

**INSTITUTO FEDERAL DO ESPIRITO SANTO
SISTEMAS DE INFORMAÇÃO**

JONATHAS GONÇALVES PICOLI

**DESENVOLVIMENTO DE UM SOFTWARE EDUCACIONAL PARA AUXÍLIO NA
APRENDIZAGEM DA TEORIA DOS GRAFOS EM DISCIPLINAS DE
COMPUTAÇÃO**

Cachoeiro de Itapemirim

2018

JONATHAS GONÇALVES PICOLI

**DESENVOLVIMENTO DE UM SOFTWARE EDUCACIONAL PARA AUXÍLIO NA
APRENDIZAGEM DA TEORIA DOS GRAFOS EM DISCIPLINAS DE
COMPUTAÇÃO**

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Sistemas de Informação do Instituto Federal do Espírito Santo, Campus Cachoeiro de Itapemirim, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: DSc. Rafael Vargas Mesquita dos Santos

Cachoeiro de Itapemirim

2018

JONATHAS GONÇALVES PICOLI

**DESENVOLVIMENTO DE UM SOFTWARE EDUCACIONAL PARA AUXÍLIO NA
APRENDIZAGEM DA TEORIA DOS GRAFOS EM DISCIPLINAS DE
COMPUTAÇÃO**

Trabalho de Conclusão de Curso apresentado à
Coordenadoria do Curso de Sistemas de Informação
do Instituto Federal do Espírito Santo, Campus
Cachoeiro de Itapemirim, como requisito parcial
para a obtenção do título de Bacharel em Sistemas
de Informação.

Aprovado em 28 de Novembro de 2018.

COMISSÃO EXAMINADORA

DSc. Rafael Vargas Mesquita dos Santos
Instituto Federal do Espírito Santo -
Cachoeiro de Itapemirim

DSc. Eros Estevão Moura
Instituto Federal do Espírito Santo -
Cachoeiro de Itapemirim

DSc. Flávio Izo
Instituto Federal do Espírito Santo -
Cachoeiro de Itapemirim

DECLARAÇÃO DO AUTOR

Declaro, para fins de pesquisa acadêmica, didática e técnico-científica, que este Trabalho de Conclusão de Curso pode ser parcialmente utilizado, desde que se faça referência à fonte e ao autor.

Cachoeiro de Itapemirim, 28 de Novembro de 2018.

JONATHAS GONÇALVES PICOLI

Dedico esse trabalho primeiramente a Deus e a todos que de alguma forma contribuíram para que o mesmo fosse realizado.

AGRADECIMENTOS

"Fala fiote, tá bão?".
Bola

RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação desktop, que permita ao professor deter ferramentas gráficas que possam auxiliá-lo a lecionar disciplinas de programação que incluam a teoria dos grafos, buscando tornar o processo mais claro e otimizando o resultado da aula. Para tal o software utiliza uma interface gráfica composta por elementos que se comportam de forma dinâmica, de acordo com a execução do algoritmo escolhido, destacando-se a exibição do código fonte ao lado do desenho do grafo, evidenciando seu funcionamento e facilitando o seu entendimento por parte dos alunos. Foram implementadas as funcionalidades de exploração para dois algoritmos de busca, sendo um de busca em largura (BFS) e outro de busca em profundidade (DFS). Através da metodologia para avaliação de software educacional desenvolvida por Thomas Reeves, verificou-se que a ferramenta atingiu um grau satisfatório de desempenho, tendo pontuação acima de 6 em todas as questões avaliadas.

Palavras-chave: Teoria dos Grafos. Educação. Software.

ABSTRACT

This work presents the development of a desktop application that allows the teacher to hold graphical tools that can help him to teach programming disciplines that include graph theory, seeking to make the process clearer and optimizing the result of the lesson. For this the software uses a graphical interface composed by elements that behave dynamically, according to the execution of the chosen algorithm, highlighting the display of the source code next to the drawing of the graph, evidencing its operation and facilitating its understanding by the students. Exploration features were implemented for two search algorithms, one search in width (BFS) and one search in depth (DFS). Through the methodology for evaluation of educational software developed by Thomas Reeves, it was verified that the tool reached a satisfactory degree of performance, having score above 6 in all the questions evaluated.

Keywords: Theory of graphs. Education. Software.

LISTA DE FIGURAS

Figura 1 – Abstração do problema da pontes de Königsberg	18
Figura 2 – Representação gráfica ilustrativa de uma rede social	19
Figura 3 – Resultado de árvore geradora Mínima	20
Figura 4 – Tela de resultado da aplicação AlgoDeGrafos	21
Figura 5 – Tela do jogo WarGrafos durante uma partida	22
Figura 6 – Tela durante a execução do algoritmo Dijkstra	23
Figura 7 – Tela de resultado da aplicação A-Graph	23
Figura 8 – Tela principal da aplicação TGrafos	24
Figura 9 – Pseudocódigo do algoritmo DFS	28
Figura 10 – Pseudocódigo do algoritmo BFS	29
Figura 11 – Diagrama de caso de uso	30
Figura 12 – Diagrama de caso de uso	30
Figura 13 – Estrutura do código	31
Figura 14 – Tela inicial	34
Figura 15 – Legenda de cor dos status	34
Figura 16 – Tela do algoritmo BFS em execução	35
Figura 17 – Tela do algoritmo DFS em execução	36
Figura 18 – Tela final do algoritmo BFS	37
Figura 19 – Tela final do algoritmo DFS	37
Figura 20 – Média geral das notas	40
Figura 21 – Média dos módulos do formulário	41

LISTA DE TABELAS

Tabela 1 – Exemplos de aplicação de grafos	18
Tabela 2 – Comparativo entre ferramentas similares	24

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Objetivos	13
1.1.1	Geral	13
1.1.2	Específicos	14
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	Software Educacional	15
2.1.1	O que é	15
2.1.1.1	Tipos de software educacional	15
2.1.2	Por que utilizar um software educacional?	16
2.1.3	Barreiras na utilização	16
2.1.4	Potencialidades	17
2.2	Grafos	17
2.2.1	Conceito	17
2.2.2	Contexto Histórico	17
2.2.3	Aplicabilidade	18
2.2.3.1	Relações entre os pontos	19
2.2.3.2	Árvore Geradora Mínima	20
2.3	Trabalhos Relacionados	20
2.3.1	Comparativo	24
3	METODOLOGIA DA PESQUISA	26
3.1	Tecnologias e ferramentas	27
3.2	Escolha dos algoritmos	27
3.2.1	DFS	28
3.2.2	BFS	29
4	RESULTADOS	30
4.1	Modelagem	30
4.2	Código	30
4.2.1	Funcionamento do código	32
4.3	Aplicação	33
4.3.1	Tela Inicial	33

4.3.2	Coloração dos elementos	34
4.3.3	Algoritmo BFS e DFS	35
4.4	Validação	38
4.5	Questionário aplicado	38
4.5.1	Atributos Pedagógicos	38
4.5.2	Atributos de interface gráfica	39
4.6	Principais Inferências	40
5	CONCLUSÃO	42
	REFERÊNCIAS	43

1 INTRODUÇÃO

O ensino superior é a porta de entrada dos estudantes para o mercado de trabalho. Ele ajuda a definir o quão qualificados serão os profissionais inseridos nas mais diversas áreas. Atualmente o mercado de trabalho exige cada vez mais dos profissionais, tornando a graduação requisito mínimo para diversas funções. Esta necessidade de profissionais mais capacitados incentiva as Instituições de Ensino Superior (IES), a ofertarem mais vagas.

O censo da educação superior, revelou que em 2017 haviam mais de 8 milhões de egressos na graduação, sendo 3,6 milhões novos egressos. Neste mesmo ano mais 10,7 milhões de vagas foram ofertadas (INEP, 2017), deixando clara a importância do investimento no setor.

Em contrapartida a números positivos tão expressivos, um dado gera preocupação. Em 2017, apenas 30% das vagas ofertadas foram preenchidas (INEP, 2017). Este número expressa um alto grau de desinteresse em ingressar nos cursos superiores no Brasil. Um ponto a ser avaliado são os avanços tecnológicos. Eles tornam o cotidiano dos alunos algo extremamente dinâmico, e isso gera um choque de realidade em relação a sala de aula, onde métodos rígidos e com poucas variações são utilizados para fazer com que o aluno absorva a informação. Esta realidade torna o ensino desinteressante e incentiva o abandono dos estudos.

Os alunos necessitam dominar o processo de aprendizagem para o desenvolvimento de suas competências, e não mais absorver somente o conteúdo. Faz-se necessária uma educação permanente, dinâmica e desafiadora visando o desenvolvimento de habilidades para a obtenção e utilização das informações (MORATORI, 2003).

Diante da dificuldade em manter os alunos interessados em sua formação superior, da alta porcentagem de vagas ociosas e cientes das necessidades de dinamizar o processo de ensino, este trabalho apresenta a proposta de desenvolvimento da aplicação EasyGrafos.

1.1 OBJETIVOS

1.1.1 Geral

Desenvolver um software educacional para auxílio na aprendizagem da teoria dos grafos em disciplinas de programação.

1.1.2 Específicos

1. Ilustrar de forma gráfica a ação dos algoritmos no grafo, bem como, o status de suas variáveis.
2. Associar cada linha do código fonte do algoritmo escolhido com uma ação na interface gráfica.
3. Disponibilizar controles para possibilitar um modo debug.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SOFTWARE EDUCACIONAL

Com o aumento do interesse no ingresso em universidades, andando em paralelo com um alto índice de evasão escolar, torna-se necessário novas formas de apresentar os conteúdos na sala de aula. Uma dessas formas pode ser exemplificada através dos softwares educacionais. Um exemplo é a plataforma virtual MOODLE, que hoje é uma ferramenta disseminada no âmbito acadêmico mundial, sendo disponibilizada em mais de 100 idiomas para milhões de usuários.

2.1.1 O que é

Várias são as definições para o termo software educacional, porém o mais adaptável a solução proposta por esse trabalho afirma que este é todo aquele programa que possa ser usado para algum objetivo educacional, pedagogicamente defensável, por professores e alunos, qualquer que seja a natureza ou finalidade para o qual tenha sido criado (TEIXEIRA, 2003).

2.1.1.1 Tipos de software educacional

Existem algumas categorias de software educacional, basicamente suas diferenças se dão no objetivo, assim de acordo com o objetivo, estas devem adotar metodologias específicas. Podemos destacar três tipos de software educacionais (GIRAFFA, 2009), sendo eles:

- Programas de reforço ou exercício - o aluno pratica e testa conhecimentos de forma dirigida e mais procedural. As versões mais atuais destes programas utilizam recursos de hipermídia mantendo estas características.
- Tutoriais - seguem o padrão de ensino da sala de aula tradicional onde o conteúdo é previamente organizado numa estrutura definida pelo professor e o aluno seleciona dentre as diversas opções disponíveis o conteúdo que deseja estudar.

- Jogos educacionais e as simulações - São tarefas a serem resolvidas, onde os jogos possuem caráter competitivo e as simulações são meramente repetições de exercícios.

