Grafos - Caminhos mais Curtos: BFS e Dijkstra

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

*Material baseado em aulas dos professores: Elaine Parros Machado de Souza, Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr., Maria Cristina Oliveira e Cristina Ciferri.

Caminho de um Grafo

Caminho de um Grafo

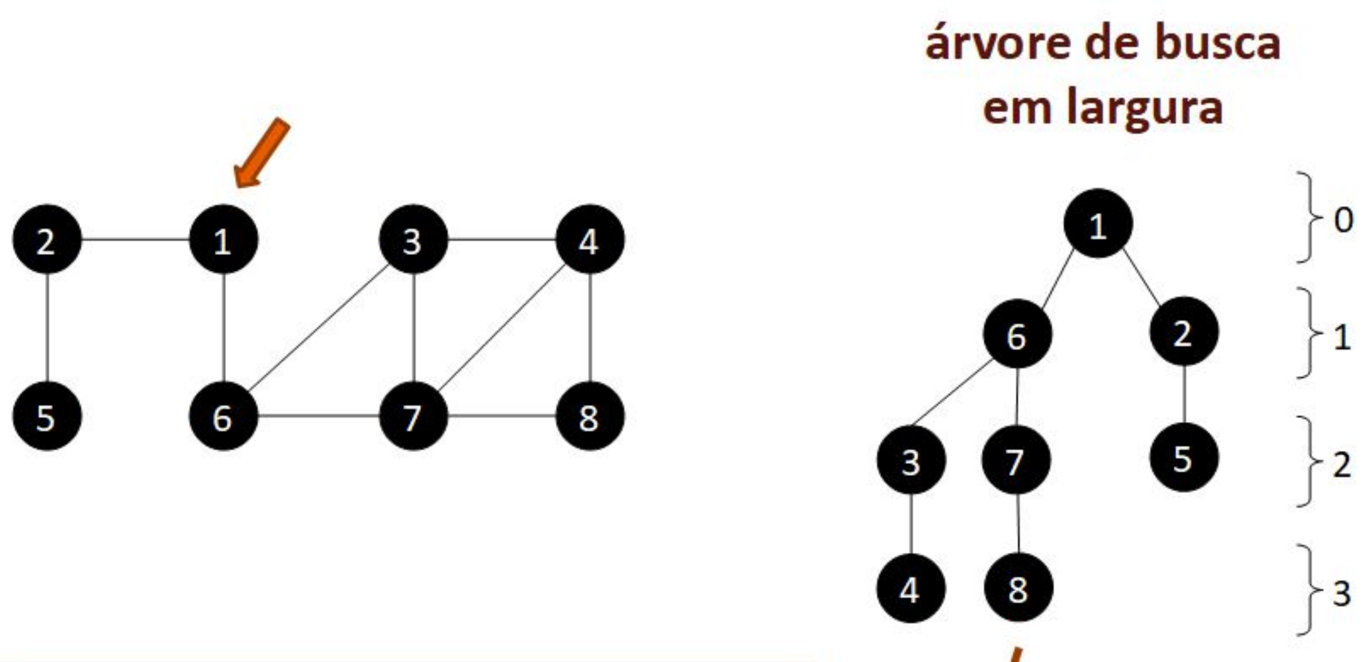
- Eu estou ligado a uma celebridade por uma cadeia de amigos?
  - Existe um caminho entre mim e uma celebridade?
  - Caminho
    - Sequência de arestas que conectam dois vértices.

Caminhos mais Curtos com Busca em Largura (BFS)

Caminhos mais Curtos - BFS

- Em grafos não ponderados, a busca em largura permite encontrar o caminho mais curto (menor número de arestas) entre o vértice de origem da busca e qualquer outro vértice alcançável a partir da origem.

Caminhos mais Curtos - BFS

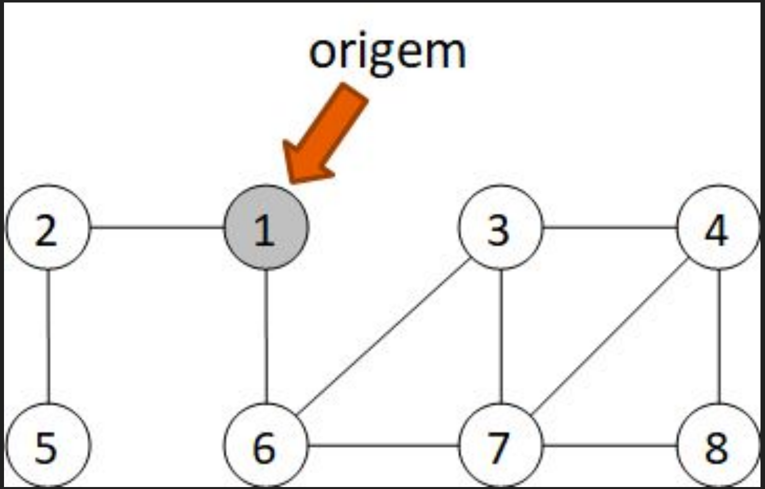


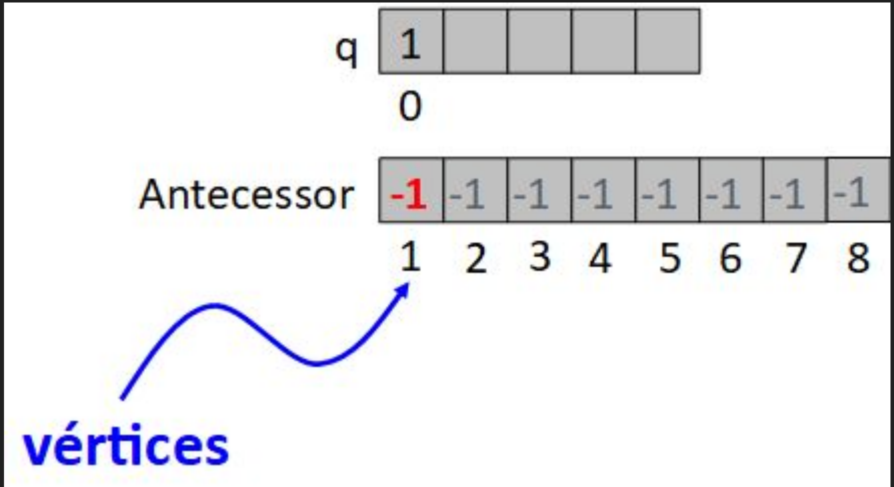
pode ser representada por um
vetor de antecessores

Caminhos mais Curtos - BFS

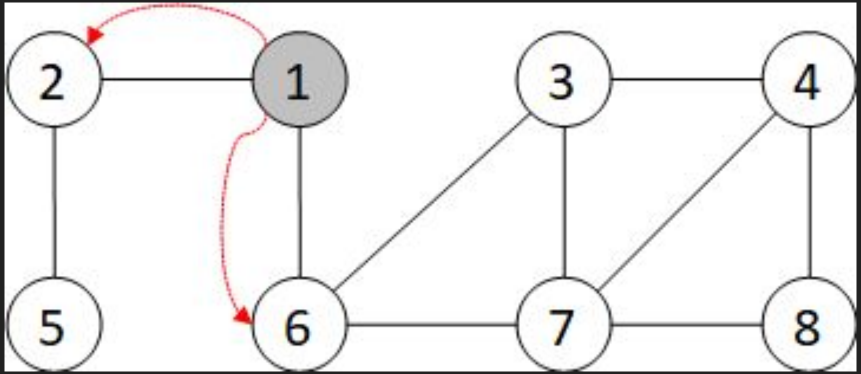
- Vetor de antecessores pode ser usado para reconstruir o caminho mais curto entre o vértice origem e cada vértice
    - Antecessor[v] contém o vértice imediatamente anterior ao vértice v no caminho mais curto.

