicações da Busca em Profundidade
  - Determinar se um grafo é acíclico
  - Determinar se um grafo é conexo
  - Determinar se um grafo é bipartido
  - Ordenação topológica
  - Determinar componentes conexas de um grafo

Busca em Profundidade

Encontrar componentes conexas

Variável global
visitado[ ]

```
COMPONENTES-CONEXAS(G(V, E))
1.  para cada vértice v em V faça
2.      visitado[v] ← 0
3.  fim-para
4.  marca ← 0
5.  para cada vértice v em V faça
6.      se visitado[v] = 0 então
7.          marca ← marca + 1
8.          PROF-CON(G(V, E), v, marca)
9.      fim-se
10. fim-para
FIM
```

```
PROF-CON(G(V, E), u, marca)
1.  visitado[u] ← marca
2.  imprimir u e marca
3.  para cada nó v adjacente a u faça
4.      se visitado[v] = 0 então
5.          PROF-CON(G(V, E), v, marca)
6.      fim-se
7.  fim-para
FIM
```

Ao final, visitado[ ] será preenchido com uma marca para cada vértice. Essa marca indica a qual componente conexa o vértice pertence.

Revisão

- Busca em largura
  - Processar os vértices por níveis de distância
- Busca em profundidade
  - Pesquisar o grafo sempre o mais profundo possível

Bibliografia

- Goldbarg, M.; Goldbarg, E. **Grafos: Conceitos, algoritmos e aplicações**. 1ª edição. Elsevier, 2012.

- Boaventura Netto, P. O. **Grafos: Teoria, Modelos, Algoritmos**. 4a edição. Edgar Blucher, 2012.