Disciplina:

Análise de Algoritmos e Estrutura de Dados

Discente Especial:

Wagner Lopes Cardozo

Docente Orientadora da UC:

D.Sc. Lilian Berton

UNIFESP

Tema da Pesquisa:

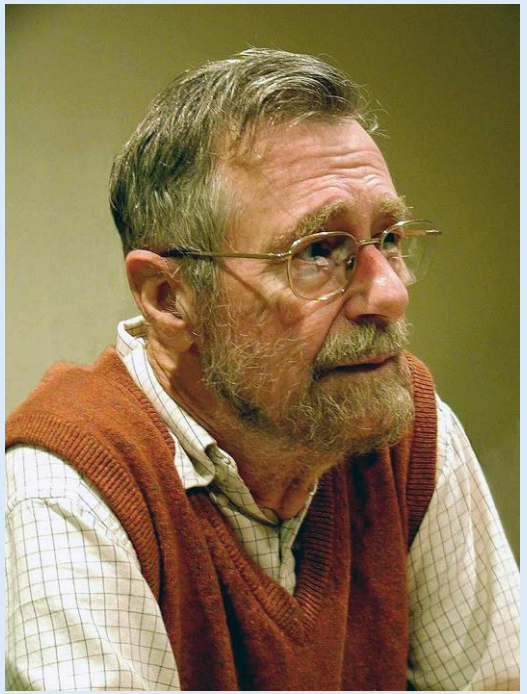
Algoritmo de Dijkstra e Bellman-Ford.

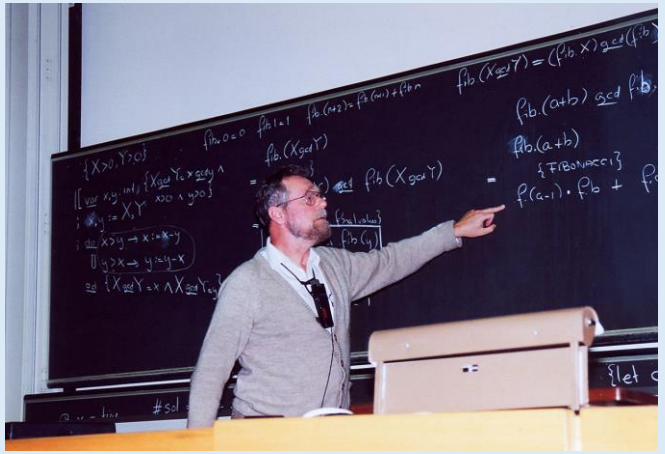
Sumário:

- ❑ História – Biografia
- ❑ Teoria dos Grafos
- ❑ Algoritmo de Dijkstra (Lógica de Funcionamento)
- ❑ Algoritmo de Bellman-Ford (Lógica de Funcionamento)
- ❑ Características
- ❑ Implementação Experimentais e Comparações
- ❑ Aplicações Práticas Para os Algoritmos de Dijkstra e Bellman-Ford
- ❑ Referências

História - Biografia:

- Edsger Wybe Dijkstra foi um importante pesquisador na área da Ciência da Computação.
- Dijkstra nasceu em 1930 em Roterdã, na Holanda.
- Apesar de inicialmente pensar em estudar Direito, ele acabou decidindo cursar Física Teórica.
- Apesar de seu gosto pela física, Dijkstra estudava matemática e computação em paralelo.





Fonte:
https://horizontes.sbc.org.br/index.php/2017/10/807/

História - Biografia:

- Dijkstra se casou com Maria Debets e nos documentos do casamento colocou “programador” como sua profissão.
- O que não foi aceito pelas autoridades, já que essa ainda não era uma profissão reconhecida na época.
- Alguns anos mais tarde, em 1962, Dijkstra... se mudou para Eindhoven, no sul da Holanda, para dar aulas no Departamento de Matemática da Universidade Tecnológica de Eindhoven.
- Criou um grupo de estudos voltado à sua área de interesse na Universidade Tecnológica de Eindhoven, que atualmente possui um dos melhores cursos de Computação do mundo.

Fonte: https://horizontes.sbc.org.br/index.php/2017/10/807/

UNIFESP

História – Biografia:

- Richard E. Bellman foi um matemático e cientista da computação americano.
- Ele nasceu em 1920 em Nova York e obteve seu doutorado em Matemática na Universidade de Wisconsin-Madison.
- Bellman é mais reconhecido por sua formulação do princípio da programação dinâmica, que revolucionou a abordagem para resolver problemas complexos de otimização.
- Ele também foi uma figura importante na história da pesquisa operacional e foi um defensor do uso de métodos computacionais para resolver problemas de otimização em diversas áreas.

Fonte: https://www.nae.edu/189177/RICHARD-E-BELLMAN-19201984

História – Biografia:

- Lester Randolph Ford Jr. foi um matemático e educador americano.
- Amplamente reconhecido por suas contribuições à teoria dos números e ao estudo das séries de funções.
- Nascido em 1900, Ford teve uma carreira acadêmica distinta e passou grande parte de sua vida ensinando matemática, principalmente na Universidade de Michigan.
- Ford é talvez mais conhecido por seu trabalho sobre frações contínuas e aproximações racionais para números irracionais.
- Lester Ford recebeu vários prêmios, incluindo o prestigioso Prêmio Leroy P. Steele, concedido pela American Mathematical Society (AMS), que destacou suas contribuições fundamentais à matemática.

Fonte: https://mathshistory.st-andrews.ac.uk/Biographies/Ford/poster/died/

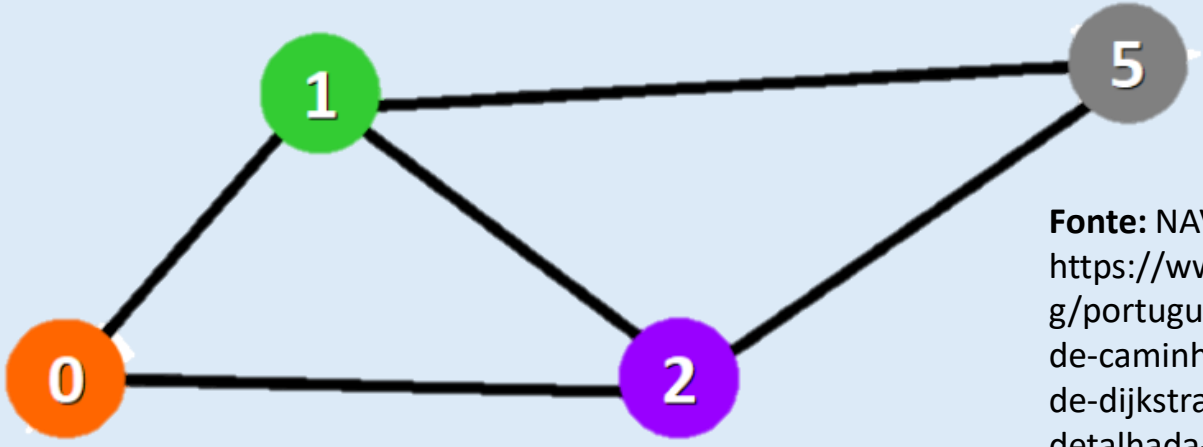
Para Refletir:

- O algoritmo de Dijkstra e o algoritmo de Bellman-Ford são ambos algoritmos clássicos para encontrar o caminho mais curto em um grafo.
- Mas diferem em sua abordagem, eficiência e nas características dos grafos que conseguem processar.