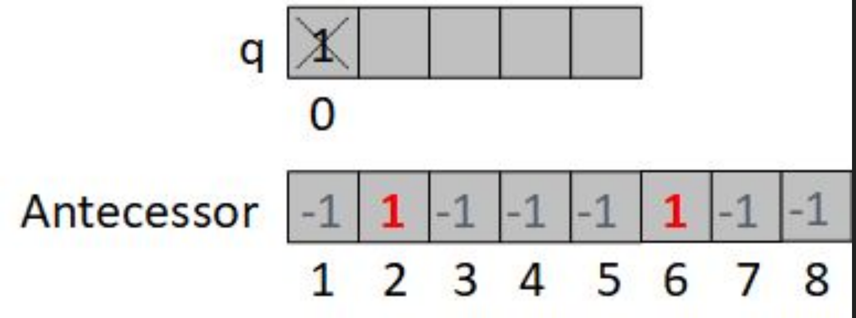
Caminhos mais Curtos - BFS



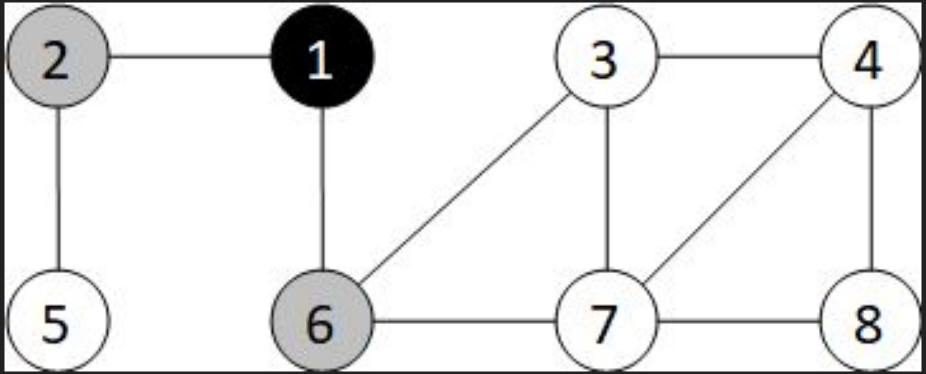


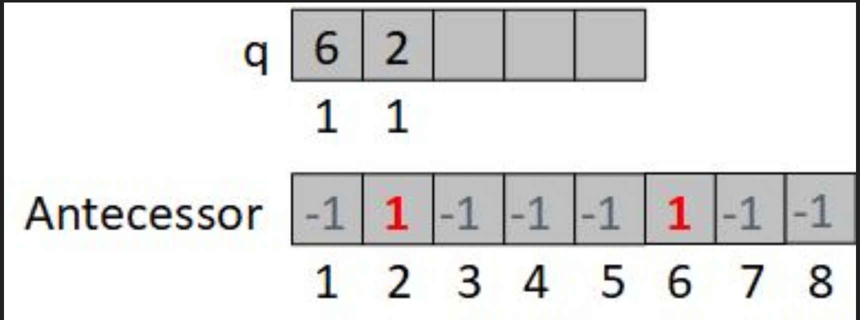
Caminhos mais Curtos - BFS



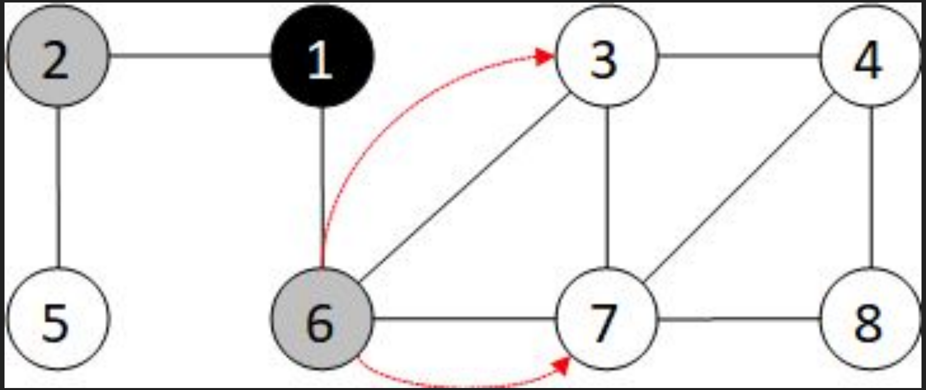


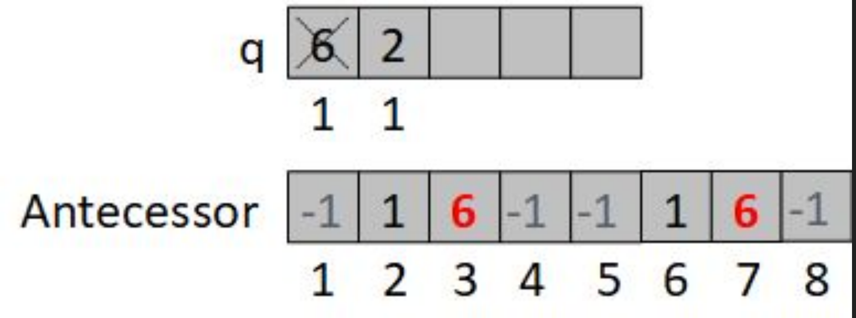
Caminhos mais Curtos - BFS



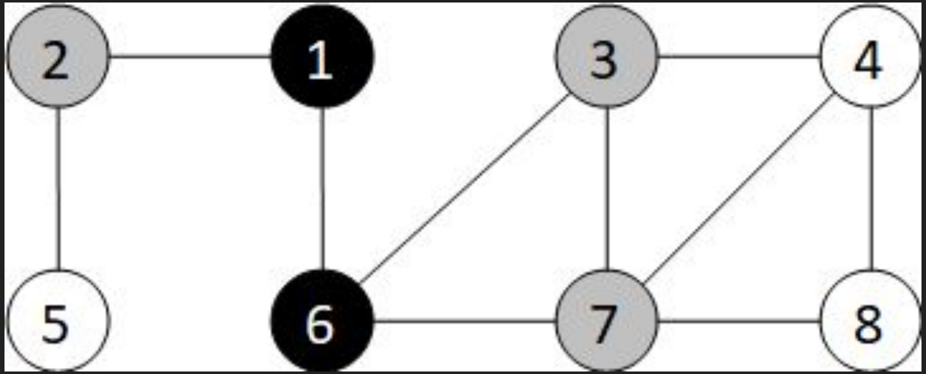


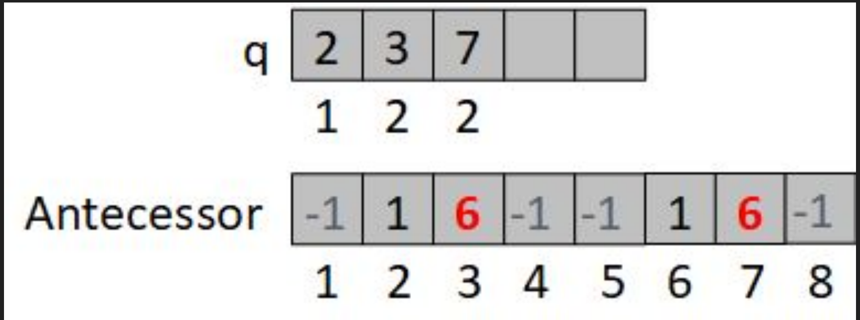
Caminhos mais Curtos - BFS



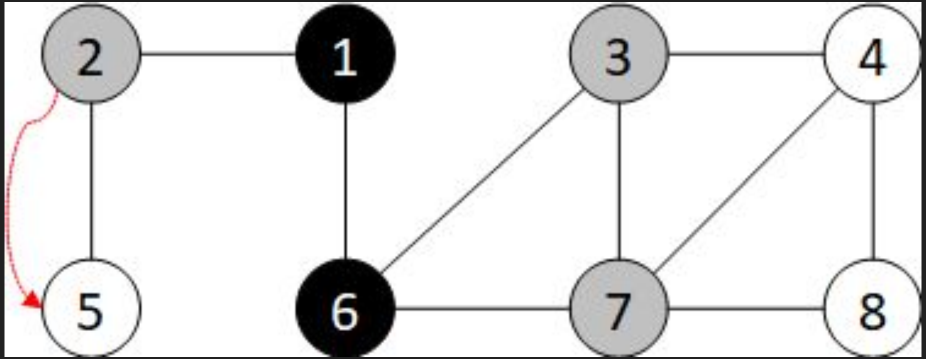


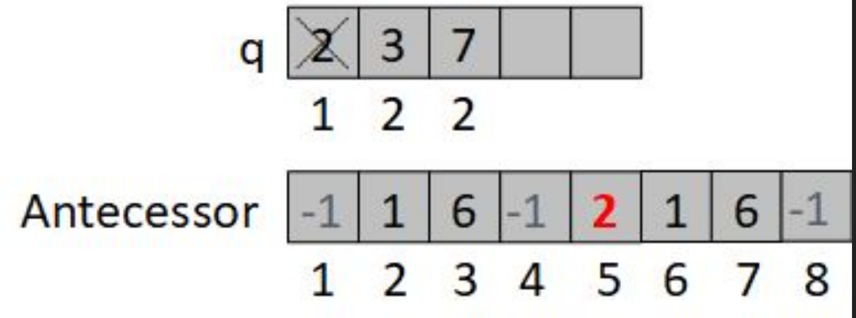
Caminhos mais Curtos - BFS



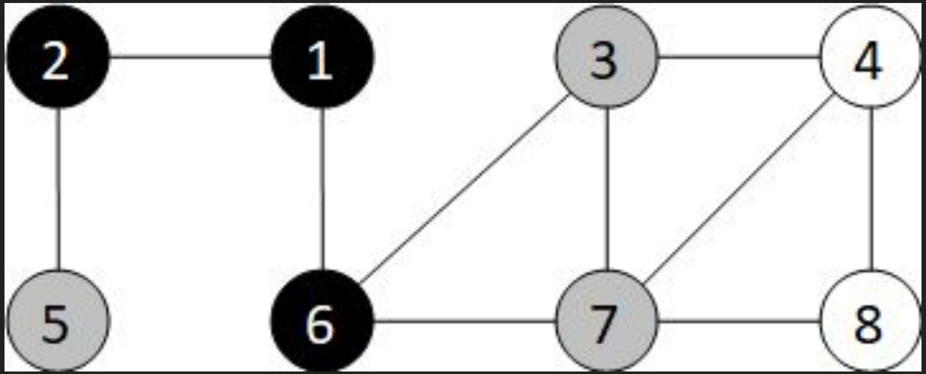


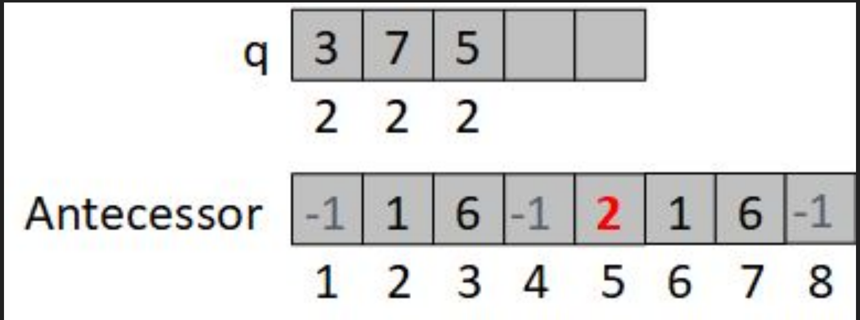
Caminhos mais Curtos - BFS



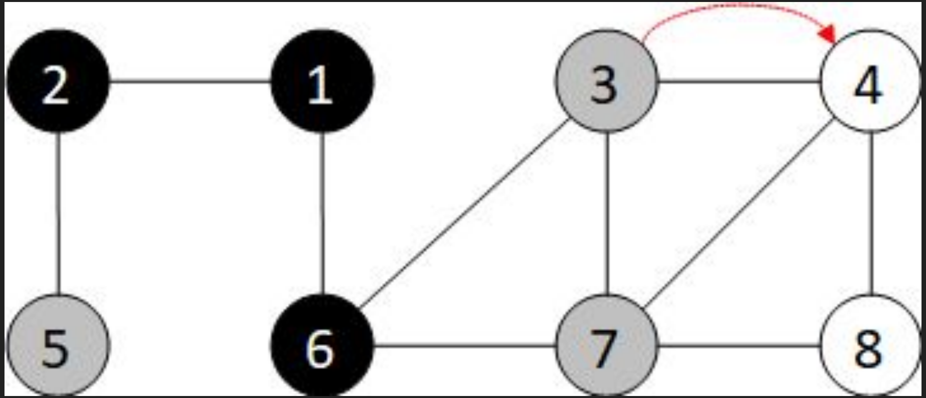


Caminhos mais Curtos - BFS