2.1.2 Por que utilizar um software educacional?

O computador está em praticamente todos os ambientes que frequentamos, e isso também é válido para os alunos. O motivo disso está na forma como eles chamam a atenção das pessoas, devido as variadas formas de expor a informação, seja ela, propagandas, músicas, notícias e etc. Logo, faz-se necessário criar esse efeito também dentro da sala de aula, caso contrário esta será sempre a segunda opção. É preciso refletir sobre as metodologias de ensino levando em consideração a realidade vivida dessa nova sociedade (SILVA, 2014).

O computador pode ser também utilizado para enriquecer ambientes de aprendizagem e auxiliar o aprendiz no processo de construção do seu conhecimento (VALENTE, 1999). Não faltam variações na forma de utilizar tais ferramentas, já que a tecnologia avança rapidamente criando novos recursos para torna-la ainda mais atrativa, como a hoje popular realidade virtual.

2.1.3 Barreiras na utilização

Entretanto essas ferramentas vem apresentando alguns problemas que as impossibilitam de atingir todo o potencial, tornando assim responsabilidade do professor explorar os espaços virtuais e suas possibilidades. A matemática, por exemplo, é uma área que possui grande variedade de aplicações que auxiliam no processo de aprendizagem, entretanto essa diversificação não implica em qualidade. Muitos programas possuem conteúdo mal formulado, problemas na execução do sistema, interfaces (telas) confusas e assim por diante (GIRAFFA, 2009).

Outro fator que deve ser observado é a dispersão. Os professores em alguns momentos não possuem ferramentas atrativas o suficiente para manter a atenção dos alunos durante a aula. Os aparelhos portáteis por exemplo oferecem dificuldade quanto a fiscalização da utilização na sala e em contrapartida, possibilita ao usuário muitas formas de distração. A falta de atenção, a desmotivação dos alunos é algo frustrante

para todo o corpo o docente. A utilização dos celulares, é apontada como motivo para dispersão e falta de interesse dos alunos (SILVA, 2014).

2.1.4 Potencialidades

Apesar dos problemas enfrentados os softwares educacionais possuem uma vantagem em relação a didática comum, que é a visualização em imagens e outras mídias, que são métodos importantes no auxílio da compreensão da abstração que as disciplinas de programação exigem.

O não mostrar equivale a não existir, a não acontecer. O que não se vê perde a existência. Um fato mostrado com imagem e palavra tem mais força, do que se for mostrado somente com palavra. Muitas situações importantes do cotidiano perdem força por não ter sido valorizadas pela imagem-palavra televisiva (MORAN, 2000).

2.2 GRAFOS

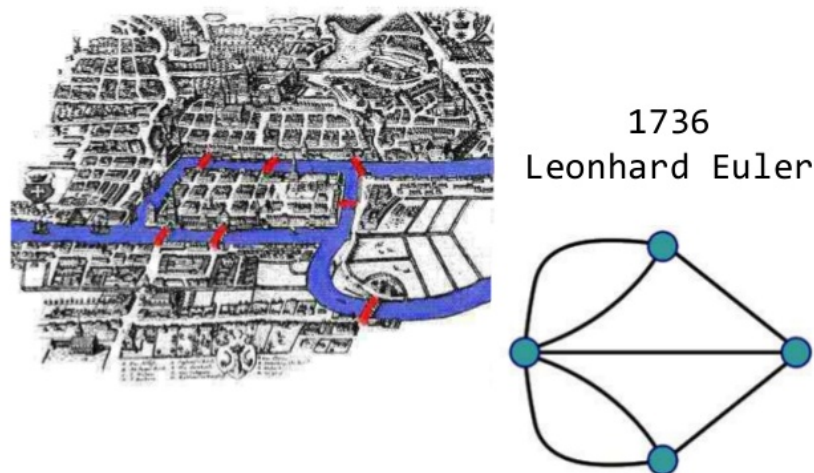
2.2.1 Conceito

Um grafo pode ser definido como uma estrutura composta de pontos e ligações entre eles (JURKIEWICZ, 2009). Sua representação pode ser dada como sendo um grafo(G), composto por um conjunto finito de vértices $V(G)$, ligados por um conjunto finito de arestas $A(G)$.

2.2.2 Contexto Histórico

O primeiro estudo que se caracterizou como teoria dos grafos, ocorreu em 1736 na cidade de Königsberg, e foi realizado pelo matemático suíço Leonhard Euler. O estudo se baseava na estrutura da cidade, a mesma possuía duas ilhas e 7 pontes ligavam suas margens. O problema a ser resolvido era avaliar a possibilidade de sair de um ponto, passar por todas as 7 pontes exatamente uma vez retornando ao ponto de origem.

Para resolver o problema Euler montou um diagrama para representar a cidade, que pode ser visualizado na figura 1. Nele cada margem foi associada a um vértice e cada ponte a uma aresta. Sua conclusão foi que para realizar o percurso com as condições impostas, cada vértice deveria ter um número par de vértices ligados a ele, como haviam vértices ligados a três arestas o percurso se mostrou impossível(COSTA, 2011).



As pontes de Königsberg

Fonte: http://www.obm.org.br/content/uploads/2017/01/Nivel1_grafos_bruno.pdf

Figura 1 – Abstração do problema da pontes de Königsberg

2.2.3 Aplicabilidade

O conceito de grafos pode ser empregado em diversos problemas da atualidade. Suas características o tornam modelo para situações complexas em inúmeras áreas de estudo, e assim possibilitam a aplicação de métodos da Teoria dos Grafos para analisar de forma mais precisa tais situações. A tabela 1, apresenta algumas das muitas abstrações possíveis em problemas das mais diversas áreas.

grafo	vértice	aresta
comunicação	telefone, computador	cabo de fibra óptica
circuito	porta, processador	corrente
mecânica	articulação	haste, viga, mola
financeiro	estoque, moeda	transações
transporte	intercessão	rua
internet	rede de classe C	conexão
jogos	posição do tabuleiro	movimento legal
rede social	pessoa	amizade
rede neural	neurônio	sinapse
rede proteica	proteína	interação proteína-proteína
molécula	átomo	ligação

Tabela 1 – Exemplos de aplicação de grafos

Fonte: Material da disciplina de TPA

Para ilustrar usos destes conceitos, serão explanadas duas categorias de prolemas presentes na tabela 1, a relação entre 2 pontos, presente em todos os exemplos e a árvore geradora mínima, que pode se fazer necessária dependendo do objetivo ao explorar a estrutura.

2.2.3.1 Relações entre os pontos

Atualmente este conceito pode ser claramente associado as redes sociais digitais, representadas de forma simplificada pela figura 2, onde existem usuários, que são representados pelos pontos e suas amizades representadas pelas relações. Assim novas possibilidades de análise desses grupos podem ser realizadas.

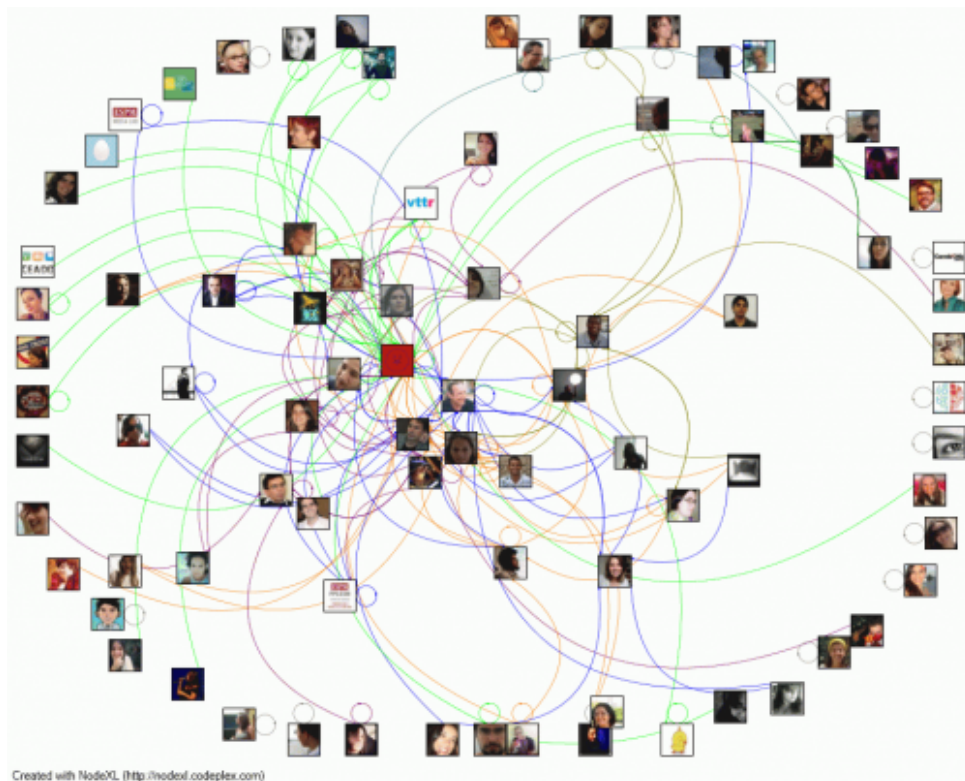


Figura 2 – Representação gráfica ilustrativa de uma rede social

Fonte: http://www.raquelrecuero.com/assets_c/2013/06/compowsfinalcluster-thumb-500x403-501.png

Cartwright junto com o matemático Harary ainda na década de 50 iniciou estudos sobre os comportamentos dos grupos, propondo que pessoas próximas tendem a agir de forma similar de frente a terceiros. Estes estudos possibilitaram a criação de modelos da interdependência sistemática entre pessoas diferentes dentro de um mesmo grupo (BRAGA; GOMES; RUEDIGER, 2008).

A aplicação deste conceito esta inserida nas mais variadas áreas de estudo. Na

sociologia por exemplo, a teoria dos grafos é uma das bases no estudo das redes sociais, onde esta possibilita a análise das redes como um todo e não somente em parte, possibilitando a exploração de propriedades antes não expostas (RECUERO, 2004).

2.2.3.2 Árvore Geradora Mínima

O conceito de árvore geradora mínima (MST) pode ser descrito como um caminho que liga todos os nós da rede sem gerar ciclos, porém devemos observar que um mesmo grafo pode ter várias árvores geradoras mínimas, sendo a solução ideal a de menor custo, sabendo que o custo é a soma dos pesos das aresta que fazem parte da solução (ALMEIDA et al., 2006). Para exemplificar esse conceito suponhamos o seguinte problema. A figura 3, ilustra com linhas cinza claro no primeiro quadro da imagem todas as possibilidades de caminho. Nesse momento ao aplicarmos um algoritmo MST, este nos dará qual o caminho de menor custo, sendo este ilustrado com linhas pretas.

