

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

Jonathas Gonçalves Picoli¹

Rafael Vargas Mesquita dos Santos¹

Thiago Machado Mendes¹

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.

DEVELOPMENT OF AN EDUCATIONAL SOFTWARE FOR AID IN LEARNING THE GRAPH THEORY IN DISCIPLINES OF COMPUTER ABSTRACT

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

¹ Instituto Federal do Espírito Santo, Campus Cachoeiro de Itapemirim, Cachoeiro de Itapemirim-ES.

* Autor para correspondência: jonathasgoncalvespicoli@gmail.com

Manuscrito submetido em: xx / xx / xxxx

Aceito para publicação em: xx / xx / xxxx

Revista Ifes Ciência, v.xx, n.xx, 20xx – Instituto Federal do Espírito Santo

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 a score above 6 in all the evaluated questions.

Keywords: Theory of graphs; Education; Software.

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.

A teoria dos Grafos

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 $A(G)$.

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).

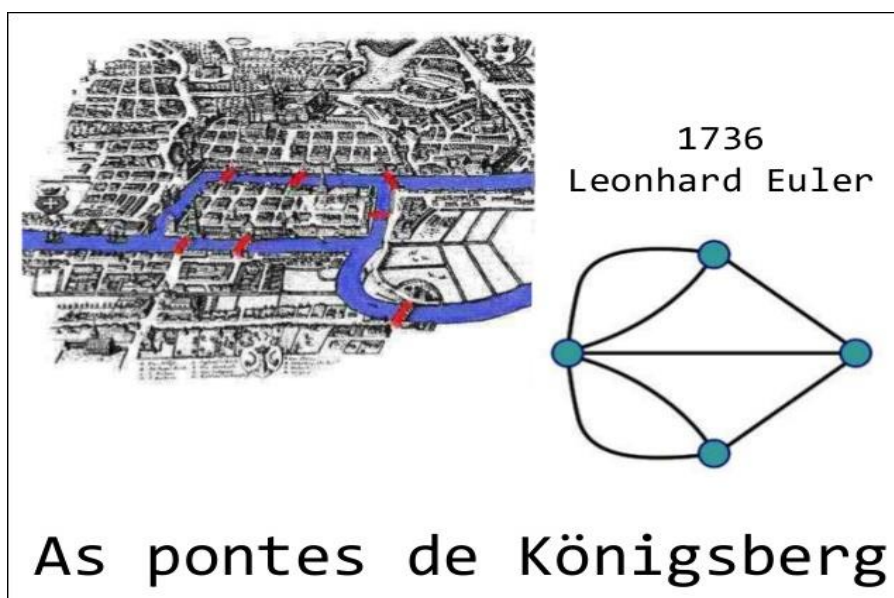


Figura 1 - Abstração do problema das pontes de Königsberg

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

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 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

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 selecionadas 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 2 apresenta a tela do resultado final da execução de um algoritmo na aplicação AlgoDeGrafos.

¹ <https://www.google.com>

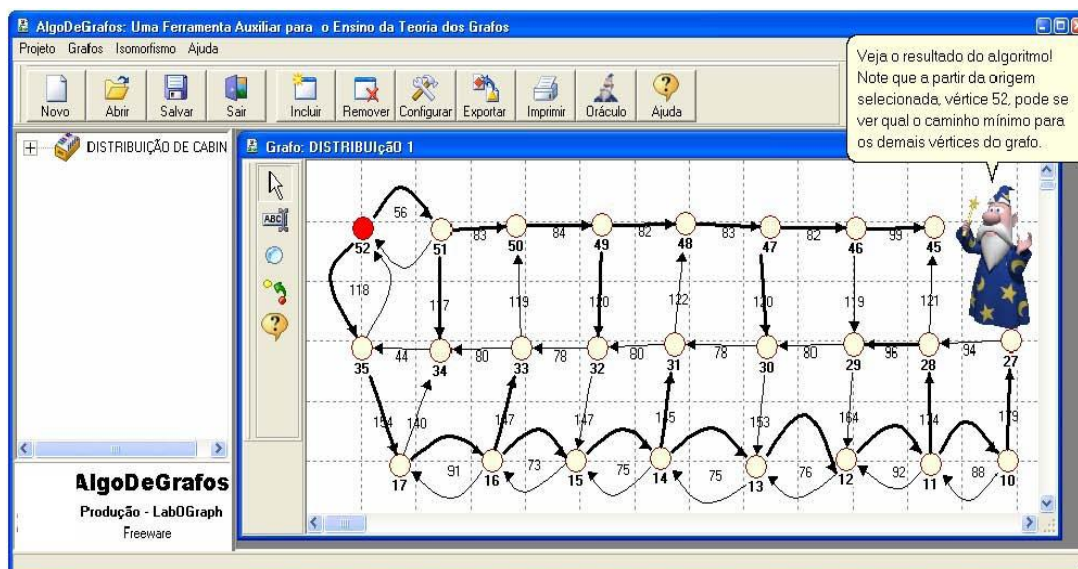


Figura 2 – Tela de resultado da aplicação AlgoDeGrafos

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 grafos (FIGUEIREDO; FIGUEIREDO, 2011). A figura 3 apresenta a tela durante uma partida do jogo.