Fonte: https://www.murabei.com/conectando-o-mundo-com-grafos-desvendando-as-estruturas-da-conexao/

Teoria dos Grafos:

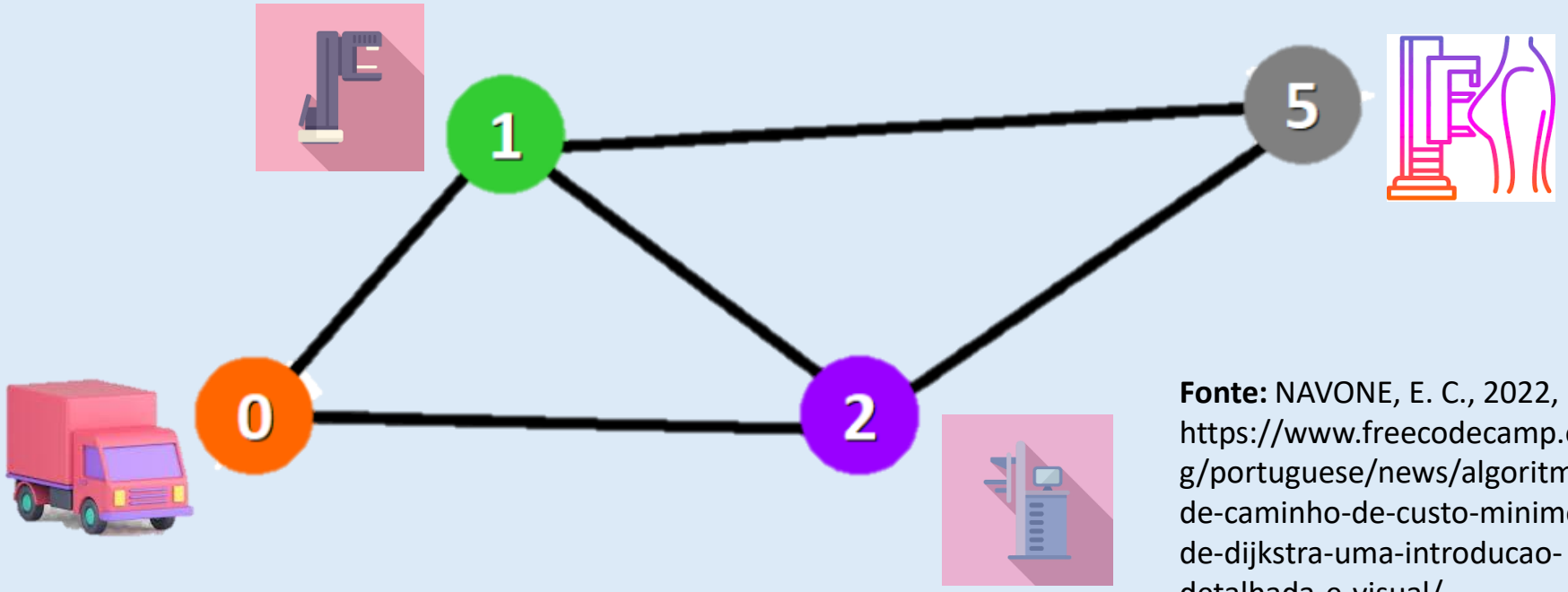
- Grafos são estruturas de dados usadas para representar "conexões" entre pares de elementos.
- Esses elementos são chamados de nós (ou vértices). Eles representam objetos, pessoas ou entidades da vida real.
- As conexões entre os nós são chamadas de arestas.
- Dois nós estarão conectados se houver uma aresta entre eles.



Fonte: NAVONE, E. C., 2022, https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/

Teoria dos Grafos:

- Os grafos são diretamente aplicáveis a cenários do mundo real. Por exemplo, poderíamos usá-los para modelar uma rede de caminhões (carretas) com toda estrutura para fazer exames de mamografia, atendendo várias regiões geográficas, onde os nós representariam municípios de um estado e arestas representariam estradas, avenidas e ruas que são os caminhos que conectam os município.

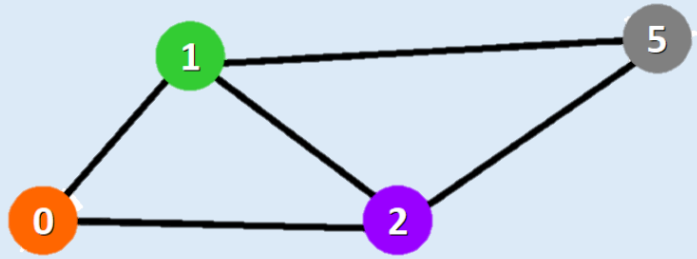


Fonte: NAVONE, E. C., 2022, https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/

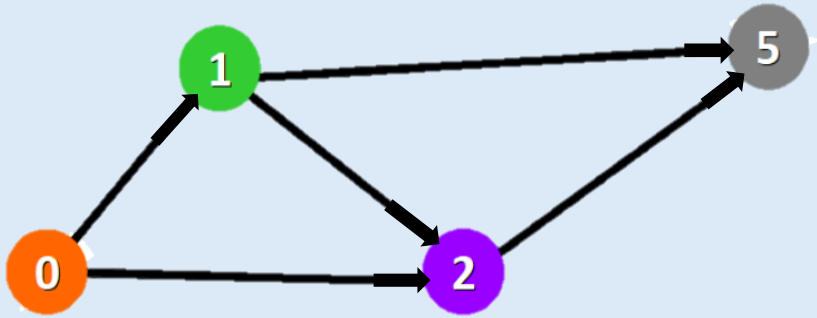
Teoria dos Grafos:

Os grafos podem ser:

- Não dirigidos: se para cada par de nós conectados, você pode ir de um nó para o outro em ambas as direções.



- Dirigidos: se para cada par de nós conectados, você só pode ir de um nó para outro em uma direção específica. Usamos setas em vez de linhas simples para representar arestas dirigidas.



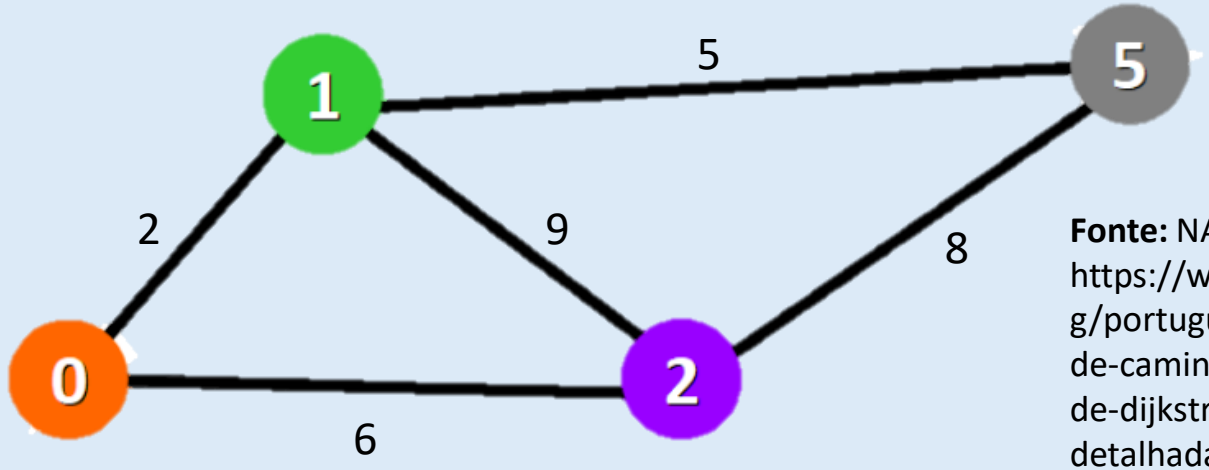
Fonte: NAVONE, E. C., 2022, https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/

Teoria dos Grafos:

Grafos ponderados:

- Um grafo ponderado é um grafo cujas arestas têm um "peso" ou "custo". O peso de uma aresta pode representar distância, tempo ou qualquer coisa que modele a "conexão" entre o par de nós que ela conecta.

