Algoritmos

Pedro Hokama

Fontes

- [*clrs*] 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=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

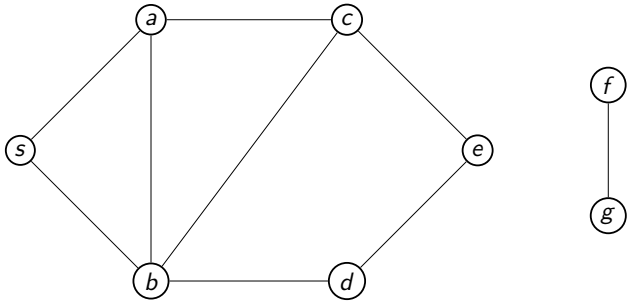
Qualquer erro é de minha responsabilidade.

BFS

Algoritmo 1: 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.



Algoritmo 2: Caminho mais curto

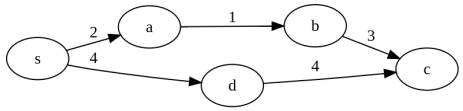
Entrada: Um Grafo G, e um vértice fonte s

```
1  Marcar s visitado e V − {s} como não-visitado;
2  dist(s) = 0; dist(v) = ∞, ∀v ∈ V\{s};
3  Seja F uma Fila inicializada com s;
4  enquanto F ≠ ∅ 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;
```

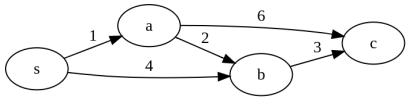

Caminhos mais Curtos de Única Fonte

Problema do Caminho mais Curto de Única fonte

Dado um grafo direcionado $G = (V, A)$, em que cada arco $a \in A$ tem um comprimento c_a, e um vértice fonte $s \in V$. Desejamos calcular para cada vértice $v \in V$ o valor $D(v)$ da distância do $s - v$ caminho mais curto.



- No exemplo acima, $D(c) = 6$.
- Os comprimentos tem que ser **positivos**, ou seja, $c_a \geq 0, \forall a \in A$.

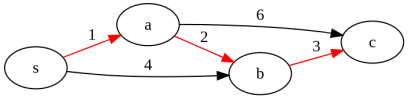


No grafo acima, qual a distância do caminho mais curto de s até s, a, b, c respectivamente?

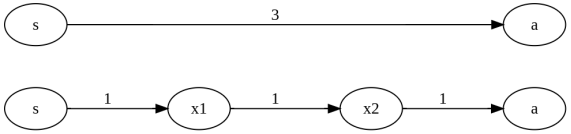
a) 0, 1, 2, 3
b) 0, 1, 4, 7
c) 0, 1, 4, 6
d) 0, 1, 3, 6

Considerações

- Já não tinhamos visto um algoritmo de caminho mais curto? A BFS? 🤔
- Sim, mas ela encontra o caminho com o menor número de arcos, o que é útil apenas se os comprimentos dos arcos forem iguais! Note que não é o caso, no exemplo abaixo o caminho mais curto de s até c tem 3 arcos, mas existe um caminho com menos arcos.



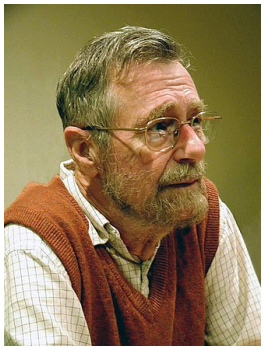
- E se substituirmos um caminho de distância 3 por 3 arcos de tamanho 1?



- Funciona! Porém pode ficar muito complexo, imagine arcos com distâncias 1000 ou mais.

Edsger W. Dijkstra

- Edsger W. Dijkstra (1930 - 2002) foi um Cientista da Computação holandês.
- A maior parte de sua carreira foi como Professor na Eindhoven University of Technology e na University of Texas at Austin.
- Um dos fundadores da ciência da computação, conhecido por contribuições em diversas áreas: Compiladores, Sistemas Operacionais, Sistemas Distribuídos, Teoria da Computação, etc
- Ganhador do Turing Award (1972).



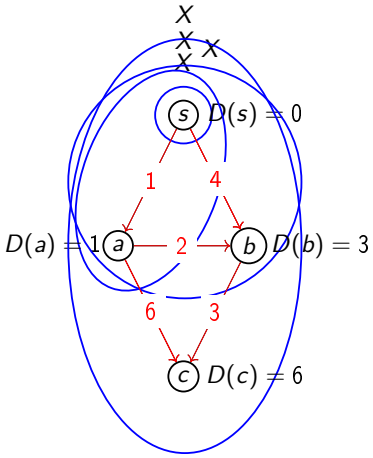
Edsger W. Dijkstra

> *What is the shortest way to travel from Rotterdam to Groningen, in general: from given city to given city. It is the algorithm for the shortest path, which I designed in about twenty minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path. As I said, it was a twenty-minute invention. In fact, it was published in '59, three years later. The publication is still readable, it is, in fact, quite nice. One of the reasons that it is so nice was that I designed it without pencil and paper. I learned later that one of the advantages of designing without pencil and paper is that you are almost forced to avoid all avoidable complexities. Eventually, that algorithm became to my great amazement, one of the cornerstones of my fame.*

Edsger Dijkstra, Communications of the ACM, 2001.