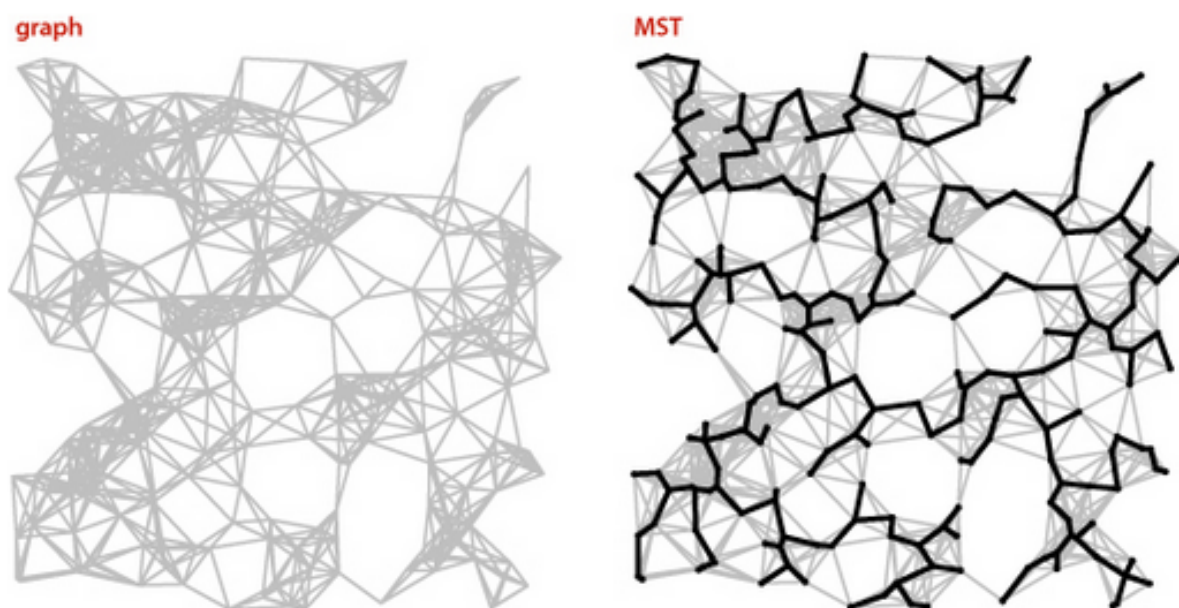


Figura 3 – Resultado de árvore geradora Mínima

Fonte: https://www.ime.usp.br/pf/algoritmos_para_grafos/aulas/mst.html

2.3 TRABALHOS RELACIONADOS

Existem hoje alguns softwares para auxiliarem no estudo da teoria dos grafos. Estes utilizam diferentes ferramentas para facilitar a compreensão da disciplina. Foram seleci-

onadas algumas dessas aplicações afim de verificar a necessidade de desenvolvimento de uma nova aplicação para o gênero. Estas foram encontradas através de pesquisas na plataforma Google¹, utilizando os termos de pesquisa "ensino e aprendizagem de grafo" e "aplicativo para aprendizagem de grafos em computação". As ferramentas deveriam possuir o mesmo propósito da ferramenta deste presente trabalho. A seguir serão citadas estas ferramentas e suas principais características.

- **AlgoDeGrafos** é uma aplicação que utiliza interface gráfica para ilustrar o resultado da execução de algoritmos para exploração de grafos, não apresentando ao usuário o código do algoritmo e não possibilitando a execução passo a passo. Há uma grande variedade de algoritmos habilitados na ferramenta e o usuário pode criar seus próprios grafos, podendo salva-los para posterior estudo. O principal objetivo da ferramenta é o auxílio no modelo EAD de educação (MELO; SILVEIRA; JURKIEWICZ, 2009). A figura 4 apresenta a tela do resultado final da execução de um algoritmo na aplicação AlgoDeGrafos.

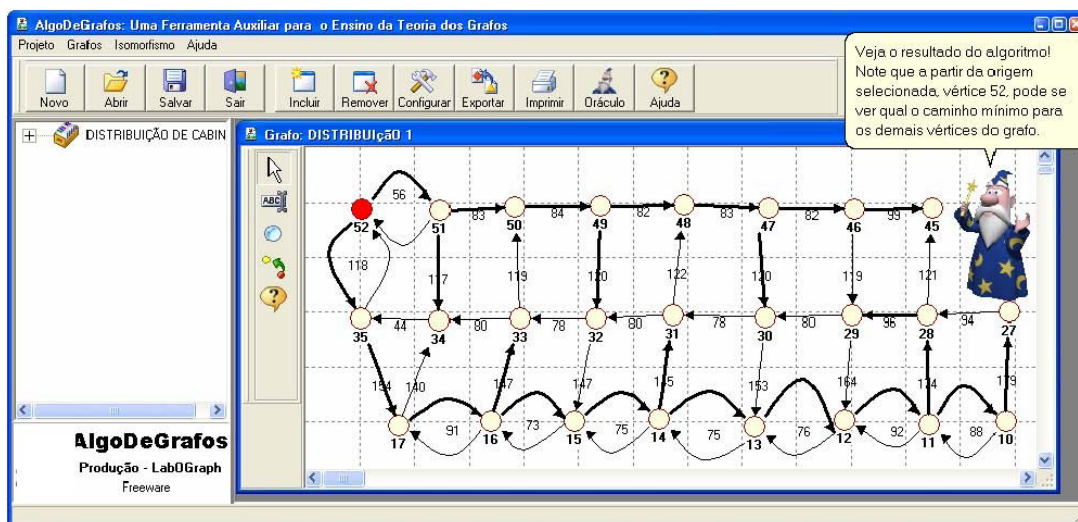


Figura 4 – Tela de resultado da aplicação AlgoDeGrafos

Fonte: (MELO; SILVEIRA; JURKIEWICZ, 2009)

- **WarGrafos** é um jogo baseado no jogo de tabuleiro War lançado pela empresa Grow. Nele os alunos são divididos em equipes e cada uma recebe um objetivo, sendo que o objetivo das equipes inimigas não é conhecido. Para alcançar o objetivo e vencer a partida a equipe deve conhecer e empregar conceitos da teoria dos

¹ <https://www.google.com>

grafos (FIGUEIREDO; FIGUEIREDO, 2011). A figura 5 apresenta a tela durante uma partida do jogo.

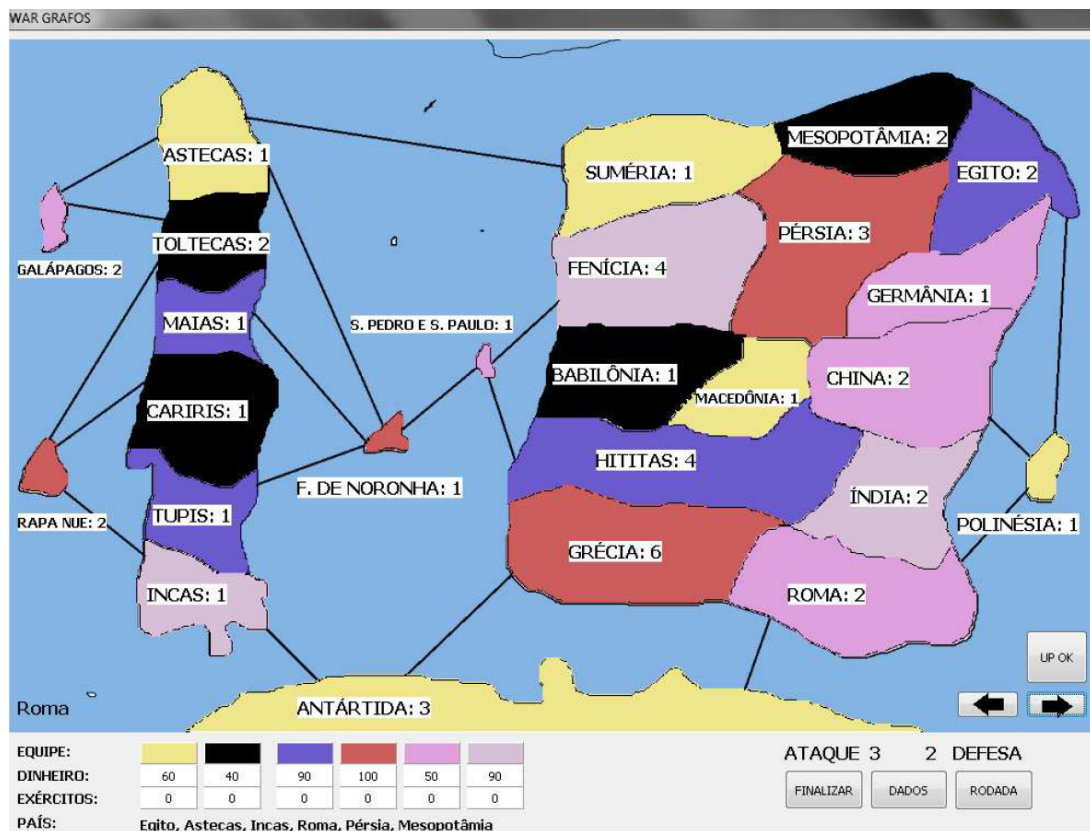


Figura 5 – Tela do jogo WarGrafos durante uma partida

Fonte: (FIGUEIREDO; FIGUEIREDO, 2011)

- **TBC-GRAFOS** é um software que visa facilitar a visualização da execução de algoritmos em estruturas de grafos. Ele se utiliza da apresentação de alguns conceitos teóricos simples, possibilidade de execução passo a passo, legendas explicativas e apresentação de um Pseudocódigo do algoritmo escolhido (SANTOS et al., 2008). Ele aborda os seguintes conceitos:

1. Busca em grafos (percurso em profundidade e em largura)
2. Árvore geradora mínima (algoritmos de Kruskal e de Prim)
3. Caminho mínimo entre vértices (algoritmos de Dijkstra e de Bellman-Ford)

A figura 6 apresenta um exemplo de execução do algoritmo Dijkstra na aplicação TBC-GRAFOS

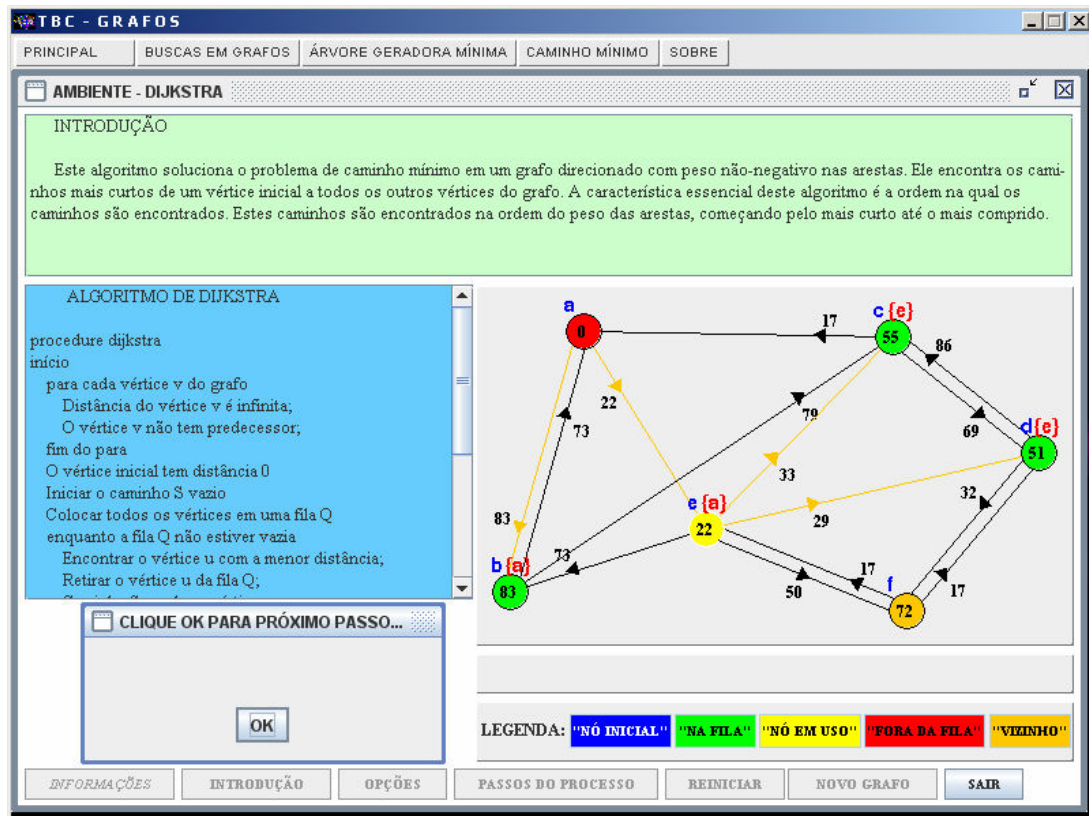


Figura 6 – Tela durante a execução do algoritmo Dijkstra

Fonte: (SANTOS et al., 2008)

- **A-Graph** visa a criação de grafos utilizando interface gráfica e possibilita a execução de dois algoritmos a busca em largura e em profundidade. A aplicação não exibe o código ao usuário e não possibilita a execução passo a passo (LOZADA, 2014). A figura 7 ilustra um exemplo de execução na ferramenta.

