Departamento de Sistemas e Computação – FURB
Curso de Ciência da Computação
Trabalho de Conclusão de Curso – 2016/1

FURB GRAPHS: uma ferramenta de ensino para a disciplina de teoria dos grafos

Acadêmico: Luiz Henrique Bernardes
luizhbernardes@hotmail.com

Orientador: Prof. Aurélio Faustino Hoppe
aurelio.hoppe@gmail.com

Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
http://www.inf.furb.br/gcg

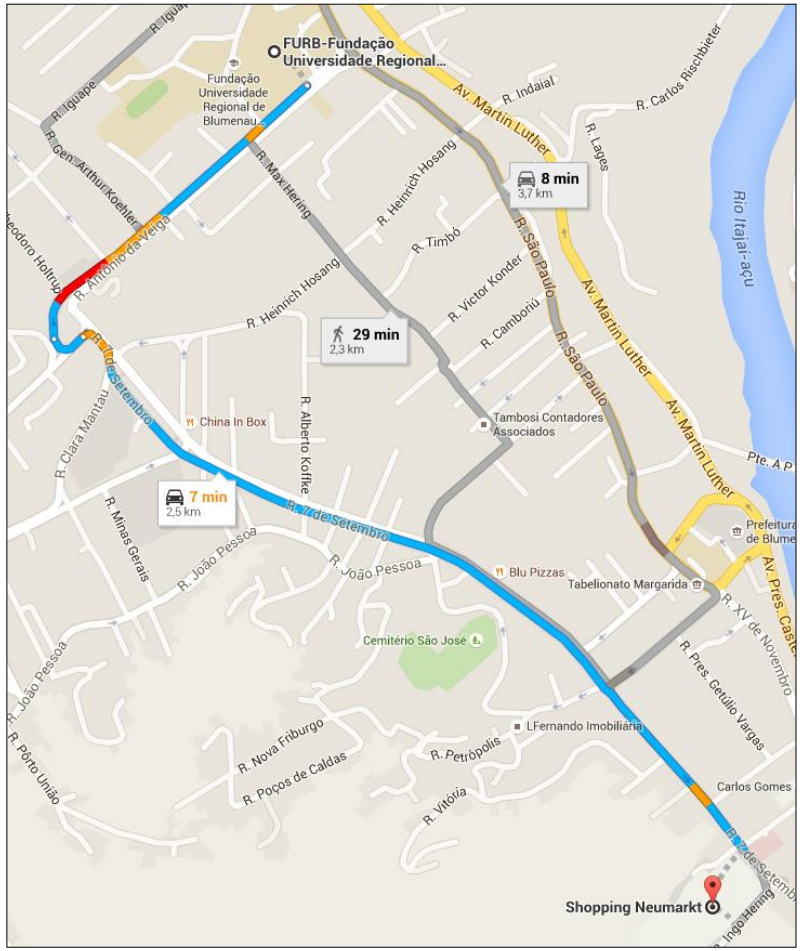
Roteiro

- Introdução
- Motivação
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Requisitos
- Especificação
- Desenvolvimento da ferramenta
- Conclusões
- Demonstração

Introdução

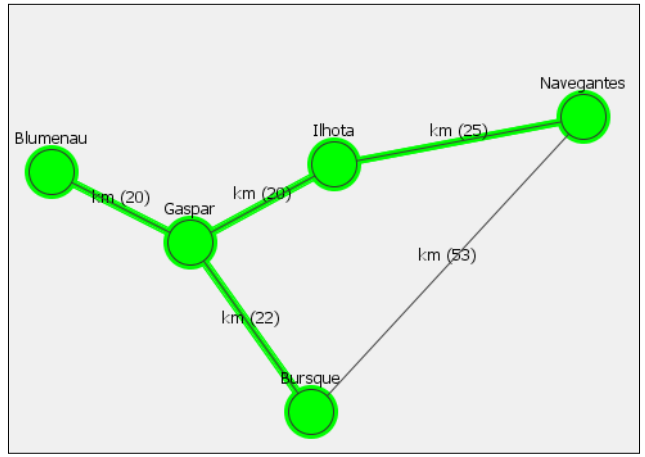
- A importância da teoria dos grafos.
  - Impacta em diversas outras áreas (Engenharia, Matemática, Logística, Física, etc...).

  - Diversas aplicações e ferramentas utilizam grafos, tanto para encontrar o menor caminho entre duas cidades quanto para mapear computadores em uma rede.