Algoritmo de Dijkstra

- Manter um conjunto X com os vértices já processados.
- Considerar os arcos que começam em X e terminam em $V \setminus X$
- Escolher o arco (u, v) que minimiza $D(u) + c_{(u,v)}$, ou seja, o arco que minimiza o caminho para um vértice de $V \setminus X$.
- Note que não existe um caminho menor para chegar em v.
- Computa $D(v)$ e inclui v em X.
- Repita até que todos os vértices estejam em X, ou não tenha nenhum arco.



Algoritmo 3: Dijkstra

Entrada: Um Grafo Direcionado G, custos c_a e um vértice fonte s

Saída: Comprimento do menor caminho para todo vértice v

1 $X = \{s\}$;
2 $D(s) = 0$;
3 **enquanto** $X \neq V$ **faça**
4 Escolha o arco (u, v), tal que $u \in X$, $v \notin X$, que minimiza $D(u) + c_{(u,v)}$;
5 $X = X \cup \{v\}$;
6 $D(v) = D(u) + c_{(u,v)}$;
7 **devolva** D ;

Corretude do Algoritmo de Dijkstra

Teorema

Para qualquer grafo direcionado com arcos de comprimento não negativos, o Algoritmo de Dijkstra calcula corretamente as distâncias dos caminhos mais curtos a partir de um vértice fonte.

Prova:

- Durante todo o algoritmo, todos os vértices $v \in X$ tem $D(v)$ corretamente calculado.
- Por indução no número de iterações.
- Caso Base: no começo do algoritmo $D(s) = 0$ que está correto.
- Hipótese: o Algoritmo calcula corretamente para todas as iterações anteriores, ou seja, para todo vértice $v \in X$, $D(v)$ está computado corretamente.
- Passo: (no próximo slide)

- O algoritmo escolhe o arco (u, v) para adicionar v à X.
- Pela hipótese de indução $D(u)$ é de fato a distância de um $s - u$ caminho mais curto. Denotaremos esse $s - v$ caminho de P
- Suponha por absurdo que existe um outro caminho P' mais curto de s para v.
- Esse caminho precisou passar pela fronteira de X. Digamos que entre o vértice j e k.
- O arco j e k era um candidato a ser escolhido pelo algoritmo. Porem não foi. Logo $P' \geq D(j) + c_{(j,k)} \geq D(u) + c_{(u,v)} = P$.
- Logo o caminho P' é maior que P o que é um absurdo!

Implementação e Complexidade

Algoritmo 4: Dijkstra

Entrada: Um Grafo Direcionado G, custos c_a e um vértice fonte s

Saída: Comprimento do menor caminho para todo vértice v

1 $X = \{s\}$;
2 $D(s) = 0$;
3 **enquanto** $X \neq V$ **faça**
4 Escolha o arco (u, v), tal que $u \in X$, $v \notin X$, que minimiza $D(u) + c_{(u,v)}$;
5 $X = X \cup \{v\}$;
6 $D(v) = D(u) + c_{(u,v)}$;
7 **devolva** D ;

- Uma implementação ingênua do Algoritmo de Dijkstra, onde a cada iteração analisamos todas os arcos, teria uma complexidade de $O(|V| \times |A|)$, o que pode ser melhorado.
- O que fazemos em toda iteração é escolher o mínimo de algum conjunto.
- Será que temos um jeito interessante de escolher mínimos várias vezes? 🤔

Heaps (Fila de Prioridade)

- Inserir, Pegar o Mínimo e Deletar em $O(\log n)$.
- **Relembrando:** é uma árvore binária perfeitamente balanceada (apesar disso, normalmente é implementada em um array).
- A chave de cada nó é sempre menor que a chave dos nós filhos.

- No algoritmo de Dijkstra colocamos no Heap os vértices v que podem ser adicionados em X usando como chave o valor de $D(u) + c_{(u,v)}$.
- O problema: quando um novo caminho é descoberto para um vértice que já está no Heap, ele precisa ser Deletado e Inserido novamente.
- No total temos $|V|$ operações de Pegar o Mínimo, e $|A|$ operações de Inserir, ou (Deletar e Inserir). Todas as operações são $O(\log |V|)$.
- Se considerarmos grafos fracamente conexos então $|V| \leq |A|$
- Então o tempo total de execução do algoritmo é $O(|A| \log |V|)$