UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO - UFERSA
CURSO DE BACHARELADO EM CIÊNCIA E TECNOLOGIA
Trabalho de Conclusão de Curso (2023).

UMA VISUALIZAÇÃO EDUCACIONAL GRÁFICA DO ALGORITMO DE DIJKSTRA.

Wilton Costa de Morais[1] , Amanda Gondim de Oliveira[2], Paulo Gabriel Gadelha Queiroz[3]

Resumo: A visualização de algoritmos desempenha um papel crucial na educação no estudo da programação e em diversas áreas da tecnologia. O algoritmo de Dijkstra, criado por Edsger Dijkstra, é fundamental para a teoria dos grafos e a otimização de rotas. No entanto, compreender o funcionamento interno desse algoritmo pode ser desafiador. Este artigo aborda a criação de uma ferramenta educacional gráfica em forma de um *website* que facilite a compreensão do algoritmo de Dijkstra por meio de visualização gráfica. O *website* foi desenvolvido usando JavaScript, HTML, CSS e a biblioteca vis.js, oferecendo uma visualização interativa do algoritmo. Os usuários podem criar grafos, calcular caminhos mais curtos e explorar visualmente o funcionamento do algoritmo. Os resultados demonstram que o *website* atingiu com sucesso seu objetivo de fornecer uma ferramenta educacional eficaz para a visualização e o aprendizado do algoritmo de Dijkstra.

Palavras-chave: algoritmo de Dijkstra, ferramentas educacionais, grafos.

1. INTRODUÇÃO

A visualização de algoritmos desempenha um papel fundamental na aprendizagem em qualquer curso de ciências exatas. Os recursos visuais são amplamente reconhecidos como amplificadores das capacidades de aprendizagem dos alunos, facilitando o processamento de informações complexas e permitindo que os estudantes compreendam de forma mais efetiva os conceitos subjacentes e o funcionamento interno dos algoritmos [1]. Neste contexto, o algoritmo de Dijkstra concebido pelo cientista da computação holandês Edsger Dijkstra em 1956 e publicado em 1959, é uma ferramenta valiosa para a compreensão dos fundamentos da teoria dos grafos e da otimização de rotas [2], ele busca o caminho mais curto em um grafo ponderado através da comparação do menor peso a partir do nó inicial até o último nó ou destino, a fim de encontrar o caminho mais eficaz e eficiente a ser percorrido [3]. No entanto, compreender o funcionamento interno e as etapas desse algoritmo pode ser um desafio para estudantes e iniciantes na área da programação. A falta de recursos educacionais eficazes que demonstrem visualmente o processo de execução do algoritmo de Dijkstra pode dificultar o aprendizado e a compreensão desse conceito fundamental [4].

Diante desse contexto, a questão de pesquisa deste estudo é “Como desenvolver uma página *web* que auxilie em uma visualização educacional gráfica do algoritmo de Dijkstra ?”

A partir disto, o presente artigo, pretende atingir o objetivo por meio do desenvolvimento de um *website* utilizando as linguagens JavaScript, HTML e CSS, juntamente com a biblioteca vis.js para que facilite o aprendizado e a compreensão do algoritmo de Dijkstra através de uma visualização gráfica interativa. Os usuários poderão desfrutar do acesso a uma página *web* interativa que ofereça uma visualização do funcionamento deste algoritmo complexo em um grafo, com o intuito de facilitar o seu aprendizado e a sua compreensão de forma educacional em um ambiente visual.

A organização do restante do trabalho pode ser representada da seguinte forma: na seção 2, apresenta-se todas as definições e fundamentos; na seção 3, encontram-se todos os procedimentos e métodos utilizados para o desenvolvimento do *website*; na seção 4, são apresentados os resultados obtidos da seção anterior; e na seção 5, são descritas as conclusões e as considerações finais deste trabalho.

[1] Graduando do curso Interdisciplinar em Ciência e Tecnologia. Departamento de Ciências Exatas e Naturais – UFERSA. E-mail: wilton.morais@alunos.ufersa.edu.br
[2] Orientadora, Profa. Dra. Departamento de Computação – UFERSA. E-mail: amandagondim@ufersa.edu.br
[3] Coorientador, Prof. Dr. Departamento de Computação – UFERSA. E-mail: pgabriel@ufersa.edu.br

2. FUNDAMENTAÇÃO TEÓRICA

Por meio deste tópico será possível entender toda a base teórica que foi utilizada para atingir o objetivo anteriormente mencionado. Esta seção, está organizada da seguinte maneira: a subseção 2.1, contextualiza de modo geral, a Teoria dos Grafos; na subseção 2.2, o algoritmo de Dijkstra; na subseção 2.3, as ferramentas educacionais e na subseção 2.4, o desenvolvimento *web*, algumas linguagens e a biblioteca vis.js.

2.1. Teoria dos Grafos

A ideia básica dos grafos foi introduzida pela primeira vez no século XVIII pelo eminente matemático suíço Leonhard Euler. Ele trabalhou no famoso problema conhecido como "Problema das Sete Pontes de Königsberg" (1735), que é considerado a origem da teoria dos grafos. Um grafo consiste nas relações entre arestas ou ligações e vértices ou nós. Em outras palavras, grafos são estruturas matemáticas que representam qualquer tipo de entidade que possa estar relacionada umas com as outras por meio de relações de pares [5, 6]. Cada entidade é representada por um vértice do grafo, enquanto as relações de pares entre as entidades são representadas por meio de arestas que conectam pares de nós correspondentes. Além disso, é possível associar um valor numérico (peso) a cada aresta, atuando como um indicador da força da relação entre os nós correspondentes [6].

