
UNIPAC
Barbacena

# Centro Universitário Presidente Antônio Carlos

## Teoria de Grafos

# Busca em Grafos (Árvores)

Felipe Roncalli de Paula Carneiro

felipecarneiro@unipac.br

# O que vamos aprender nessa aula

- Recursividade;
- **Busca em Profundidade;**
- **Busca em Largura;**

# Recursividade

É o mecanismo de programação no qual uma definição de função ou de outro objeto refere-se ao próprio objeto sendo definido. Assim função recursiva é uma função que é definida em termos de si mesma.

# Busca em Profundidade

**A busca em profundidade, (em inglês, *depth-first search*)**

é um algoritmo para caminhar no grafo

**Estratégia**

- sempre buscar primeiro o vértice mais profundo no grafo
- após todas as arestas adjacentes forem exploradas, a busca retrocede no grafo para explorar vértices anteriores

# Busca em Profundidade

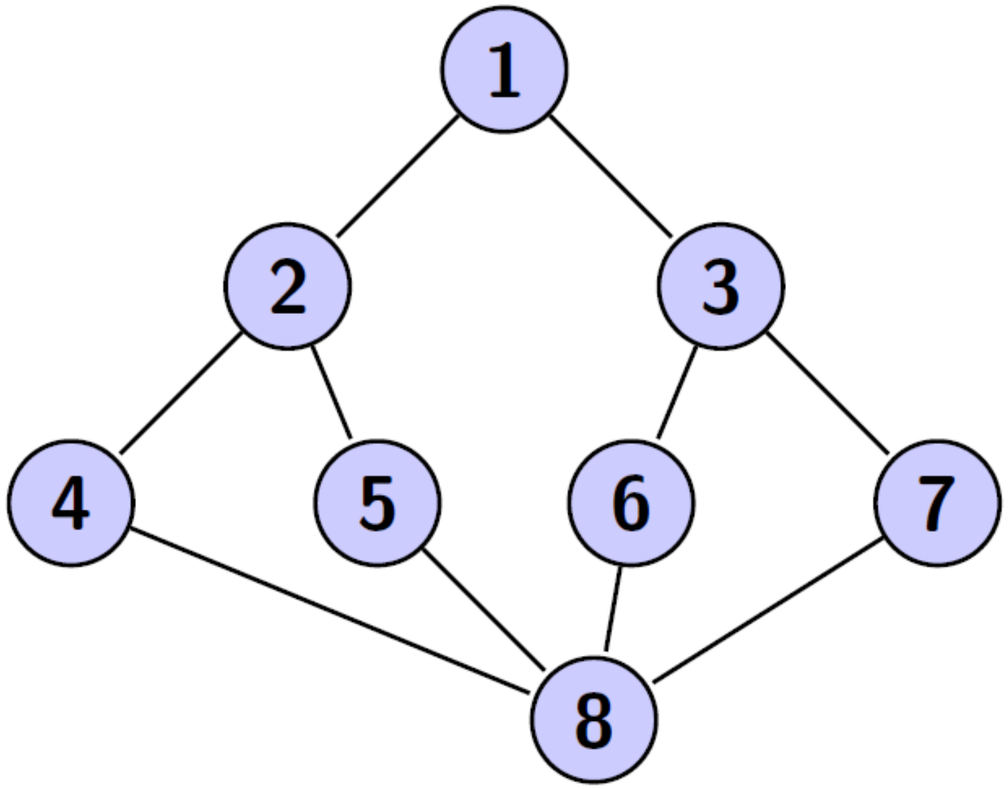
Algoritmo: iniciar em um vértice arbitrário

1. Visita vértice
2. Se vértice for o buscado, retorna que encontrou
3. Senão, faz busca em profundidade para cada vértice adjacente ainda não visitado

# Busca em Profundidade

```
1  boolean buscaProfundidade(Vertice procurado, Vertice
       proximo) {
2
3   // visita o vertice proximo
4   proximo.visitado = true;
5   if (proximo.igualA(procurado)) {
6     return(true);
7   } else {
8     for (Vertice adjacente: adjacentes)
9       if (adjacente.visitado == false) {
10        boolean encontrou = buscaProfundidade(procurado,
              adjacente);
11        if (encontrou)
12          return(true);
13      }
14  }
15  return(false);
16 }
```

# Exemplo - Busca em Profundidade

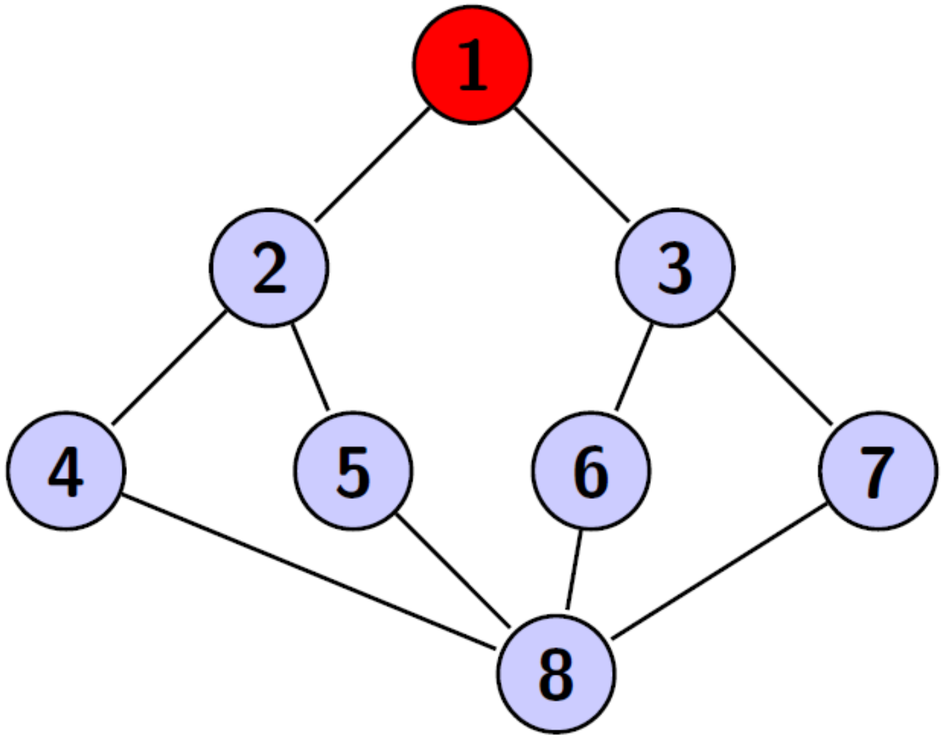



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Início com 1.

A cor verde significa que vértice foi visitado.

