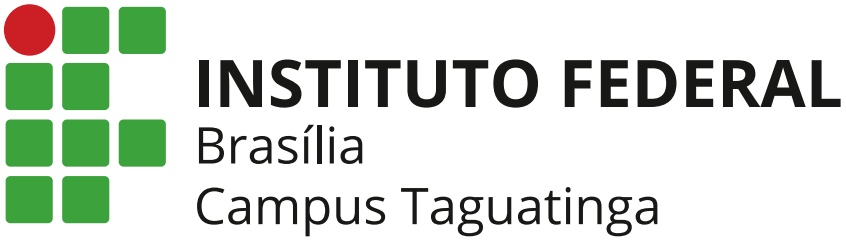
Instituto Federal de Educação, Ciência e Tecnologia de Brasília
Campus Taguatinga

Giovanni Lucas Oliveira da Silva

Linha de Pesquisa: Análise de Algoritmos e Complexidade de Computação

Análise de Desempenho do Algoritmo de Dijkstra Utilizando Diferentes Estruturas de Dados para a Fila de Prioridade



Brasília, DF

2025

Instituto Federal de Educação, Ciência e Tecnologia de Brasília
Campus Taguatinga

Giovanni Lucas Oliveira da Silva

Área de Pesquisa: Análise de Algoritmos e Complexidade de Computação

Análise de Desempenho do Algoritmo de Dijkstra Utilizando Diferentes Estruturas de Dados para a Fila de Prioridade

Documento apresentado como requisito parcial para matrícula na disciplina de Projeto de Conclusão de Curso do curso superior de Bacharelado em Ciência da Computação

Orientador: Daniel Saad Nogueira Nunes

Instituto Federal de Educação, Ciência e Tecnologia de Brasília – Campus Taguatinga

Curso Superior de Bacharelado em Ciência da Computação

Brasília, DF

2025

Introdução

Dados, no ramo da ciência da computação, têm uma importância fundamental para o computador cumprir suas finalidades (SANTOS, 2014). Podemos usar os dados para representar coisas simples como números ou textos, ou coisas mais complexas como imagens e vídeos, ou até representações de pessoas, lugares e objetos reais. Toda aquisição de dados normalmente é feita com o propósito de que sejam utilizados para alguma finalidade, seja para o armazenamento de informações, ou para a realização de algum processamento e tomada de decisões.

Esses dados precisam ser armazenados de alguma forma eficiente, para que possam ser analisados e utilizados posteriormente, e é aí que se fazem importantes as estruturas de dados. Uma estrutura de dados é um modo de armazenar e organizar dados com o objetivo de facilitar acesso e modificações (CORMEN et al., 2012), existem várias estruturas de dados, cada uma com suas características, vantagens e desvantagens. A escolha da estrutura de dados correta para um determinado problema é crucial para o desempenho e eficiência do tratamento desses dados e, consequentemente, para a realização de tarefas e resolução de problemas.

Com o avanço da internet, a quantidade de dados gerados cresceu de forma significativa. Dados esses correspondem muita das vezes a representações de relações entre diferentes objetos, como pessoas, locais e outros elementos. De acordo com o relatório *Data Age 2025*, publicado pela IDC, em 2015 cada pessoa gerava, em média, cerca de 1,3 GB de dados por dia. A previsão era que, em 2025, esse número chegasse a 5,3 GB diários, incluindo interações em redes sociais, pesquisas na web, uso de aplicativos, comunicações e outras atividades digitais (LINKAGES, 2023). Tais relações de dados podem ser modeladas por meio de uma estrutura fundamental chamada grafo. Trata-se de uma das estruturas de dados mais utilizadas justamente por permitir modelar objetos — chamados de vértices — e as conexões entre eles, denominadas arestas. Cada aresta simboliza um vínculo entre dois vértices, que pode representar, por exemplo, a amizade entre duas pessoas ou a rota que liga duas cidades. (GRONER, 2019)

Tomando como por exemplo as aplicações de geolocalização, ter um grafo representando as rotas entre cidades torna esses dados mais fáceis de serem analisados e processados, se colocarmos valores nas arestas representando a distância entre as cidades, quando um usuário quiser ir de uma cidade A para uma outra cidade B, podemos com esse grafo analisar as diversas rotas possíveis entre essas duas cidades caso haja mais de uma rota, ou se até mesmo não houver uma rota direta entre essas duas cidades, e assim determinar diversas possibilidades de rotas que o usuário pode fazer para ir de A para B. Mas dentre todas essas rotas possíveis, surge problemas: Como B é alcançável a partir de A? Qual a rota mais curta entre A e B? Quais são as rotas que passam por uma outra cidade C? Esses são problemas entre muitos outros que podem surgir quando trabalhamos

com essas questões. Nesse pré-projeto vamos focar em um dos problemas mais comuns e importantes que é o problema em determinar qual a rota mais curta entre dois vértices. Esse problema é mais referido como o problema do menor caminho.(LINTZMAYER; MOTA, 2023)(GRONER, 2019)

Esse problema é encontrado em diversas aplicações do mundo real, como nesse caso de geolocalização e navegação, em muitas outras aplicações como redes sociais, jogos, entre outros. E para solucionar esse problema, as pessoas desenvolveram soluções que chamamos de algoritmos, uma sequência de passos definidos que levam a uma solução.(CORMEN et al., 2012) Para o problema do menor caminho, existem diversos algoritmos desenvolvidos, cada um com suas características, vantagens e desvantagens.