A Teoria dos Grafos desempenha um papel crucial na resolução de problemas práticos, modelagem de sistemas e análise de redes, tornando-se uma ferramenta essencial em uma variedade de campos acadêmicos e industriais [5]. Para exemplificar melhor na Figura 1 segue um exemplo da aplicação desta teoria em um grafo ponderado, este grafo conecta algumas cidades onde os pontos são as cidades e as a arestas são as distancias entre elas:

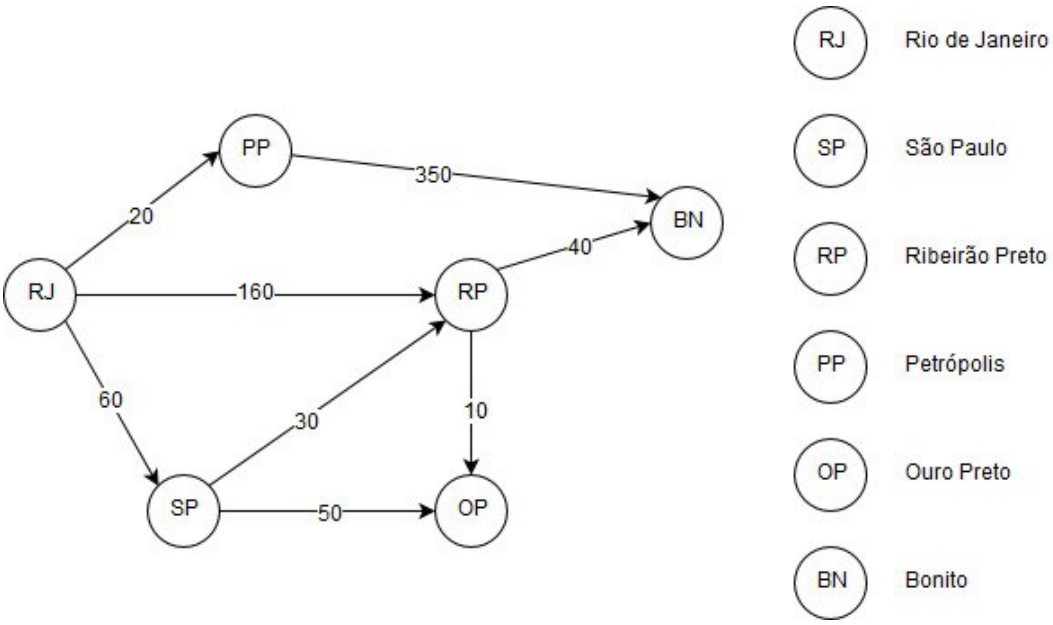

Figura 1. Mapa das cidades. Maida (2017)

2.2. Algoritmo de Dijkstra

O algoritmo de Dijkstra foi criado e publicado por Dr. Edsger W. Dijkstra, um cientista da computação e engenheiro de software holandês. Em 1959, publicou um artigo de 3 páginas intitulado "A note on two problems in connection with graphs", onde explicou seu novo algoritmo que é uma técnica fundamental na teoria dos grafos para encontrar o caminho mais curto entre dois pontos em um grafo ponderado [7]. O problema do caminho mais curto, buscando a rota mais eficiente de um ponto a outro, é um problema

fundamental de otimização com aplicações amplas, como sistemas de navegação, redes de comunicação, planejamento de rotas, entre outros [8]. Além disto, o algoritmo Dijkstra é mais rápido do que outros algoritmos, pois pode calcular o comprimento mais curto para cada ponto [9].

O algoritmo de Dijkstra parte da premissa de que um grafo pode ser representado como uma matriz quadrada NxN, onde os cabeçalhos das linhas e colunas da matriz representam os nós, e os valores dentro da matriz (Aij) representam o valor das arestas, seja ele o caminho, a distância ou o tempo [10]. Isso proporciona uma representação conveniente para calcular os caminhos mais curtos em um grafo. Para calcular, o algoritmo opera em etapas, começando com o nó de origem e expandindo-se gradualmente para os nós adjacentes, ajustando continuamente as distâncias mínimas. Esse processo continua até que todos os nós tenham sido visitados e calculado o caminho mais curto até eles ou até o nó de destino [1]. O algoritmo atribui alguns valores iniciais de distância e tenta aprimorá-los passo a passo da seguinte forma:

1º Passo: Atribua um valor de distância provisório para cada nó. O valor para o nó inicial é definido como zero e como infinito para todos os outros nós.

2º Passo: Marque todos os nós como não visitados e defina o nó inicial como o atual. Crie um conjunto de nós não visitados chamado de conjunto não visitado, que consiste em todos os nós.

3º Passo: Para o nó atual, considere todos os seus vizinhos não visitados e calcule suas distâncias provisórias. Compare a distância provisória recém-calculada com o valor atualmente atribuído e atribua o menor valor.

4º Passo: Se todos os vizinhos do nó atual forem considerados, o nó atual é marcado como visitado e removido do conjunto não visitado. Um nó visitado nunca será verificado novamente.

5º Passo: No caso em que o nó de destino foi marcado ou se a menor distância provisória entre os nós no conjunto não visitado for infinita, pare, ou seja, o algoritmo terminou.

6º Passo: Escolha o nó não visitado com a menor distância provisória e defina-o como o novo "nó atual", depois retorne ao passo 3.

Portanto, para ilustrar melhor esse passo a passo, segue um exemplo onde é possível visualizar a sequência desses passos em um grafo ponderado, onde é atualizado as distancias mínimas para os nós ligados ao nó inicial, os nós visitados e os nós não visitados. Onde a aresta vermelha é representada como o caminho mais curto. Na Figura 2 o nó 0 é escolhido como nó inicial e é removido da lista de nós não visitados:

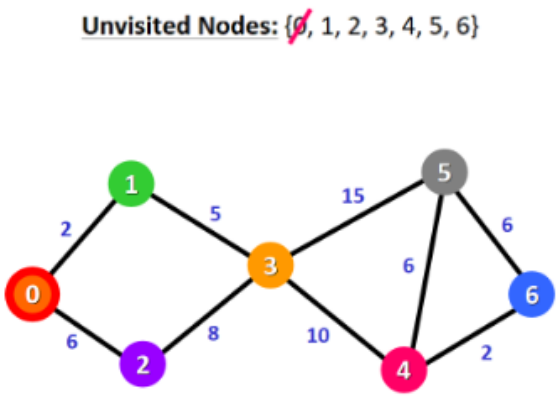

Figura 2. Grafo ponderado e lista de nós não visitados, Navone (2022)

Já na Figura 3 podemos observar a tabela com as distancias atualizadas, onde a distância para o no inicial é 0 e infinito para os nós restantes:

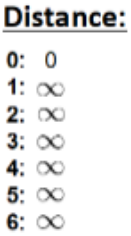

Figura 3. Tabela de distâncias do grafo da figura 2, Navone (2022)

Após isso na Figura 4 é selecionado o no 1, pois note que por enquanto ele é o caminho mais curto saindo do nó 0, então ele é retirado dos nós não visitado assim como o nó 0:

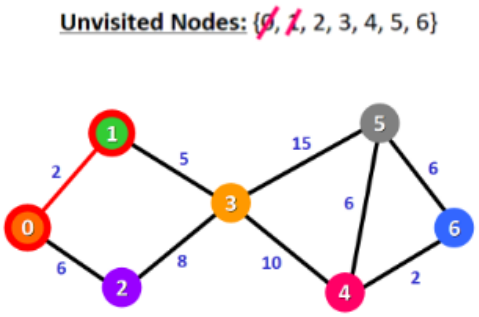

Figura 4 – Grafo ponderado incialmente atualizado e lista de nós não visitados, Navone (2022)

Assim como foi feito antes, na Figura 5 encontra-se a tabela das distancias atualizadas onde é definido provisoriamente a distância mínima partindo do no 0 para os nós 1 e 2:

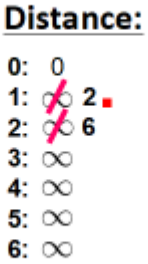

Figura 5. Tabela de distâncias do grafo da figura 4, Navone (2022)

Diante do exemplo exposto de como o algoritmo pode ser utilizado inicialmente em grafos ponderados, é possível perceber que o entendimento do algoritmo em questão não é trivial, sendo esse o principal motivo para o desenvolvimento da ferramenta gráfica educacional para auxiliar na visualização do algoritmo de Dijkstra.

2.3. Ferramentas Educacionais

Devido à dificuldade de compreensão dos passos do algoritmo de Dijkstra a utilização de ferramentas educacionais no apoio da aprendizagem desse algoritmo se mostra bastante interessante, pois a visualização de algoritmos em geral é uma parte essencial do aprendizado nas ciências exatas, e sua importância na educação é amplamente reconhecida [1]. Assim, para facilitar o aprendizado de conceitos complexos, como o algoritmo de Dijkstra, e tornar o processo mais envolvente e compreensível, é fundamental o uso de ferramentas educacionais adequadas. As ferramentas educacionais desempenham um papel fundamental na fomentação da aprendizagem interativa e na exploração ativa de conceitos. Elas oferecem recursos visuais e práticos que auxiliam os alunos a assimilar informações de forma mais eficaz, bem como a aplicar os conhecimentos adquiridos em situações do mundo real. Essas ferramentas são projetadas para tornar o processo de aprendizado mais envolvente, cativante e interativo, beneficiando tanto estudantes quanto educadores [11]. E o uso de ferramentas digitais na educação está se tornando cada vez mais comum e em constante evolução [12].

A incorporação de ferramentas educacionais no processo de ensino pode melhorar significativamente a qualidade da educação em cursos de ciências exatas e em diversas outras áreas acadêmicas. E ferramentas educacionais voltadas para a visualização de algoritmos são projetadas para tornar o aprendizado mais acessível e envolvente. Elas oferecem aos alunos a oportunidade de interagir com a lógica por trás dos algoritmos, experimentando diferentes cenários e observando os resultados em tempo real. Essa abordagem prática ajuda os alunos a internalizar os conceitos, tornando o processo de aprendizagem mais eficaz [12]. Nessa seção foi possível vislumbrar a importância das ferramentas educacionais na melhoria do processo de aprendizado, especialmente quando se trata de conceitos complexos como o algoritmo de Dijkstra. Portanto, na próxima seção do presente artigo explorará brevemente sobre o desenvolvimento *web* projetado para a visualização educacional do algoritmo e junto a isso as tecnologias empregadas para criação do *website*.

2.4. Desenvolvimento *Web*

Atualmente as ferramentas educacionais estão sendo desenvolvidas com o uso de diversas tecnologias, incluindo jogos interativos e aplicativos móveis [13]. Neste artigo, optou-se por desenvolver uma ferramenta educacional em forma de *website*. O desenvolvimento *web* é o processo de criação de sites e aplicativos usando tecnologias como HTML, CSS e JavaScript. É responsabilidade de um desenvolvedor projetar e construir sites visualmente atraentes e otimizados para mecanismos de pesquisa.

Javascript é uma linguagem de programação criada por Brendan Eich em 1995, ela é essencial no desenvolvimento *web* e foi criada com o objetivo de dar vida às páginas *web*, permitindo a criação de *scripts* que são executados diretamente no navegador do usuário. Cada *script* está diretamente ligado ao *HyperText Markup Language* (HTML) de uma página *web* e é executado imediatamente quando a página é carregada. Esta linguagem é conhecida por sua versatilidade e velocidade, uma vez que os navegadores modernos otimizam e executam o código JavaScript de forma eficiente [14]. Esta tecnologia desempenha um papel fundamental na criação de interatividade e dinamismo em páginas da *web* [15].

CSS (*Cascading Style Sheets*): O CSS é uma linguagem de estilo que trabalha em conjunto com o HTML para definir a apresentação de conteúdo em documentos *web*. Ele permite que os desenvolvedores tenham controle total sobre a formatação de documentos HTML, separando a estrutura do documento (definida no HTML) da formatação (definida no CSS). Ao vincular documentos HTML a folhas de estilo CSS compartilhadas, os desenvolvedores podem garantir consistência de design e facilitar a edição em vários elementos de uma página *web* [16].

HTML (HyperText Markup Language): O HTML é uma linguagem de marcação de hipertexto que define a estrutura e o conteúdo das páginas *web*. Está linguagem é composta por um conjunto de *tags* padronizadas que os desenvolvedores usam para indicar parágrafos, imagens, vídeos e links para outros conteúdos da *web*. Juntamente com o CSS, o HTML é a base das páginas da *web*, permitindo que os desenvolvedores criem estruturas de página e apresentações visuais [16].

Vis.js: é uma biblioteca de visualização dinâmica para uso em navegadores da *web*. A biblioteca foi projetada para ser de fácil utilização, lidar com grandes volumes de dados dinâmicos e possibilitar a manipulação e interação com esses dados. A biblioteca é composta pelos componentes DataSet, Timeline, Network, Graph2d e Graph3d. Em relação ao Vis.js - Network, a parte "*network*" (rede) na biblioteca Vis.js é composta principalmente por nós (*nodes*) e arestas (*edges*) [17].

Após essa compreensão das tecnologias fundamentais que servirão de base para o desenvolvimento da ferramenta educacional desejada. Na próxima seção, "Procedimentos e Métodos", entrarei em detalhes sobre como essas tecnologias serão aplicadas para desenvolver a ferramenta educacional proposta, explorando os procedimentos e métodos que guiarão o processo de desenvolvimento e criação do *website* de visualização educacional.

3. PROCEDIMENTOS E MÉTODOS

Nesta seção, apresenta-se os recursos utilizados que podem ser divididos da seguinte maneira: na Subseção 3.1, apresenta-se o estudo inicial realizado em artigos científicos; na Subseção 3.2, descreve-se a metodologia de desenvolvimento; na Subseção 3.3, as linguagens e tecnologias utilizadas.