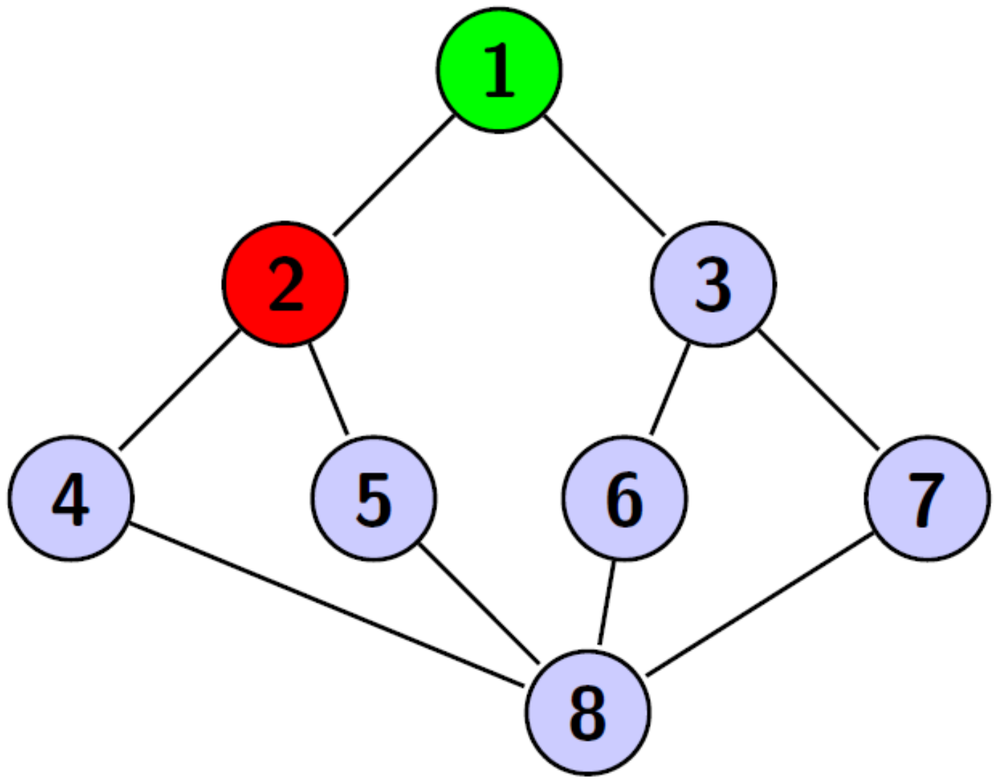



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 2.

A cor verde significa que vértice foi visitado.

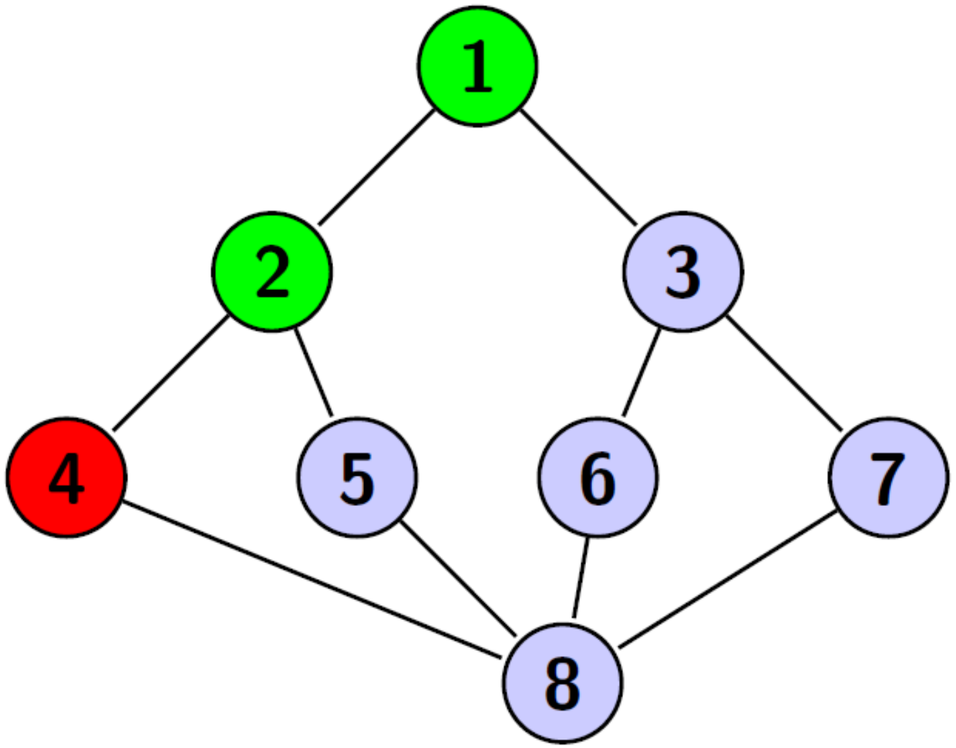



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 4.

A cor verde significa que vértice foi visitado.

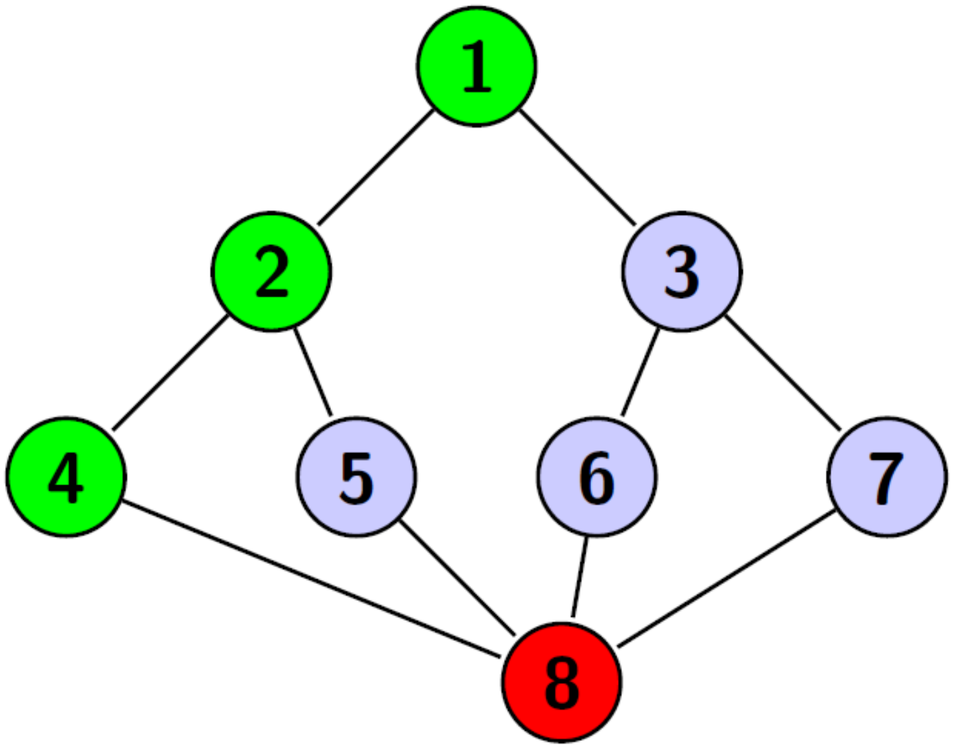



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 8.

A cor verde significa que vértice foi visitado.