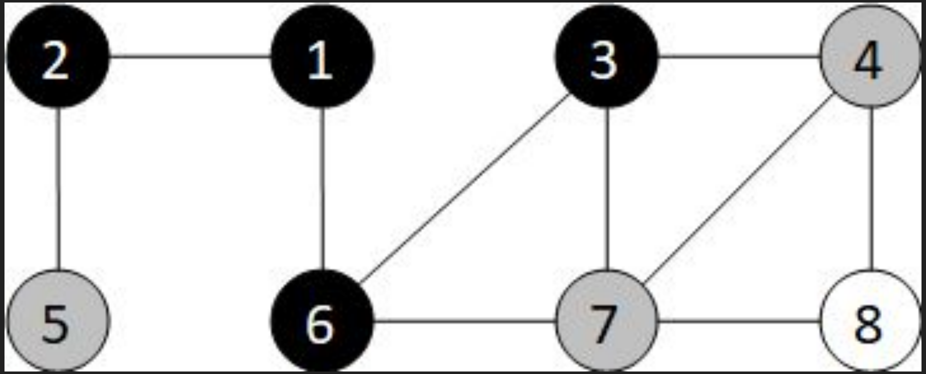
Caminhos mais Curtos - BFS

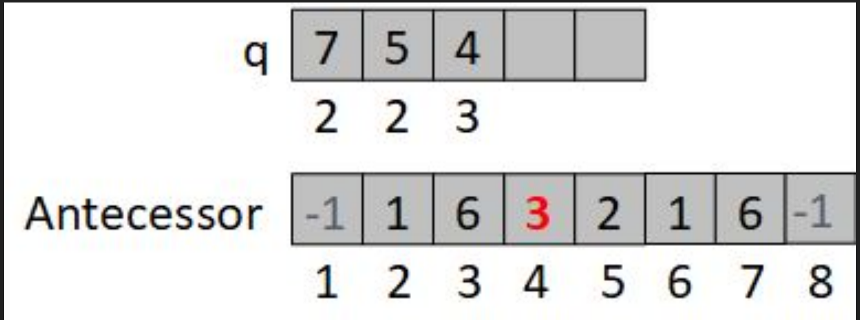


q	~~3~~	7	5		
	2	2	2		

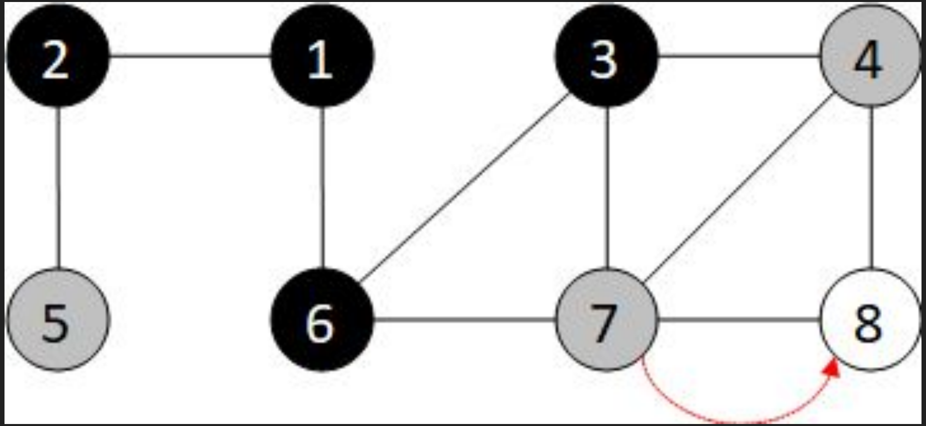
Antecessor	-1	1	6	3	2	1	6	-1
	1	2	3	4	5	6	7	8

Caminhos mais Curtos - BFS





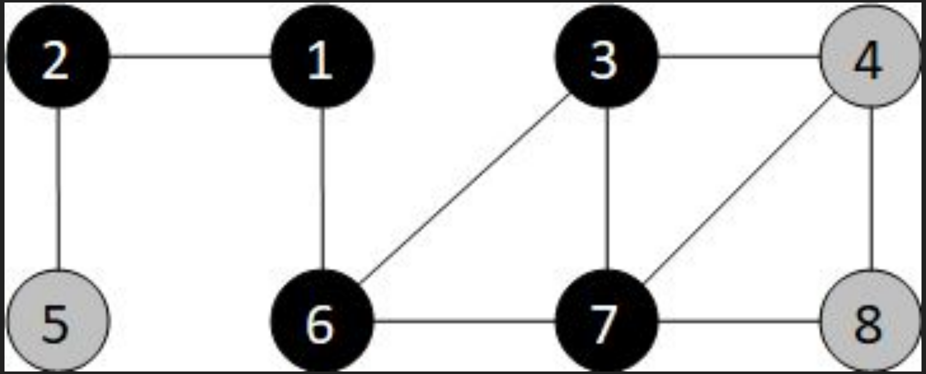
Caminhos mais Curtos - BFS

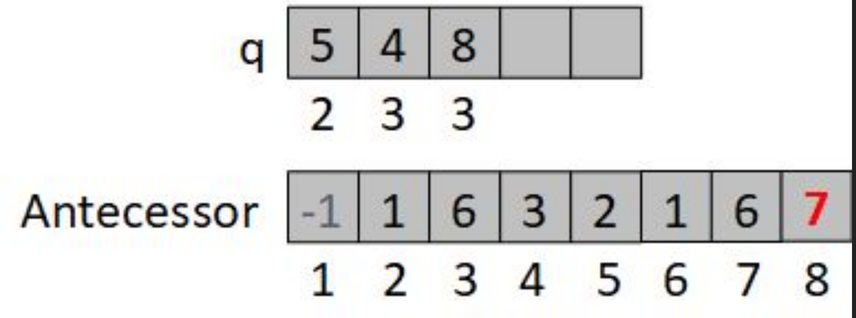


q	~~7~~	5	4		
	2	2	3		

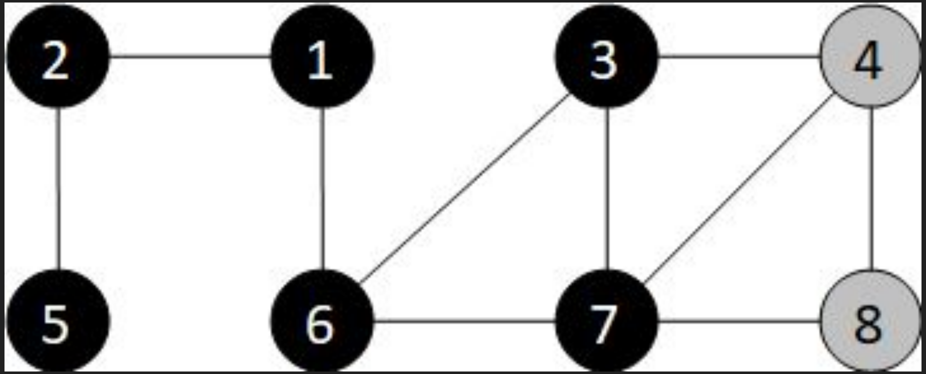
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8

Caminhos mais Curtos - BFS





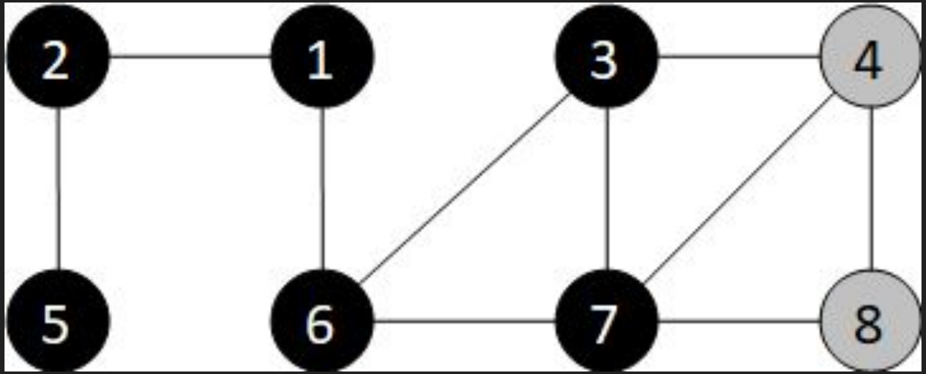
Caminhos mais Curtos - BFS

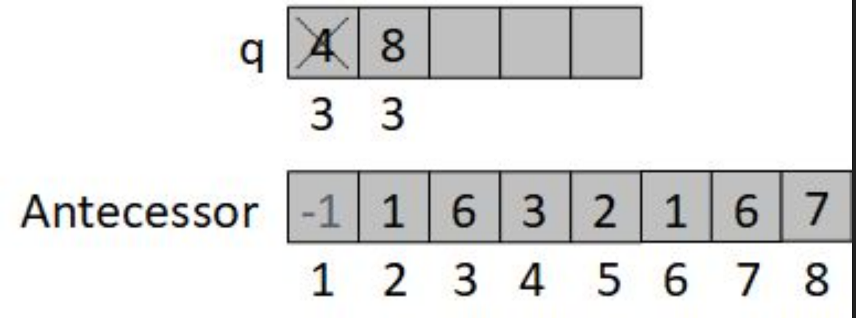


q	4	8			
	3	3			

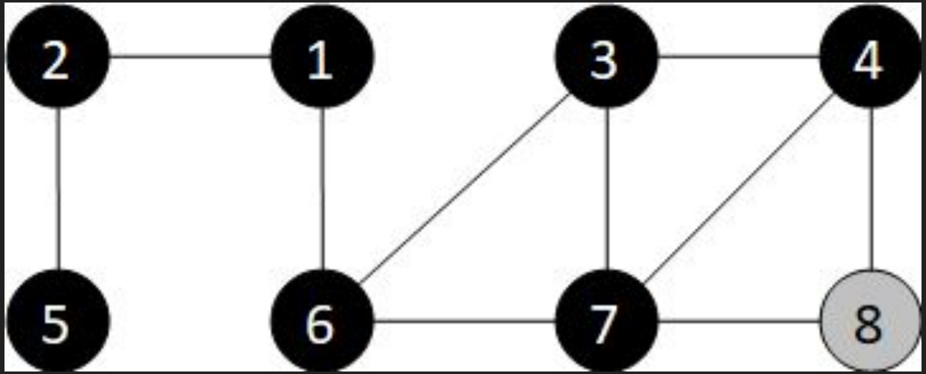
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8