3.1. Estudo Realizado em Artigos Científicos

Antes de iniciar o desenvolvimento do *website* de visualização do algoritmo de Dijkstra, foi realizado um estudo aprofundado da teoria por meio de artigos científicos relevantes. Esse estudo teve como objetivo compreender completamente o algoritmo, seus princípios subjacentes e suas aplicações na teoria dos grafos.

Neste estudo, foram explorados os seguintes aspectos:

- A origem e história do algoritmo de Dijkstra, incluindo sua criação pelo cientista da computação Edsger W. Dijkstra [18].
- Os fundamentos da teoria dos grafos e a importância do algoritmo de Dijkstra na resolução do problema do caminho mais curto.
- A representação de grafos como matrizes de adjacência e como essa representação facilita o cálculo dos caminhos mais curtos.
- A lógica e os passos do algoritmo de Dijkstra, desde a atribuição de distâncias provisórias até a seleção do caminho mais curto.
- As aplicações práticas do algoritmo de Dijkstra em sistemas de navegação, redes de comunicação e outros domínios.

Esse estudo prévio foi fundamental para embasar o desenvolvimento do *website*, garantindo uma compreensão sólida do algoritmo e de como apresentá-lo de forma educativa e interativa aos usuários.

3.2. Metodologia de Desenvolvimento

O desenvolvimento do *website* seguiu uma abordagem iterativa, conforme detalhado a seguir:

- **Menu inicial:** Foi definido um menu inicial com as opções de “Explicação do algoritmo” e “Instruções”. Onde na primeira opção ao ser clicada abre um *pop-up* com uma breve explicação do algoritmo de Dijkstra e na segunda opção também em forma de *pop-up* descreve instruções para usar o site e suas funções.
- **Definição do Grafo Inicial:** Foram definidos nós e arestas iniciais para representar o grafo. Cada nó foi associado a uma letra e uma cor aleatória. Por exemplo na Figura 6 mostra o grafo inicial no site, onde foi indicado por uma seta oque são os nós e o que são as arestas:

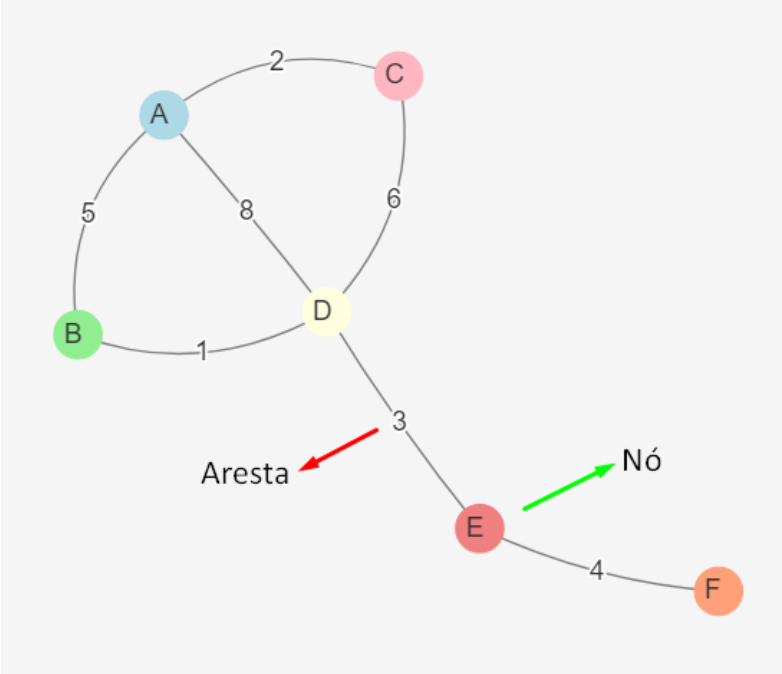

Figura 6. Representação do grafo no *website*. (Autoria própria)

- **Adição de Nós:** Implementou-se uma função que permite a adição de novos nós ao grafo de forma interativa. Ao clicar no botão "Adicionar Nó," um novo nó é criado e conectado a um nó existente aleatoriamente como pode ser observado na Figura 7, onde o nó G foi adicionado aleatoriamente ao grafo:

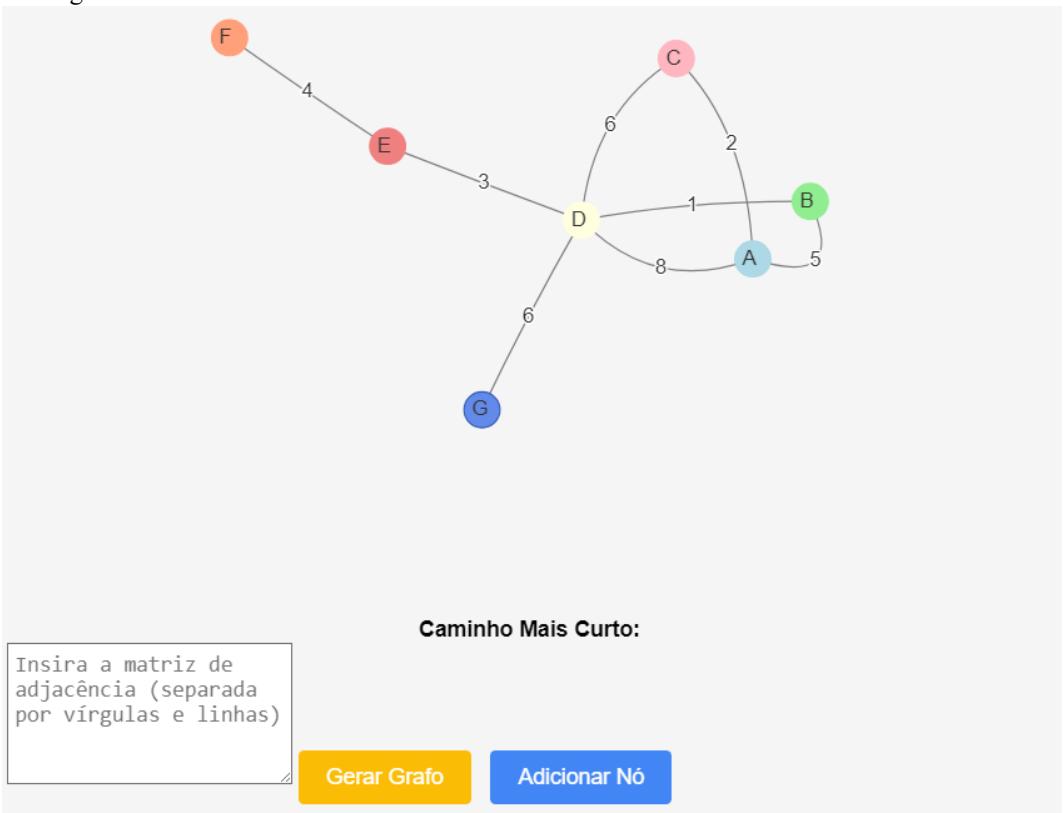

Figura 7. Grafo após acionar a função de adição de nó. (Autoria própria)

- **Visualização Interativa:** Utilizando a biblioteca Vis.js, foi possível gerar visualmente o grafo de

forma interativa no *website*. Os nós são representados por círculos coloridos, e as arestas são exibidas com rótulos que indicam a distância entre os nós que estão ligados.