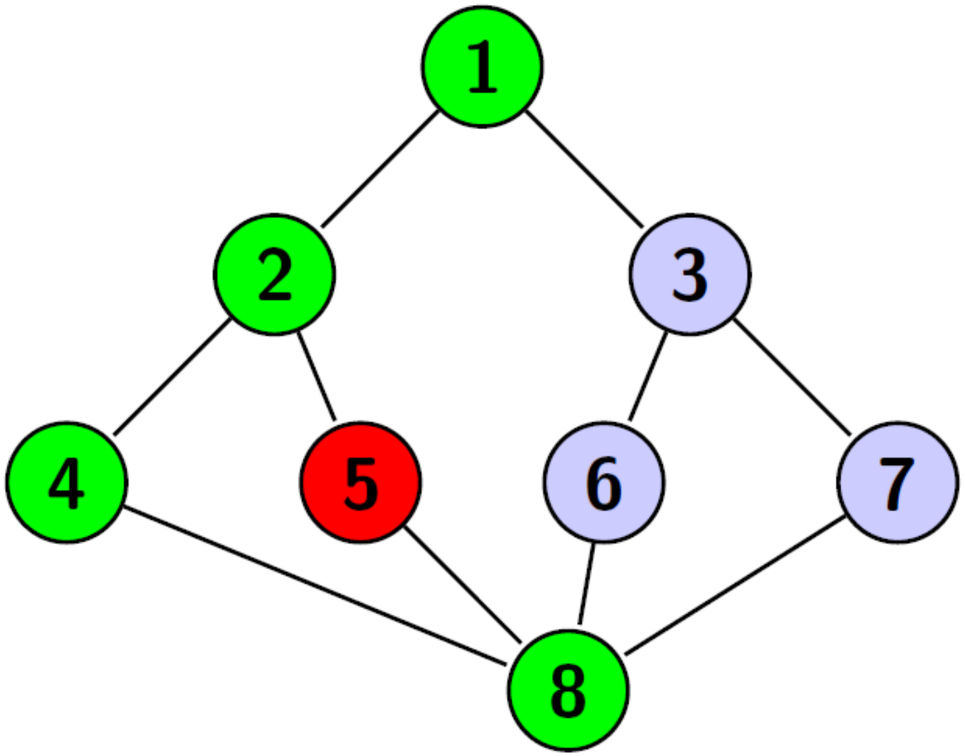



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 5.

A cor verde significa que vértice foi visitado.

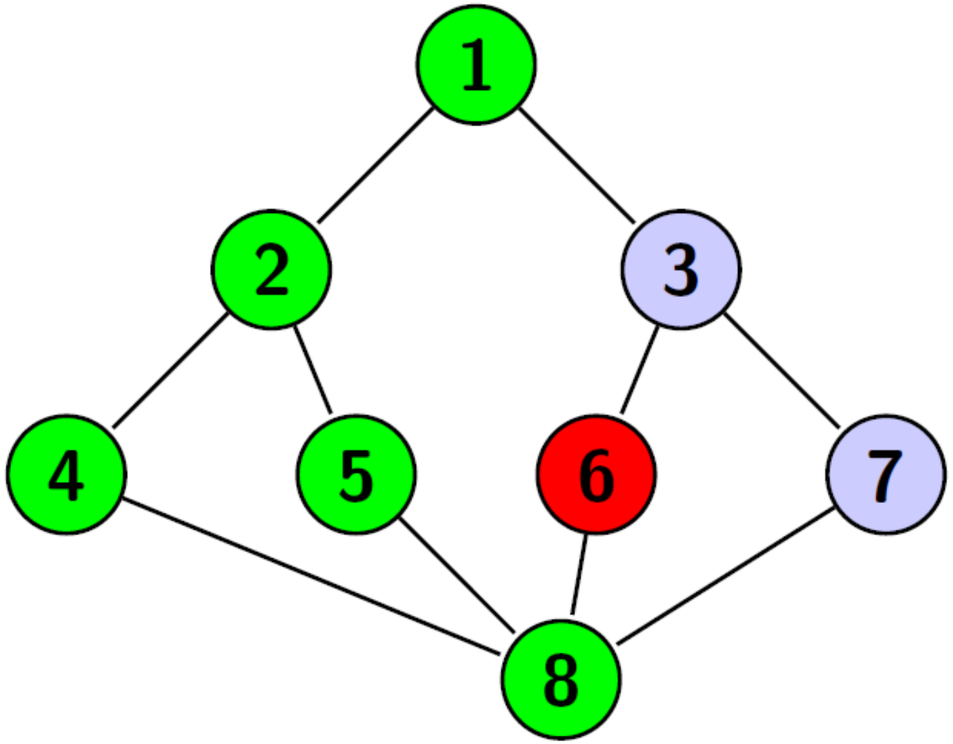



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 6.

A cor verde significa que vértice foi visitado.

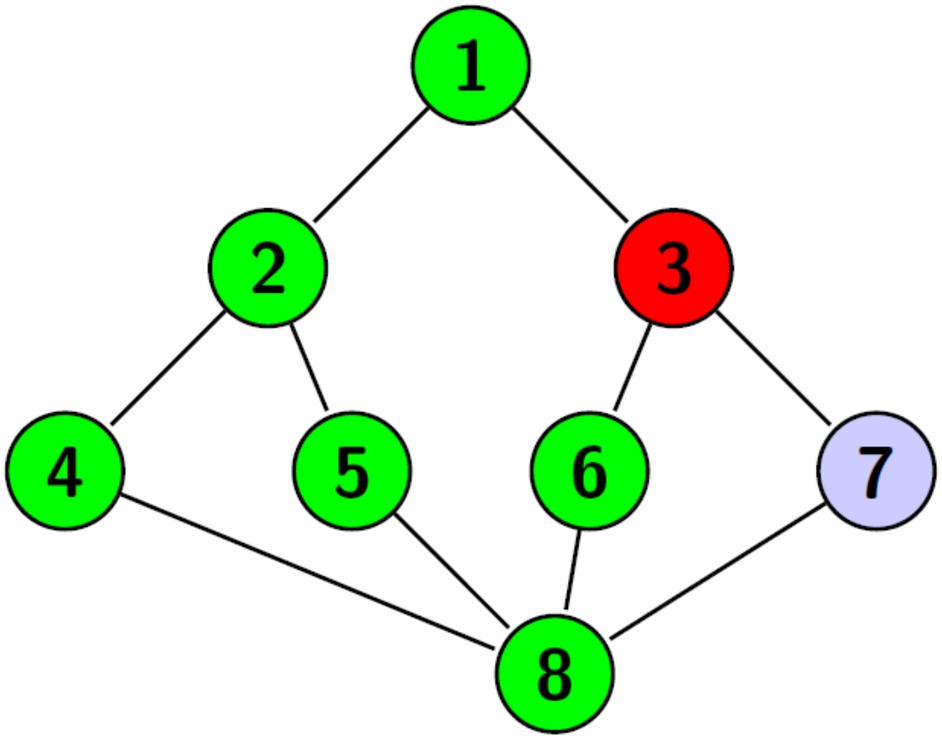



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 3.

A cor verde significa que vértice foi visitado.