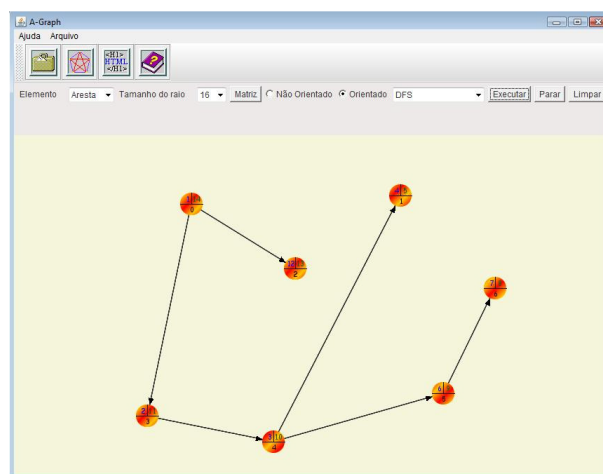


Figura 7 – Tela de resultado da aplicação A-Graph

Fonte: (LOZADA, 2014)

- **TGrafos** é um software para criação e estudo de grafos. Apresenta diversos

recursos em relação a teoria da disciplina e em relação a características do grafo criado. Não há a possibilidade de executar algoritmos nos grafos criados (SILVEIRA; SILVA, 2016). A figura 8 apresenta a tela principal da aplicação TGrafos.

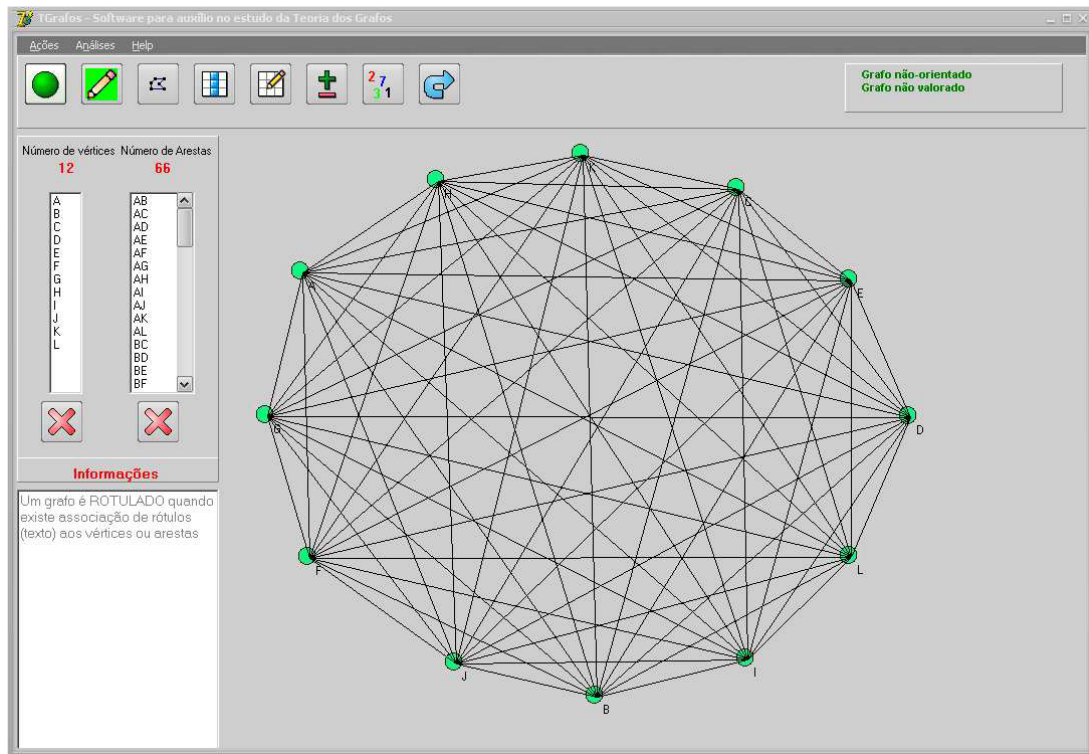


Figura 8 – Tela principal da aplicação TGrafos

Fonte: (SILVEIRA; SILVA, 2016)

2.3.1 Comparativo

Baseado nas principais características das aplicações citadas, foi possível realizar um comparativo, entre estas e a aplicação proposta pelo trabalho. O resultado pode ser visualizado na tabela 2.

Característica/Software	WarGrafos	TBC-Grafos	AlgoDeGrafos	A-Graph	Tgrafo	EasyGrafos
Interface para manipulação e análise dos grafos		X	X	X	X	X
Criar os próprios grafos			X	X	X	X
Executar algoritmos da disciplina nos grafos		X	X	X	X	X
Execução em modo debug		X	X			X
Associação do código fonte do algoritmo em estudo ao grafo						X
Utilização de técnicas de gamificação	X					

Tabela 2 – Comparativo entre ferramentas similares

Fonte: Próprio autor

O estudo destas ferramentas e a avaliação da tabela comparativa, nos possibilita aferir os diferenciais entre a aplicação proposta e as demais já disponíveis. A principal delas

é a associação do código fonte do algoritmo selecionado, ao desenho do grafo, que em conjunto com os controles de modo debug (disponíveis na aplicação EasyGrafos), entregam ao professor vastas possibilidades de ministrar a aula. Nenhuma das aplicações pesquisadas apresentou este recurso. Outro diferencial é a execução em modo debug, que só é disponibiliza por menos da metade das aplicações pesquisadas.

3 METODOLOGIA DA PESQUISA

Para o desenvolvimento da ferramenta proposta neste projeto, foram realizadas as seguintes etapas:

- Estudo sobre as principais ferramentas para o ensino de grafos na área da ciência da computação;
- Elaboração de cinco arquivos contendo dados de grafos exemplos, com diferentes características (grafos densos, grafos esparsos, etc.), para utilização na ferramenta;
- Implementação de dois algoritmos clássicos que utilizam o conceito da teoria dos grafos: busca em largura e busca em profundidade;
- Implementação de duas telas para a exibição da execução dos dois algoritmos mencionados anteriormente;
- Documentação do código em javadoc;
- Escolha das questões do formulário de avaliação: As questões escolhidas foram retiradas da metodologia para avaliação de software educacional de Thomas Reeves. Foram escolhidas 5 questões de carácter pedagógico e 6 referentes a interface com o usuário.
- Validação do objeto de aprendizagem: nesta etapa a ferramenta foi disponibilizada para a turma da disciplina de técnicas avançadas de programação do curso de sistemas de informação do Ifes Campus Cachoeiro de Itapemirim. Os alunos, após a utilização da ferramenta, responderam a um questionário de avaliação;

3.1 TECNOLOGIAS E FERRAMENTAS

O software produzido utiliza como tecnologia para o desenvolvimento da aplicação a linguagem de programação Java na versão 8, utilizando-se das bibliotecas gráficas do JavaFX, para produzir as animações necessárias.

A codificação foi realizada seguindo o paradigma de orientação a objetos e a estruturação do código segue o padrão de arquitetura de software MVC.

Os protótipos das telas foram desenvolvidos baseados em listas de exercícios utilizadas na disciplina de técnicas de programação avançada (TPA) e em materiais do livro Algoritmos, 4ª edição, dos autores Sedgewick e Wayne. Com isso foi possível adicionar ao protótipo elementos gráficos previamente pensados em auxiliar a aula, buscando focar os componentes mais importantes do algoritmo em execução.

3.2 ESCOLHA DOS ALGORITMOS

Foram escolhidos dois algoritmos para serem implementados na ferramenta, o Algoritmo de busca em largura (Breadth First Search - BFS) e o algoritmo de busca em profundidade (Depth First Search - DFS). A escolha desses algoritmos se deu baseada na ementa da disciplina de Técnicas de programação avançadas, do curso de Sistemas de Informação.

Esta disciplina aborda quatro algoritmos relacionados a teoria dos grafos, a saber: busca em largura (BFS), busca em profundidade (DFS), árvore geradora mínima e caminho mínimo. Os algoritmos BFS e DFS são base para os demais algoritmos citados. Além disso, podem servir de estratégias de caminamento em grafos nos mais diferentes domínios de problemas. Portanto, priorizou-se a implementações destes dois algoritmos.

Os dois algoritmos escolhidos possuem o mesmo objetivo que é encontrar, partindo de um vértice origem qualquer, todos os vértices acessíveis. Entretanto, cada um destes algoritmos utiliza estratégias distintas para a busca de solução, conforme será melhor detalhado nas próximas seções.

3.2.1 DFS

A principal característica desse algoritmo é sempre buscar o vértice mais profundo. Como forma de realizar esse tipo de exploração o algoritmo sempre avança no caminhar do grafo considerando uma aresta que conecta o vértice atual a um vértice ainda não visitado. Dessa maneira, a estratégia de caminhar só para de avançar ou "aprofundar" no grafo, quando o vértice explorado não possui arestas para outros vértices ainda não explorados. Um vértice é considerado finalizado quando todas as arestas adjacentes a ele estão conectadas a vértices já visitados. Para garantir que vértices não sejam visitados mais de uma vez, sempre que a execução chega em um vértice, este é marcado como visitado e adicionado a uma pilha.

O objetivo da pilha é garantir que o primeiro vértice visitado seja o último a ser finalizado. Esta estrutura é importante, pois a condição de desempilhamento no algoritmo é encontrar um vértice ligado apenas a vértices já visitados, logo, o primeiro vértice a ser finalizado deve ser o último encontrado. Ela também garante que quando o primeiro vértice encontrado for finalizado, o algoritmo concluiu sua execução (SOUZA et al., 2013).