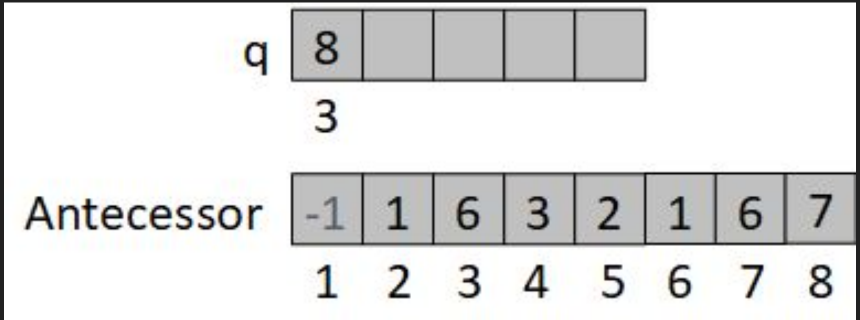
Caminhos mais Curtos - BFS



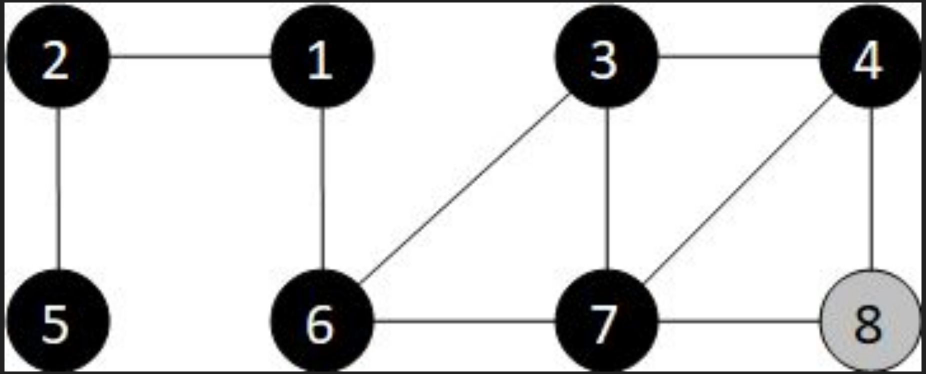


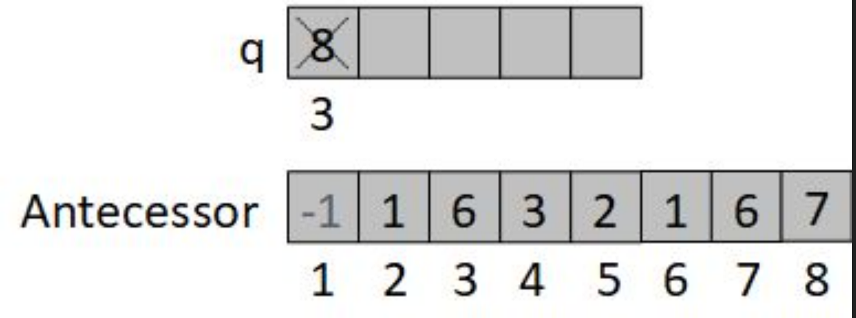
Caminhos mais Curtos - BFS



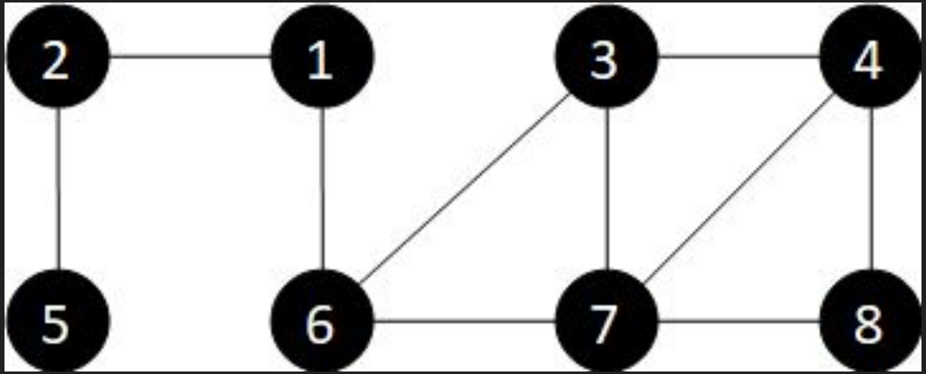


Caminhos mais Curtos - BFS



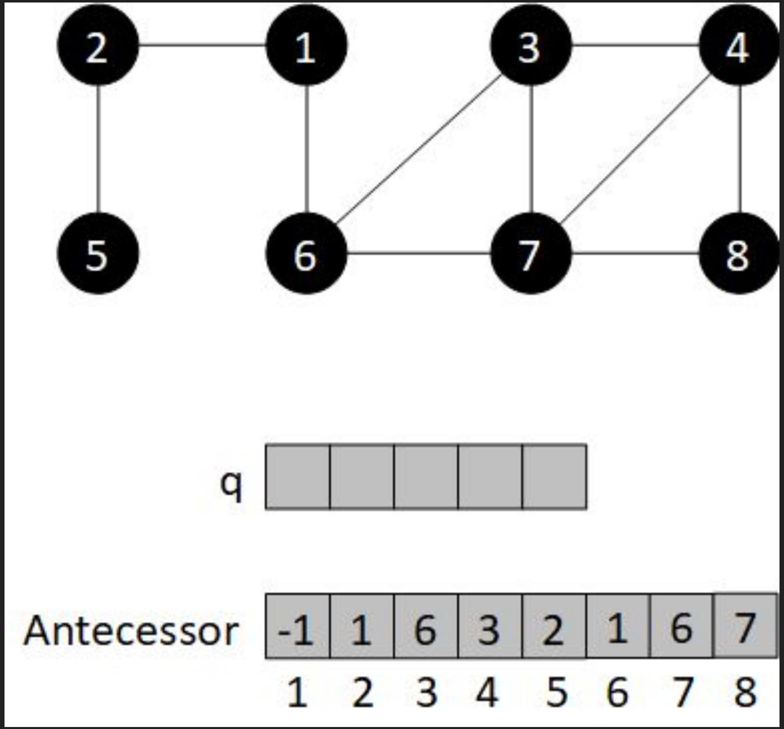


Caminhos mais Curtos - BFS

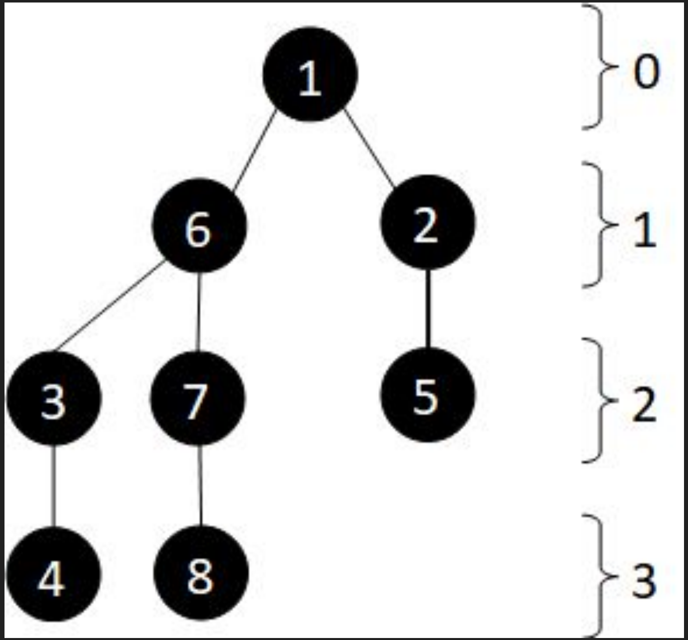


q								
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8

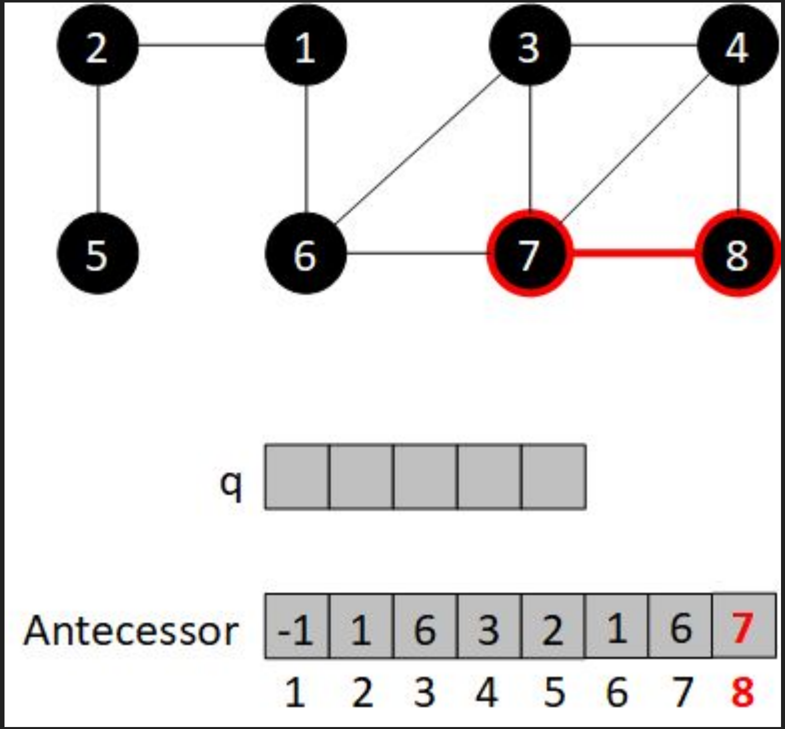
Caminhos mais Curtos - BFS



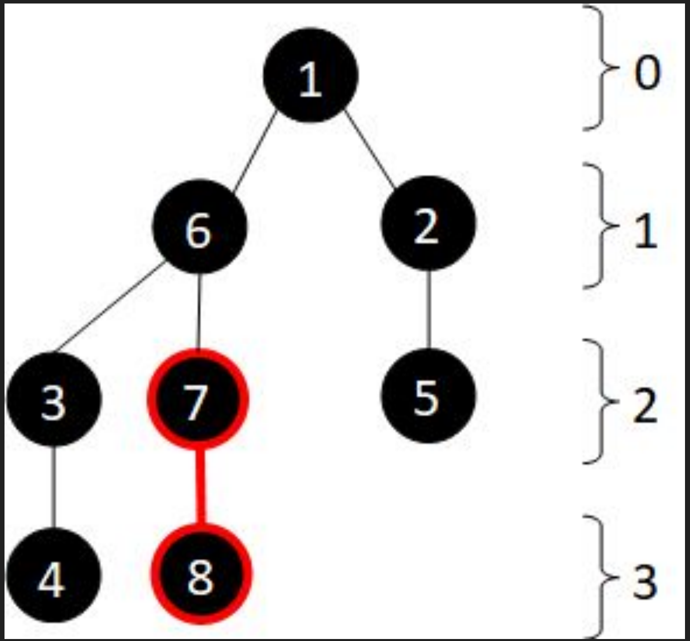
Caminho mais curto entre (1) e (8)



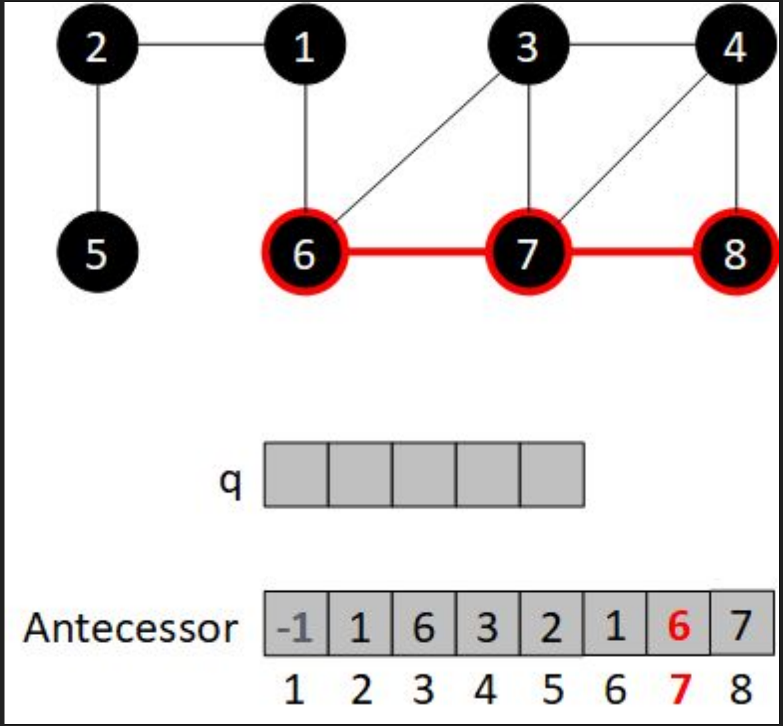
Caminhos mais Curtos - BFS



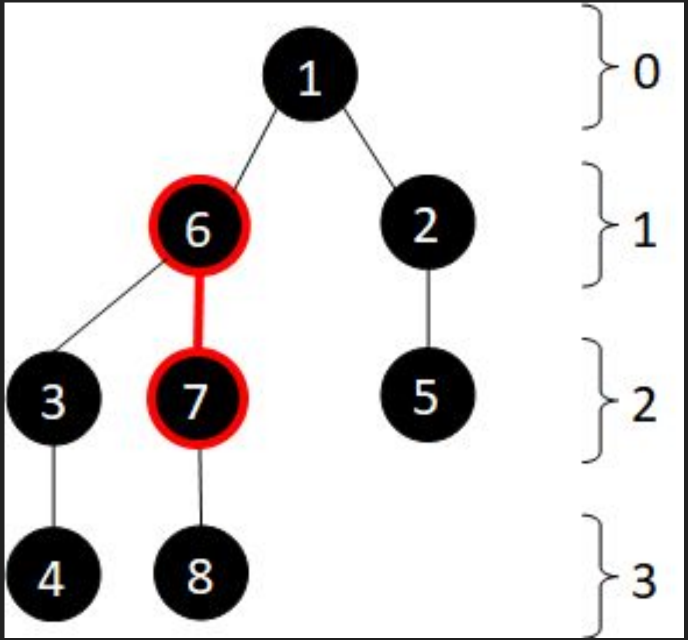
Caminho mais curto entre (1) e (8)



Caminhos mais Curtos - BFS

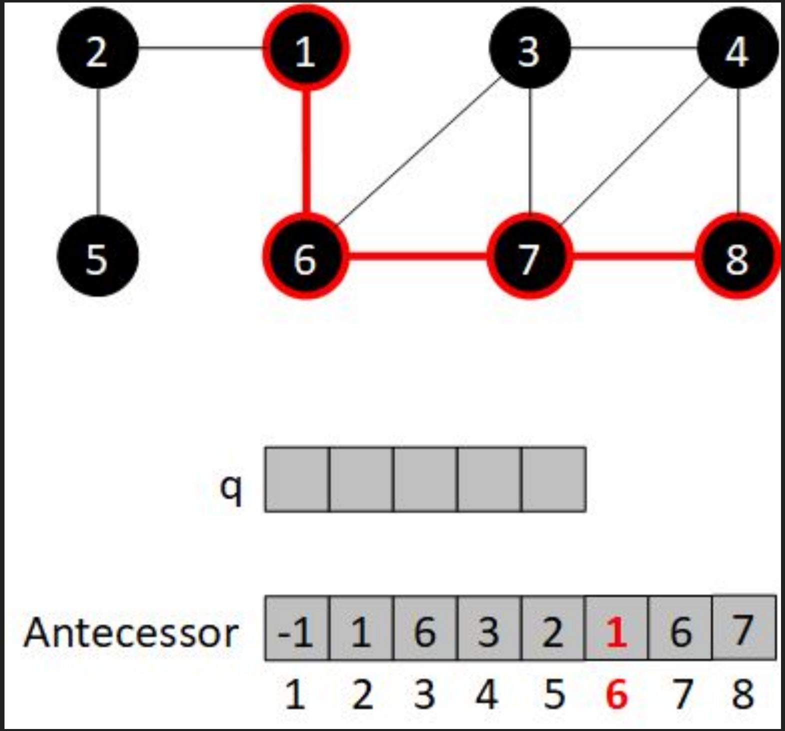


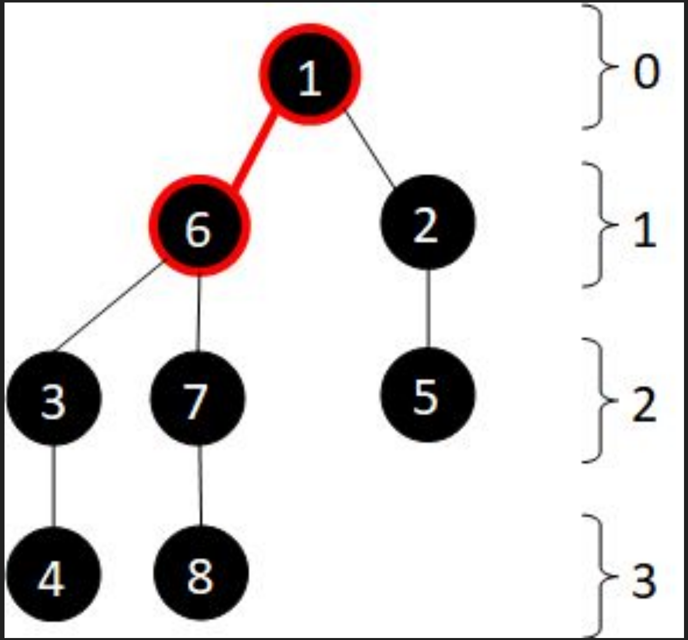
Caminho mais curto entre (1) e (8)



Caminhos mais Curtos - BFS

Caminho mais curto entre (1) e (8)





Caminhos mais Curtos - BFS

- Em grafos não ponderados (ou com arestas de mesmo peso), a busca em largura soluciona o problema de CAMINHOS MAIS CURTOS DE ORIGEM ÚNICA.

Caminhos mais Curtos - BFS

- Exemplos de problemas que podem ser resolvidos com algoritmo para encontrar caminho mais curto de origem única:
  - Caminhos mais curtos entre um par de vértices
    - Algoritmo para problema da origem única é a melhor opção.

Caminhos mais Curtos - BFS

- Exemplos de problemas que podem ser resolvidos com algoritmo para encontrar caminho mais curto de origem única:
  - Caminhos mais curtos entre todos os pares de vértices
    - Pode ser resolvido aplicando o algoritmo |V| vezes, uma vez para cada vértice origem.

Caminhos mais Curtos - BFS

- Exemplos de problemas que podem ser resolvidos com algoritmo para encontrar caminho mais curto de origem única:
  - Caminhos mais curtos com destino único
    - grafos direcionados
      - Reduzido ao problema de origem única invertendo a direção de cada aresta do grafo, ou seja, calculando o grafo transposto e iniciando a busca do vértice destino.

E para um grafo ponderado?

Algoritmo de Dijkstra!

Algoritmo de Dijkstra

- Algoritmo de Dijkstra
  - Encontra caminhos mais curtos em um grafo ponderado com pesos diferentes entre as arestas.