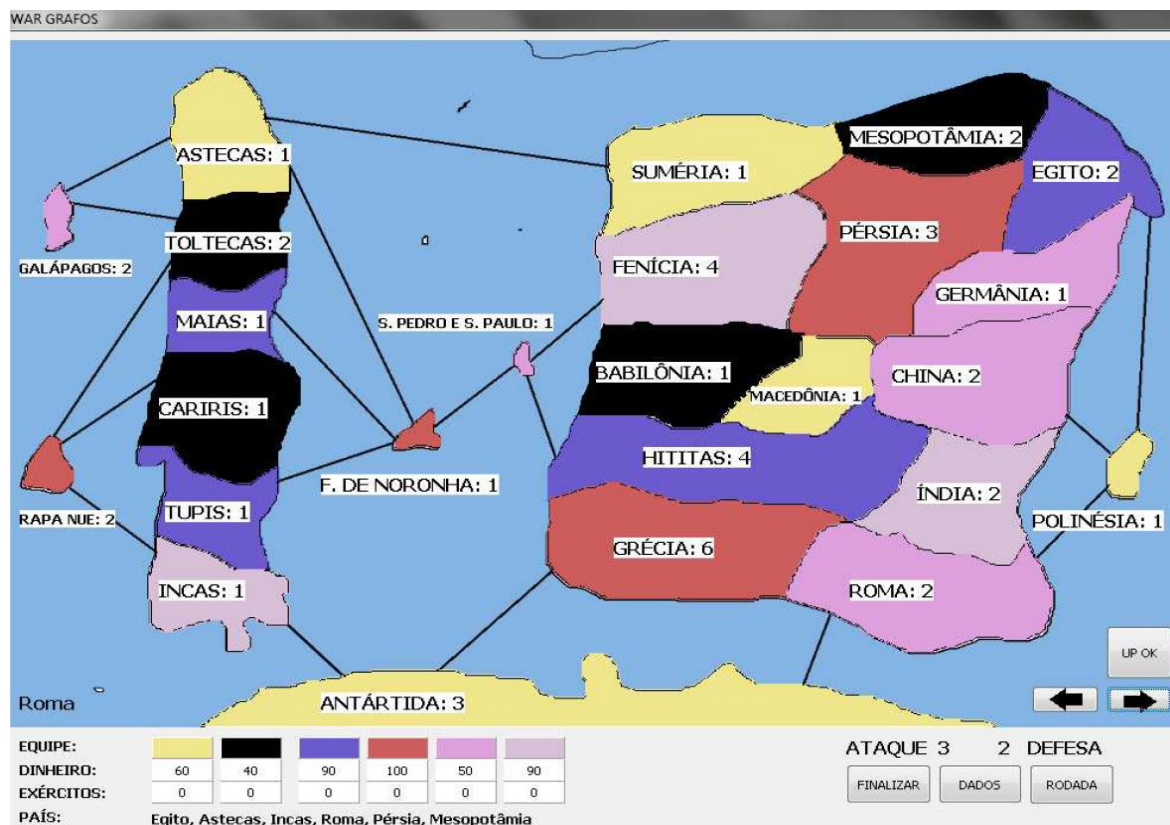


Figura 3 – Tela do jogo WarGrafos durante uma partida

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 4 apresenta um exemplo de execução do algoritmo Dijkstra na aplicação TBC-GRAFOS.