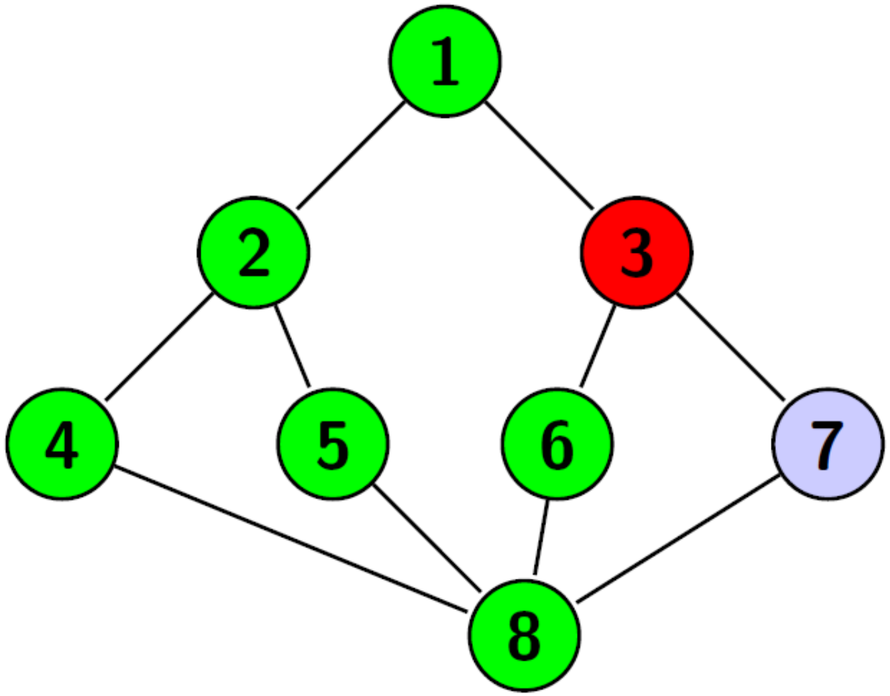



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 3.

A cor verde significa que vértice foi visitado.

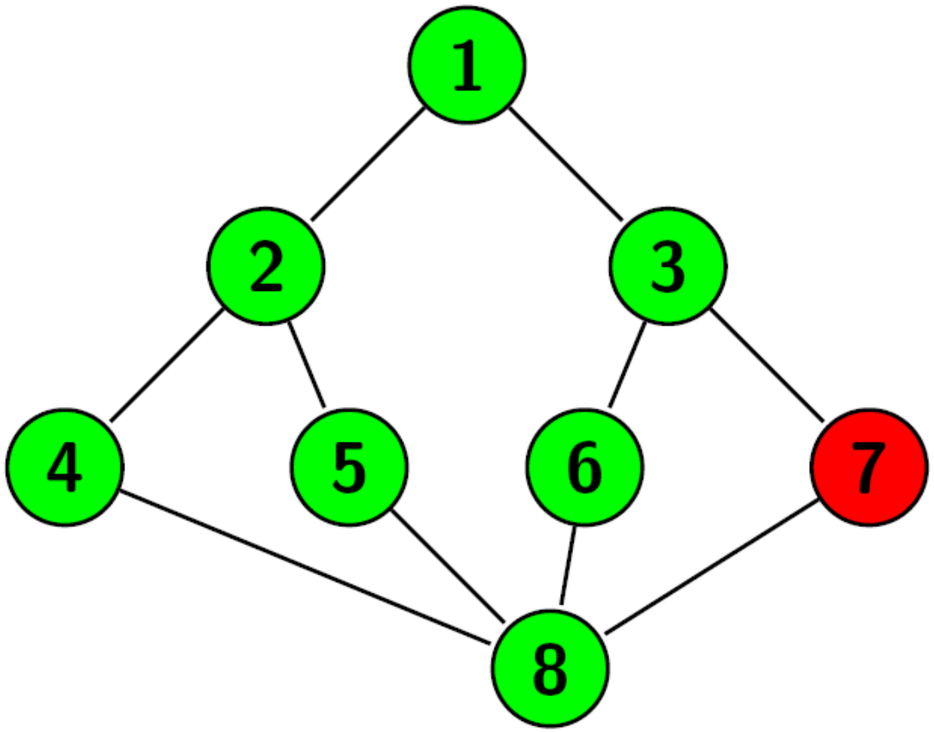



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Profundidade

Visitanto o 7.

A cor verde significa que vértice foi visitado.



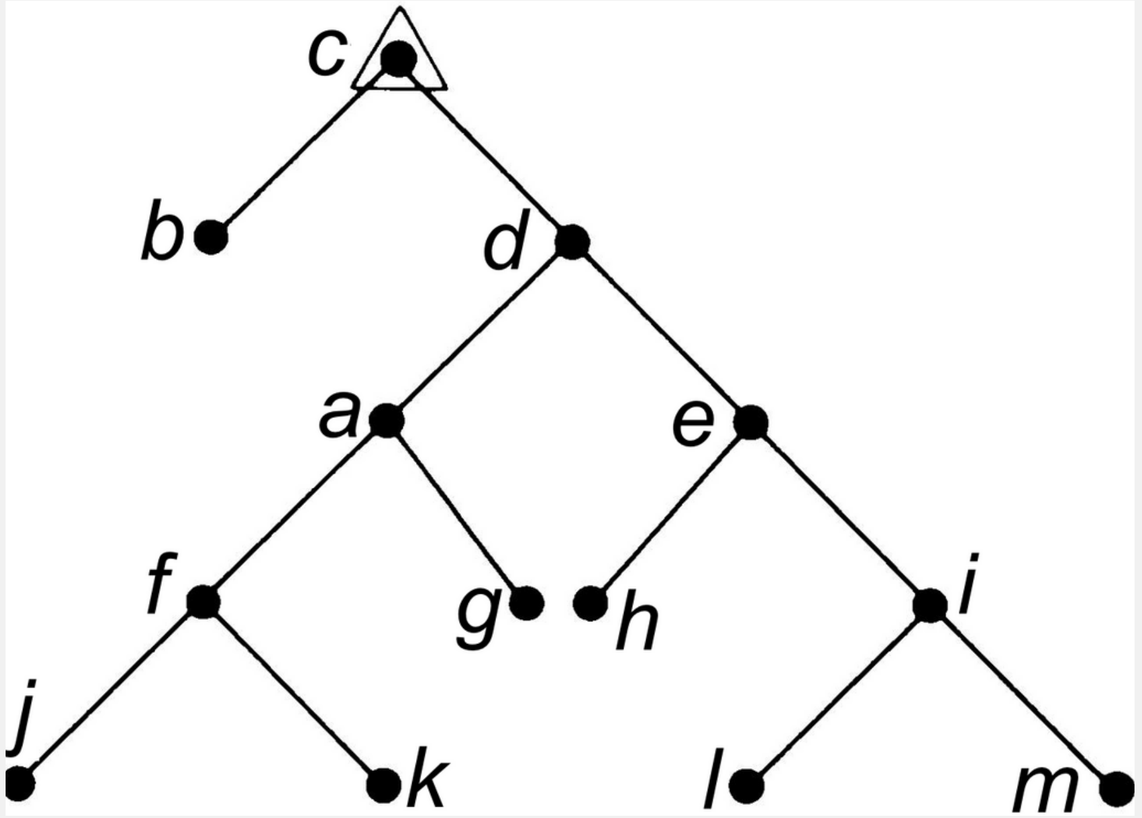

- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Qual o pior caso?