Algoritmo de Dijkstra

- Peso de um caminho *c = v0, v1, ..., vk* em um grafo ponderado
  - Soma de todos os pesos das arestas do caminho.
- Caminho mais curto do vértice *v0* para o vértice *vk*
  - Caminho de menor peso de *v0* para *vk*.
- Um caminho mais curto tem peso infinito se *vk* não é alcançável a partir de *v0*.

Algoritmo de Dijkstra

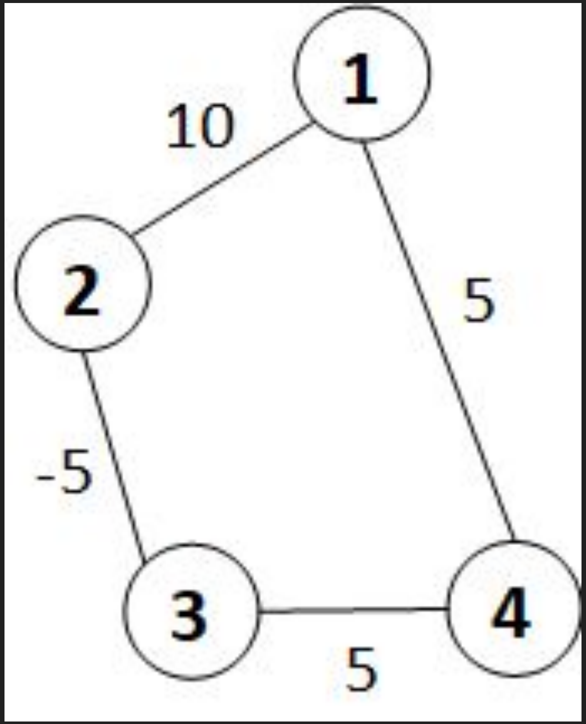
- Algoritmo de Dijkstra
    - Encontra os caminhos mais curtos para todos os vértices de um grafo G a partir de uma origem única.
- Soluciona o problema de caminhos mais curtos de origem única!

Algoritmo de Dijkstra

- Algoritmo de Dijkstra segue o Princípio da Otimalidade de Bellman
  - Soluções ótimas são encontradas a partir de sub-soluções prévias ótimas.
  - Um sub-caminho de um caminho mais curto é também um caminho mais curto por si só
    - Se não houver ciclo com aresta de peso negativo

Algoritmo de Dijkstra

- Caminho mais curto de 1 para 3:
  - 1 - 2 - 3
- Mas:
  - 1 - 2 não é o caminho mais curto de 1 para 2



Algoritmo de Dijkstra

- Estratégia:
  - Algoritmo mantém um conjunto **S** dos vértices cujos pesos finais dos caminhos mais curtos desde a origem já tenham sido determinados.
    - Inicialmente **S** contém somente o vértice origem.

Algoritmo de Dijkstra

- Estratégia:
  - Algoritmo “guloso” (greedy)
    - A cada iteração, um vértice $w \in (\mathbf{V} - \mathbf{S})$, cuja distância ao vértice origem é a menor até o momento, é adicionado a **S**.

Algoritmo de Dijkstra

- Estratégia:
  - Assumindo que todos os vértices possuem custos não negativos, sempre é possível encontrar um caminho mais curto do vértice origem até *w* que passa somente por vértices em **S**.

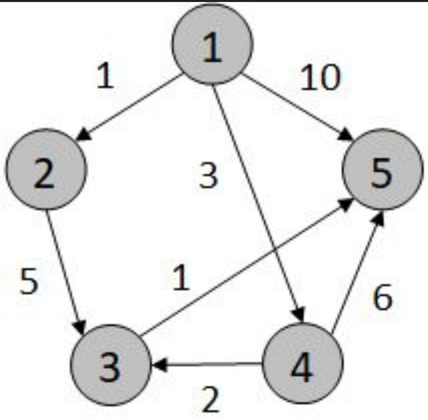
Algoritmo de Dijkstra

- Estratégia:
  - A cada iteração, um vetor D armazena o custo do caminho mais curto conhecido até o momento entre o vértice origem e os demais vértices do grafo.

Algoritmo de Dijkstra

- Estratégia:
  - Para os vértices em **S**, D possui o custo do caminho mais curto final.
  - O algoritmo termina quando todos os vértices estão em **S**.

Algoritmo de Dijkstra



$S = \varnothing$

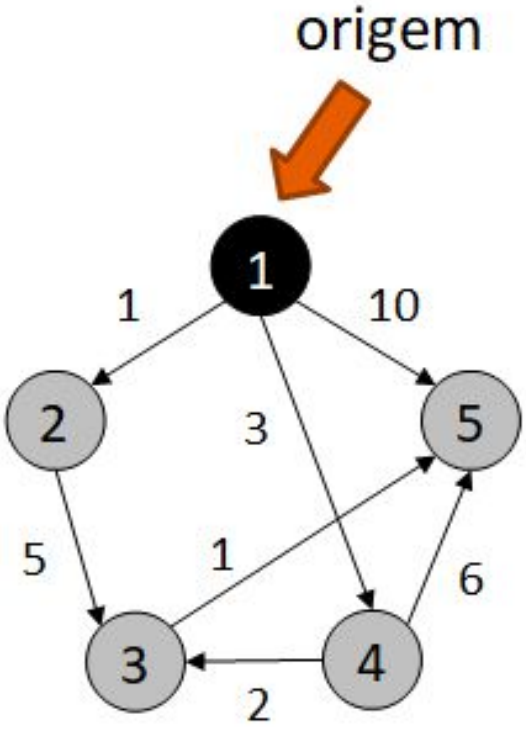
D =

1	2	3	4	5
∞	∞	∞	∞	∞

custo do menor caminho da origem até o vértice *i*

vértices

Algoritmo de Dijkstra

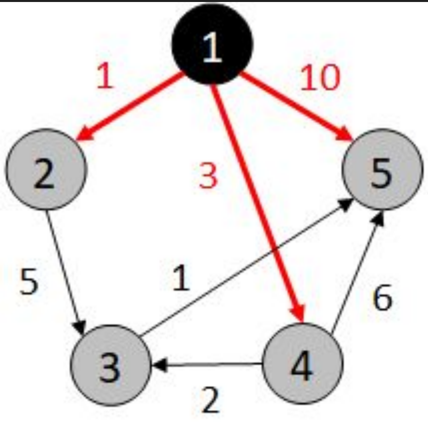


$S = \{1\}$

D =

0	∞	∞	∞	∞
1	2	3	4	5

Algoritmo de Dijkstra



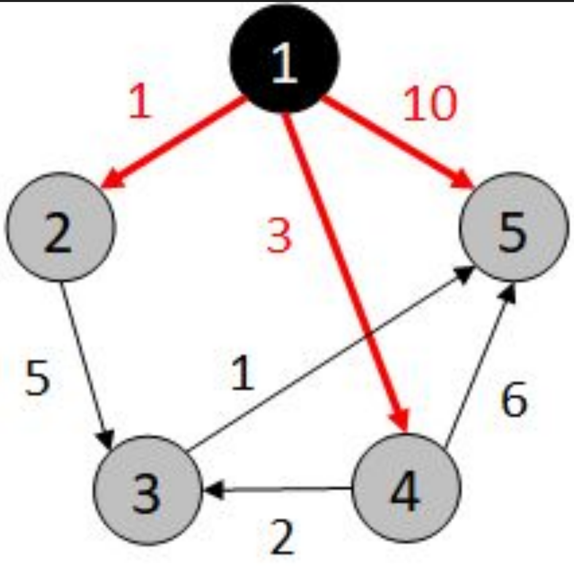
$S = \{1\}$

D =

1	2	3	4	5
0	∞	∞	∞	∞
	<?		<?	<?
	1		3	10

custo do caminho sendo explorado é menor que o armazenado em D?