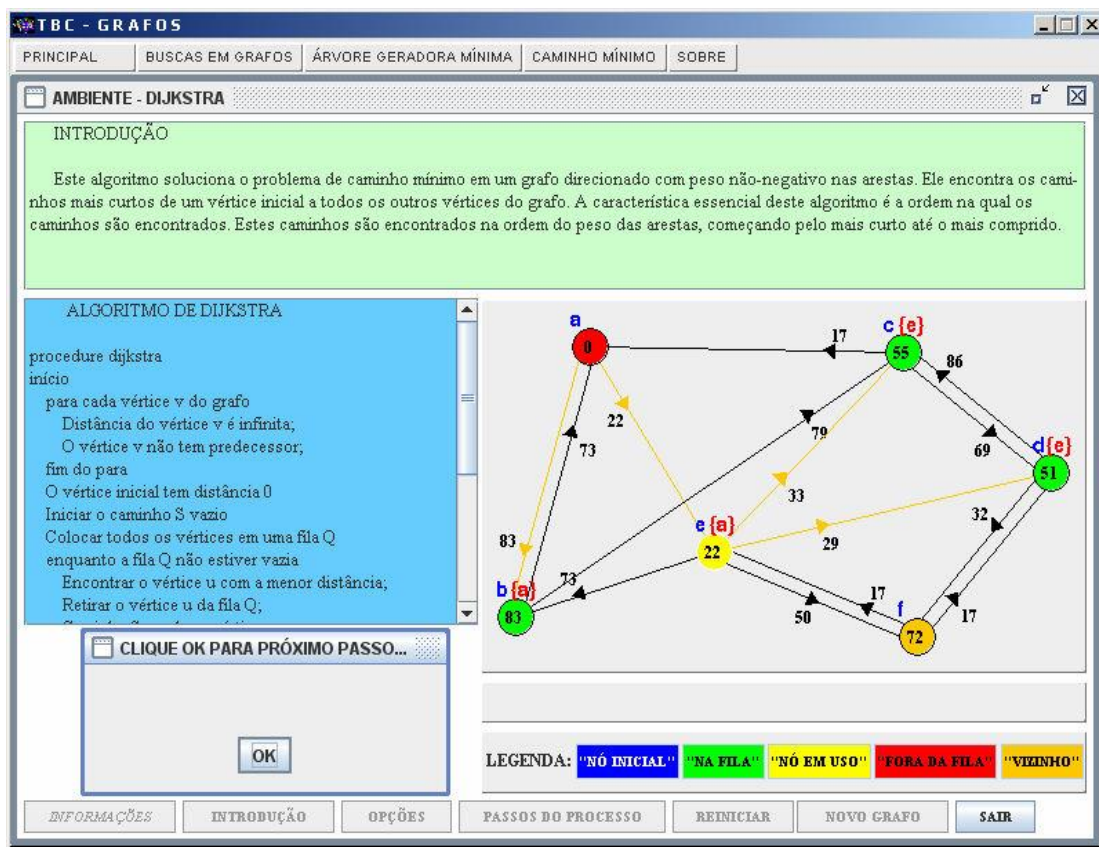


Figura 4 – Tela durante a execução do algoritmo Dijkstra

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 5 ilustra um exemplo de execução na ferramenta.

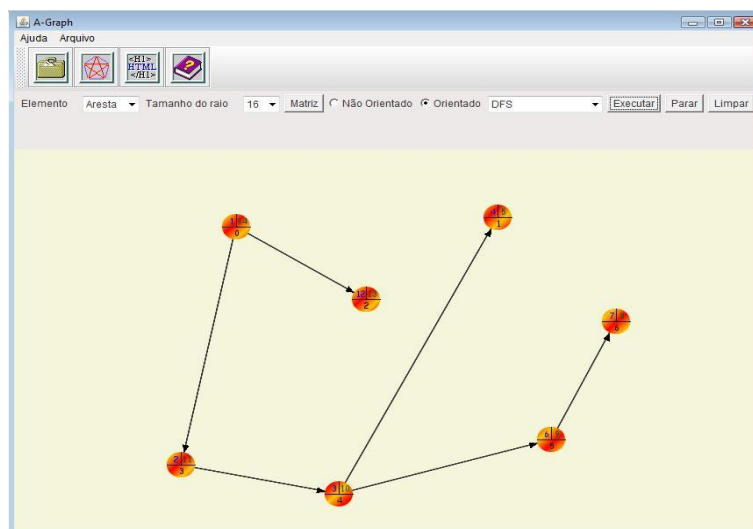


Figura 5 – Tela de resultado da aplicação A-Graph

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 6 apresenta a tela principal da aplicação TGrafos.

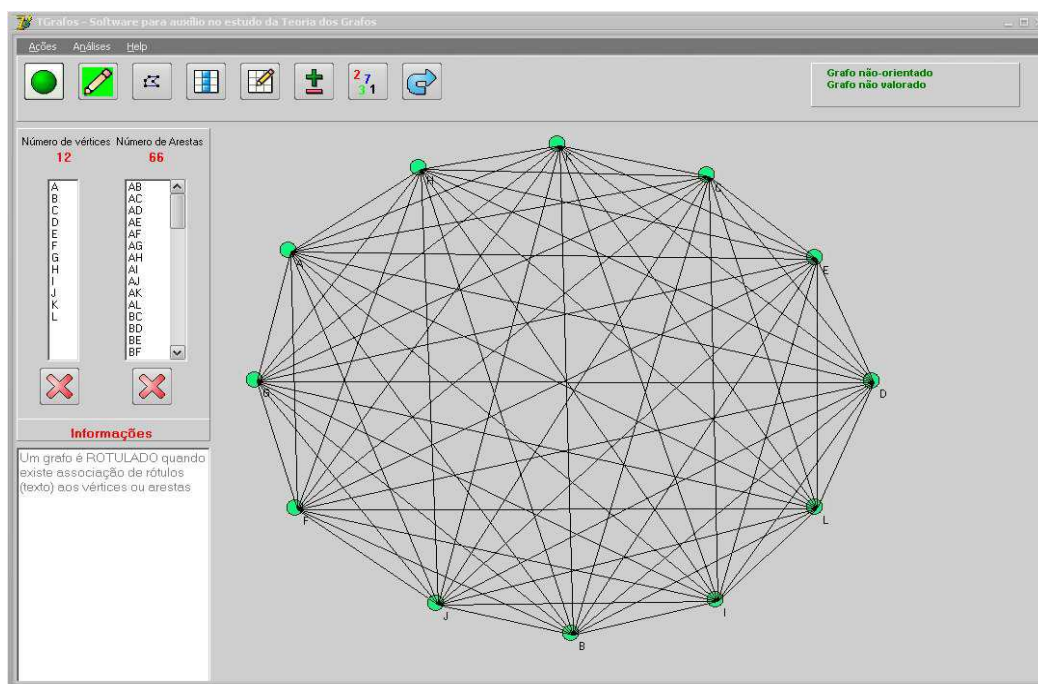


Figura 6 – Tela principal da aplicação TGrafos

Fonte: (SILVEIRA; SILVA, 2016)

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 disponibilizada por menos da metade das aplicações pesquisadas.

MATERIAIS E MÉTODOS

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;

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.

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 caminhamiento 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.