- **Encontrar o Caminho Mais Curto:** Quando um nó é clicado, a função de encontrar o caminho mais curto é acionada. Ela utiliza o algoritmo de Dijkstra para calcular o caminho mais curto do nó A até o nó clicado e destaca esse caminho no gráfico, como podemos observar na Figura 8 onde é destacado o caminho mais curto do nó A até o no E:

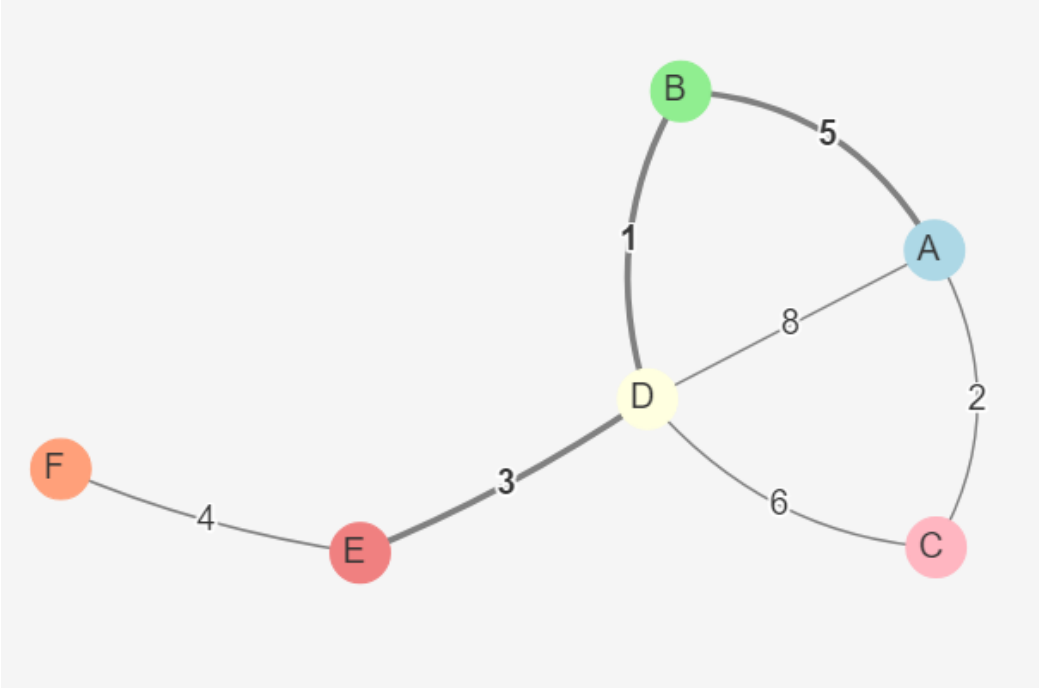

Figura 8. Grafo após clicar no nó E. (Autoria própria)

- **Geração de Gráfico a partir da Matriz de Adjacência:** Foi implementada uma função que permite aos usuários gerar o grafo a partir de uma matriz de adjacência inserida no formato adequado. Cada célula da matriz representa a distância entre os nós correspondentes. Como na Figura 9 onde o grafo foi adicionado de acordo com a matriz adjacência inserida na caixa de texto:

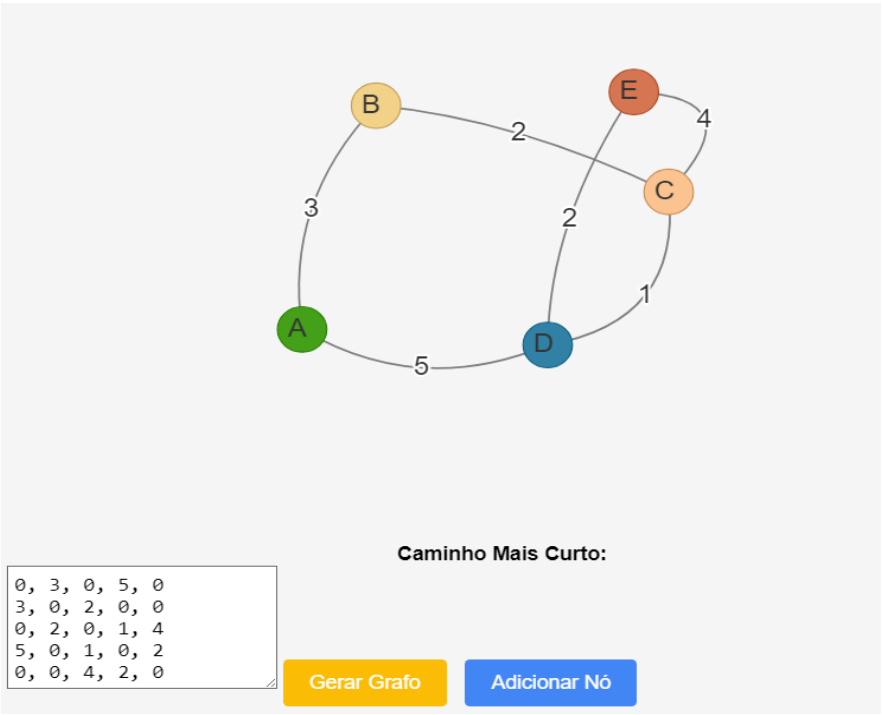

Figura 9. Exemplo de grafo gerado pela matriz adjacência mostrada (Autoria própria)