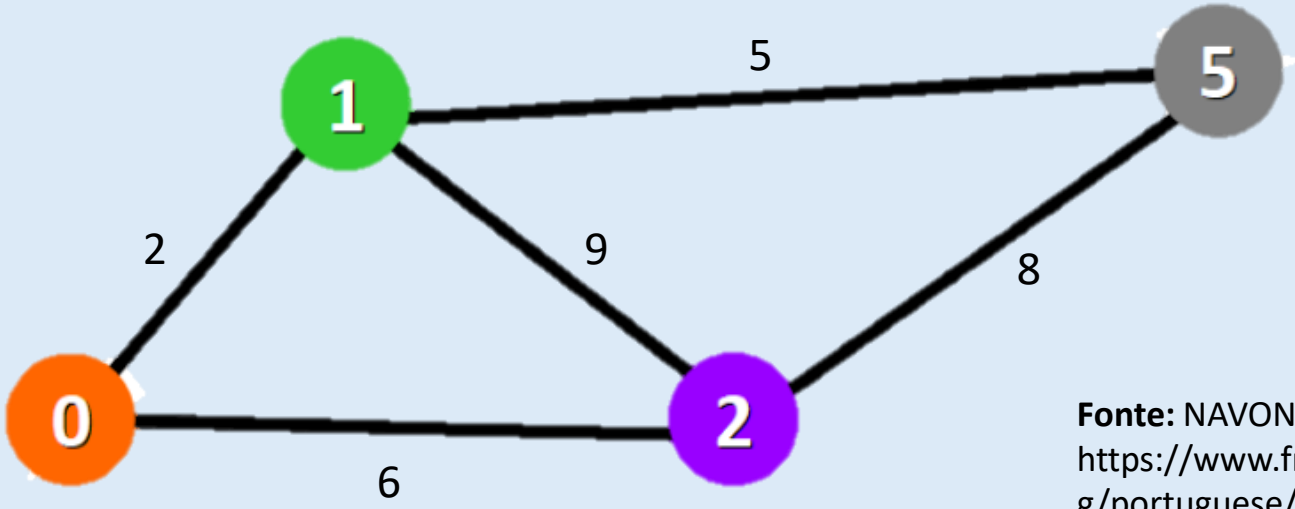
- Por exemplo, no grafo ponderado abaixo, você pode ver um número próximo de cada aresta. Este número é usado para representar o peso da aresta correspondente.



Fonte: NAVONE, E. C., 2022, https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/

Algoritmo de Dijkstra:

- Com o algoritmo de Dijkstra, você poderá encontrar o caminho de menor custo entre nós em um grafo. Particularmente, você poderá encontrar o caminho de menor custo entre um nó (denominado “origem”) e todos os outros nós do grafo, produzindo uma árvore de custo-mínimo.



Fonte: NAVONE, E. C., 2022, https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/

UNIFESP

Algoritmo de Dijkstra:

- Noções básicas do algoritmo de Dijkstra

  - ✓ O Algoritmo de Dijkstra basicamente começa no nó que você escolhe (o nó de origem) e analisa o grafo para encontrar o caminho de menor custo entre esse nó e todos os outros nós do grafo.

  - ✓ O algoritmo mantém o registro da distância mais curta atualmente conhecida de cada nó, até o nó de origem, e atualiza esses valores se encontrar um caminho de menor custo.

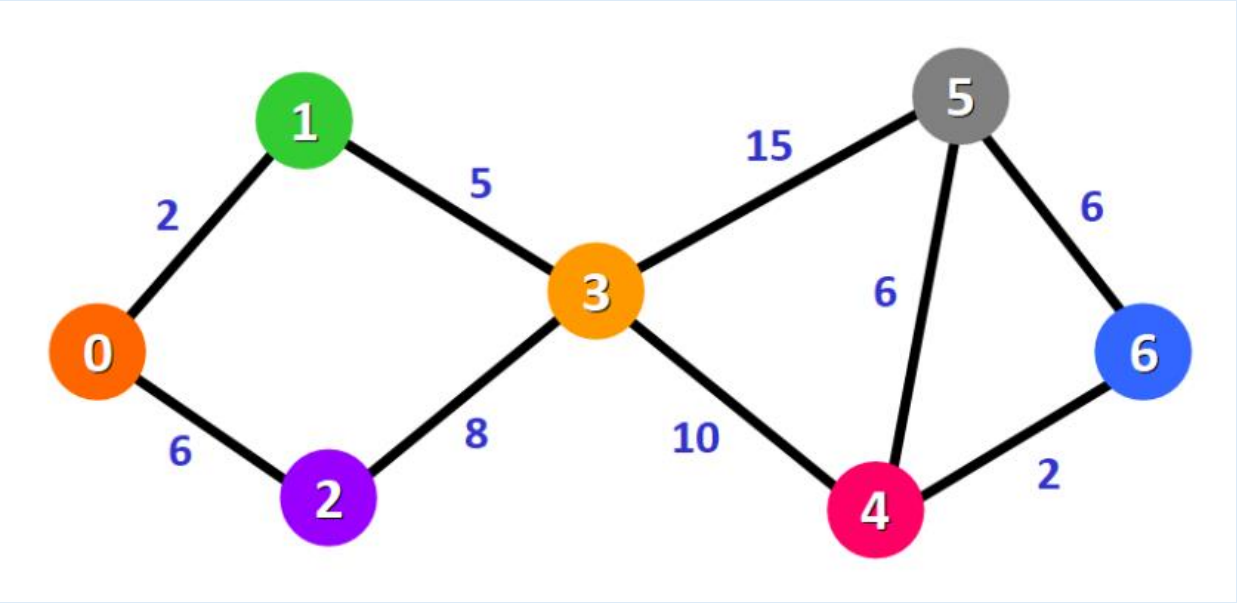
  - ✓ Uma vez que o algoritmo encontrou o caminho de menor custo entre o nó de origem e outro nó, esse nó é marcado como "visitado" e adicionado ao caminho.

  - ✓ O processo continua até que todos os nós do grafo tenham sido adicionados ao caminho. Desta forma, temos um caminho que conecta o nó de origem a todos os outros nós seguindo o caminho de menor custo possível para chegar a cada nó.

  - ✓ O Algoritmo de Dijkstra só funciona com grafos com pesos positivos.