- Qual é o pior caso?
  - Custo de ir para cada vértice é proporcional a $|V|$
  - Custo de transitar em cada aresta é proporcional $|A|$
  - Complexidade de pior caso $O(|V| + |A|)$

# Exercício

Execute a busca em Profundidade visando encontrar o Vértice l;



# Busca em Largura

## Estratégia

- avança por fronteiras
- Visita todos os adjacentes ainda não visitados primeiro

## Algoritmo

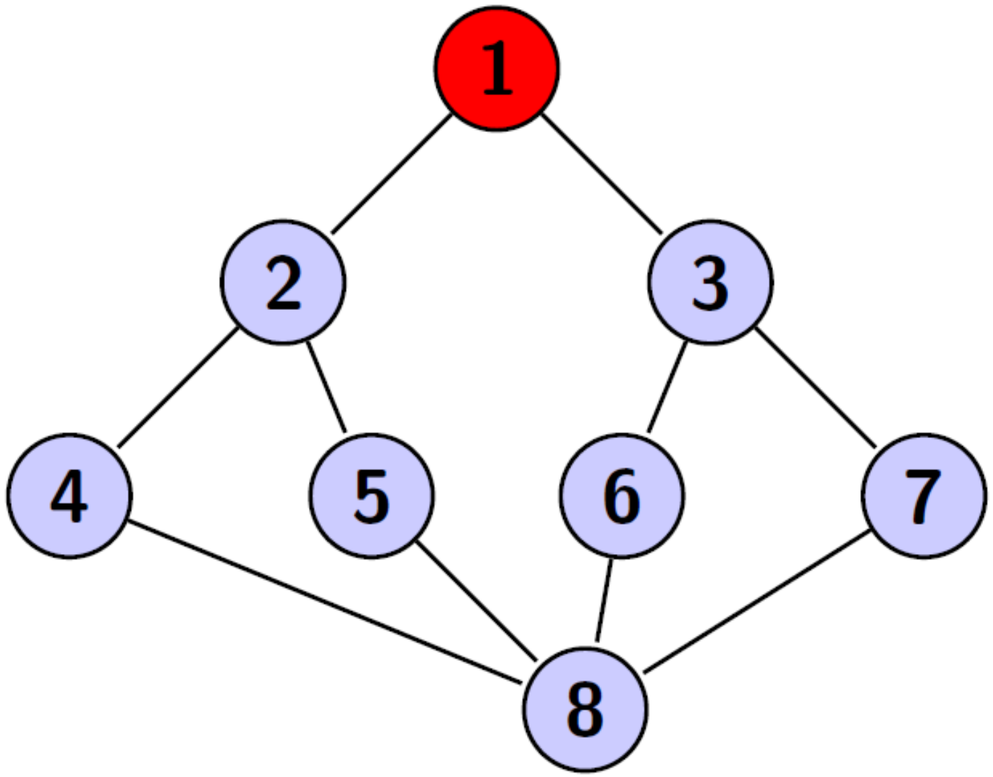
Iniciar em um vértice arbitrário $u$

1. colocar $u$ em uma fila $F$
2. enquanto a fila não for vazia fazer:
3. Pegar e remover o primeiro elemento $u$ da fila $F$
4. Para cada vértice adjacente que ainda não visitado $v$ ao $u$
    1. visitar vértice adjacente $v$
    2. colocar vértice $v$ na fila $F$

# Exemplo - Busca em Largura

Início com 1.

Fila [1]

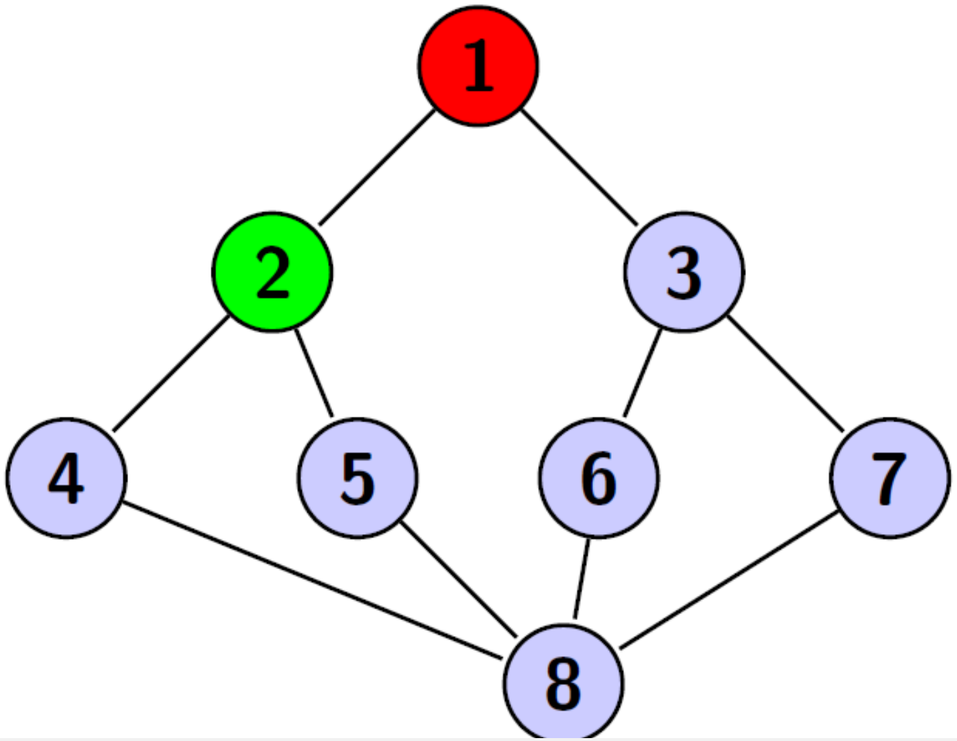



- [1] — [2] — [3]
- [2] — [1] — [4] — [5]
- [3] — [1] — [6] — [7]
- [4] — [2] — [8]
- [5] — [2] — [8]
- [6] — [3] — [8]
- [7] — [3] — [8]
- [8] — [4] — [5] — [6] — [7]

# Exemplo - Busca em Largura

Primeiro da fila é 1.

Fila [2]

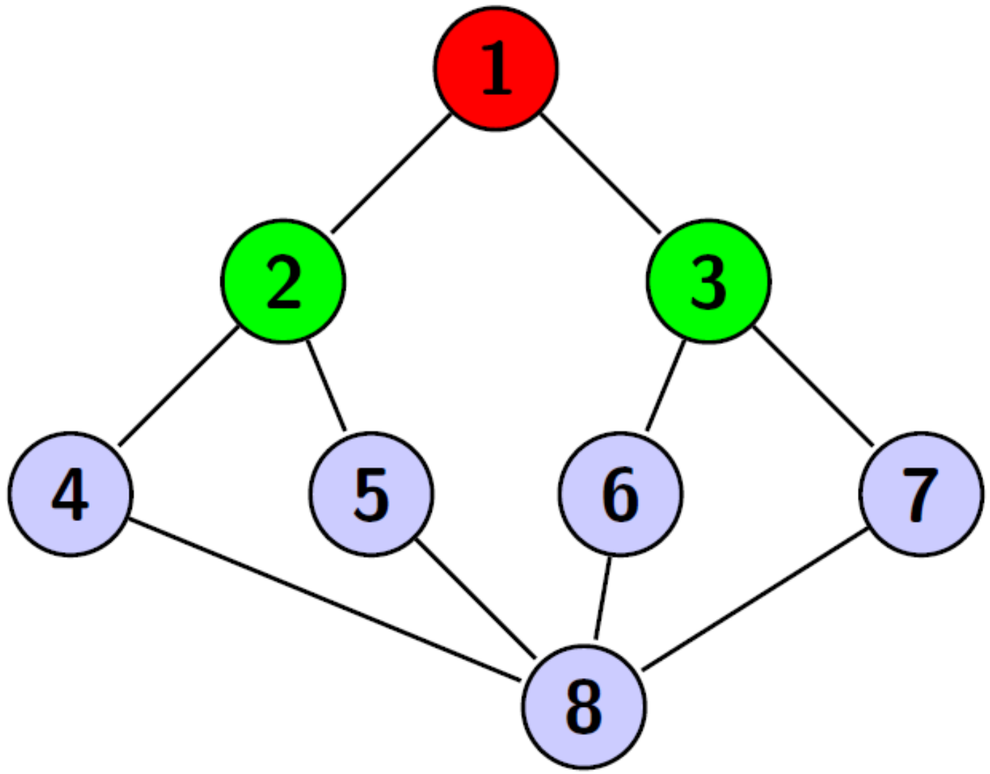



- [1] — [2] — [3]
- [2] — [1] — [4] — [5]
- [3] — [1] — [6] — [7]
- [4] — [2] — [8]
- [5] — [2] — [8]
- [6] — [3] — [8]
- [7] — [3] — [8]
- [8] — [4] — [5] — [6] — [7]

# Exemplo - Busca em Largura

Primeiro da fila é 1.