A figura 9 apresenta um pseudocódigo do algoritmo DFS, no qual podemos observar que da linha 7 a linha 11 da primeira parte, temos um laço que verifica se cada vértice do grafo foi visitado, caso não tenha sido, visita-o.

```

1 DFS(GRAFO(V, E))
2 PARA todo vértice i do GRAFO FAÇA
3   cor[i] ← BRANCO
4   pai[i] ← nulo
5 FIM_PARA
6 Tempo ← 0
7 PARA todo vértice i do GRAFO FAÇA
8   Se cor[i] = BRANCO ENTÃO
9     DFS-VISITA(i)
10  FIM-SE
11 FIM_PARA

```

```

1 DFS-VISITA(u)
2 cor[u] ← CINZA
3 tempo ← tempo + 1
4 inicio[u] ← tempo
5 PARA todo vértice i de Adj(u) FAÇA
6   Se cor[i] = BRANCO ENTÃO
7     pai[i] ← u
8     DFS-VISITA(i)
9   FIM-SE
10 FIM_PARA

```

Figura 9 – Pseudocódigo do algoritmo DFS

Fonte: (SOUZA et al., 2013)

3.2.2 BFS

Diferente do DFS, o BFS explora os vértices de nível em nível. Este tipo de busca é caracterizado por visitar todos os vértices a uma distância k do vértice de origem antes de visitar os que estão a uma distância $k + 1$.

Aqui o conceito de vértice finalizado tem o mesmo significado do DFS. Ele acontece quando todas as suas arestas já foram visitadas. Nesse momento a exploração no nível k se encerra dando início a exploração no nível $k + 1$.

Para garantir a ordem de exploração dos vértices é utilizada uma estrutura de fila. Esta garante que o primeiro vértice a ser visitado é também o primeiro a ser finalizado. Ao ser visitado o vértice é adicionado na fila. Quando é finalizado é retirado da fila. A figura 10 apresenta um pseudocódigo do algoritmo BFS. Na linha 16 realiza-se a contagem de distância do vértice encontrado para o vértice de origem, somando a distância do vértice anterior mais um.

```

1 BFS (GRAFO(V, E), v)
2  PARA todo vértice i do GRAFO FAÇA
3    cor[i] ← BRANCO
4    d_arestas[i] ← ∞
5    pai[i] ← nulo
6  FIM_PARA
7  d_arestas[v] ← 0
8  cor[v] ← CINZA
9  FILA ← ∅
10 INSERE(FILA, v)
11 ENQUANTO FILA ≠ ∅ FAÇA
12   u ← REMOVE(FILA)
13   PARA todo vértice i de Adj(u) FAÇA
14     Se cor[i] = BRANCO ENTÃO
15       cor[i] ← CINZA
16       d_arestas[i] ← d_arestas[u] + 1
17       pai[i] ← u
18       INSERE(FILA, i)
19   FIM_SE
20   FIM_PARA
21   cor[u] ← PRETO
22 FIM_ENQUANTO

```

Figura 10 – Pseudocódigo do algoritmo BFS

Fonte: (SOUZA et al., 2013)

4 RESULTADOS

Baseando-se nos conceitos apresentados na disciplina de TPA, e visando atender suas demandas, os seguintes artefatos foram desenvolvidos:

4.1 MODELAGEM

As funções disponibilizadas pela aplicação permitem ao usuário manusear a exploração do algoritmo de diversas formas. A figura 12 apresenta o diagrama de caso de uso da ferramenta.

Figura 11 – Diagrama de caso de uso

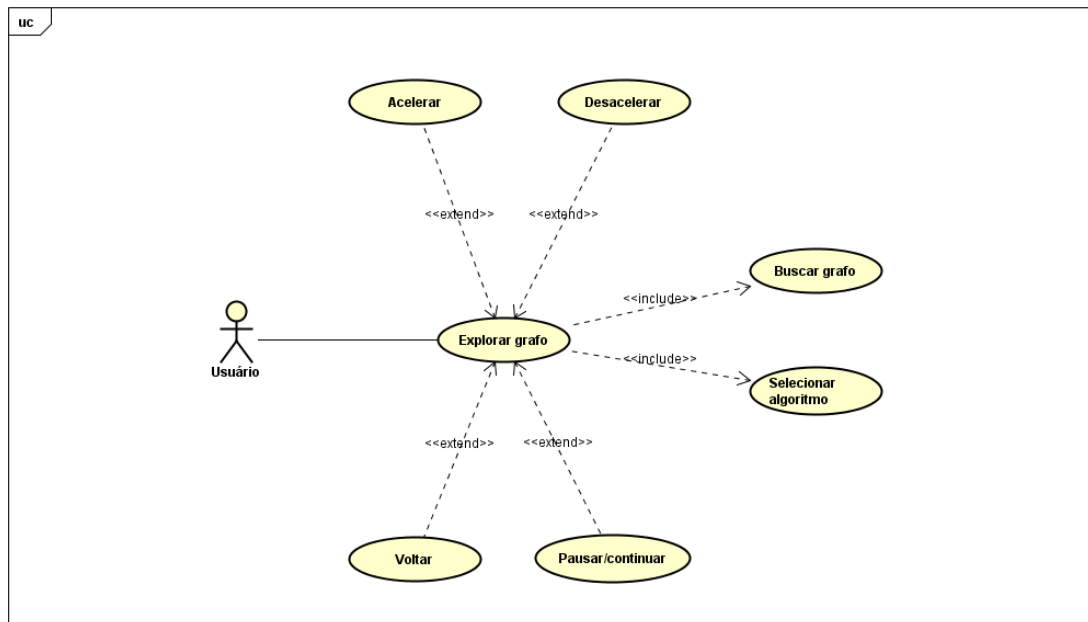


Figura 12 – Diagrama de caso de uso

Fonte: Produzido pelo próprio autor

Conforme verificado na figura 12, a ação do usuário gira em torno de Explorar grafo, que pode estender outras operações, acelerar e desacelerar a execução, pausar ou continuar e voltar para a tela anterior. Outro fator importante é que explorar grafo exige que um grafo e um algoritmo tenham sido escolhidos.

4.2 CÓDIGO

A figura 13 apresenta como a estrutura do código esta dividida. Ela pode ser resumida pelas classes Grafo, Vertice, Aresta, AlgoritmoBFS, AlgoritmoDFS, RunnableBFS e

RunnableDFS. Onde:

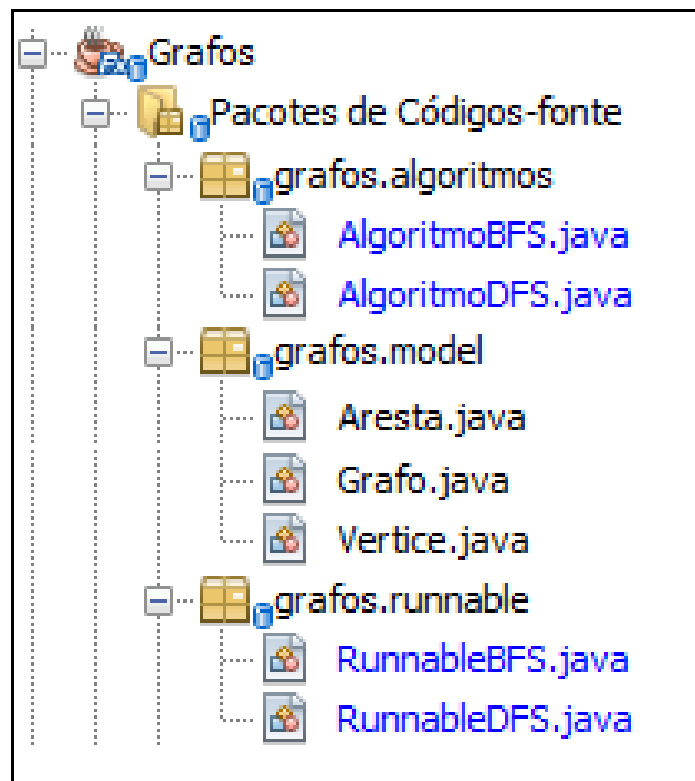


Figura 13 – Estrutura do código

Fonte: Produzido pelo próprio autor

- A classe *Grafo* é a representação computacional do elemento grafo. Contendo os atributos necessários para possibilitar aos algoritmos a sua exploração, como quantidades de vértices e arestas e como se dá suas disposições, através da lista de adjacências.
- A classe *Vertice* armazena os dados sobre os vértices, como a sua identificação e a sua posição.
- A classe *Aresta* possui como principais informações, os identificadores do vértice de origem e destino.
- A classe *AlgoritmoBFS* consiste em uma representação do código, do método de busca em largura BFS, sendo esta, efetivamente, a classe que explora os grafos.

- A classe *AlgoritmoDFS* consiste em uma representação do código, do método de busca em profundidade DFS, sendo esta, efetivamente, a classe que explora os grafos.
- A classe classe *RunnableBFS* implementa a interface *Runnable*, possibilitando assim que suas instâncias possam ser enviadas para o construtor de uma classe *Thread*. Esta *Thread* cuida das modificações na parte gráfica da aplicação, que são acionadas à partir da execução das instruções da instância da classe *AlgoritmoBFS*.
- A classe classe *RunnableDFS* também implementa a interface *Runnable*. Esta *Thread* cuida das modificações na parte gráfica da aplicação, que são acionadas à partir da execução das instruções da instância da classe *AlgoritmoDFS*.

4.2.1 Funcionamento do código

Devido ao fato do JavaFX não ser Thread-Safe¹, algumas medidas tiveram que ser tomadas para garantir o bom funcionamento da ferramenta.

Ao carregar o arquivo .txt que contém os dados do grafo escolhido, o desenho do mesmo é gerado na tela e instâncias das classes *Aresta*, *Vertice* e *Grafo*, são utilizadas para criar a estrutura computacional que representa o grafo escolhido pelo usuário. O grafo é representado internamente na aplicação através de uma lista de adjacências *vértice-vértice*, possibilitando assim a sua exploração pelo algoritmo.

Ao clicar em confirmar na primeira tela, o usuário inicia uma *Thread* passando como parâmetro a instância da classe *RunnableDFS/RunnableBFS*. No seu método *run*², uma instância da classe *AlgoritmoBFS/AlgoritmoDFS* é criada. Nesta classe o seu método construtor inicia a função *DFS()/BFS()*, que é a responsável por explorar a estrutura do grafo. Neste ponto a aplicação possui duas *Threads* ativas: a que acabou de ser criada e a JavaFX Application Thread.³

A partir de agora, cada linha de execução do método *DFS()/BFS()*, pode gerar uma alteração na interface gráfica, como colorir e/ou preencher elementos. É neste ponto que as *Threads* que implementam a *RunnableDFS/RunnableBFS* são realmente

¹ Capacidade de manipular estruturas de dados compartilhadas de uma forma que garanta uma execução segura através de várias threads ao mesmo tempo

² Método da interface *Runnable* que é executado quando a *Thread* é iniciada

³ *Thread* do JavaFX responsável pelas alterações na interface gráfica

necessárias, já que cada uma dessas linhas podem desencadear várias mudanças na interface, alterações estas aplicadas simultaneamente.