Algoritmo de Dijkstra:

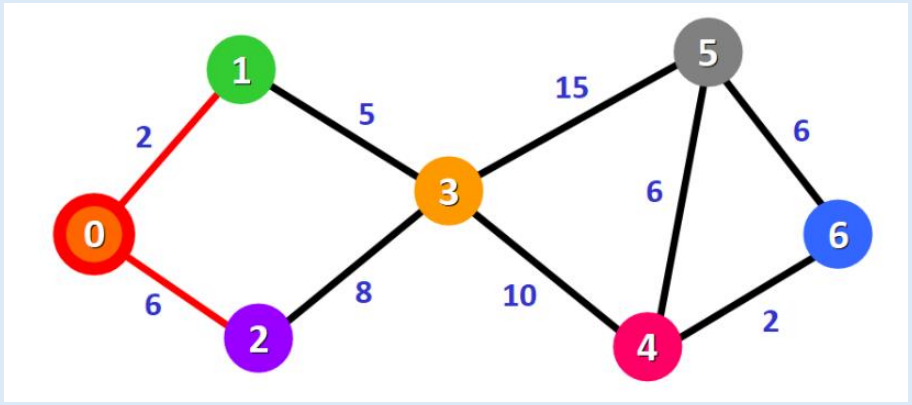
- Dinâmica do Processamento do Algoritmo:

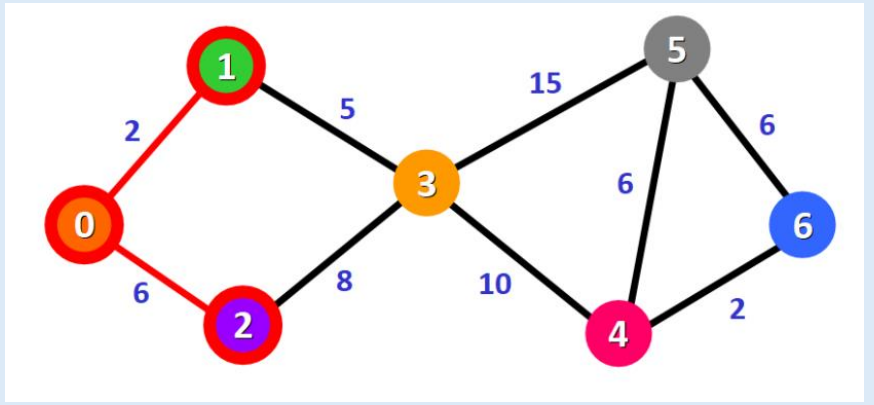


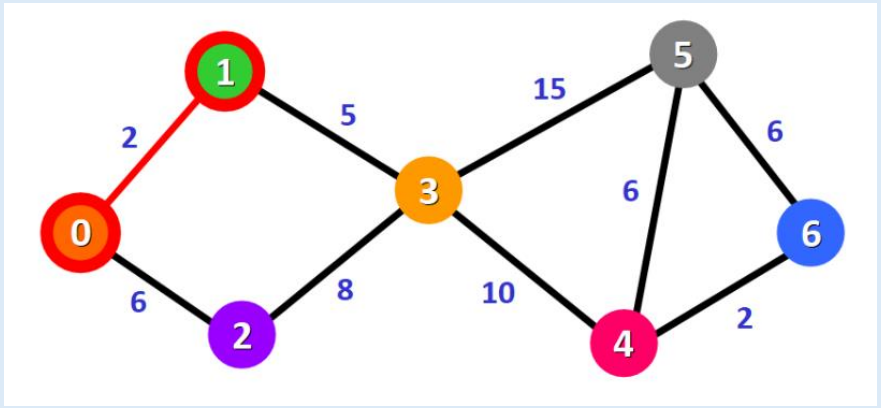
Fonte: NAVONE, E. C., 2022, https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/

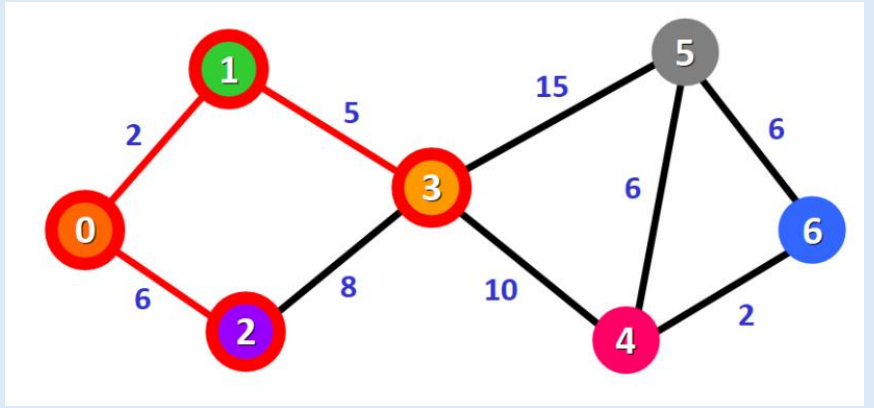
UNIFESP

Algoritmo de Dijkstra:



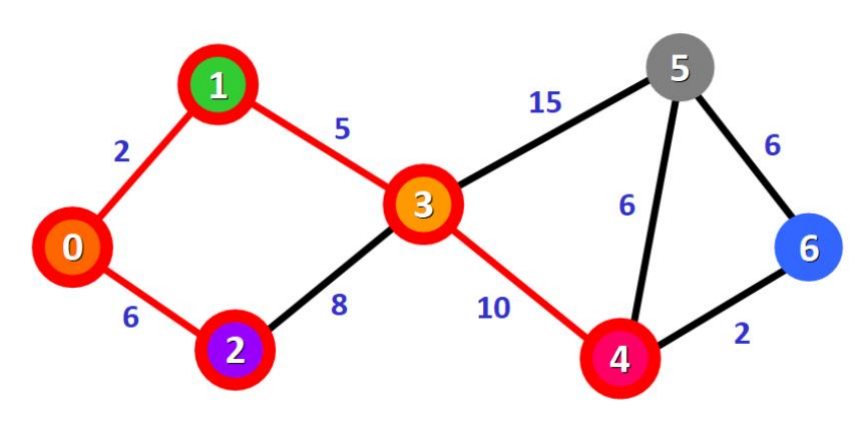


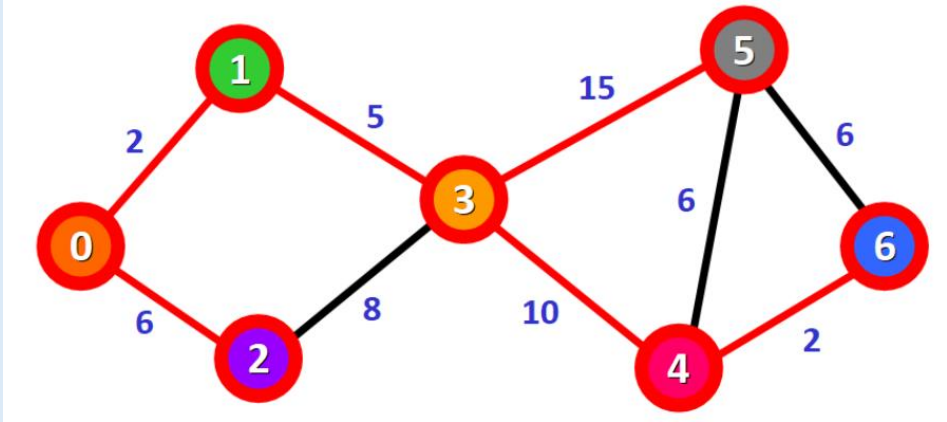


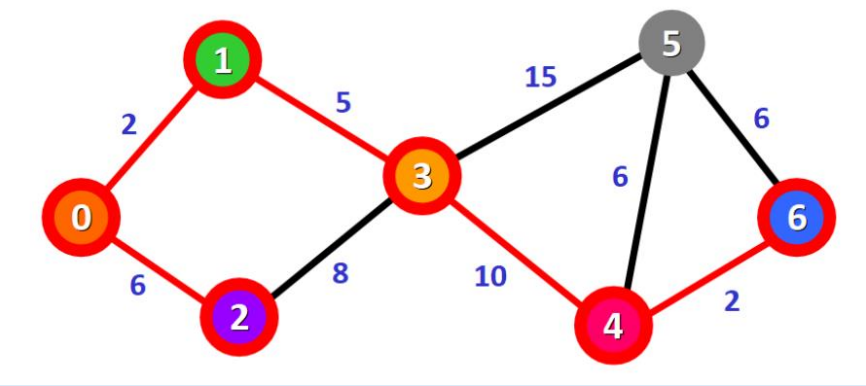


UNIFESP

Algoritmo de Dijkstra:



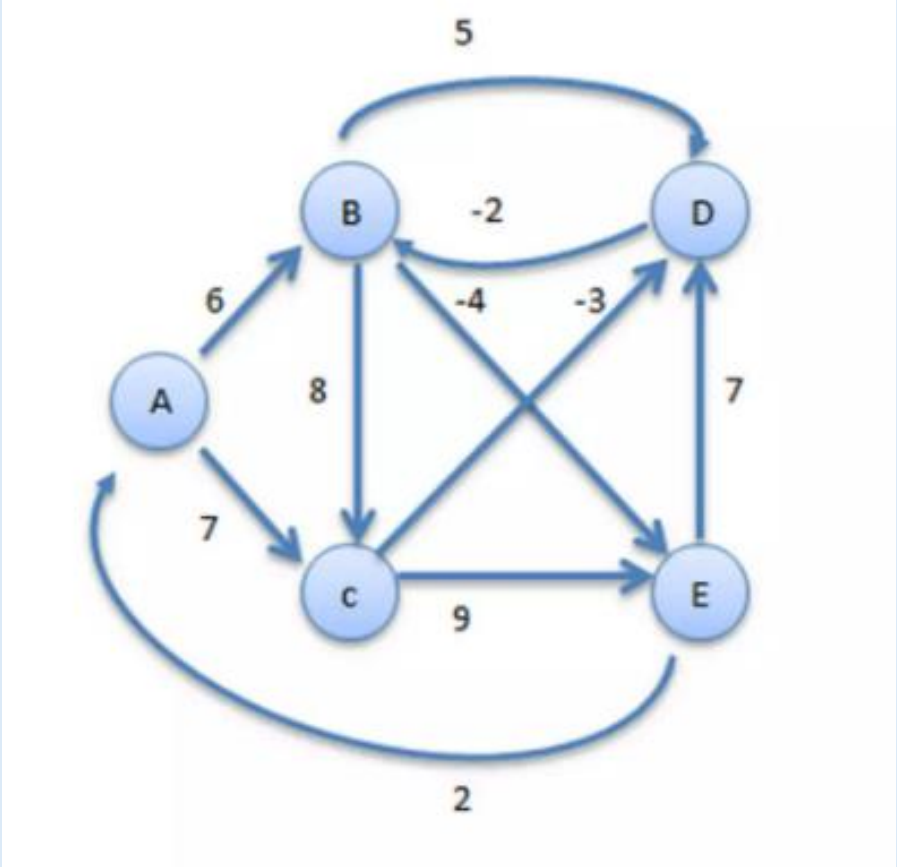




- ✓ **Menor caminho nó:** 0, 1, 3, 4, 5 e 6
- ✓ **Menor caminho arestas:** 2+7+10+2=19

Algoritmo de Bellman-Ford:

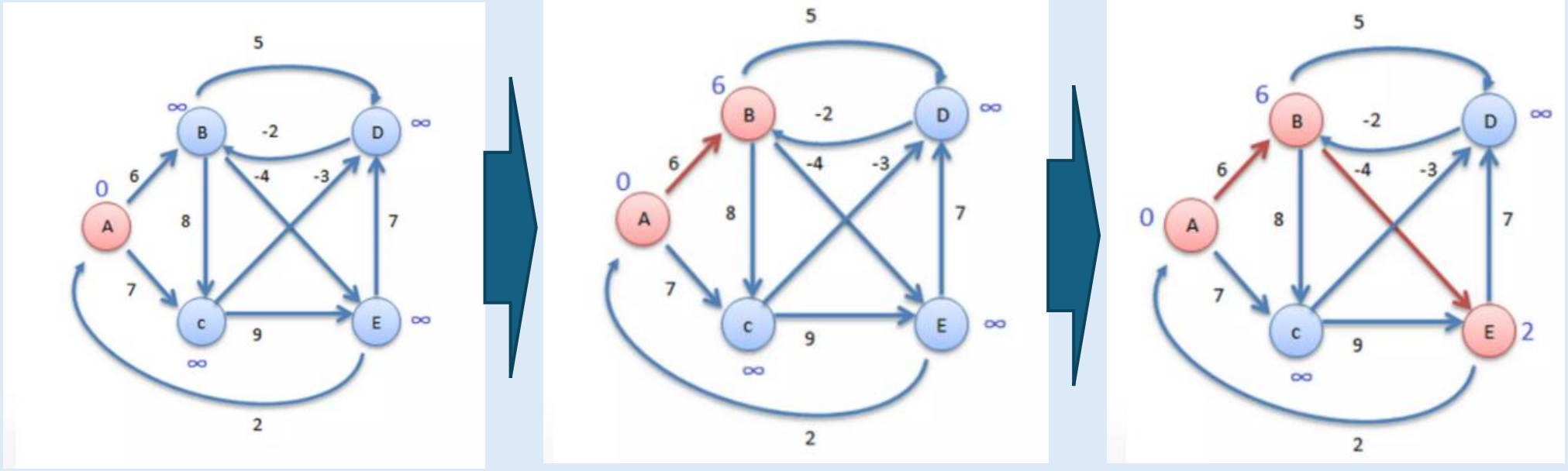
- **Relaxamento:** consiste em verificar se pode ser encontrado um caminho mais curto do que aquele encontrado até o momento, passando pelos vértices, ou seja, o algoritmo verifica se o custo do caminho calculado anteriormente é menor do que o caminho calculado atualmente, mesmo que o caminho atual tenha que percorrer um número maior de arestas.



Fonte: RAMALHO, G., https://pt.slideshare.net/slideshow/anlise-de-algoritmos-problemas-em-grafos-caminho-mnimo-algoritmo-de-bellmanford/34528524#17

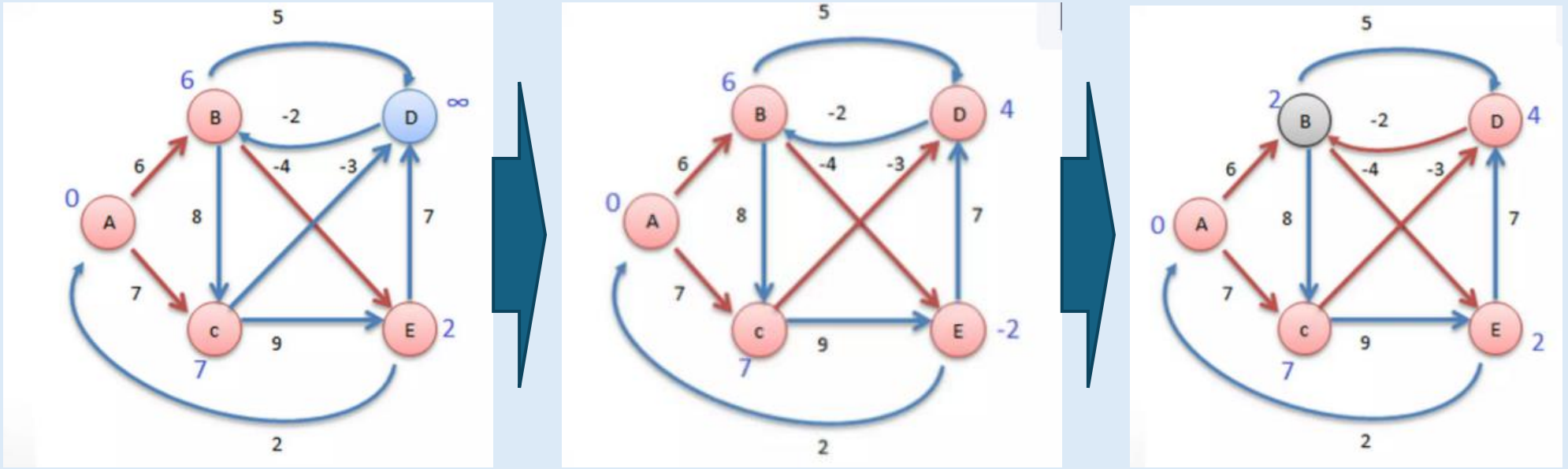
UNIFESP

Algoritmo de Bellman-Ford:



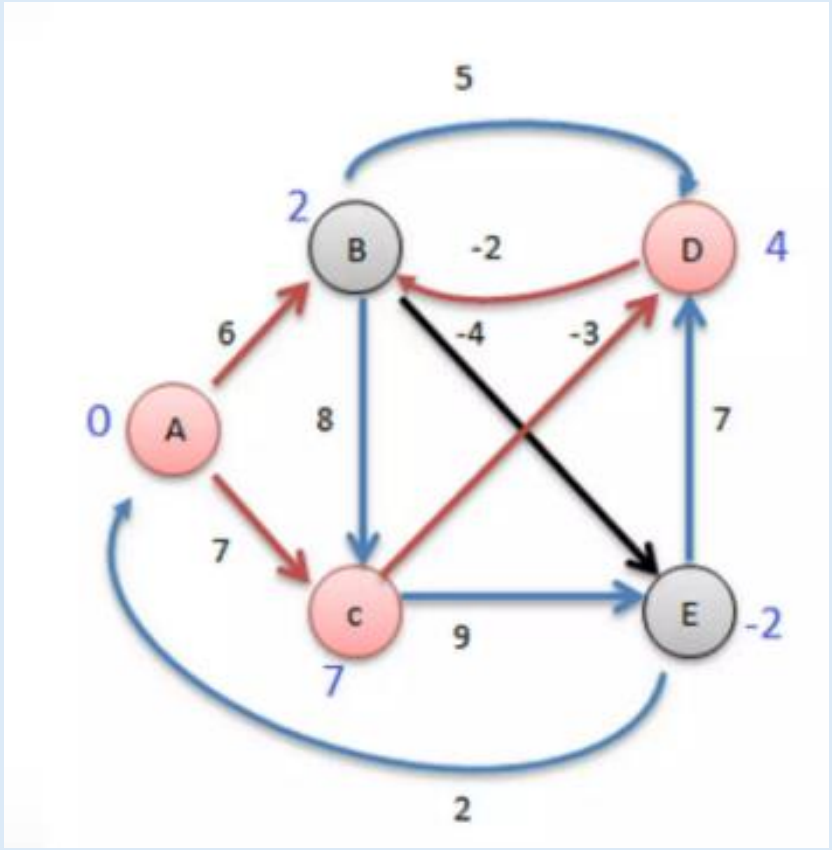
UNIFESP

Algoritmo de Bellman-Ford:



Algoritmo de Bellman-Ford:

Vértice	Distância de A
A	0
B	2
C	7
D	4
E	-2



Características:

Objetivo e Aplicações:

- Ambos os algoritmos buscam o caminho mais curto de um único vértice de origem para todos os outros vértices de um grafo. São muito usados em redes de transporte, redes de comunicação e sistemas de navegação.

- **Dijkstra:** Funciona melhor em grafos onde todas as arestas têm pesos não negativos.

- **Bellman-Ford:** É mais flexível e pode lidar com grafos que contenham arestas de peso negativo, algo que o algoritmo de Dijkstra não consegue processar corretamente.

UNIFESP

Características:

Complexidade Assintótica Big (O):

- Dijkstra: Sua complexidade é O(E + V logV) (usando uma fila de prioridade), onde V é o número de vértices e E é o número de arestas. Isso o torna mais eficiente para grafos densos, especialmente se implementado com estruturas de dados otimizadas.

- Bellman-Ford: Tem uma complexidade de O(V x E), tornando-o mais lento em grafos grandes. Ele é mais adequado para grafos esparsos, onde E é relativamente pequeno em relação a V.

Características:

Método de Funcionamento:

- **Dijkstra:** Utiliza uma abordagem gulosa, expandindo sempre o vértice com a menor distância acumulada em relação ao vértice de origem. Ele atualiza as distâncias dos vértices adjacentes de um vértice processado, mas nunca revisita ou relaxa as arestas de um vértice já processado.

- **Bellman-Ford:** Baseia-se em uma abordagem iterativa (relaxamento). Ele repete a operação de relaxamento de todas as arestas do grafo $V-1$ vezes, onde V é o número de vértices. Isso garante que, ao final, as distâncias mínimas de todos os vértices são encontradas, mesmo em presença de arestas negativas.

Características:

Tratamento de Ciclos de Peso Negativo:

- **Dijkstra:** Não é capaz de detectar ciclos de peso negativo e pode falhar ao encontrar o caminho correto se houverem arestas com pesos negativos, uma vez que esses valores podem fazer com que ele produza distâncias incorretas.

- **Bellman-Ford:** Pode detectar ciclos de peso negativo, o que é uma vantagem significativa. Após V−1 iterações de relaxamento, uma passagem adicional pelas arestas revela se há um ciclo negativo (se ainda houver relaxamento possível).

Características:

Escolha do Algoritmo:

- **Dijkstra:** É geralmente a escolha preferida quando o grafo não possui pesos negativos, especialmente quando se busca alta eficiência em grafos densos. Sua rapidez e implementação relativamente simples o tornam uma solução comum em sistemas de roteamento e redes de comunicação.

- **Bellman-Ford:** É mais indicado para cenários onde os pesos negativos são uma possibilidade ou quando há a necessidade de detectar ciclos negativos. É especialmente útil em grafos direcionados com pesos variados, como em certas simulações econômicas ou em modelos de fluxo de rede.

UNIFESP

Características:

Exemplo Comparativo:

- Suponha um grafo em que há caminhos alternativos que contêm pesos negativos, como em alguns modelos de custo onde descontos ou penalidades podem ser aplicados.
- O Bellman-Ford garante o cálculo do caminho mais curto correto, mesmo levando em conta os pesos negativos.
- Nesse caso, o uso do Dijkstra levaria a resultados incorretos.
- Por outro lado, em um grafo de rede de tráfego com pesos positivos (representando distâncias ou tempos de viagem), o algoritmo de Dijkstra oferece uma solução rápida e precisa, porém se do destino não voltar para origem e for necessário ir para outro destino pode ser que o algoritmo de Bellman-Ford seja mais indicado.