- **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 caminhamiento do grafo considerando uma aresta que conecta o vértice atual à um vértice ainda não visitado. Dessa maneira, a estratégia de caminhamiento 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 7 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 7 – Pseudocódigo do algoritmo DFS

Fonte: (SOUZA et al., 2013)

- **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 8 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 8 – Pseudocódigo do algoritmo BFS

Fonte: (SOUZA et al., 2013)

RESULTADOS E DISCUSSÃO

Baseado nas lacunas apresentados pelos softwares educacionais utilizados no ensino da teoria dos grafos, os seguintes artefatos foram desenvolvidos:

Modelagem

As funções disponibilizadas pela aplicação permitem ao usuário manusear a exploração do algoritmo de diversas formas. A figura 9 apresenta o diagrama de caso de uso da ferramenta.

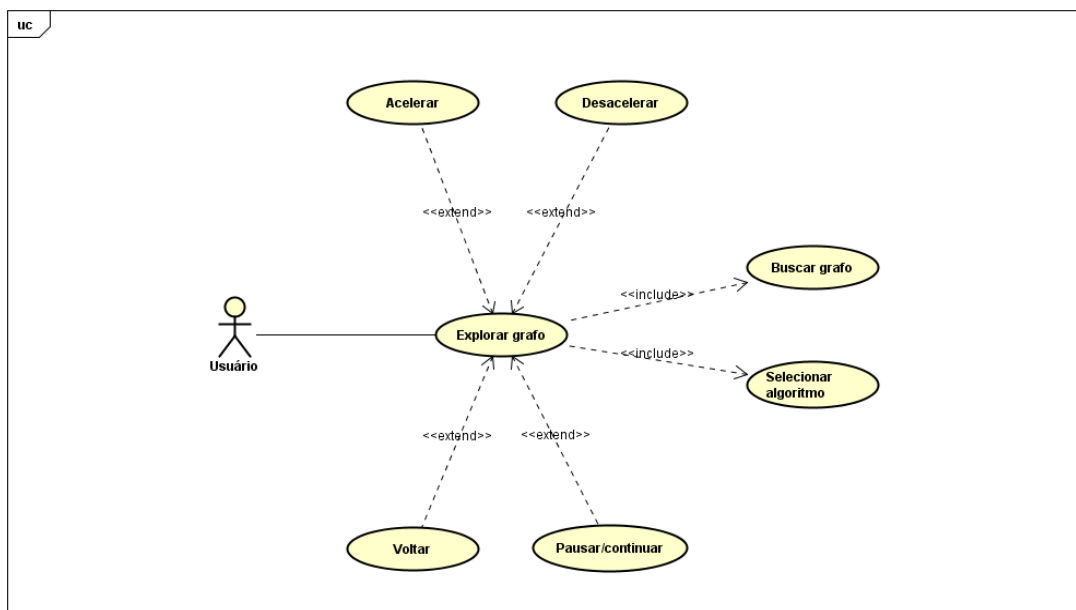


Figura 9 – Diagrama de caso de uso

Fonte: Produzido pelo próprio autor

Conforme verificado na figura 9, 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.

Código

A figura 10 apresenta como a estrutura do código está dividida.

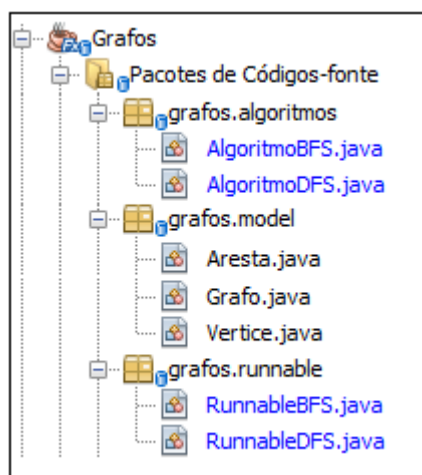


Figura 10 – Estrutura do código

Fonte: Produzido pelo próprio autor

Ela pode ser resumida pelas classes Grafo, Vertice, Aresta, AlgoritmoBFS, AlgoritmoDFS, RunnableBFS e RunnableDFS, onde:

- 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 *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 *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*.

Funcionamento do código

Devido ao fato de o JavaFX não ser Thread-Safe², 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()*³, 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*⁴.

² Capacidade de manipular estruturas de dados compartilhadas de uma forma que garanta uma execução segura através de várias threads ao mesmo tempo

³ Método da interface *Runnable* que é executado quando a *Thread* é iniciada

⁴ *Thread* do JavaFX responsável pelas alterações na interface gráfica

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 necessárias, já que cada uma dessas linhas pode 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 Task⁵ 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.

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.

Tela Principal

A figura 11 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. Em seguida os parâmetros necessários para execução do algoritmo serão solicitados para que a função confirmar possa ser acionada.