Motivação

- O problema atualmente.
  - Dificuldade dos alunos em compreender a definição teórica e implementar os algoritmos.



  - São necessárias diversas instruções, anotações e desenhos para se resolver um problema de grafos no papel.

	Passo 1		Passo 2		Passo 3		Passo 4		Passo 5	
Cidades/Passos	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo
Blumenau	Null	0	*	*	*	*	*	*	*	*
Gaspar	Blumenau	20								
Ilhota	Null	∞								
Brusque	Null	∞								
Navegantes	Null	∞								

	Passo 1		Passo 2		Passo 3		Passo 4		Passo 5	
Cidades/Passos	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo
Blumenau	Null	0	*	*	*	*	*	*	*	*
Gaspar	Blumenau	20	Blumenau	20	*	*	*	*	*	*
Ilhota	Null	∞	Gaspar	40						
Brusque	Null	∞	Gaspar	44						
Navegantes	Null	∞	Null	∞						

	Passo 1		Passo 2		Passo 3		Passo 4		Passo 5	
Cidades/Passos	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo
Blumenau	Null	0	*	*	*	*	*	*	*	*
Gaspar	Blumenau	20	Blumenau	20	*	*	*	*	*	*
Ilhota	Null	∞	Gaspar	40	Gaspar	40	*	*	*	*
Brusque	Null	∞	Gaspar	42	Gaspar	42				
Navegantes	Null	∞	Null	∞	Ilhota	65				

	Passo 1		Passo 2		Passo 3		Passo 4		Passo 5	
Cidades/Passos	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo
Blumenau	Null	0	*	*	*	*	*	*	*	*
Gaspar	Blumenau	20	Blumenau	20	*	*	*	*	*	*
Ilhota	Null	∞	Gaspar	40	Gaspar	40	*	*	*	*
Brusque	Null	∞	Gaspar	42	Gaspar	42	Gaspar	42	*	*
Navegantes	Null	∞	Null	∞	Ilhota	65	Ilhota	65		

	Passo 1		Passo 2		Passo 3		Passo 4		Passo 5	
Cidades/Passos	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo	Pai	Custo
Blumenau	Null	0	*	*	*	*	*	*	*	*
Gaspar	Blumenau	20	Blumenau	20	*	*	*	*	*	*
Ilhota	Null	∞	Gaspar	40	Gaspar	40	*	*	*	*
Brusque	Null	∞	Gaspar	42	Gaspar	42	Gaspar	42	*	*
Navegantes	Null	∞	Null	∞	Ilhota	65	Ilhota	65	Ilhota	65

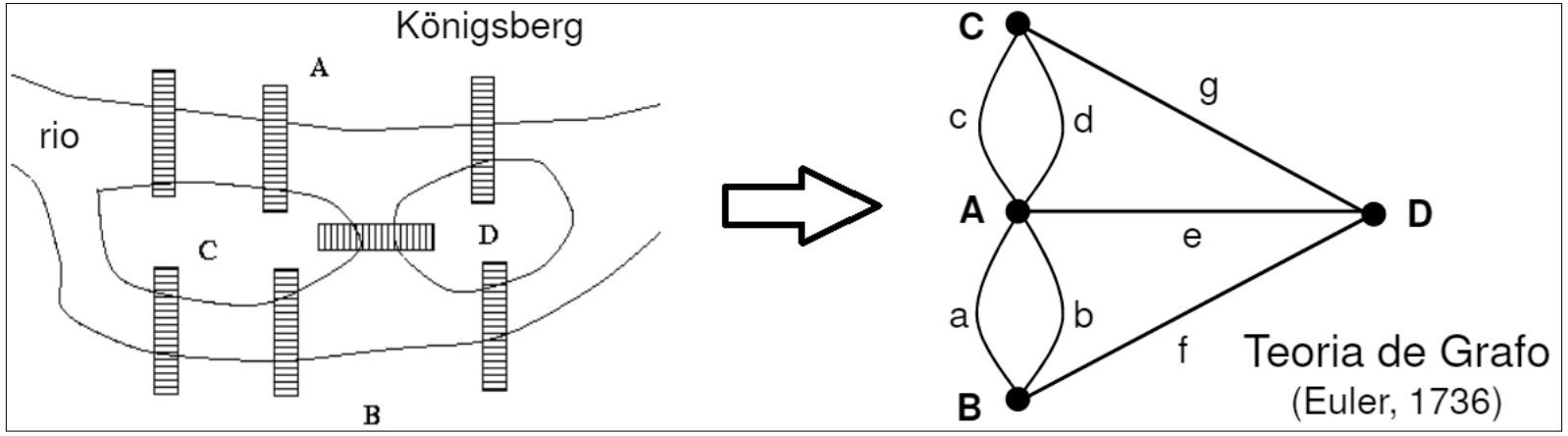
Objetivos

Adaptar a aplicação FURB Graphs de Borba (2014) tornando-a uma ferramenta de ensino para a disciplina de teoria dos grafos.