- **Encontrar o Caminho entre Dois Nós Específicos:** Os usuários podem especificar dois nós para encontrar o caminho mais curto entre eles. As letras correspondentes aos nós são convertidas em *IDs* de nó, e o caminho é destacado no gráfico. Por exemplo na Figura 10 o nó A foi definido como ponto inicial e o nó E como ponto final e após clicar em no botão "Encontrar Caminho" é indicado o caminho mais curto entre eles:

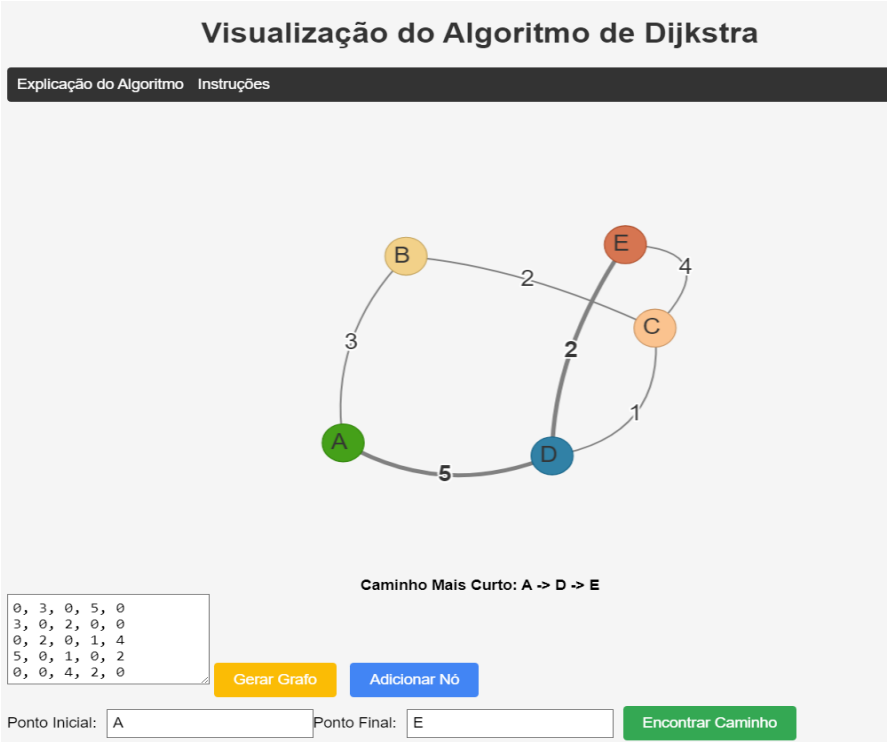

Figura 10. Grafo indicando o caminho mais curto com nós específicos. (Autoria própria)

- **Descrição do Caminho Percorrido:** Para que fique ainda melhor visível também foi implementado uma indicação em forma de sequência dos nós que compõem o caminho mais curto.

3.3. Linguagens e Tecnologias utilizadas

- **JavaScript:** Utilizado para implementar a lógica do algoritmo de Dijkstra e todas as outras funções necessárias para as interações com o usuário e manipulação de dados.
- **HTML e CSS:** Utilizados para criar a estrutura e a aparência do *website*.
- **Vis.js:** Utilizada para criar a visualização gráfica interativa do algoritmo de Dijkstra. A biblioteca foi incluída como um *script* externo e forneceu as funcionalidades necessárias para criar e manipular o grafo.

4. RESULTADOS

Após a conclusão do desenvolvimento do *website* de visualização do algoritmo de Dijkstra, foram alcançados resultados significativos que atenderam aos objetivos propostos no início deste artigo. Abaixo estão os principais resultados e funcionalidades alcançados:

4.1. Visualização Interativa do Algoritmo de Dijkstra

O *website* permite aos usuários uma visualização interativa do funcionamento do algoritmo de Dijkstra em um grafo ponderado. Isso inclui a capacidade de adicionar, remover e modificar nós e arestas usando a matriz adjacência, proporcionando uma experiência dinâmica e abrangente.

4.2. Geração de Grafos a partir de Matrizes de Adjacência

Os usuários podem gerar grafos automaticamente, inserindo uma matriz de adjacência NxN na interface. Essa funcionalidade torna mais acessível a criação e exploração de diferentes configurações de grafos, permitindo que os usuários experimentem cenários diversos.

4.3. Encontrar o Caminho Mais Curto

Uma das características mais importantes do *website* é a capacidade de encontrar o caminho mais curto entre dois nós selecionados. Ao escolher o nó de partida e o nó de destino ou ao clicar em um nó de destino considerando que o nó de partida será o primeiro nó (nó A), o algoritmo de Dijkstra é aplicado automaticamente, destacando o caminho mais curto no grafo. Isso proporciona uma compreensão visual instantânea das rotas mais eficientes entre os pontos desejados.

4.4. Demonstração de todas funções após a conclusão do *website*

Portanto, para demonstrar todas as funcionalidades do *website* irei propor um problema passo a passo onde serão utilizadas todas as funções do site:

1º Passo: Na Figura 11 mostra a geração de um novo grafo usando a matriz adjacência e clicando no botão “Gerar Grafo”:

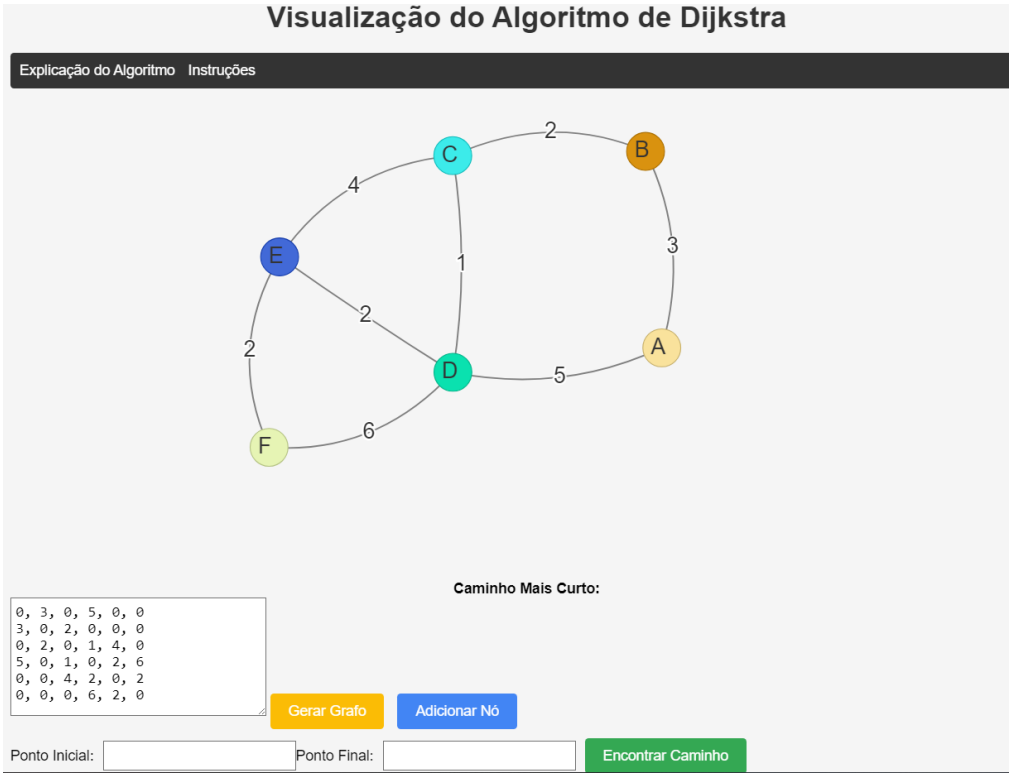

Figura 11. Grafo gerado com matriz adjacência. (Autoria própria)

2º Passo: Na Figura 12 mostra o segundo passo do problema proposto onde foi adicionado o nó G ligado a outro de forma aleatória e também de peso aleatório com o botão “Adicionar Nó”:

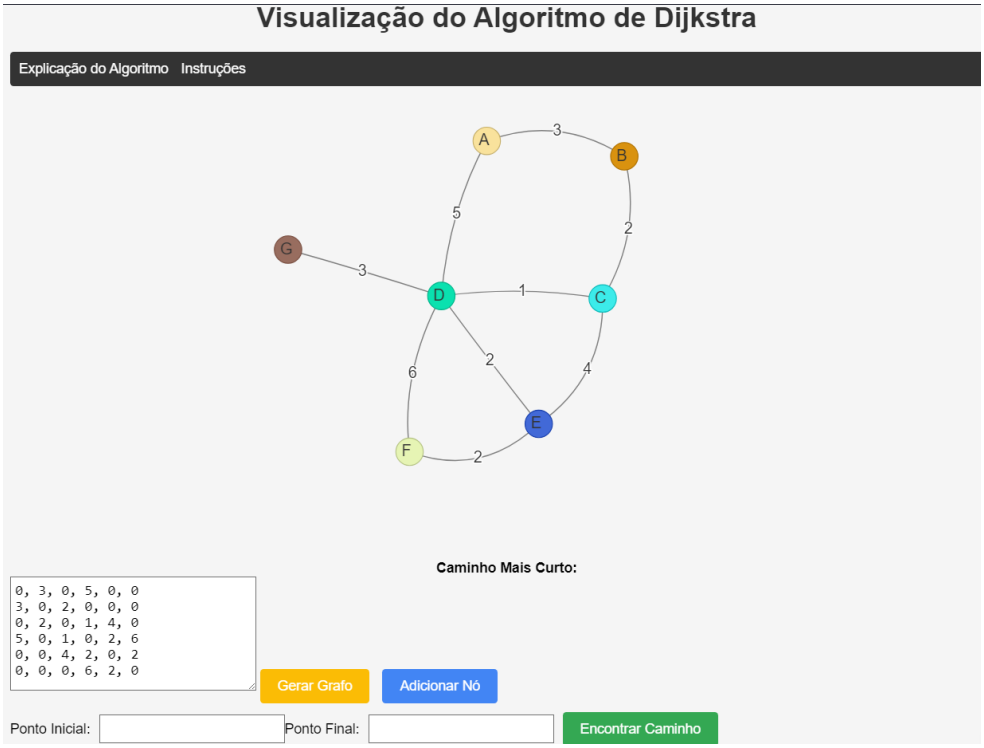

Figura 12. Grafo relacionado ao 2º passo do problema proposto. (Autoria própria)

3º Passo: E na Figura 13 foi definido um ponto inicial e final, onde após clicar no botão “Encontrar Caminho” foi destacado o caminho mais curto no grafo e também descrito embaixo do grafo em sequência os nós que pertencem a esse caminho:

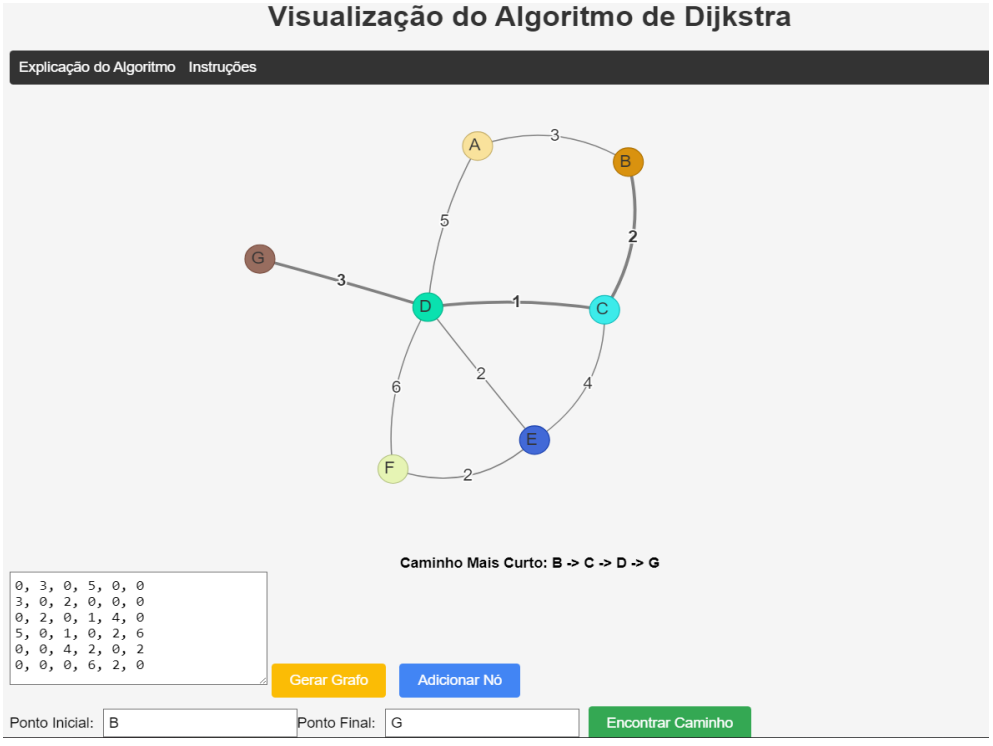

Figura 13. Grafo relacionado ao 3º passo do problema proposto. (Autoria própria)

Diante da resolução deste problema pode-se concluir como essa ferramenta educacional pode ser usada para visualizar o funcionamento do algoritmo de Dijkstra nos mais variados grafos ponderados, pois além

de destacar o caminho mais curto, também é capaz de gerar qualquer grafo com a utilização de uma matriz adjacência.