⁵ Implementação da classe *javafx.concurrent*. Assim como a *Runnable* possibilita execuções assíncronas

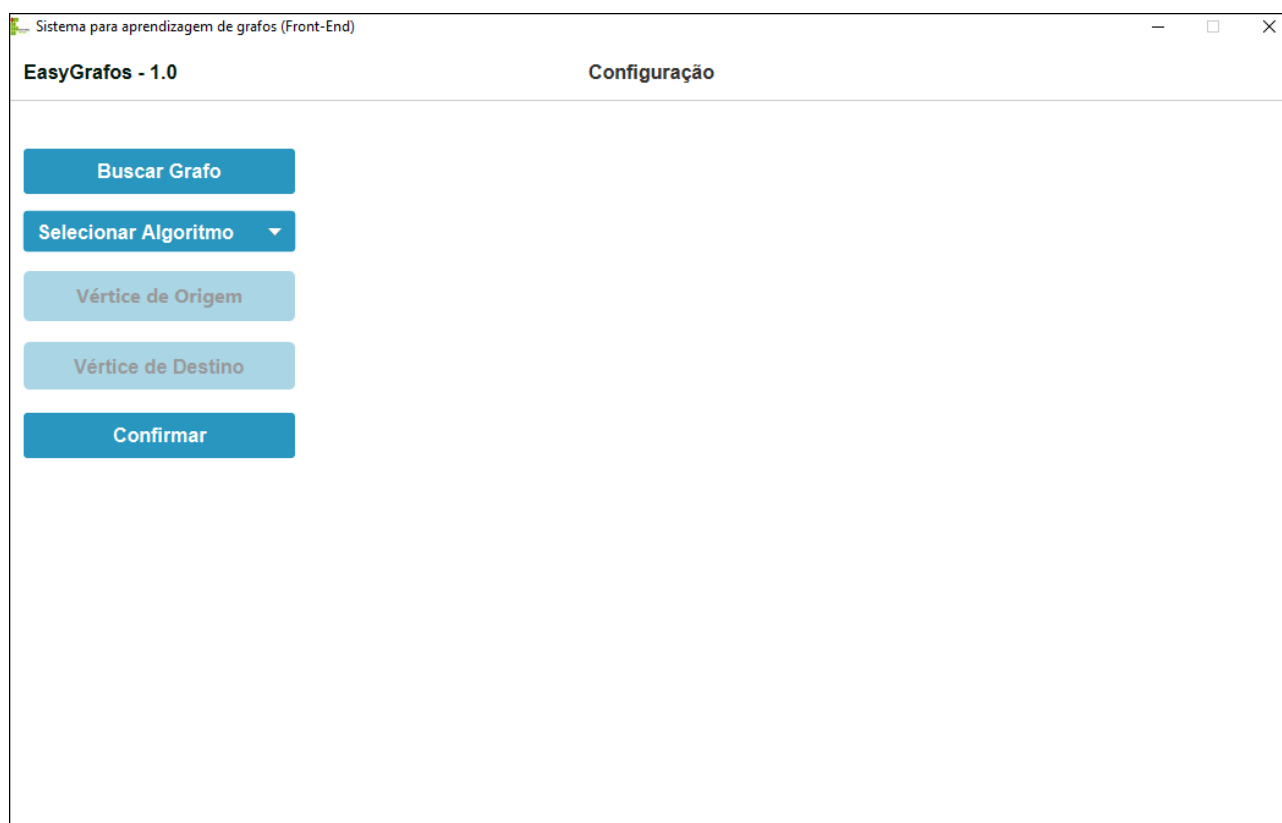


Figura 11 – Tela inicial

Fonte: Produzido pelo próprio autor

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 12 apresenta uma legenda com as cores de cada status destes elementos.




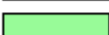

Status		Aresta	Vértice
Branco		Não aplicável	Não visitado
Cinza		Descartada	Descartado
Preto		Não visitada	Não aplicável
Verde Claro		Visitada	Visitado
Verde Escuro		Visitando	Visitando

Figura 12 – Legenda de cor dos status

Fonte: Produzido pelo 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.

Algoritmos 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 13 e 14 apresentam respectivamente as telas da ferramenta durante a execução dos algoritmos BFS e DFS.

Sistema para aprendizagem de grafos (Front-End)

EasyGrafos - 1.0

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

Execução do Algoritmo Busca em Largura (BFS)

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

Explorando vértice 3

Lista de Adjacências	Fila
0 -- 3 - 2 - 1	2
1 -- 6 - 3 - 0	1
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	