Um outro ponto específico do código que exige a existência dessas Threads é a coloração das arestas. Quando a coloração de uma aresta é solicitada, uma nova Thread implementando uma *Task*⁴ é iniciada. Neste ponto da execução temos três threads ativas: Uma que utiliza a implementação da classe *RunnableDFS/RunnableBFS*, a JavaFX Application Thread e a que acabamos de criar para colorir as arestas.

O objetivo da terceira thread, é somente representar a execução da linha de código, que indica a visitação de uma aresta. Enquanto o algoritmo esta visitando uma aresta, nenhuma outra instrução é executada, logo, nenhuma alteração nos elementos gráficos deve ser feita. Para que isso seja garantido a thread que implementa a classe *RunnableDFS/RunnableBFS* deve parar e esperar o fim da execução da terceira thread, para retomar seu trabalho. Para isso o método *join()* da classe Thread é utilizado.

4.3 APLICAÇÃO

O principal objetivo da ferramenta é preencher as lacunas deixadas por softwares similares. A maior delas é a apresentação do código fonte do algoritmo que o usuário escolheu. A ferramenta proposta irá proporcionar um modo debug de execução , onde a linha que está sendo executada ficará em destaque. A seguir serão apresentadas as telas da aplicação EasyGrafos, demonstrando os principais estados da ferramenta durante sua execução.

4.3.1 Tela Inicial

A figura 14 apresenta a primeira tela da ferramenta. Nela o usuário irá preencher os parâmetros necessários para utilização da ferramenta. Nesta etapa o usuário deve escolher um grafo para ser explorado e um algoritmo para explorá-lo. Após, os parâmetros necessários para execução do algoritmo serão solicitados para que a função confirmar possa ser acionada.

⁴ Implementação da classe *javafx.concurrent*. Assim como a *Runnable* possibilita execuções assíncronas

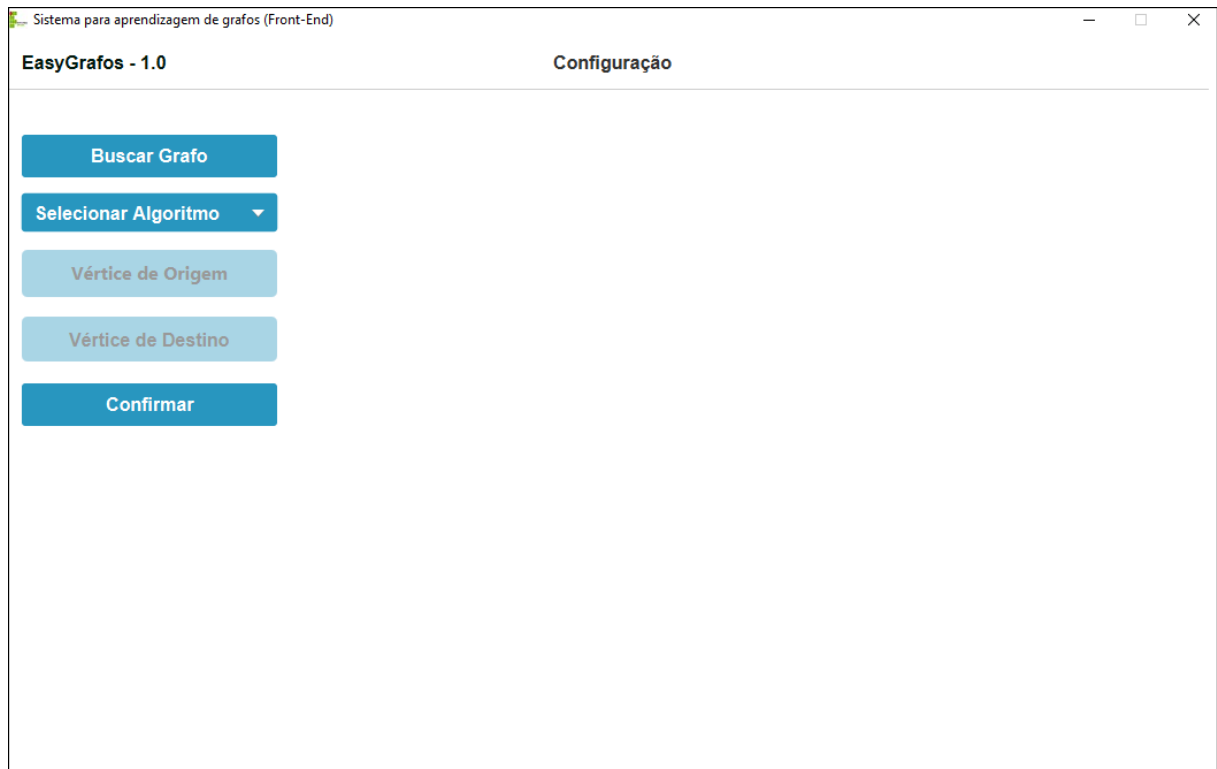


Figura 14 – Tela inicial

Fonte: Próprio autor

4.3.2 Coloração dos elementos

A compressão do funcionamento da ferramenta e dos resultados apresentados por ela, estão diretamente ligados a coloração atribuída aos elementos gráficos. A figura 15 apresenta uma legenda com as cores de cada status destes elementos.






Status		Aresta	Vértice
Branco		Não aplicável	Não visitado
Cinza		Descartada	Descartado
Preto		Não visitada	Não aplicável
Verde Claro		Visitada	Visitado
Verde Escuro		Visitando	Visitando

Figura 15 – Legenda de cor dos status

Fonte: Próprio autor

Os status atingidos pelos elementos durante a execução do algoritmo, podem ser explicados da seguinte forma:

- Não aplicável : O elemento nunca é representado com esta cor.

- Não visitado : Indica que o algoritmo ainda não passou por este elemento.
- Visitando : Indica que a execução do algoritmo está verificando este elemento neste exato momento.
- Visitado : Indica que a execução do algoritmo já passou por este elemento, porém, passará por ele novamente.
- Descartado : Indica que o algoritmo já realizou todas as verificações necessárias neste elemento.

4.3.3 Algoritmo BFS e DFS

A seguir serão apresentadas quatro figuras no total, sendo duas para cada algoritmo. Serão, portando, duas ilustrando a tela da aplicação durante a execução e duas após o término da execução. As figuras 16 e 17 apresentam respectivamente as telas da ferramenta durante a execução dos algoritmos BFS e DFS.

Código Fonte

```
public class AlgoritmoBFSGrafo {
    private boolean[] marcado;
    private int[] arestaPara;
    private int[] distPara;
    private int vo;

    public AlgoritmoBFSGrafo(Grafo G, int vo){
        this.vo = vo;
        arestaPara = new int[G.V()];
        marcado = new boolean[G.V()];
        distPara = new Integer[G.V()];
        bfs(G, vo);
    }

    private void bfs(Grafo G, int vo) {
        Fila<Integer> f = new Fila<Integer>();
        f.enfileira(vo);
        marcado[vo] = true;
        distPara[vo] = 0;
        while (!f.isEmpty()) {
            int v = f.desenfileira();
            for (Aresta a : G.adj(v)) {
                int x = a.getV2();
                if (!marcado[x]) {
                    arestaPara[x] = v;
                    distPara[x] = distPara[v] + 1;
                    marcado[x] = true;
                    f.enfileira(x);
                }
            }
        }
    }
}
```

Execução do Algoritmo Busca em Largura (BFS)

Explorando vértice 3

Lista de Adjacências

0	-- 3 - 2 - 1
1	-- 6 - 3 - 0
2	-- 4 - 3 - 0
3	-- 6 - 5 - 4 - 2 - 1 - 0
4	-- 6 - 5 - 3 - 2
5	-- 6 - 4 - 3
6	-- 5 - 4 - 3 - 1

Fila

2
1

Atributos do algoritmo

v	marcado[v]	arestaPara[v]	distPara[v]
0	TRUE	--	0
1	TRUE	0	1
2	TRUE	0	1
3	TRUE	0	1
4	FALSE	--	-1
5	FALSE	--	-1
6	FALSE	--	-1

Variáveis Locais

Váriável v: 3
Variável x: --

Figura 16 – Tela do algoritmo BFS em execução

Fonte: Próprio autor

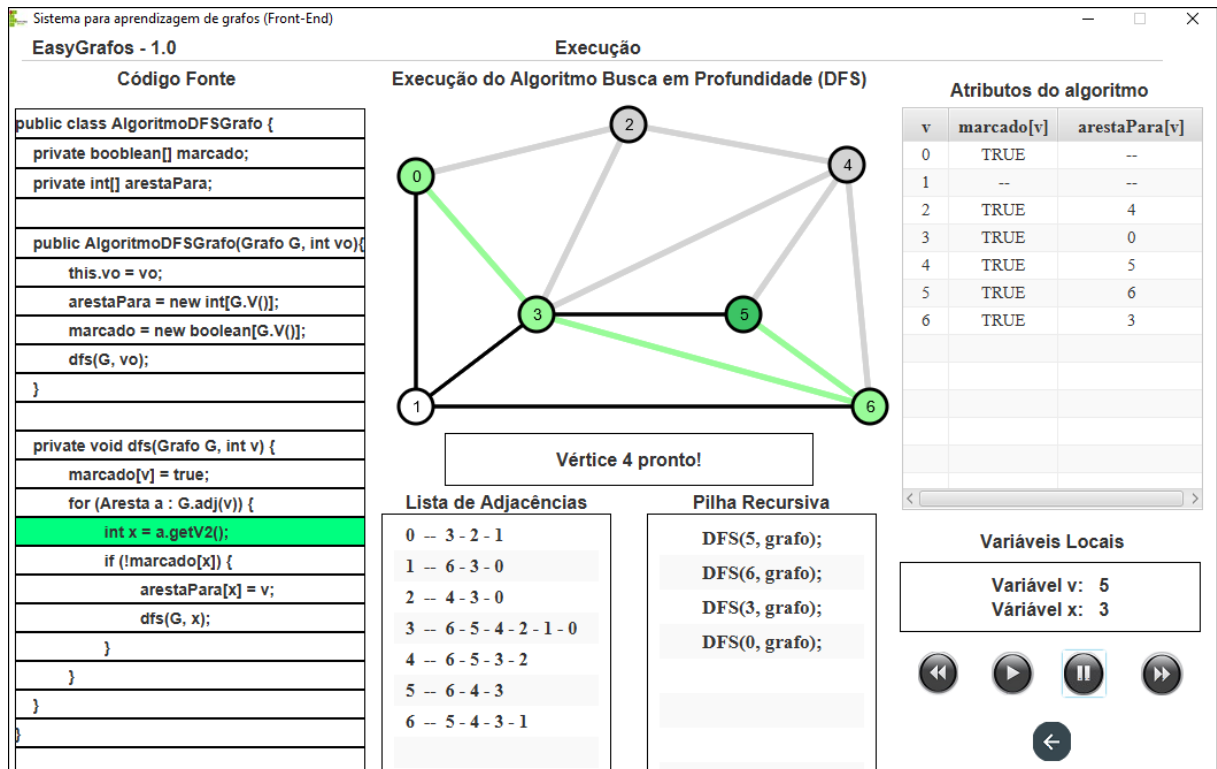


Figura 17 – Tela do algoritmo DFS em execução

Fonte: Próprio autor

É importante observar que existe uma linha de código colorida de verde em ambas as telas. Esta representa a instrução que está sendo executada no momento. Ainda nesta etapa outras ações podem ocorrer, como: a mudança no desenho do grafo (referente as cores), bem como, a alteração nos valores das variáveis dispostas na tabela do lado direito e no retângulo abaixo desta. A principal diferença que pode ser notada entre as telas é que na figura 17 temos a exibição de uma pilha de chamadas recursivas, enquanto na figura 16 temos a exibição de uma fila de vértices a serem visitados, ambas abaixo do desenho do grafo.