Algoritmo de Dijkstra

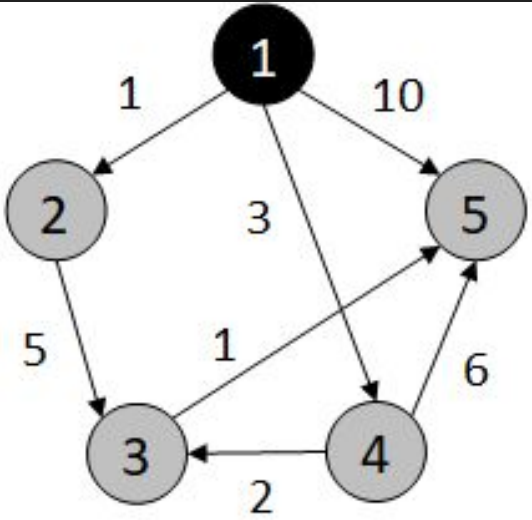


$S = \{1\}$

D =

0	1	∞	3	10
1	2	3	4	5

Algoritmo de Dijkstra



$S = \{1\}$

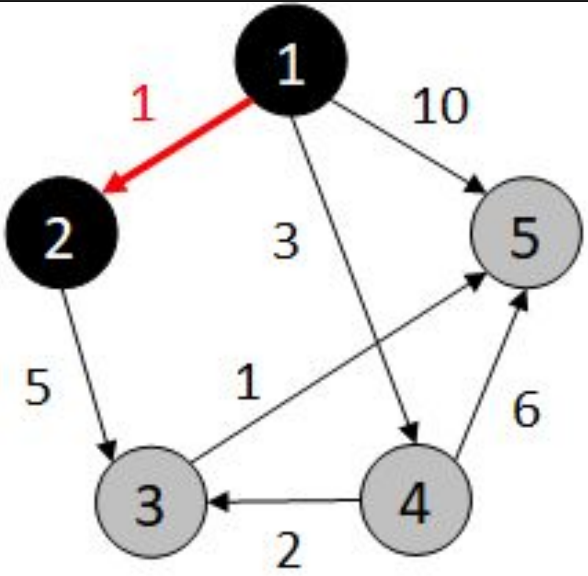
$D =$

0	1	∞	3	10
1	2	3	4	5

menor custo

$(w \in V - S)$

Algoritmo de Dijkstra

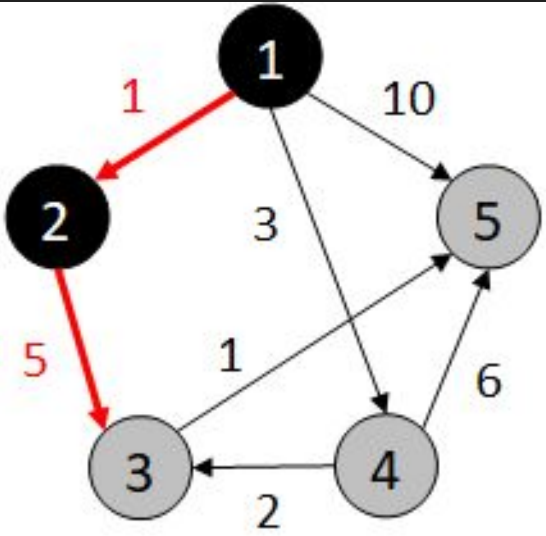


$S = \{1, \mathbf{2}\}$

$D =$

0	1	∞	3	10
1	2	3	4	5

Algoritmo de Dijkstra



$S = \{1, 2\}$

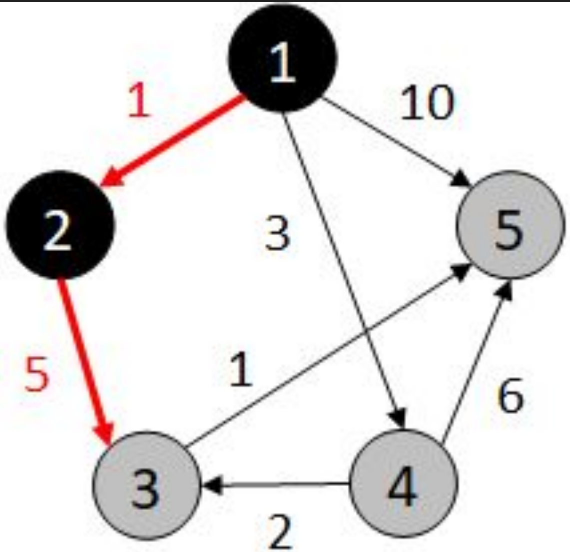
D =

0	1	∞	3	10
1	2	3	4	5

<?