Variáveis Locais

Váriável v: 3
Váriável x: --

Controles: << >> || <

Figura 13 – Tela do algoritmo BFS em execução

Fonte: Produzido pelo próprio autor

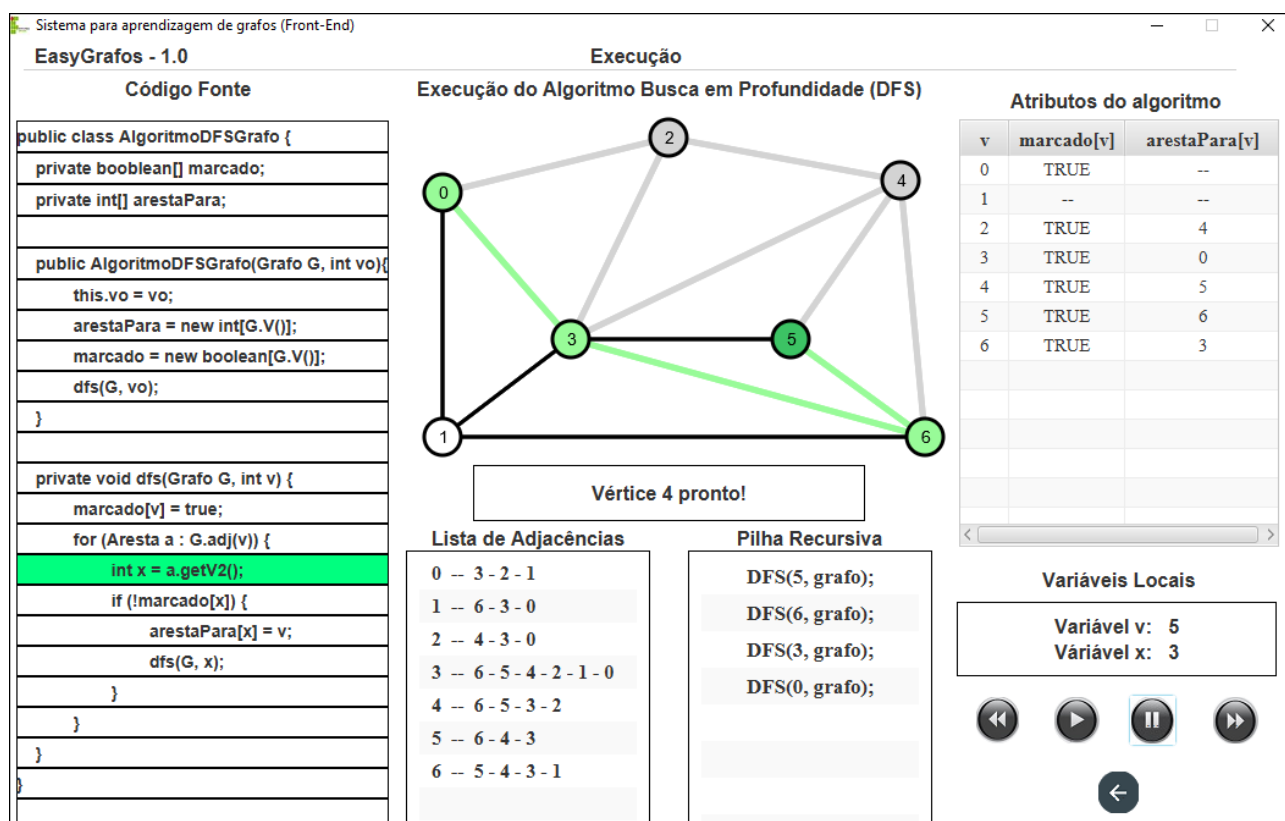


Figura 14 – Tela do algoritmo DFS em execução

Fonte: Produzido pelo 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 14 temos a exibição de uma pilha de chamadas recursivas, enquanto na figura 13 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 15 e 16 ilustram respectivamente o resultado final da execução dos algoritmos BFS e DFS.