Alguns dos algoritmos mais conhecidos para resolver o problema do menor caminho são os algoritmos de Dijkstra (DIJKSTRA, 1959), Bellman-Ford (BELLMAN, 1958) (FORD, 1956) e o Floyd-Warshall (FLOYD, 1962) (WARSHALL, 1962), cada um desses algoritmos tem uma analise assintótica diferente, essas analises nos permitem prever o desempenho de um algoritmo sem a necessidade de implementá-lo e testá-lo, o que é muito útil quando estamos lidando com grandes quantidades de dados. (LINTZMAYER; MOTA, 2023) Porém, essas análises são apenas previsões teóricas, e o desempenho real de um algoritmo pode variar dependendo de muitos fatores. Por isso, é importante não apenas analisar os algoritmos teoricamente, mas também testá-los na prática.

Recentemente um novo algoritmo de menor caminho foi proposto em uma pesquisa intitulada “Breaking the Sorting Barrier for Directed Single-Source Shortest Paths”(DUAN et al., 2025) este algoritmo tem um tempo de execução determinístico assintoticamente menor que o Dijkstra, o que é uma grande proposta, visto que ele é um dos algoritmos mais eficientes para resolver problemas de menor caminho em grafos. (LINTZMAYER; MOTA, 2023) A analise assintotica descreve como o tempo de execucao de um algoritmo escala conforme o tamanho da entrada cresce. Ela nao mede o tempo em segundos, mas sim a ordem de crescimento, ignorando constantes e termos de ordem inferior. Embora seja essencial para comparar algoritmos teoricamente, em situacoes praticas pode levar a conclusoes enganosas. Por exemplo, algoritmos de complexidade $\mathcal{O}(n \log n)$, como o *MERGE SORT*, podem ser mais lentos em entradas pequenas do que algoritmos de $\mathcal{O}(n^2)$, como o *INSERTION SORT*, devido ao peso das constantes escondidas. De forma semelhante, o algoritmo de Dijkstra com fila de prioridade implementada por *Fibonacci Heap* possui melhor complexidade assintotica, mas na pratica costuma ser superado pela versao com *Binary Heap*, que e mais simples e eficiente em cenarios reais. Teoricamente esse algoritmo quebrou uma barreira que a muito tempo era considerada a mais eficiente nesse problema, mas como falado anteriormente, o desempenho real desse algoritmo será diferente do desempenho teórico, e com isso podemos indagar: Será que esse algoritmo vai trazer ganhos práticos em relação ás praticas atuais? Será que é de fato mais rápido que o

Dijkstra em grafos reais?

Proposta

Objetivo Geral

O trabalho se propõe a investigar a discrepância entre o ganho de eficiência teórico e prático do novo algoritmo para o Problema do Menor Caminho de Fonte Única (SSSP), em comparação com o algoritmo de Dijkstra. Para isso, o cerne deste projeto foca na realização de estudos de benchmark. Quantificando e analisando o tempo de execução de cada algoritmo nessas diferentes situações em grafos. Dessa forma, será possível determinar se o avanço assintótico do novo algoritmo se traduz em ganhos de desempenho prático significativos, e em quais condições específica ele consegue, de fato, superar a eficiência consolidada do algoritmo de Dijkstra.

Objetivos Específicos

- Analisar a Complexidade: Realizar a comparação assintótica formal entre o novo algoritmo e os algoritmos clássicos de SSSP, estabelecendo o potencial ganho de eficiência em termos teóricos.
- Testar a Performance Prática: Determinar o comportamento prático do novo algoritmo através de testes em uma ampla variedade de benchmarks de grafos, medindo e quantificando seu tempo de execução e consumo de recursos.
- Quantificar o Ganho de Desempenho: Mensurar o grau de avanço prático do algoritmo de DUAN et al. em relação ao algoritmo de Dijkstra.

Justificativa

O problema do menor caminho não é um problema puramente acadêmico, ele é um problema que é presente em vários ambitos descritos anteriormente, e por isso, ter algoritmos mais eficientes para resolver esse problema é de grande importância. E para que possamos ter certa clareza sobre a eficiência de algoritmos que se propoem a serem mais rápidos que os já utilizados atualmente, é necessário que seja verificada a sua validade prática, ou seja, criando evidências empíricas que comprovem a sua superioridade. Esse estudo baseado em benchmarks permite o fornecimento de dados concretos sobre o desempenho real, aplicando os fatores ignorados pela notação assintótica em seu resultado.