5. CONCLUSÕES

Os resultados deste artigo demonstram que o *website* atingiu com sucesso seu objetivo de proporcionar uma visualização educacional gráfica do algoritmo de Dijkstra. A combinação de funcionalidades de criação de grafos, cálculo de caminhos mais curtos e representação visual eficaz contribui para uma ferramenta valiosa no contexto da aprendizagem deste algoritmo tão importante.

A questão de pesquisa que orientou este artigo foi: "Como desenvolver uma página *web* que auxilie em uma visualização educacional gráfica do algoritmo de Dijkstra?" A resposta a essa pergunta é clara: o desenvolvimento do *website* utilizando as linguagens JavaScript, HTML e CSS, juntamente com a biblioteca Vis.js, foi eficaz na criação de uma ferramenta que facilita o aprendizado e a compreensão deste algoritmo por meio de uma visualização gráfica interativa.

Além destas conclusões acima, o presente artigo sugere algumas recomendações e áreas para trabalhos futuros como:

- **Melhorias na Interface do Usuário:** Embora o *website* seja funcional, melhorias na interface do usuário podem torná-lo ainda mais amigável e atraente.
- **Compatibilidade com Dispositivos Móveis:** Atualizar o *website* para garantir que seja responsivo e funcione bem em dispositivos móveis pode aumentar ainda mais sua utilidade e acessibilidade.
- **Expansão de Recursos:** Adicionar recursos adicionais, como a capacidade de salvar e carregar grafos, permitiria que os usuários continuassem a explorar e aprender com suas criações.
- **Integração de Outros Algoritmos:** Considerar a inclusão de outros algoritmos de grafos, além do Dijkstra, para proporcionar uma experiência mais abrangente de aprendizado.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] BORISSOVA D.; MUSTAKEROV, I.. E-learning tool for visualization of shortest paths algorithms. Trends Journal of Sciences Research. Vol. 2, No. 3, 2015, pp. 84-89.

[2] RACHMAWATI, Dian; GUSTIN, Lysander. Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem. Journal of Physics: Conference Series. 2020

[3] SERAJ, M.; WONG, C. Y. Lecturers' and Students' Perception on Learning Dijkstra's Shortest Path Algorithm Through Mobile Devices. International Journal of Interactive Mobile Technologies (2014), Disponível em: https://online-journals.org/index.php/i-jim/article/view/3745. Acesso em: 10 sep. 2023.

[4] LU, Y., MAO, X., WANG, T. et al. Improving students' programming quality with the continuous inspection process: a social coding perspective. Front. Comput. Sci. 14, 145205 (2020).

[5] PANDEY, Ambrish; SINGH, Chitra. Application of Graph Theory in Real Life to Develop Routes. Volume 5, Abril 2023.

[6] PAVLOU, Orestis; MICHOS, Ioannis; PAPADOPOULOU, Vicky L.; PAPADOPOULOS, Michalis; PAPAEFTHYMIOU, Evangelos S.; EFSTATHIOU, Andreas. Graph Theoretical Analysis of Local Ultraluminous Infrared Galaxies and Quasars. Astronomy and Computing. 2023. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4455994. Acesso em: 08 de Agosto de 2023

[7] SIPAYUNG, Liskedame; SINAGA, Chaston; SAGALA, Angelica. (2023). Application of Dijkstra's Algorithm to Determine the Shortest Route from City Center to Medan City Tourist Attractions. Journal of Computer Networks, Architecture and High Performance Computing. 5. 648-655. 10.47709/cnahpc.v5i2.2699.

[8] KEMPEPATIL, Ramesh; RUDRASWAMYMATH, Vishwas. Advanced study of Shortest Route Problem and its applications. INTERNATIONAL JOURNAL OF SCIENTIFIC DEVELOPMENT AND RESEARCH, Volume 8, P.1640, 2023

[9] HE, Baoyi. Application of Dijkstra algorithm in finding the shortest path. Journal of Physics: Conference Series. 2181. 012005. 10.1088/1742-6596/2181/1/012005.

[10] IVAN, Makohon; DUC T. Nguyen; MASHA Sosonkina; YUZHONG Shen. JAVA BASED VISUALIZATION AND ANIMATION FOR TEACHING THE DIJKSTRA SHORTEST PATH ALGORITHM IN TRANSPORTATION NETWORKS. International Journal of Software Engineering & Applications (IJSEA), Vol.7, No.3, Maio 2016

[11] KRYUKOV, Vladimir; GORIN, Alexey. DIGITAL TECHNOLOGIES AS EDUCATION INNOVATION AT UNIVERSITIES, Journal of Internet Banking and Commerce; Ottawa Vol. 21, Ed. 3, (Dec 2016): 1-19.

[12] ABID, Haleem; MOHD Javaid; MOHD Asim Qadri; RAJIV Suman. Understanding the role of digital technologies in education: A review. Sustainable Operations and Computers, Volume 3, 2022, Pages 275-285, ISSN 2666-4127,

[13] LI, Yuanzhe; XU, Zezheng; HAO, Yu; XIAO, Peng; LIU, Jingyan. Psychosocial Impacts of Mobile Game on K12 Students and Trend Exploration for Future Educational Mobile Games. Frontiers in Education. 7. 843090. 10.3389/feduc.2022.843090.

[14] TILKOV, S.; VINOSKI, S. Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing, 14(6), 80–83. doi:10.1109/mic.2010.145

[15] SHOIKHEDBROD, Michael. Using JavaScript in web design. BOHR International Journal of Engineering. 2023, Vol. 2, No. 1, pp. 7–14

[16] DAUBS, Michael. (2019). HTML and CSS. Disponível em: https://www.researchgate.net/publication/337956828_HTML_and_CSS. Acesso em: 08 de Agosto de 2023

[17] GAJDOŠ, Ľuboš; GAJDOŠOVA, Elena. Examples of Corpus Data Visualization: Collocations in Chinese. Acta Linguistica Asiatica. 12. 9-26. 10.4312/ala.12.2.9-26, 2022

[18] DIJKSTRA, E. W. A note on two problems in connexion with graphs. Numerische Mathematik, 1959. 269–271.