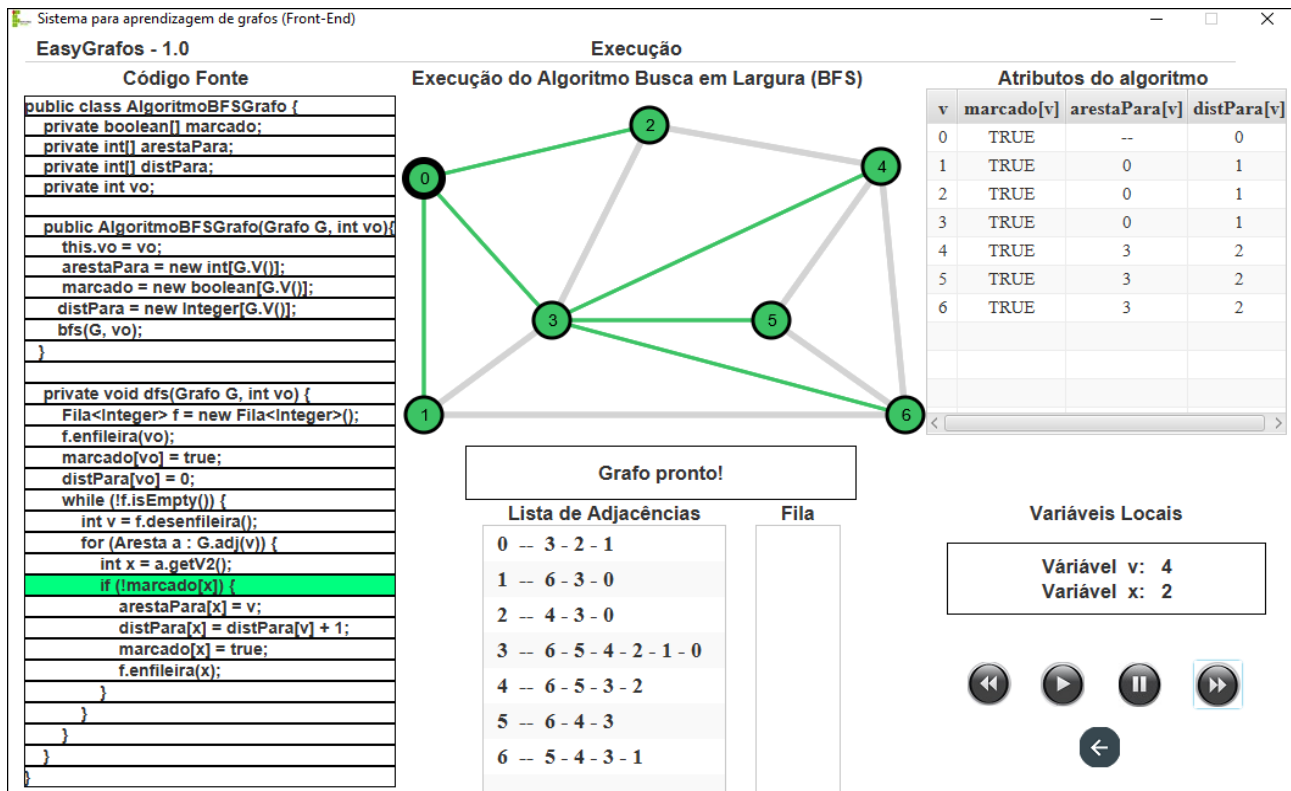


Figura 15 – Tela final do algoritmo BFS

Fonte: Produzido pelo próprio autor

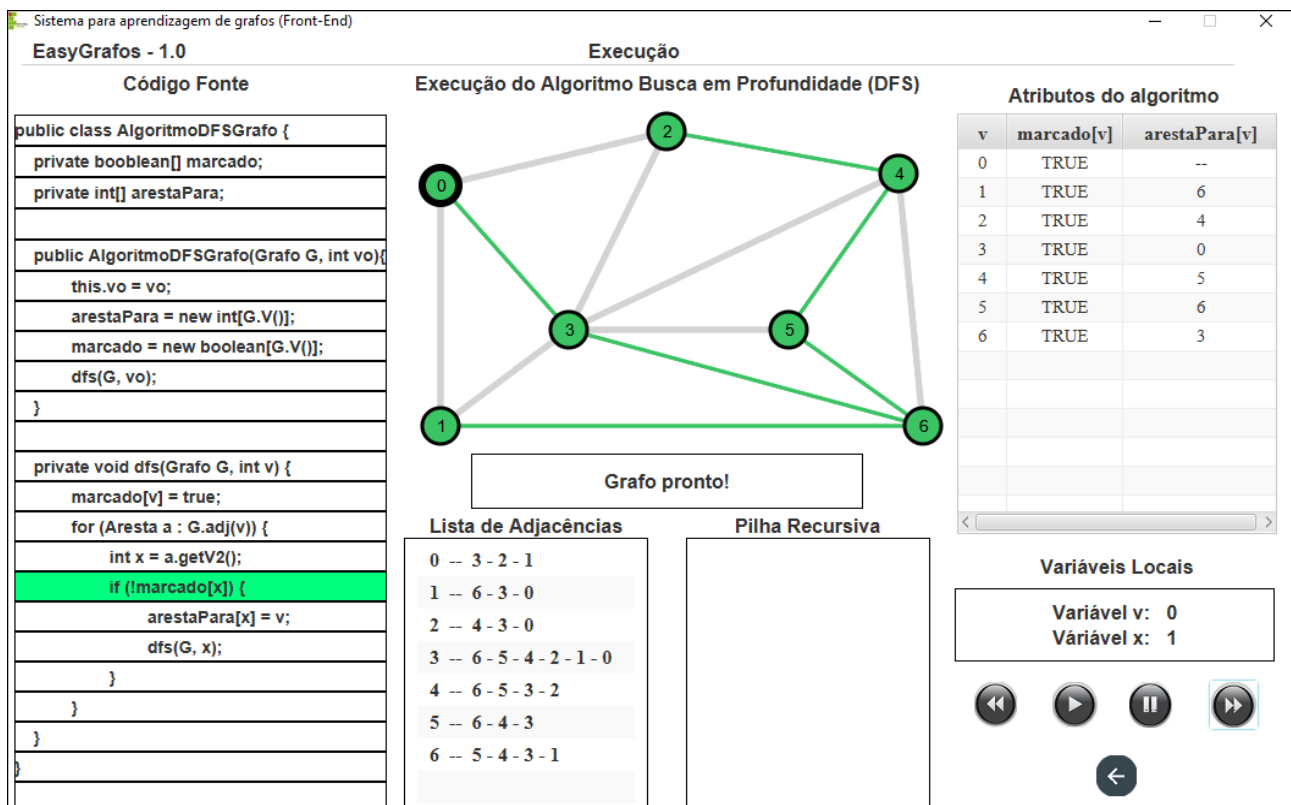


Figura 16 – Tela final do algoritmo DFS

Fonte: Produzido pelo próprio autor

As figuras 15 e 16 apresentam os vértices acessíveis a partir de um dado vértice de origem, neste caso o vértice 0 (colorido com borda mais grossa). Ambas têm 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 13 e 14, ao perceber que os vértices que são descartados primeiro (coloração cinza), não são os mesmos. Assim como, nas figuras 15 e 16, onde as arestas coloridas não são as mesmas.

Validação

A ferramenta EasyGrafos foi avaliada utilizando um formulário⁶ baseando no formulário de validação de software educacional desenvolvido por Thomas Reeves⁷, 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 está mais associada a característica negativa está a implementação deste atributo e quanto mais próxima do 10, mais associada a característica positiva está a implementação deste atributo.

CONCLUSÕES

O formulário desenvolvido para validar a ferramenta, 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 17 apresenta um gráfico com a média de cada quesito avaliado.