O outro ponto são os controles disponibilizados no canto inferior direito da tela. Com eles o usuário poderá parar e continuar a execução da ferramenta, acelerar e desacelerar a execução e retornar para a tela anterior. O principal objetivo neste ponto é disponibilizar ao usuário da ferramenta, mecanismos de controle do ritmo de execução do programa, facilitando o entendimento.

A seguir as figuras 18 e 19 ilustram respectivamente o resultado final da execução dos algoritmos BFS e DFS.

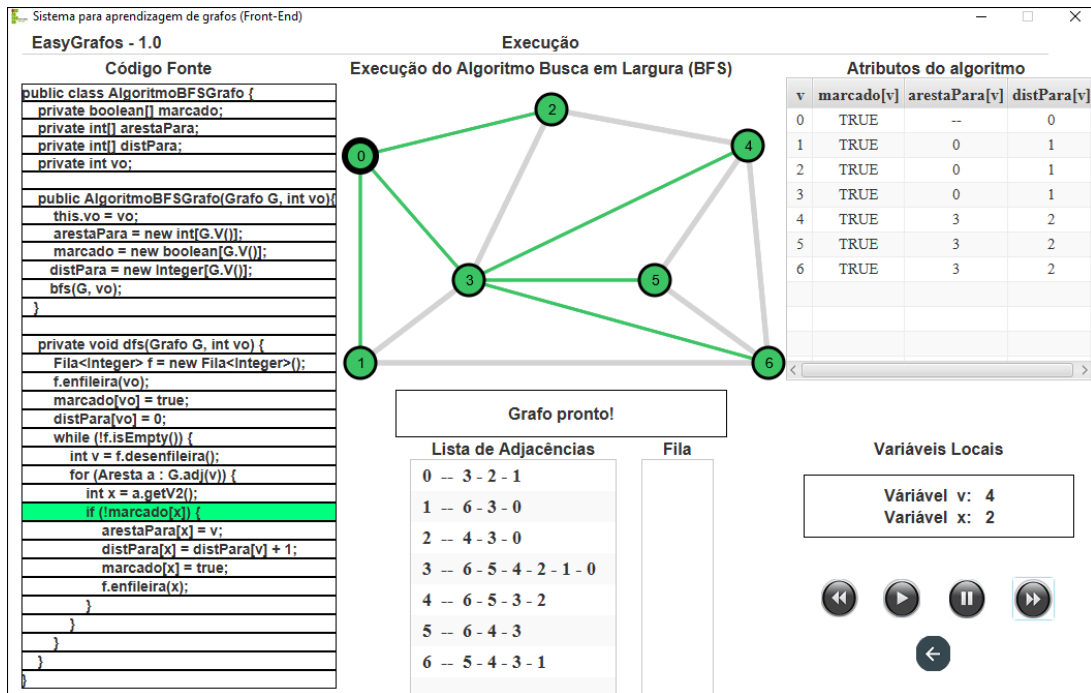


Figura 18 – Tela final do algoritmo BFS

Fonte: Próprio autor

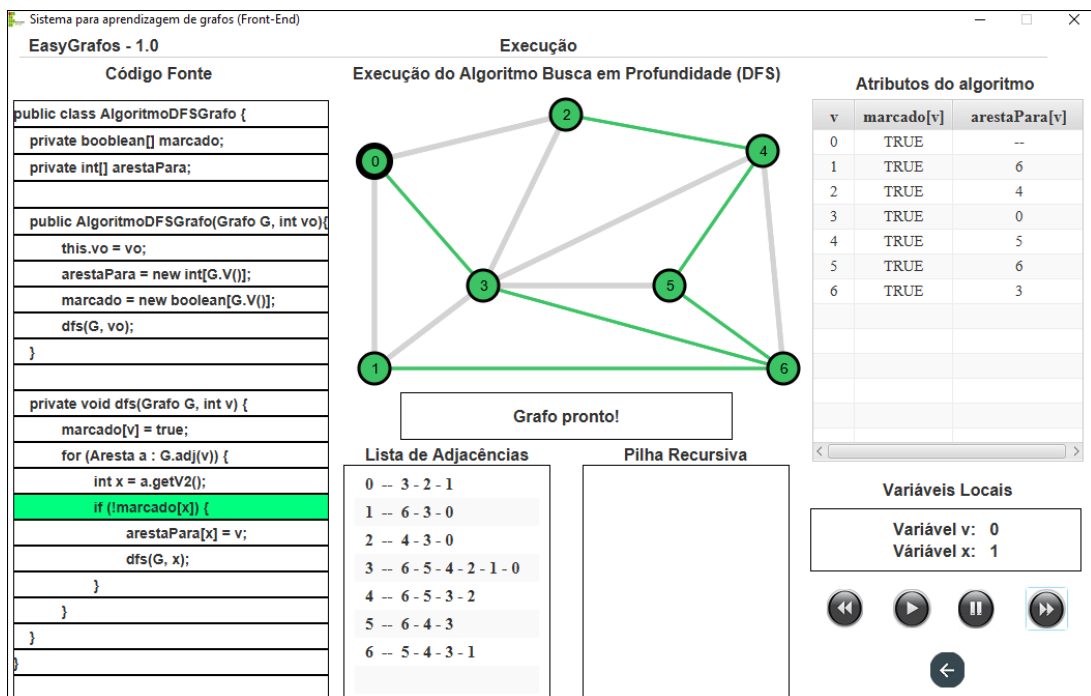


Figura 19 – Tela final do algoritmo DFS

Fonte: Próprio autor

As figuras 18 e 19 apresentam os vértices acessíveis à partir de um dado vértice de origem, neste caso o vértice 0 (colorido com borda mais grossa). Ambas tem o mesmo resultado final, no entanto, o caminhamento no grafo é realizado de maneira diferente durante a execução. Este fato pode ser observado nas figuras 16 e 17, ao perceber

que os vértices que são descartados primeiro (coloração cinza), não são os mesmos. Assim como, nas figuras 18 e 19, onde as arestas coloridas não são as mesmas.

4.4 VALIDAÇÃO

A ferramenta EasyGrafos foi avaliada baseando-se no formulário de validação de software educacional desenvolvido por Thomas Reeves⁵, aplicando 5 das 14 questões pedagógicas e 6 das 10 questões de interface gráfica.

Cada questão consiste em um atributo do software que pode ser pontuado de 0 a 10. A nota é associada a duas características, uma é negativa, ou seja informa que este atributo não foi implementado satisfatoriamente, e outra positiva, informando que o software implementou de forma satisfatória este atributo. Quanto mais próxima do 0 a nota esta, mais associada a característica negativa esta a implementação deste atributo e quanto mais próxima do 10, mais associada a característica positiva esta a implementação deste atributo.

4.5 QUESTIONÁRIO APLICADO

A seguir serão listadas cada uma das questões do formulário utilizado⁶, sendo o primeiro atributo em negrito a característica associada a nota 0 e a segunda associada a nota 10.

4.5.1 Atributos Pedagógicos

1. Filosofia Pedagógica:

Instrutivista é aquela que enfatiza a importância de metas independentes do aluno, embasada na teoria comportamentalista.

Construtivista enfatiza a intenção, a estratégia, a experiência do aluno, sendo ele visto como um indivíduo detentor de conhecimentos e motivações.

⁵ Professor emérito em Aprendizagem, Design e Tecnologia pela Universidade de Georgia, EUA. Página pessoal disponível em: <http://treeves.coe.uga.edu/edit8350/UIRF.html>.

⁶ https://docs.google.com/forms/d/e/1FAIpQLSfDvTgPTK-HSh0lnOSr7D9wdv2HzES6k9pMP0cUF8lcZj-bsQ/viewform?usp=sf_link.

2. Psicologia Subjacente:

Comportamental significa que os fatores de aprendizado não são comportamentos que podem ser diretamente observados (comportamento desejável obtido através de estímulo-resposta).

Cognitiva é a psicologia que dá ênfase aos estados mentais internos, reconhece que uma ampla variedade de estratégias de aprendizagem deve ser empregada.

3. Objetividade:

Precisamente focalizado é a forma utilizada nos tutoriais e treinamentos.

Não focalizado é a forma empregada nas simulações virtuais e ambientes de aprendizado.

4. O papel do instrutor:

Provedor de materiais, o instrutor é considerado o detentor do conhecimento.

Agente facilitador, o instrutor é uma fonte de orientação e consulta.

5. Aprendizado Cooperativo:

Não suportado, quando não permite trabalho em pares ou grupos.

Integral, quando permite o trabalho cooperativo, fazendo com que os objetivos sejam compartilhados.

4.5.2 Atributos de interface gráfica

1. Facilidade de Utilização:

Difícil ou **Fácil** do ponto de vista do programa ser de fácil entendimento.

2. Navegação:

Difícil ou **Fácil**, refere-se ao ato de ir de um tópico a outro dentro do programa.

3. Compatibilidade Espacial do Conhecimento:

Incompatível ou **Compatível**, verifica a compatibilidade do sistema com as expectativas e necessidades do usuário em sua tarefa (informação apresentada é compatível com o nível de conhecimento do usuário).

4. Apresentação da informação:

Confusa ou **Clara**, verifica se a a forma como a informação foi apresentada, possibilita sua compreensão.

5. Estética:

Desagradável ou **Agradável**, quanto a beleza e elegância.

6. Funcionalidade Geral:

Não funcional ou **Altamente funcional**, quanto a utilidade percebida do programa, que é altamente relacionada com o objetivo de uso.

4.6 PRINCIPAIS INFERÊNCIAS

Um formulário com as questões descritas na seção 5.1, foi aplicado a turma do 6º período de Sistemas de Informação, do Ifes campus Cachoeiro de Itapemirim, no segundo semestre de 2018, após uma aula da disciplina de TPA, utilizando a ferramenta EasyGrafos. A turma é composta de 6 alunos e todos responderam ao questionário. A figura 20 apresenta um gráfico com a média de cada quesito avaliado.

