

## Fontes

### Algoritmos

Pedro Hokama

- [cls] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest, Ronald L. Rivest e Clifford Stein.
- [timr] Algorithms Illuminated Series, Tim Roughgarden

Apresentação Baseada:

- Stanford Algorithms  
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V>  
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZ17A5HEK6>
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende

1 / 15

2 / 15

### Busca em Grafos

- O **Problema de Busca em grafos** é um dos problemas mais fundamentais em Grafos
- É base para milhares de outras aplicações, incluindo o também fundamental problema de **caminho mais curto em grafos**

3 / 15

### Busca em Grafos

Motivação:

- Verificar se sua rede é conexa. Telefonia, Estradas, Redes de Computadores.
- Conexidade/Caminhos em redes não físicas. Por exemplo um Grafo em que vértices são atores e existe uma aresta entre eles se já atuaram em uma mesma produção.
- Bacon-Number, o menor número de saltos que você precisa dar até chegar no ator Kevin Bacon<sup>1</sup>
- Erdős-Number <sup>2</sup>
- Rota no GPS
- Pode ser algo ainda mais abstrato, um caminho pode ser uma sequência de decisões. Então você pode querer saber qual a sequência de decisões deve ser feita até uma determinada solução de um problema.

<sup>1</sup><https://oracleofbacon.org/>

<sup>2</sup><https://mathscinet.ams.org/mathscinet/freeTools.html?version=2>

4 / 15

## Busca em Grafos

Veremos dois métodos fundamentais em Grafos.

- Busca em Largura (*Breadth-First Search*, BFS)
- Busca em Profundidade (*Depth-First Search*, DFS)
- Geralmente nossa busca começa por um vértice inicial, chamado fonte (*source*), normalmente representado por  $s$ .
- Objetivos:
  - 1 Encontrar tudo que puder ser alcançado pela fonte. (Grafos direcionados ou não)
  - 2 Não explorar nada duas vezes ( $O(m + n)$ ). Ou seja queremos resolver em tempo linear no tamanho do Grafo!

5 / 15

## Busca em Grafo Genérica

### Teorema

Ao final do algoritmo,  $v$  está visitado se e somente se  $G$  tem um caminho de  $s$  até  $v$ .

Para provar um **se e somente se** temos sempre que provar as duas direções.

( $\implies$ ) Se  $v$  está visitado então existe um caminho de  $s$  até  $v$ .

( $\impliedby$ ) Se existe um caminho de  $s$  até  $v$  então  $v$  será visitado.

- Por contradição, suponha que  $G$  tem um caminho  $p$  de  $s$  até  $v$  mas  $v$  está não-visitado no final do algoritmo.
- Então existe uma aresta  $\{u, x\}$  em  $P$  com  $u$  visitado e  $x$  não visitado. (talvez  $u = s$  ou ainda  $x = v$ ).
- Mas se houvesse tal aresta o algoritmo não teria terminado. Encontramos uma Contradição!  $\square$

7 / 15

## Busca em Grafo Genérica

Algoritmo Genérico

---

**Algoritmo 1:** Busca Genérica

---

**Entrada:** Um Grafo  $G$ , e um vértice fonte  $s$

- 1 Marcar  $s$  como visitado ;
  - 2 Marcar todos os outros vértices como não-visitado;
  - 3 **enquanto** *for possível* **faça**
  - 4     escolha uma aresta  $\{u, v\}$  com  $u$  visitado e  $v$  não visitado, se não houver tal aresta, **pare**;
  - 5     Marcar  $v$  como visitado;
- 

6 / 15

## Busca em Largura (BFS)

- Explorar os vértices em camadas.
- Poderemos usar esse algoritmo para encontrar caminhos mais curtos.
- Encontrar componentes Conexas de grafos não direcionado.
- Pode ser feita em tempo linear  $O(m + n)$  😎

8 / 15

## Propriedades da Busca em Largura

---

### Algoritmo 2: Busca em Largura (BFS)

---

**Entrada:** Um Grafo  $G$ , e um vértice fonte  $s$

- 1 Marcar  $s$  como visitado;
  - 2 Marcar todos os outros vértices como não-visitado;
  - 3 Seja  $F$  uma Fila inicializada com  $s$ ;
  - 4 **enquanto**  $F \neq \emptyset$  **faça**
  - 5     remova o primeiro vértice de  $F$ , denotado por  $u$ ;
  - 6     **para cada aresta**  $\{u, v\}$  **faça**
  - 7         **se**  $v$  *está não-visitado* **então**
  - 8             Marcar  $v$  como visitado;
  - 9             Adicionar  $v$  em  $F$ .
- 

### Teorema

Ao final do BFS,  $v$  é visitado  $\iff$  existem um caminho de  $s$  até  $v$  em  $G$

- A BFS é um caso especial da Busca Genérica.

### Teorema

O tempo de execução total do loop principal é  $O(n_s + m_s)$  em que  $n_s$  é o número de vértices alcançáveis por  $s$ ,  $m_s$  é o número de arestas alcançáveis por  $s$ .

- Ver pseudo-código.

9 / 15

10 / 15

## Usando a BFS para Caminhos Mais Curtos

- Objetivo: encontrar, para todo vértice  $v \in V$ ,  $dist(v)$ , o menor número de arestas de um  $s - v$  caminho
- Por convenção se não existe um caminho de  $s$  até  $v$ , dizemos que  $dist(v) = \infty$

---

### Algoritmo 3: Busca em Largura (BFS)

---

**Entrada:** Um Grafo  $G$ , e um vértice fonte  $s$

- 1 Marcar  $s$  visitado e  $V - \{s\}$  como não-visitado;
  - 2 Seja  $F$  uma Fila inicializada com  $s$ ;
  - 3 **enquanto**  $F \neq \emptyset$  **faça**
  - 4     remova o primeiro vértice de  $F$ , denotado  $u$ ;
  - 5     **para cada aresta**  $\{u, v\}$  **faça**
  - 6         **se**  $v$  *está não-visitado* **então**
  - 7             Marcar  $v$  como visitado;
  - 8             Adicionar  $v$  em  $F$ ;
- 

---

### Algoritmo 4: Caminho mais curto

---

**Entrada:** Um Grafo  $G$ , e um vértice fonte  $s$

- 1 Marcar  $s$  visitado e  $V - \{s\}$  como não-visitado;
  - 2  $dist(s) = 0$ ;  $dist(v) = \infty, \forall v \in V \setminus \{s\}$ ;
  - 3 Seja  $F$  uma Fila inicializada com  $s$ ;
  - 4 **enquanto**  $F \neq \emptyset$  **faça**
  - 5     remova o primeiro vértice de  $F$ , denotado  $u$ ;
  - 6     **para cada aresta**  $\{u, v\}$  **faça**
  - 7         **se**  $v$  *está não-visitado* **então**
  - 8             Marcar  $v$  como visitado;
  - 9             Adicionar  $v$  em  $F$ ;
  - 10              $dist(v) = dist(u) + 1$ ;
- 

11 / 15

12 / 15

## Caminho mais curto

### Teorema

Ao final do algoritmo  $\text{dist}(v) = i \iff v$  está na  $i$ -ésima camada.

- Se  $v$  está na  $i$ -ésima camada o  $s - v$  caminho mais curto tem  $i$  arestas.
- Toda vértice  $v$  da camada  $i$  é adicionado em  $F$  por um vértice  $u$  da camada  $i - 1$ , através da aresta  $\{u, v\}$ .  $\square$