Fila | 2 | 3 |.

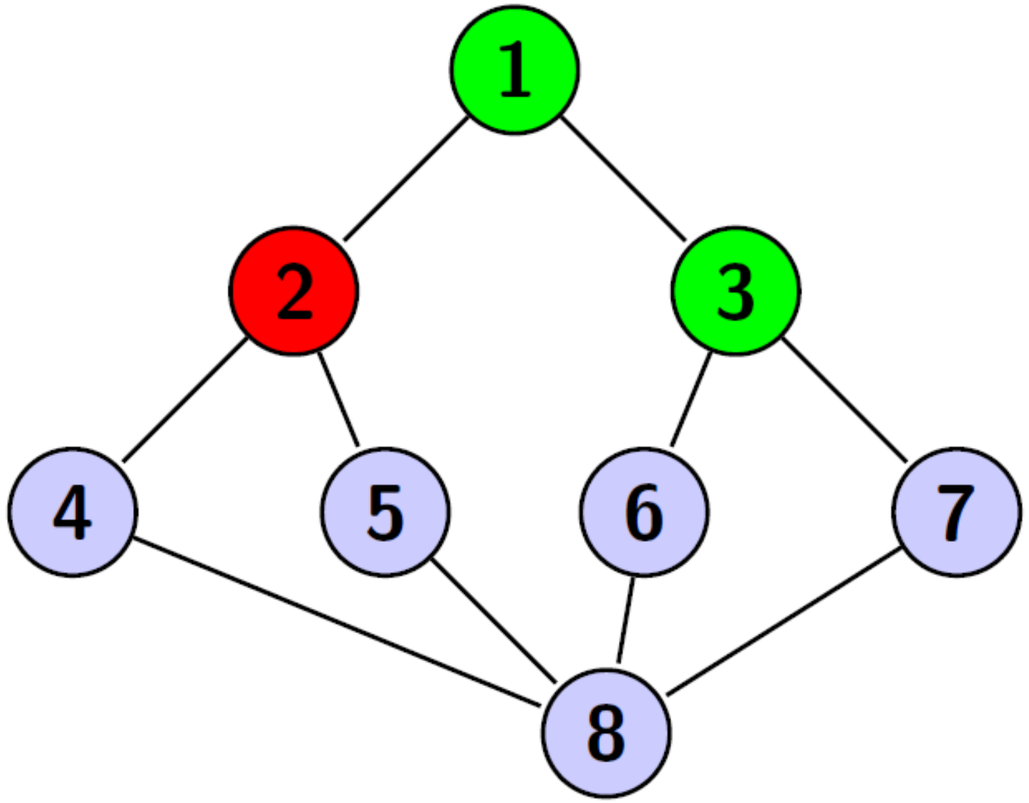



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Largura

Primeiro da fila é 2.

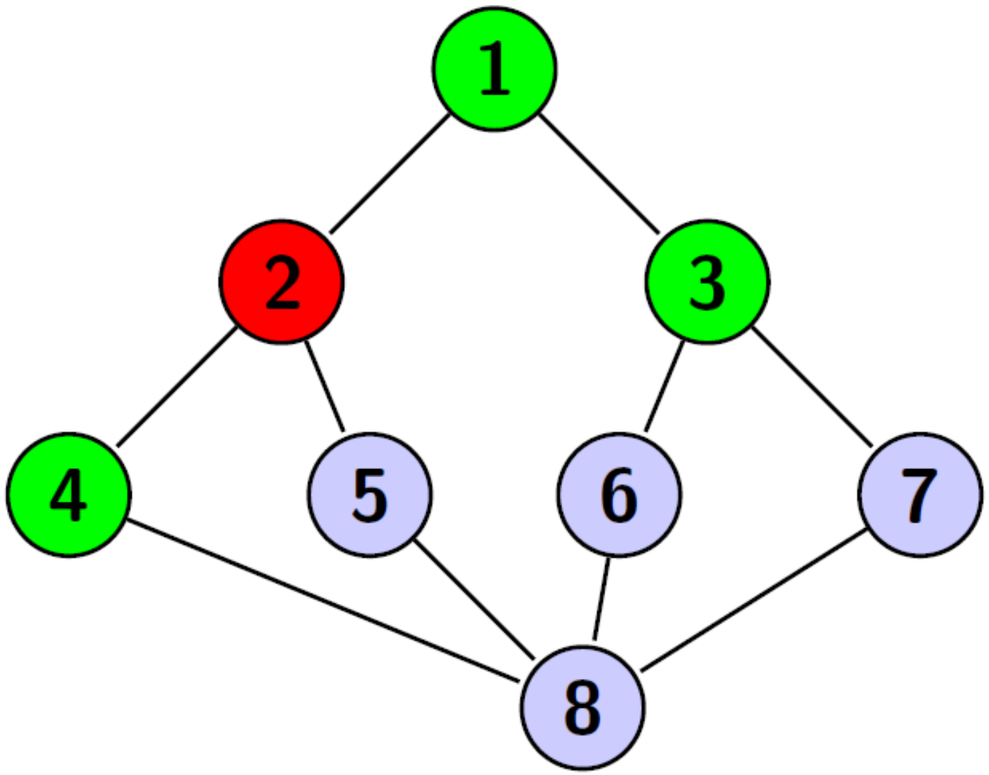
Fila [3].




- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Largura

Primeiro da fila é 2.
Fila de [2] é [4].