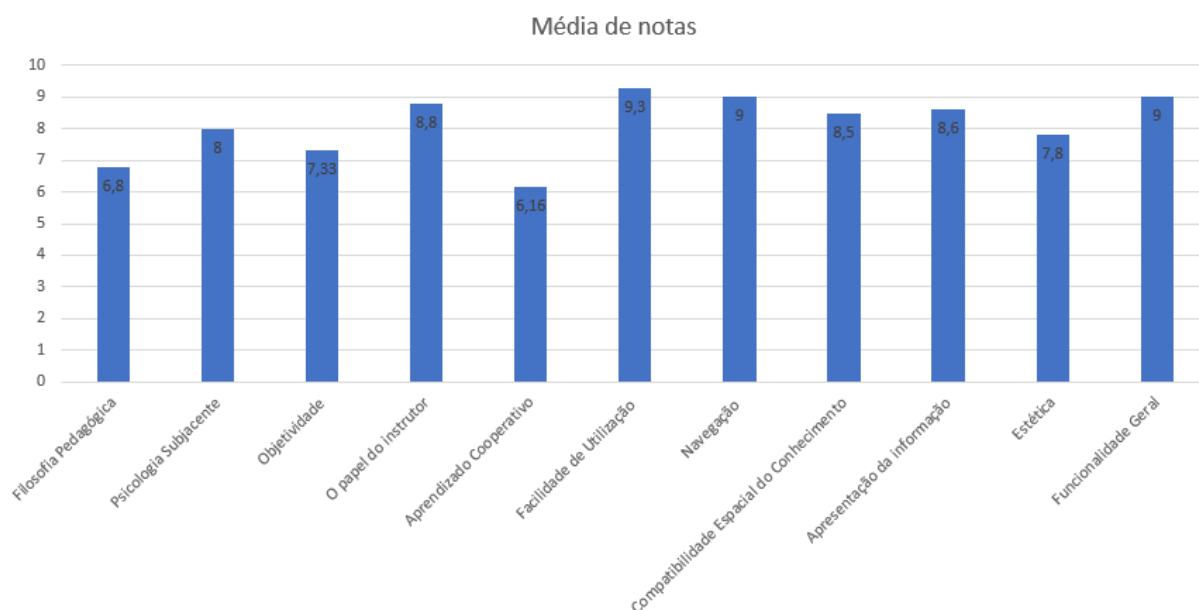


Figura 20 – Média geral das notas

Fonte: Elaborado pelo próprio autor

Observando os resultados apresentados, pode-se verificar que nenhum quesito foi

implementado de forma insatisfatória, já que, todas as médias ultrapassaram a nota 6. É necessário enfatizar que o atributo aprendizado cooperativo recebeu as notas mais baixas, o que representa uma necessidade de desenvolver mecanismos que integrem os alunos da sala e possibilite que estes interajam entre si durante os exemplos da ferramenta. Em contrapartida, o atributo facilidade de utilização recebeu média de 9,3, demonstrando que a ferramenta proporciona um ambiente de simples entendimento, oferecendo poucos ou nenhum empecilho ao aluno, quanto a facilidade de utilização ou de compreensão de como a ferramenta funciona.

A figura 21 apresenta um gráfico comparativo entre os dois módulos do questionário, o referente aos atributos pedagógicos e o referente aos atributos de interface gráfica.

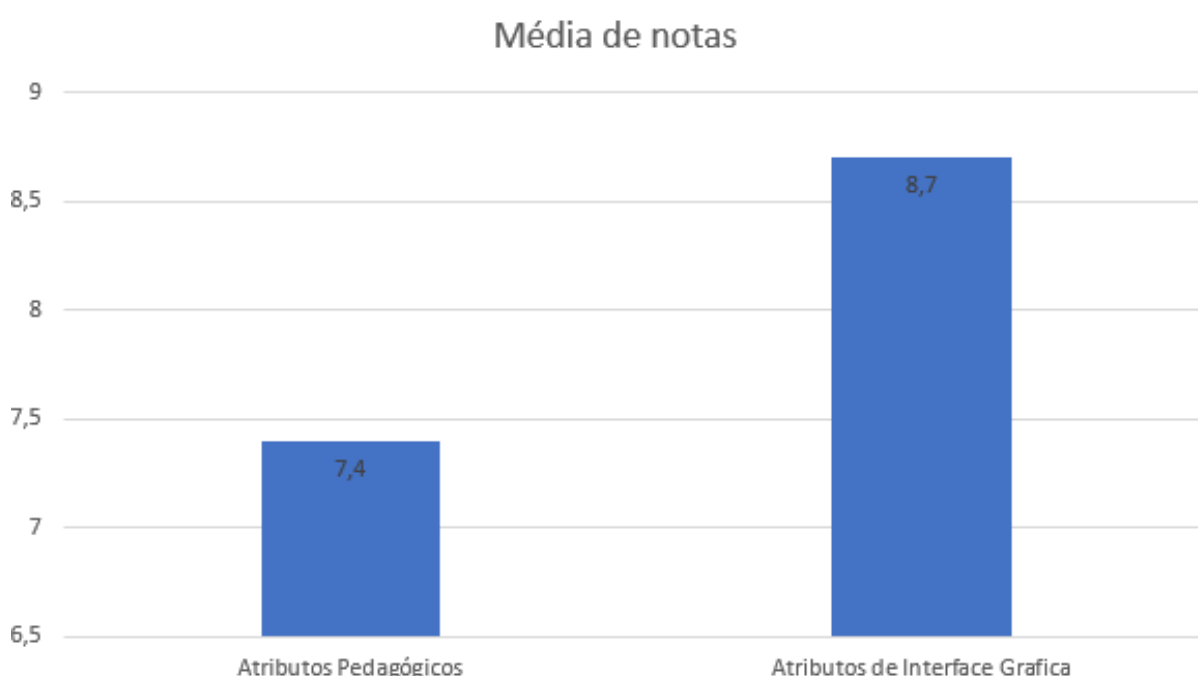


Figura 21 – Média dos módulos do formulário

Fonte: Elaborado pelo próprio autor

Analisando as duas médias podemos observar que tanto a parte pedagógica quanto a parte de interface da ferramenta EasyGrafos, atingiu resultados satisfatórios nas avaliações. Visto que ambas ultrapassaram a nota 7 de média, destacando-se os atributos de interface gráfica que passaram de 8,5.

5 CONCLUSÃO

O presente trabalho proporcionou o desenvolvimento de um software educacional intitulado EasyGrafos, que auxilia a aprendizagem da teoria dos grafos em disciplinas de computação, se utilizando de componentes gráficos para proporcionar ao aluno o grau de abstração necessário para compreender os conceitos da disciplina.

Um estudo comparativo possibilitou levantar as lacunas deixadas pelas ferramentas similares disponíveis. A maior delas é não apresentar o código fonte do algoritmo integrado ao desenho do grafo e suas modificações em tempo de execução.

A aplicação possibilitou apresentar ao aluno de forma simultânea todas as alterações que o algoritmo exerce no grafo durante sua execução, através de colorações aplicadas nas arestas e nos vértices. Também é possível analisar o código do algoritmo enquanto essas alterações são feitas, já que os elementos gráficos que os representam estão dispostos lado a lado na tela. Outra funcionalidade disponibilizada são os controles de execução. Estes geram um modo debug na ferramenta, possibilitando ao usuário, parar ou acelerar e desacelerar a execução do algoritmo. Estes ambientes foram implementados para dois algoritmos clássicos da teoria dos grafos, o algoritmo de busca em Largura (BFS) e o algoritmo de busca em profundidade (DFS).

Um questionário avaliativo baseado no método de avaliação de software educacional desenvolvido por Thomas Reeves, foi aplicado a alunos da disciplina de TPA, do curso de Sistemas de Informação, do Ifes campus cachoeiro de Itapemirim, e os resultados puderam comprovar que a aplicação satisfaz os objetivos propostos pela mesma.

Como trabalhos futuros, analisando a ementa da disciplina de TPA do curso de Sistemas de Informação, verificou-se a possibilidade de implementar os algoritmos de árvore geradora mínima e caminho mínimo, que são os outros conceitos da teoria dos grafos abordados na disciplina.

REFERÊNCIAS

- ALMEIDA, T. A. d. et al. Computação evolutiva aplicada a resolução do problema da árvore geradora mínima com parâmetros fuzzy. [sn], 2006.
- BRAGA, M. J. d. C.; GOMES, L. F. A. M.; RUEDIGER, M. A. Mundos pequenos, produção acadêmica e grafos de colaboração: um estudo de caso dos enanpads. *Revista de Administração Pública-RAP*, SciELO Public Health, v. 42, n. 1, 2008.
- COSTA, P. P. d. Teoria dos grafos e suas aplicações. Universidade Estadual Paulista (UNESP), 2011.
- FIGUEIREDO, R. T.; FIGUEIREDO, C. Wargrafos—jogo para auxílio na aprendizagem da disciplina de teoria dos grafos. *X Simpósio Brasileiro de Games e Entretenimento Digital (SBGames 2011)*, 2011.
- GIRAFFA, L. M. M. Uma odisséia no ciberespaço: O software educacional dos tutoriais aos mundos virtuais. *Brazilian Journal of Computers in Education*, v. 17, n. 01, p. 20, 2009.
- INEP. *Resumo técnico censo da educação superior 2017*. 2017. Disponível em <http://download.inep.gov.br/educacao_superior/censo_superior/documentos/2018/censo_da_educacao_superio>
- JURKIEWICZ, S. Grafos—uma introdução. *Programa de Iniciação Científica da*, 2009.
- LOZADA, L. A. P. A-graph: Uma ferramenta computacional de suporte para o ensino-aprendizado da disciplina teoria dos grafos e seus algoritmos. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2014. v. 3, n. 1, p. 61.
- MELO, V. A. de; SILVEIRA, D. S. da; JURKIEWICZ, S. Teoria de grafos: Uma proposta de objeto de aprendizagem para o modelo ead. *XLI SBPO 2009 - Pesquisa Operacional na Gestão do Conhecimento*, 2009.
- MORAN, J. M. *Novas tecnologias e mediação pedagógica*. [S.l.]: Papyrus Editora, 2000.
- MORATORI, P. B. Por que utilizar jogos educativos no processo de ensino aprendizagem. *UFRJ. Rio de Janeiro*, 2003.
- RECUERO, R. da C. Teoria das redes e redes sociais na internet: considerações sobre o orkut, os weblogs e os fotologs. In: *XXVII Congresso Brasileiro de Ciências da Comunicação ao. XXVII INTERCOM*. [S.l.: s.n.], 2004.
- SANTOS, R. P. et al. O uso de ambientes gráficos para ensino e aprendizagem de estruturas de dados e de algoritmos em grafos. In: *Anais do XVI Workshop sobre Educação em Computação, XXVIII Congresso da Sociedade Brasileira de Computação*. [S.l.: s.n.], 2008. p. 157–166.
- SILVA, K. M. F. d. S. O uso de novas tecnologias em sala de aula: um estudo de caso. 2014.

SILVEIRA, E. B. de A.; SILVA, M. O. da. Desenvolvimento de um aplicativo educacional para o estudo de teoria dos grafos. 2016.

SOUZA, A. L. et al. Teoria dos grafos e aplicações. Universidade Federal do Amazonas, 2013.

TEIXEIRA, A. C. Software educacional: o difícil começo. *RENOTE*, v. 1, n. 1, 2003.

VALENTE, J. A. Análise dos diferentes tipos de softwares usados na educação. *O computador na sociedade do conhecimento*, Gráfica da UNICAMP Campinas^ eSP SP, p. 71, 1999.