⁶ https://docs.google.com/forms/d/e/1FAIpQLSfDvTgPTK-HSh0lnOSr7D9wdv2HzES6k9pMP0cUF8IcZj-bsQ/viewform?usp=sf_link

⁷ Professor emérito em Aprendizagem, Design e Tecnologia pela Universidade de Georgia, EUA. Página pessoal disponível em: <http://treeves.coe.uga.edu/edit8350/UIRF.html>

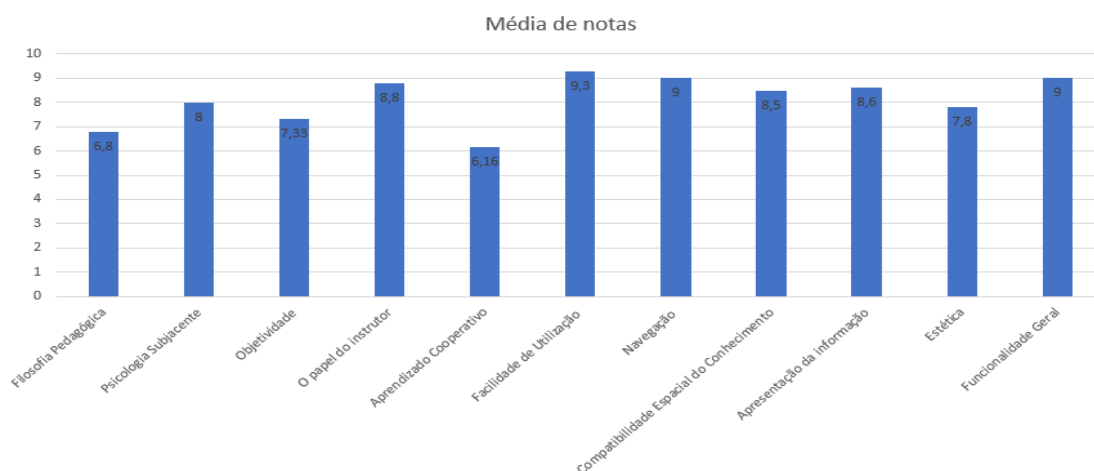


Figura 17 – Média geral das notas

Fonte: Produzido 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 18 apresenta um gráfico comparativo entre os dois módulos do questionário, o referente aos atributos pedagógicos e o referente aos atributos de interface gráfica.

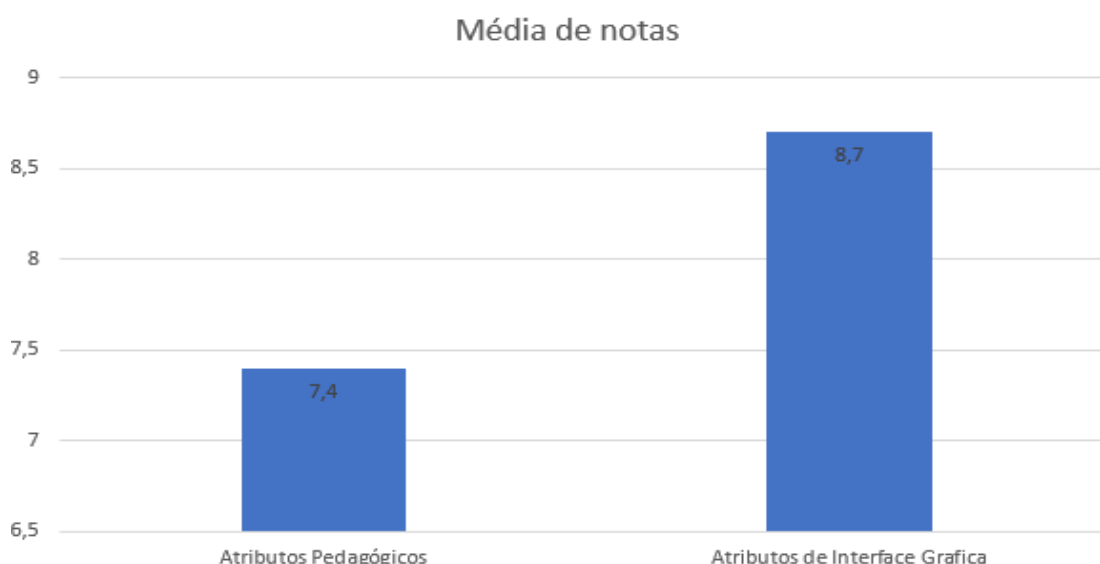


Figura 18 – Média dos módulos do formulário

Fonte: Produzido 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.

CONSIDERAÇÕES FINAIS

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. Como forma de motivar os alunos a utilizarem a ferramenta, a possibilidade de implementar o conceito gameificação otimizar os resultados da mesma.

AGRADECIMENTOS

Agradecimentos e créditos a instituições de financiamento deverão aparecer no final do texto e antes do item Referências.

REFERÊNCIAS

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.

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_superior>

JURKIEWICZ, Samuel. Grafos—uma introdução. São Paulo: **OBMEP**, 2009. COSTA, P. P. d. **Teoria dos grafos e suas aplicações**. Universidade Estadual Paulista (UNESP), 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

MORATORI, P. B. **Por que utilizar jogos educativos no processo de ensino aprendizagem**. UFRJ. Rio de Janeiro, 2003.

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.

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.

SOUZA, A. L. et al. **Teoria dos grafos e aplicações**. Universidade Federal do Amazonas, 2013. SILVEIRA, E. B. de A.; SILVA, M. O. da. **Desenvolvimento de um aplicativo educacional para o estudo de teoria dos grafos**. 2016.