Implementação Experimental e Comparações:

- Implementando Manualmente:

Algoritmo	Tempo (s)	Memória (bytes)
Dijkstra	0.00711131	28320
Bellman-Ford	2.57838	14779

Algoritmo	Tempo (s)	Memória (bytes)
Prim	5.67436e-05	512
Kruskal	4.88758e-05	2984
Dijkstra	1.66893e-05	640
Bellman-Ford	1.50204e-05	512
Ford-Fulkerson	2.36034e-05	1096
Kosaraju	7.84397e-05	3024

UNIFESP

Implementação Experimental e Comparações:

- Implementando com as bibliotecas NetworkX e Scipy:

Algoritmo	Tempo de Execução (s)	Memória Usada (MB)
NetworkX Dijkstra - shortest_path	0.000336	0.005150 MB
NetworkX Dijkstra - single_source_dijkstra	0.000158	0.003784 MB
NetworkX Bellman-Ford - bellman_ford_path	0.000242	0.005295 MB
NetworkX Bellman-Ford - single_source_bellman_ford_path	0.000369	0.006027 MB
Scipy Dijkstra	0.00206	0.007298 MB

- Implementando Manualmente:

Algoritmo	Tempo (s)	Memória (bytes)
Dijkstra	0.00711131	28320
Bellman-Ford	2.57838	14779

Implementação Experimental e Comparações:

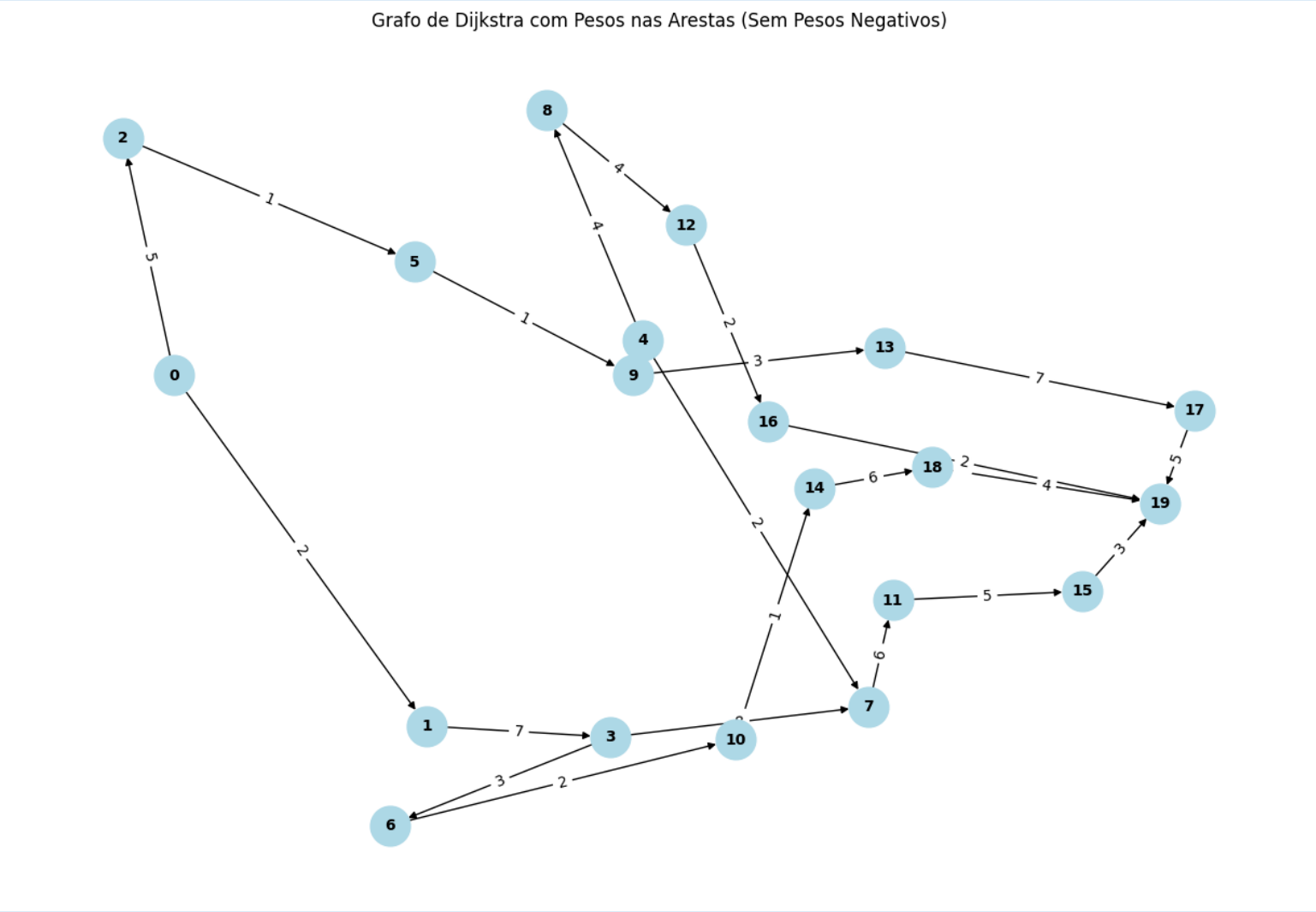
```
Algoritmo de Dijkstra:
Tempo de execução: 0.105971 segundos
Memória usada: 0.00 MB

Distâncias mínimas de cada nó a partir do nó inicial:
Nó 0: 0
Nó 1: 2
Nó 2: 5
Nó 3: 9
Nó 5: 6
Nó 6: 12
Nó 7: 17
Nó 4: inf
Nó 8: inf
Nó 9: 7
Nó 10: 14
Nó 11: 23
Nó 12: inf
Nó 13: 10
Nó 14: 15
Nó 15: 28
Nó 16: inf
Nó 17: 17
Nó 18: 21
Nó 19: 22

Caminho de menor custo:
0 -> 2 -> 5 -> 9 -> 13 -> 17 -> 19
```

UNIFESP

Implementação Experimental e Comparações:



UNIFESP

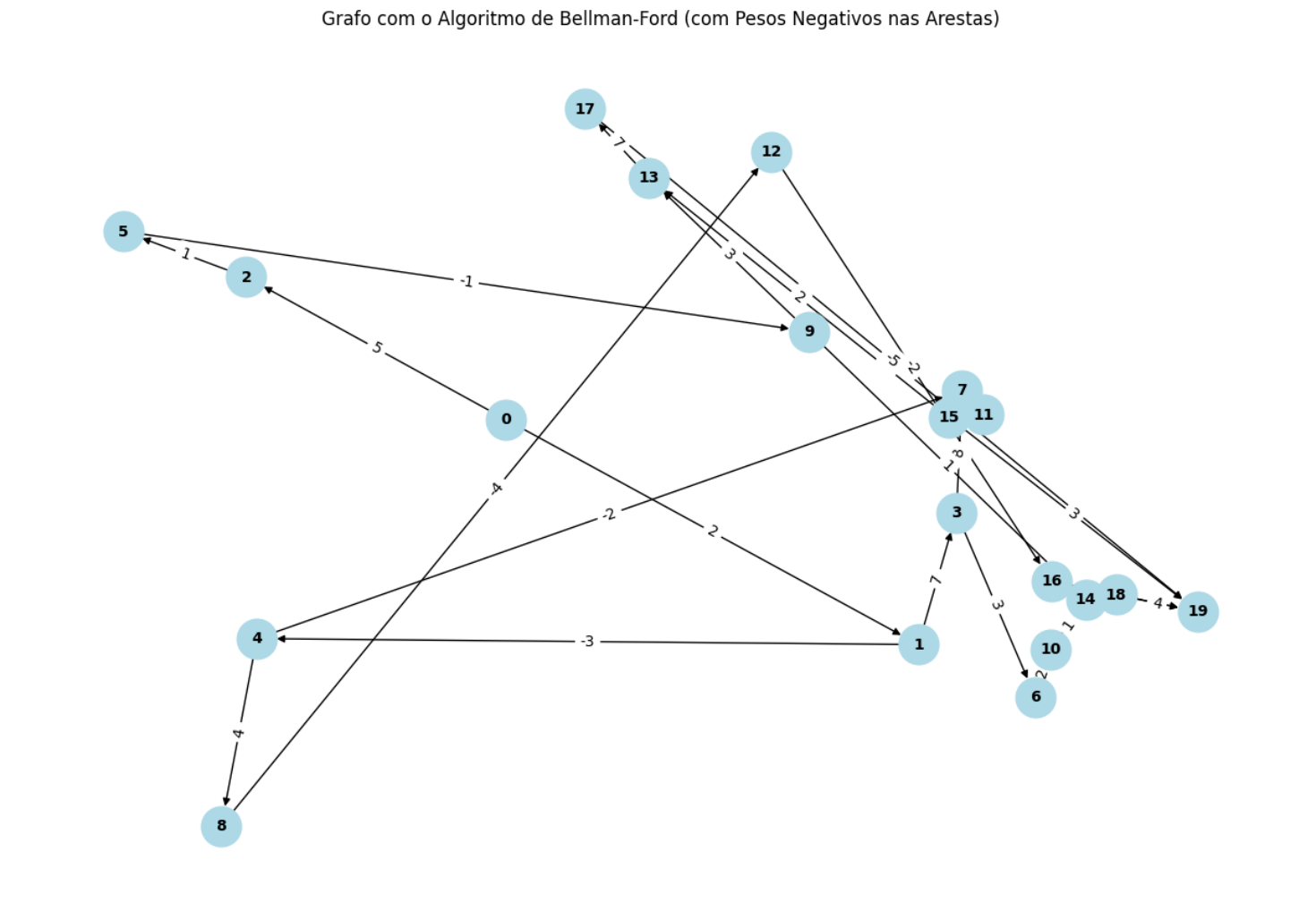
Implementação Experimental e Comparações:

```
Algoritmo de Bellman-Ford:
Tempo de execução: 0.003042 segundos
Memória usada: 0.00 MB

Distâncias mínimas de cada nó a partir do nó inicial:
Nó 0: 0
Nó 1: 2
Nó 2: 5
Nó 3: 9
Nó 4: -1
Nó 5: 6
Nó 6: 12
Nó 7: -3
Nó 8: 3
Nó 9: 5
Nó 10: 14
Nó 11: 3
Nó 12: -1
Nó 13: 8
Nó 14: 6
Nó 15: 8
Nó 16: -3
Nó 17: 15
Nó 18: 12
Nó 19: -1

Caminho de menor custo:
0 -> 1 -> 4 -> 8 -> 12 -> 16 -> 19
```