6

Algoritmo de Dijkstra

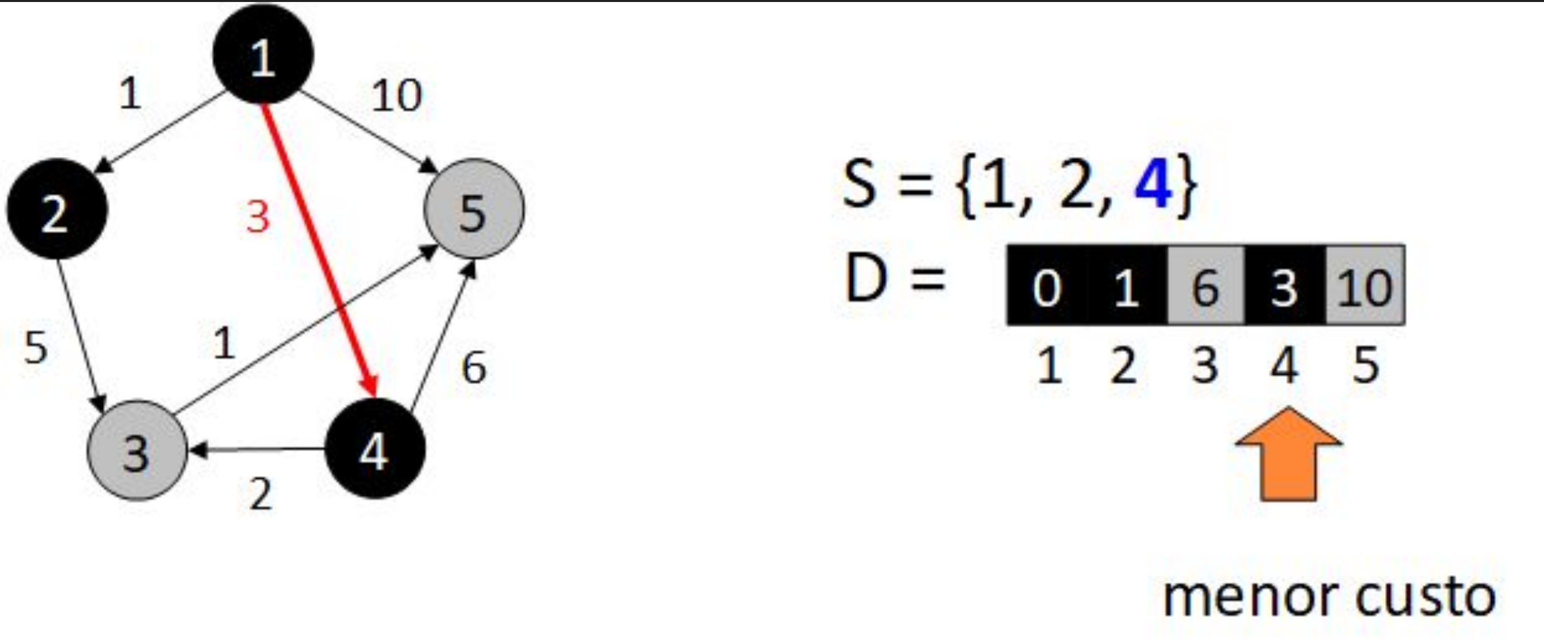


$S = \{1, 2\}$

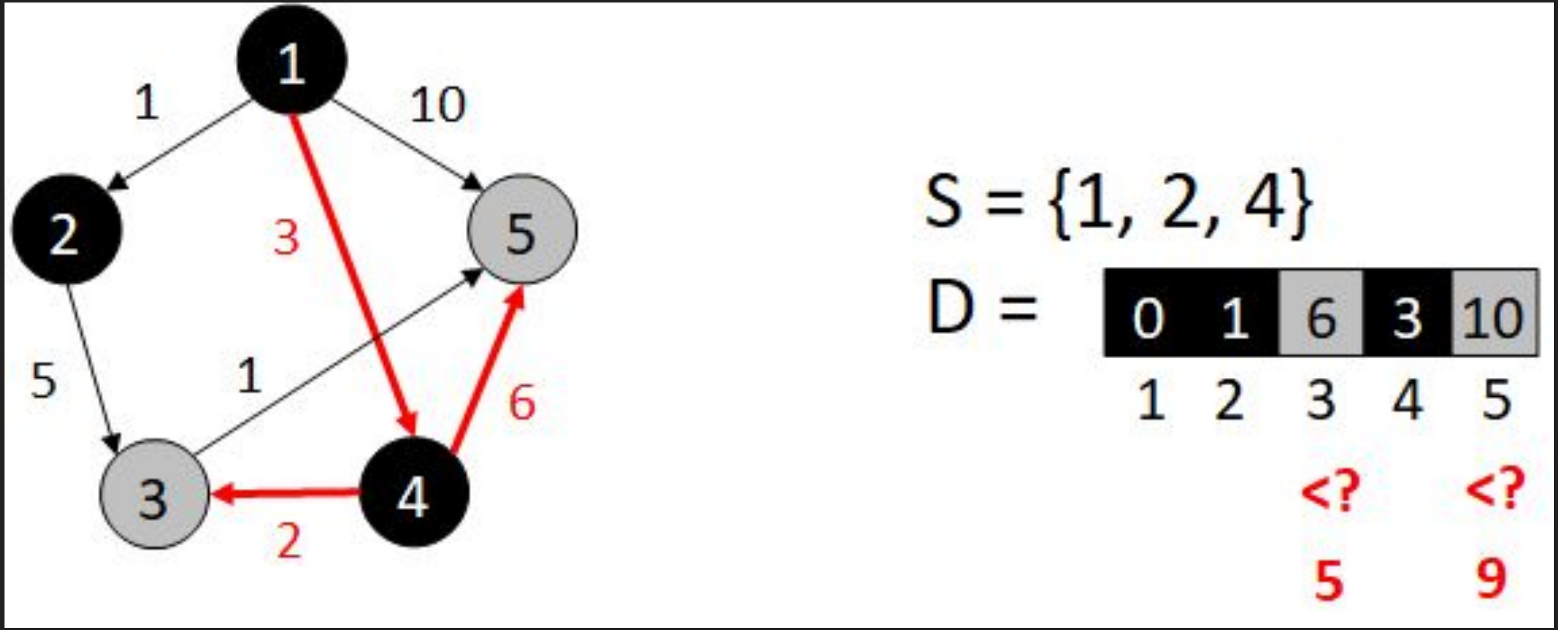
D =

1	2	3	4	5
0	1	6	3	10

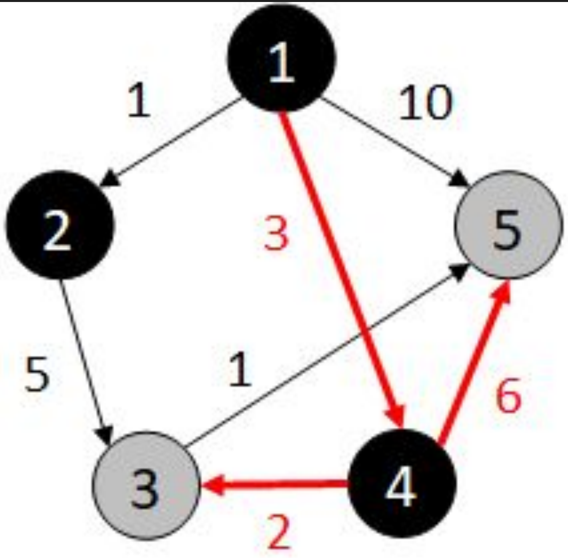
Algoritmo de Dijkstra



Algoritmo de Dijkstra



Algoritmo de Dijkstra

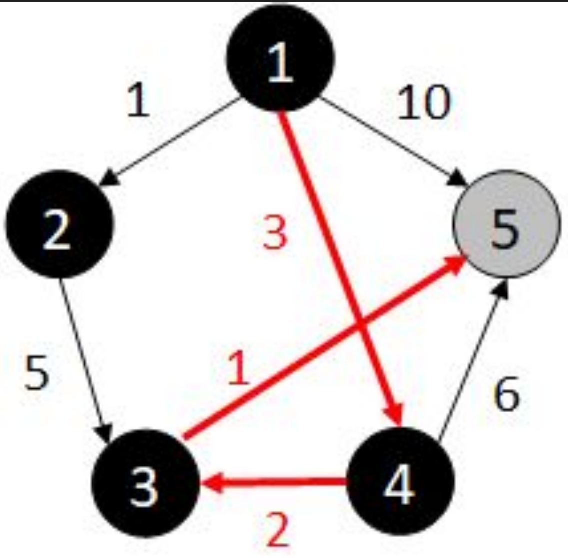


$S = \{1, 2, 4\}$

D =

0	1	5	3	9
1	2	3	4	5

Algoritmo de Dijkstra

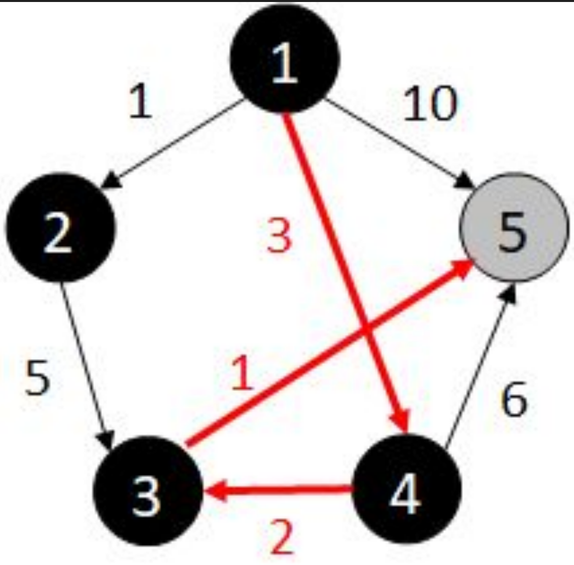


$S = \{1, 2, 4, 3\}$

D =					
	0	1	5	3	9
	1	2	3	4	5

menor custo

Algoritmo de Dijkstra

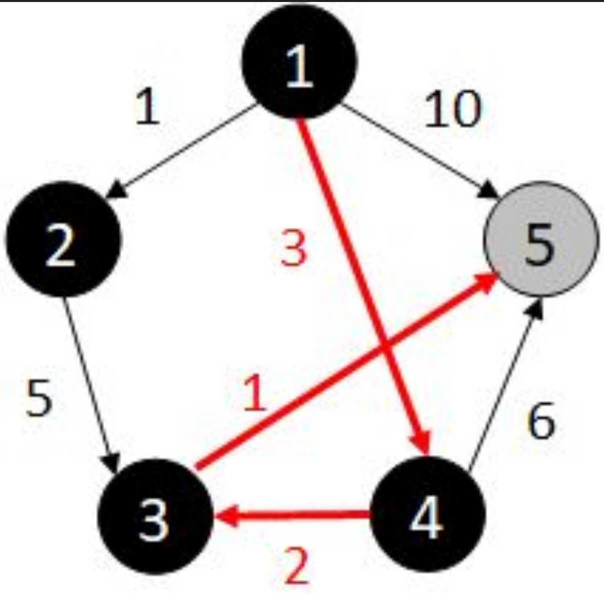


$S = \{1, 2, 4, \mathbf{3}\}$

D =

0	1	5	3	9
1	2	3	4	5
				<?
				6

Algoritmo de Dijkstra

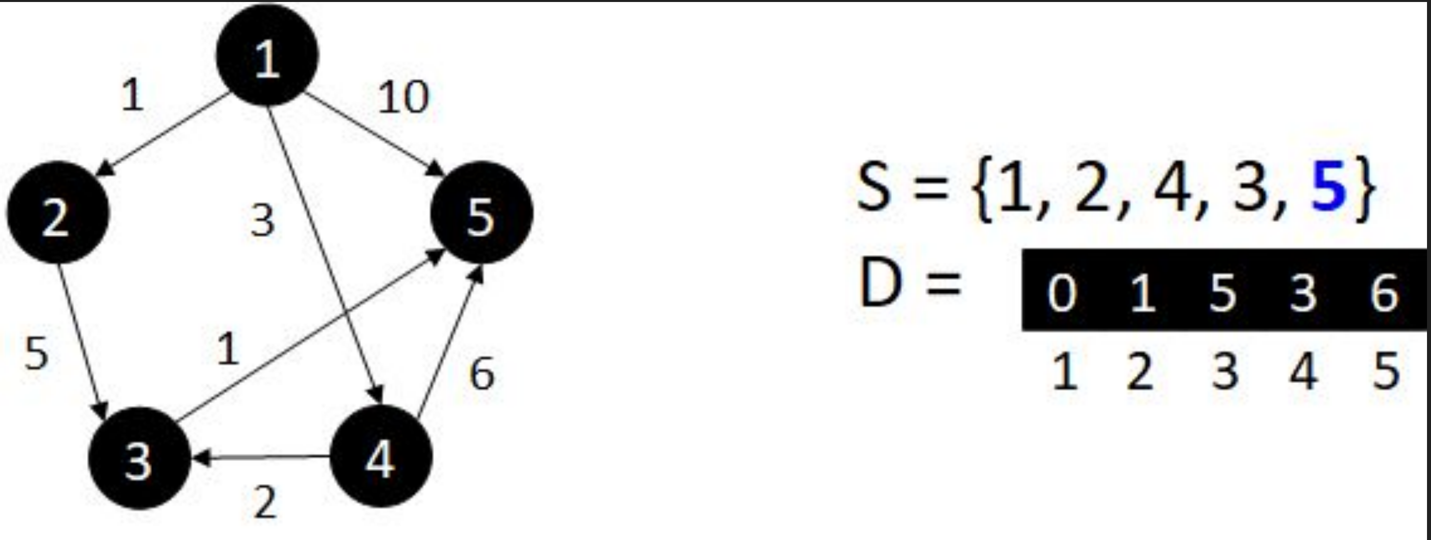


S = {1, 2, 4, 3}

D =

0	1	5	3	6
1	2	3	4	5

Algoritmo de Dijkstra



D armazena o custo do menor caminho entre a origem (1) e cada vértice do grafo.

E como saber qual é o menor caminho?

Algoritmo de Dijkstra - Menor Caminho

- Vetor de antecessores A pode ser usado para reconstruir o caminho mais curto entre o vértice origem e cada vértice
  - Técnica similar à utilizada na busca em largura
  - A possui os caminhos mínimos conhecidos até a iteração corrente

```
procedimento Dijkstra(origem: TVertice, var G: TGrafo)
variáveis
   D: vetor[TVertice] de TPeso;
   w: TVertice;
   S, V: conjunto de TVertice;
início
   S := {origem};
   V := {todos os vértices de G};
   D[origem] := 0;
   para i:=1 até G.NumVertices faça
   início
      se i != origem e existe a aresta (origem, i)
         então D[i] := Peso da aresta (origem, i)
      senão  D[i] := ∞;
   fim;
   enquanto S ≠ V faça
   início
      encontre um vértice w ∈ V - S tal que D[w] é mínimo;
      S := S ∪ {w};
      para todo v adjacente a w faça
         D[v] := min(D[v], D[w] + Peso da aresta (w,v));
   fim;
fim;
```

Algoritmo de Dijkstra - Menor Caminho

- Relembrando pontos importantes...
  - Vértices adicionados a **S** possuem o seu caminho mínimo definitivo
    - Não precisam mais serem revisitados
  - Não pode haver ciclos com arestas de custo negativo
    - Isso tornaria necessário revisitar os vértices em **S**

Algoritmo de Dijkstra - Menor Caminho

- Relembrando pontos importantes...
  - D possui os custos dos caminhos mínimos conhecidos até a iteração corrente
    - As atualizações de D a cada novo vértice inserido em **S** se certifica disso.
    - A possui os antecessores dos caminhos mínimos conhecidos até a iteração corrente

Complexidade

Algoritmo de Dijkstra - Complexidade

- Fila de prioridades para organizar os vértices de $V - S$
- Busca e remoção, na fila, do vértice w de menor custo
- Atualização da fila de prioridades a cada alteração em D
- Atualização/inserção/remoção em heap
  - $O(\log |V|)$
- Custo total de busca e remoção de w
  - $O(|V| \log |V|)$
- Custo total de atualizações na heap
  - $O(|A| \log |V|)$

Algoritmo de Dijkstra - Complexidade

- Custo final:
  - O (|A| log |V|)

Caminhos mais curtos de todos os pares

Caminhos mais curtos de todos os pares

- Exemplo: grafo direcionado ponderado representando as possíveis rotas de uma companhia aérea conectando diversas cidades.
- Objetivo: construir uma tabela com os custos dos menores caminhos entre todas as cidades.
- Problema: encontrar os caminhos mais curtos para todos os pares de vértices.

Caminhos mais curtos de todos os pares

- Uma possível solução
  - Algoritmo de Dijkstra executado para cada vértice como origem alternadamente.
- Solução alternativa (mais direta)
  - Algoritmo de Floyd-Warshall.
  - Utiliza uma matriz $A\,[|V| \times |V|]$ para calcular e armazenar os tamanhos dos caminhos mais curtos
  - Complexidade é $O(|V|^3)$

Referências

- WIRTH,N. Algorithms and Data Structures, Englewood Cliffs, Prentice-Hall, 1986.
- CORMEN, H.T.; LEISERSON, C.E.; RIVEST, R.L. Introduction to Algorithms, MIT Press, McGraw-Hill, 1999.
- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- SZWARCFITER,J.L. Grafos e Algoritmos Computacionais. Editora Campus, 1983.
- Van Steen, Maarten. "Graph theory and complex networks." An introduction 144 (2010).
- Gross, Jonathan L., and Jay Yellen. Graph theory and its applications. CRC press, 2005.
- Barabási, A.-L., Pósfai, M. (2016). Network science. Cambridge: Cambridge University Press. ISBN: 9781107076266 1107076269