Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

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**

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[1]
- Erdös-Number [2]
- Rota no GPS
- 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.

[1] `https://oracleofbacon.org/`

[2] `https://mathscinet.ams.org/mathscinet/freeTools.html?version=2`

Busca em Grafos

Veremos dois métodos fundamentais em Grafos.

- Busca em Largura (*Breadth-First Search*, BFS)
- Busca em Profundidade (*Depth-First Search*, DFS)
- Dado um Grafo $G = (V, E)$ com $|V| = n$ e $|E| = m$.
- Geralmente nossa busca começa por um vértice inicial, chamado fonte (*source*), normalmente representado por $s \in V$.
- 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!

Busca em Grafo Genérica

Algoritmo Genérico

Algoritmo 1: Busca Genérica

Entrada: Um Grafo G, e um vértice fonte s

```
Marcar s como visitado ;
Marcar todos os outros vértices como não-visitado;
enquanto for possível faça
    escolha uma aresta {u, v} com u visitado e v não visitado,
    se não houver tal aresta, pare;
    Marcar v como visitado;
```


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.
($\Longrightarrow$) Se v está visitado então existe um caminho de s até v.
($\Longleftarrow$) 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! $\square$

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 $O(m + n)$ 😎

Algoritmo 2: Busca em Largura (BFS)

Entrada: Um Grafo G, e um vértice fonte s

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

Propriedades da Busca em Largura

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.

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

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

Algoritmo 4: Caminho mais curto

Entrada: Um Grafo G, e um vértice fonte s

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

Caminho mais curto

Teorema

Ao final do algoritmo $dist(v) = i \iff v$ 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 $\{u, v\}$. $\square$