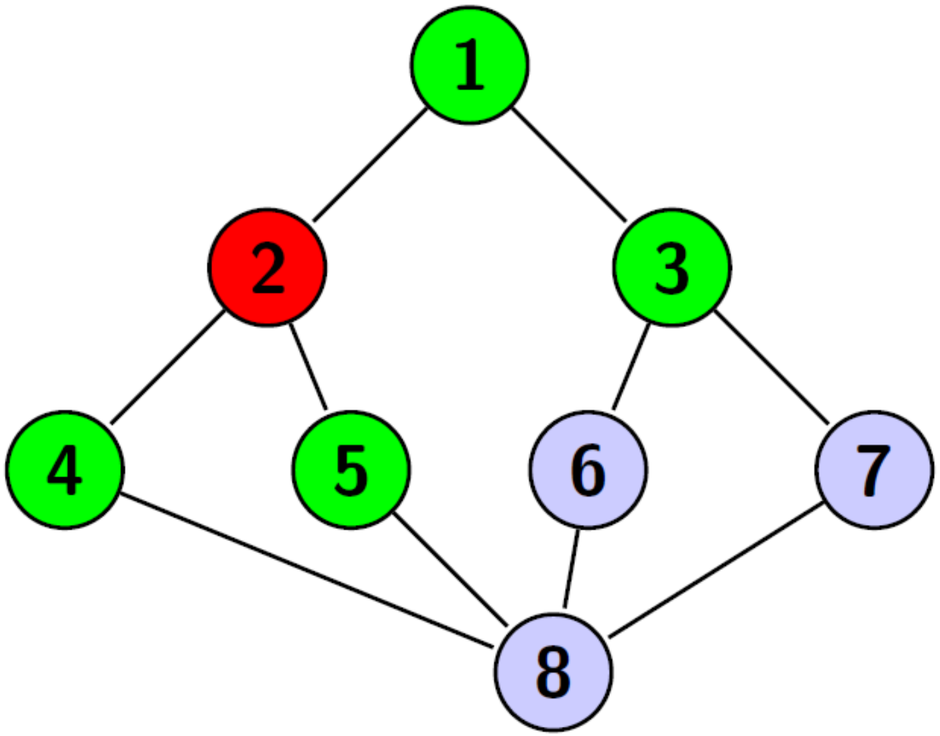



- [1] — [2] — [3]
- [2] — [1] — [4] — [5]
- [3] — [1] — [6] — [7]
- [4] — [2] — [8]
- [5] — [2] — [8]
- [6] — [3] — [8]
- [7] — [3] — [8]
- [8] — [4] — [5] — [6] — [7]

# Exemplo - Busca em Largura

Primeiro da fila é 2.

Fila | 3 | 4 | 5 |.

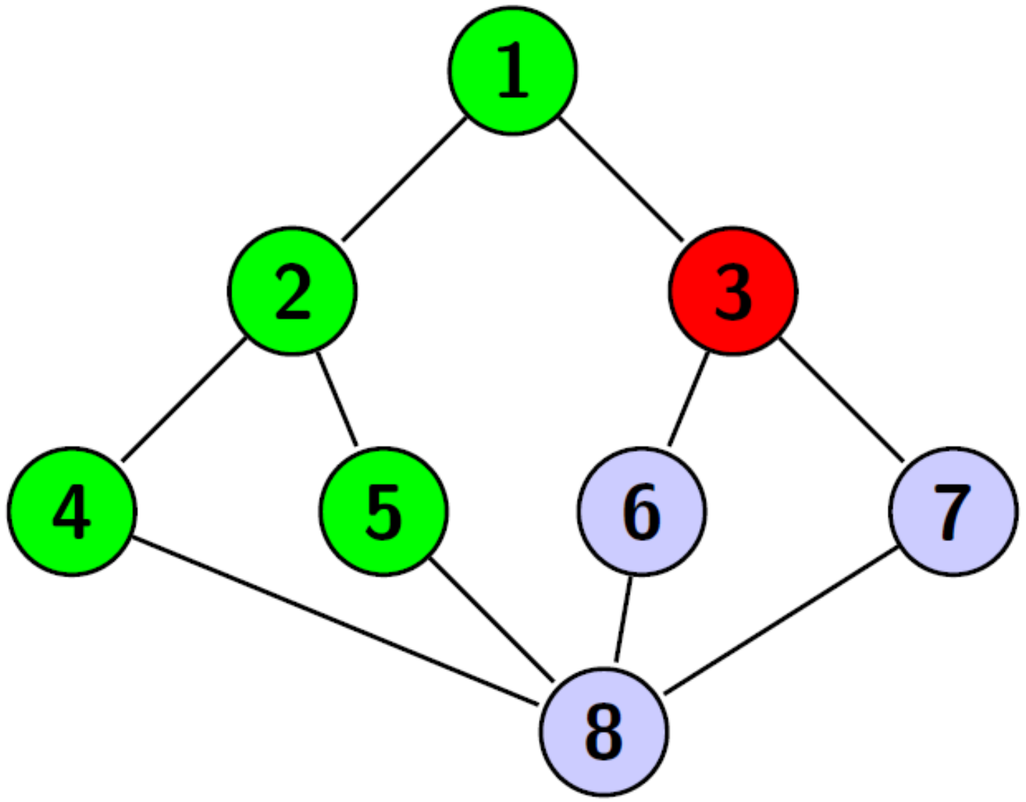



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Largura

Primeiro da fila é 3.

Fila | 4 | 5 |.

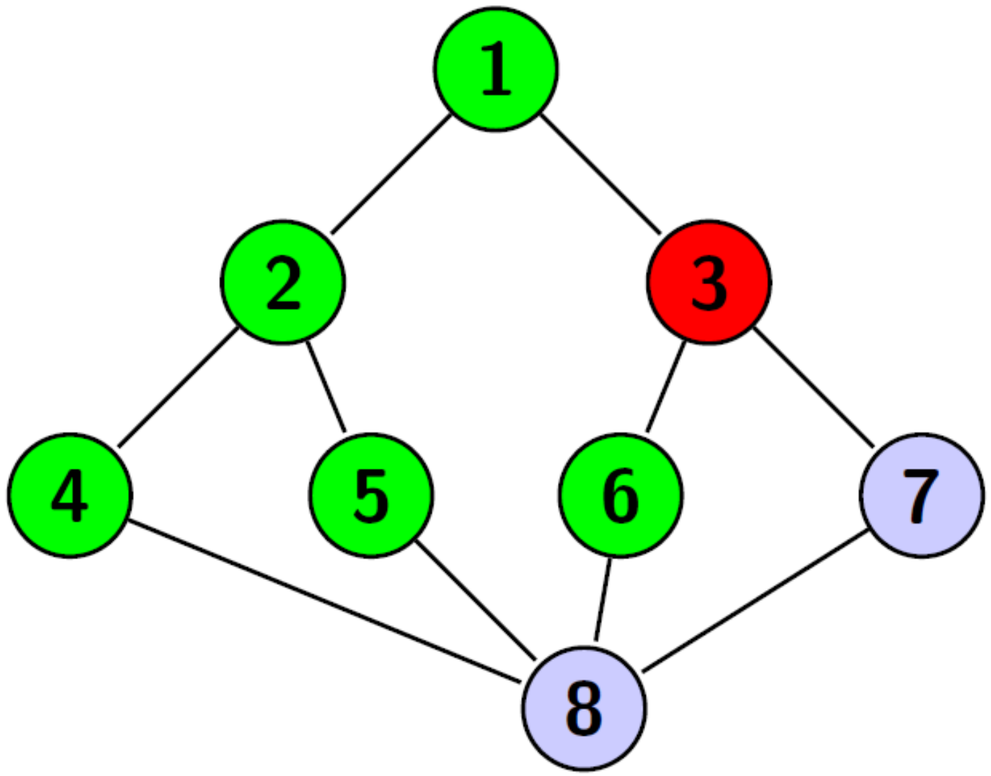



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Largura

Primeiro da fila é 3.

Fila | 4 | 5 | 6 |.