Implementação Experimental e Comparações:



UNIFESP

Aplicações práticas para o algoritmo de Dijkstra:

- **Roteamento em Redes de Computadores:** é amplamente utilizado para encontrar as rotas mais curtas em redes de computadores.

- **GPS e Navegação:** aplicativos de navegação, como Google Maps e Waze, utilizam variações do algoritmo de Dijkstra para calcular as rotas mais rápidas entre dois pontos, considerando a distância e, às vezes, o tempo estimado.

- **Planejamento de Transporte Público:** é útil para encontrar rotas ideais entre diferentes estações de transporte público, como metrô, ônibus e trens.

- **Jogos e IA em Jogos:** Muitos jogos que envolvem mapas, como jogos de estratégia em tempo real e RPGs, usam o algoritmo de Dijkstra para calcular os caminhos mais curtos para os personagens do jogo, facilitando a movimentação pelo ambiente do jogo.

UNIFESP

Aplicações práticas para o algoritmo de Bellman-Ford:

- **Análise de Risco em Finanças:** em mercados financeiros, o Bellman-Ford pode ser usado para detectar oportunidades de arbitragem em mercados com valores de câmbio ou taxas de juros variáveis, onde os pesos podem ser negativos, representando ganhos potenciais.

- **Sistemas de Recomendação e Comparação de Preços:** o algoritmo é aplicado em plataformas que analisam produtos com diferentes preços e descontos. Como lida com pesos negativos, ele pode detectar a sequência de itens ou serviços que maximiza o valor total considerando taxas ou descontos.

- **Planejamento de Rotas em Redes de Transporte com Custos Variáveis:** Em redes de transporte, o Bellman-Ford permite encontrar caminhos ótimos mesmo com custos variáveis.

UNIFESP

Referências:

- CORMEM, T. H., LEISERSON C. E., RIVEST R. L., STEIN C., **Algoritmos Teoria e Prática** – Gen LTC, 3ª Edição, 2012, ISBN-13: 978-8535236996
- ZIVIANI, N., **Projetos de Algoritmos com Implementação e Java e C++,** Cengage Learning, 2006
- FEOFILOFF, P. **Minicurso de Análise de Algoritmos**, 2010, http://www.ime.usp.br/~pf/livrinho-AA/
- GOODRICH M. T., TAMASSIA R., GOLDWASSER M. H., **Data Structures and Algorithms in Python**, 2013
- DIJKSTRA, E. W., A Note on Two Problems in Connexion with Graphs, Numerische Mathematik 1, 269-271 (1959), https://ir.cwi.nl/pub/9256/9256D.pdf
- OLIVEIRA, V. A., RANGEL, S., ARAUJO, S. A., Teoria dos Grafos, Departamento de Matemática Aplicada, Capitulo 10 (Notas de Aulas), UNESP 2013, https://www.ibilce.unesp.br/Home/Departamentos/MatematicaAplicada/docentes/socorro/caminhominimo.pdf

UNIFESP

Referências:

- INTERIAN, R., Projeto e Análise de Algoritmos II (MC558)Caminhos Mínimos (Notas de Aula), Unicamp,https://ic.unicamp.br/~ruben/2024/S1MC558/files/Slides_MC558_Aula_12_Grafos_12.pdf
- https://dl.acm.org/doi/pdf/10.1145/1787234.1787249
- https://akiradev.netlify.app/posts/algoritmo-dijkstra/
- https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/
- https://horizontes.sbc.org.br/index.php/2017/10/807/
- https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dijkstra.html
- http://www.dsc.ufcg.edu.br/~pet/jornal/fevereiro2012/materias/historia_da_computacao.html
- https://www.cs.utexas.edu/~EWD/welcome.html

UNIFESP

Referências:

- https://en.wikipedia.org/wiki/Edsger_W._Dijkstra
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- https://pt.slideshare.net/slideshow/anlise-de-algoritmos-problemas-em-grafos-caminho-mnimo-algoritmo-de-bellmanford/34528524
- https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/bellman-ford.html
- https://pt.wikipedia.org/wiki/Algoritmo_de_Bellman-Ford
- https://en.wikipedia.org/wiki/Lester_R._Ford
- https://en.wikipedia.org/wiki/Richard_E._Bellman
- https://www.bcc.unifal-mg.edu.br/~humberto/disciplinas/2010_2_grafos/pdf_aulas/aula_12.pdf
- https://pt.slideshare.net/slideshow/anlise-de-algoritmos-problemas-em-grafos-caminho-mnimo-algoritmo-de-bellmanford/34528524

UNIFESP

“Computer Science is no more about computers than astronomy is about telescopes.”

— Edsger Wybe Dijkstra

“A Ciência da Computação não tem mais a ver com computadores do que a astronomia tem a ver com telescópios.”

“Qual é o caminho mais curto para viajar de Roterdã a Groningen?”

***Fonte: Questionamento feito por Dijkstra**, relatado no artigo:* An Interview with Edsger W. Dijkstra, august-2010, vol.53, no.8, Thomas J. Misa – Editor, communications of the acm - https://dl.acm.org/doi/pdf/10.1145/1787234.1787249

UNIFESP

Muito Obrigado!!!

Duvidas, Perguntas ou Questionamentos

E-mail: wagner.cardozo72@gmail.com

LinkedIn: www.linkedin.com/in/wagner-lopes-cardozo-8b4a031ab