Esses resultados terão impacto prático direto pois, se o novo algoritmo se mostrar de fato mais eficiente, essa informação influenciará decisões de engenharia e desenvolvimento de

software, levando a escolhas mais informadas sobre quais algoritmos utilizar em aplicações que envolvem o problema do menor caminho. Além disso, esse estudo contribuirá para a comunidade acadêmica e profissional ao fornecer uma análise crítica e detalhada sobre a aplicabilidade prática de avanços teóricos em algoritmos, incentivando futuras pesquisas e inovações na área.

Proposta Metodológica Preliminar

A metodologia de pesquisa será de natureza experimental e quantitativa, focada na implementação detalhada de ambos os algoritmos: o novo algoritmo assintoticamente mais rápido e o algoritmo clássico de Dijkstra. Ambos serão implementados fielmente na linguagem **C++** para garantir máxima eficiência e controle sobre o overhead. A avaliação prática será direcionada por meio de um ambiente de testes realista e controlado, utilizando conjuntos de dados e *frameworks* de *benchmark* de grafos. Ferramentas como o **BGGS (*Big Graph Generator Suite*)** serão empregadas para a geração de grafos sintéticos, simulando cenários que variam em escala, densidade e estrutura, refletindo a complexidade de aplicações do mundo real. O tempo de execução será mensurado com alta precisão e repetido múltiplas vezes. Por fim, os resultados serão analisados para comparar o desempenho dos algoritmos em diferentes cenários, validando a aplicabilidade prática do avanço teórico do algoritmo. (DUAN et al., 2025)

Referências

BELLMAN, R. E. On a routing problem. *Quarterly of Applied Mathematics*, v. 16, n. 1, p. 87–90, 1958. Disponível em: ⟨https://www.ams.org/journals/qam/1958-16-01/S0033-569X-1958-0102435-2/⟩. Citado na página 3.

CORMEN, T. H. et al. *Algoritmos: Teoria e Prática*. 3. ed. Rio de Janeiro: Elsevier, 2012. Tradução da 3ª edição de *Introduction to Algorithms*. Citado 2 vezes nas páginas 2 e 3.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, v. 1, p. 269–271, 1959. Disponível em: ⟨https://ir.cwi.nl/pub/9256/9256D.pdf⟩. Citado na página 3.

DUAN, R. et al. Breaking the sorting barrier for directed single–source shortest paths. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25)*. New York, NY, USA: ACM, 2025. p. 36–44. Disponível em: ⟨https://doi.org/10.1145/3717823.3718179⟩. Citado 2 vezes nas páginas 3 e 5.

FLOYD, R. W. Algorithm 97: Shortest path. *Communications of the ACM*, v. 5, n. 6, p. 345, 1962. Citado na página 3.

FORD, L. R. J. *Network Flow Theory*. [S.l.], 1956. Disponível em: ⟨https://apps.dtic.mil/sti/tr/pdf/AD0422842.pdf⟩. Citado na página 3.

GRONER, L. *Estruturas de dados e algoritmos com JavaScript*. 2. ed. São Paulo: Novatec Editora, 2019. Tradução da obra *Learning JavaScript Data Structures and Algorithms - Third Edition* (Packt Publishing, 2018). Tradução de Lúcia A. Kinoshita. Revisão gramatical de Tássia Carvalho. ISBN 978-85-7522-728-2. Disponível em: ⟨https://www.kufunda.net/publicdocs/Estruturas%20de%20dados%20e%20algoritmos%20com%20JavaScript%20(Loiane%20Groner).pdf⟩. Citado 2 vezes nas páginas 2 e 3.

LINKAGES. *Dados: quantos geramos e como isso impacta nossa vida*. 2023. Acesso em: 26 set. 2025. Disponível em: ⟨https://linkages.com.br/2023/03/29/dados-quantos-geramos-e-como-isso-impacta-nossa-vida/⟩. Citado na página 2.

LINTZMAYER, C. N.; MOTA, G. O. *Análise de Algoritmos e de Estruturas de Dados*. [S.l.], 2023. Versão de 4 de agosto de 2023. Disponível em: ⟨https://www.ime.usp.br/~mota/bookFiles/livro_AAED.pdf⟩. Citado na página 3.

SANTOS, A. C. dos. *Algoritmo e Estrutura de Dados I*. [S.l.], 2014. Texto da disciplina produzido em 2008 e revisado em 2010, 2012 e 2014. Licenciado sob Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional. Disponível em: ⟨https://educapes.capes.gov.br/bitstream/capes/176522/2/TextoAEDI_SI_UFAL_AiltonCruz.pdf⟩. Citado na página 2.

WARSHALL, S. A theorem on boolean matrices. *Journal of the ACM*, v. 9, n. 1, p. 11–12, 1962. Citado na página 3.