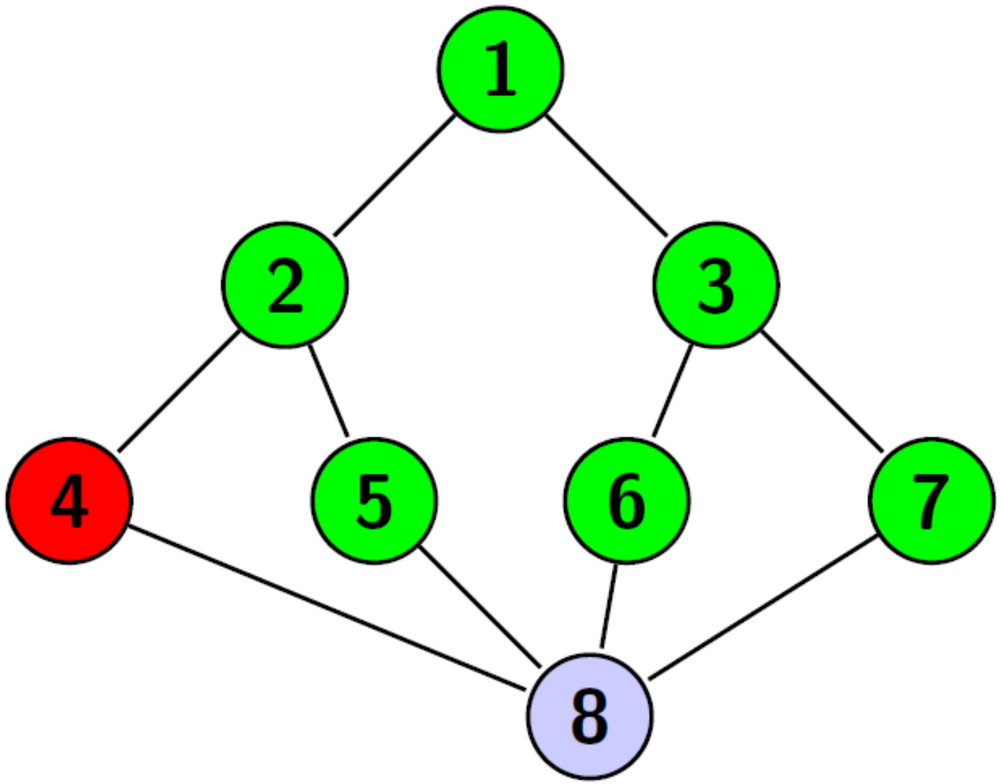



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Largura

Primeiro da fila é 4.

Fila | 5 | 6 | 7 |.

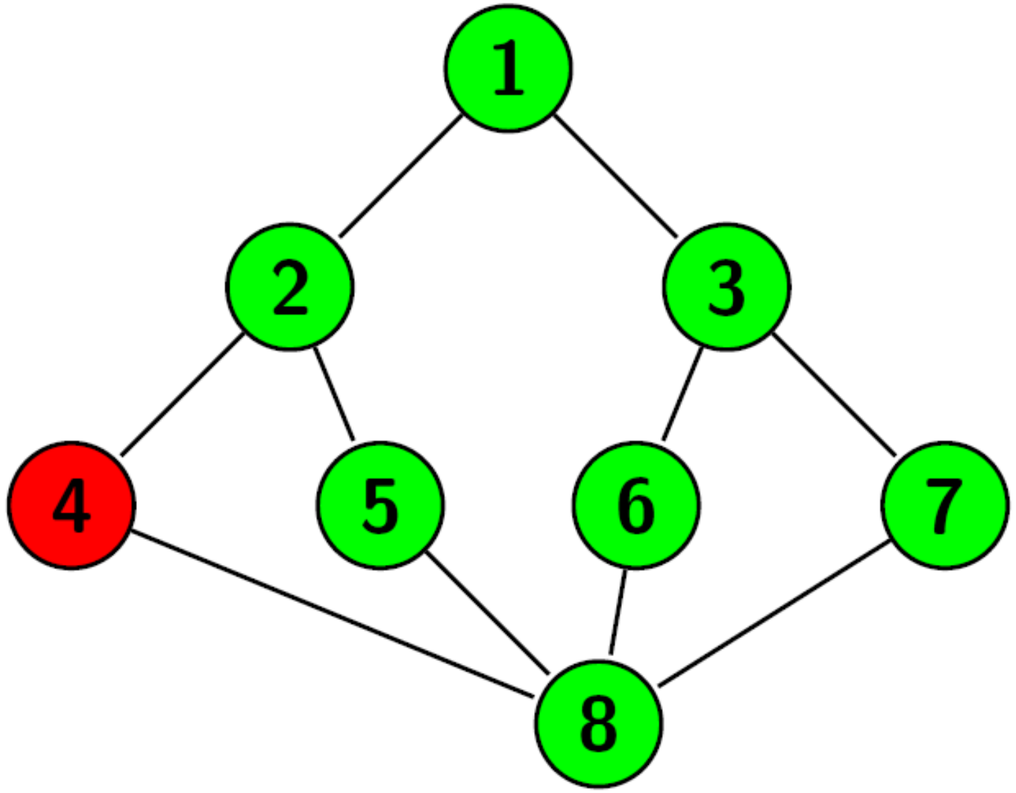



- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Exemplo - Busca em Largura

Primeiro da fila é 4.

Fila | 5 | 6 | 7 |.



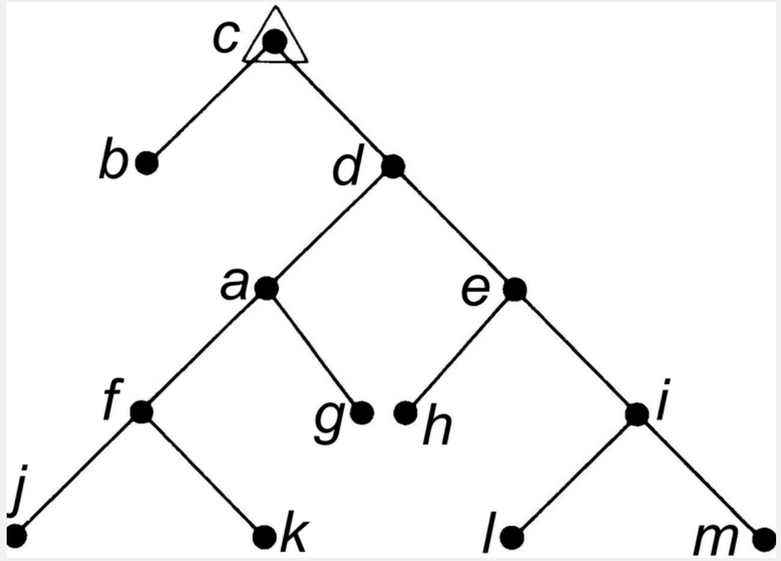

- 1 — 2 — 3
- 2 — 1 — 4 — 5
- 3 — 1 — 6 — 7
- 4 — 2 — 8
- 5 — 2 — 8
- 6 — 3 — 8
- 7 — 3 — 8
- 8 — 4 — 5 — 6 — 7

# Qual o pior caso?

- Qual é o pior caso?
  - Custo de inserção e remoção em fila é constante
  - Todos os vértices são enfilerados e removidos uma vez $O(|V|)$
  - Todas as arestas são utilizadas $|A|$
  - Complexidade de pior caso $O(|V| + |A|)$

# Exercício

1) Execute a busca em Largura visando encontrar o Vértice l;




2) Escreva o PseudoCódigo do método de busca em Largura.

# Dúvidas??