- **Objetivos específicos**
  - Adaptar a aplicação de Borba (2014) para demonstrar o funcionamento passo a passo dos algoritmos BFS, DFS e Dijkstra através da interface gráfica atual.
  - Apresentar informações e instruções quanto aos algoritmos executados.

Fundamentação teórica

- Teoria dos grafos
  - Leonhard Paul Euler
  - Problema das pontes de Königsberg
  - Vértices
  - Arestas

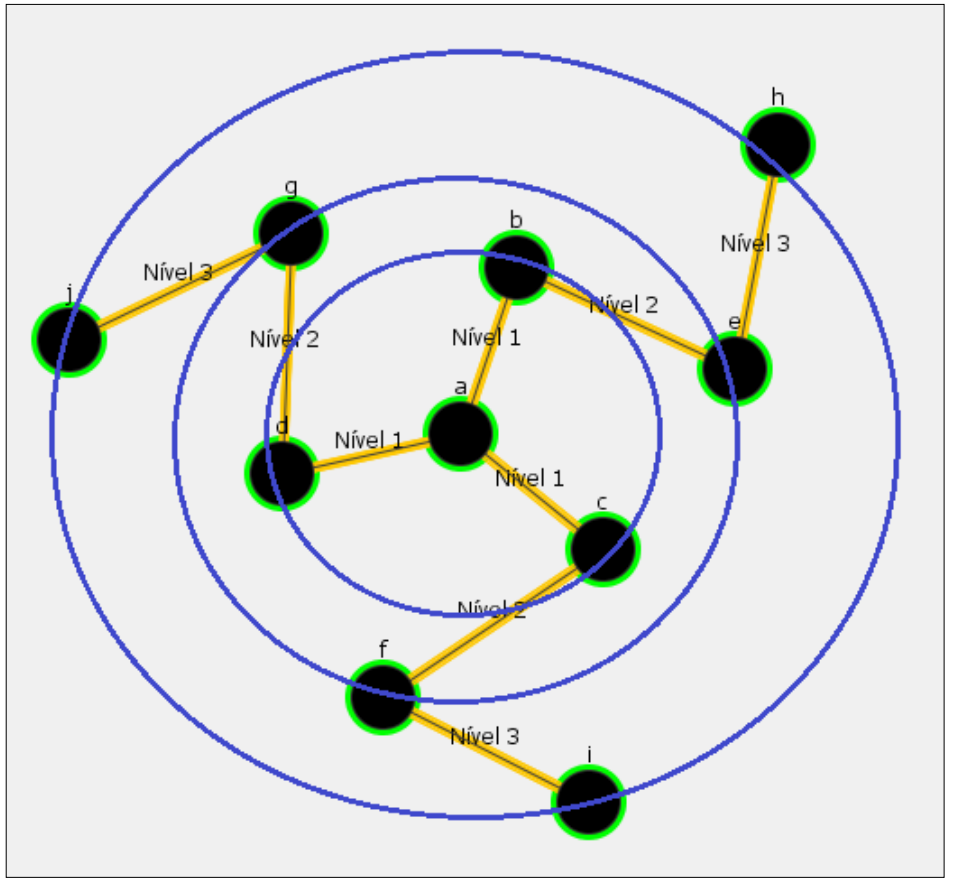


Fundamentação teórica

- Busca em largura (*Breadth First Search*)
  - Utiliza uma fila como estrutura auxiliar
  - A busca ocorre de forma expansiva

```
PROCESSANDO VERTICE >>> a
VERTICES VIZINHOS:
>> b
>> c
>> d

Vetor de roteamento - <a >

VISITANDO VERTICE > d
FILA >> [a, d]

VISITANDO VERTICE > c
FILA >> [a, d, c]
```

```
VISITANDO VERTICE > b
FILA >> [a, d, c, b]

VERTICE >>> a JÁ FOI EXPLORADO!
REMOVENDO VERTICE >>> a
FILA >> [d, c, b]

PROCESSANDO VERTICE >>> d
VERTICES VIZINHOS:
>> g

Vetor de roteamento - <a <b , c , d>>
```

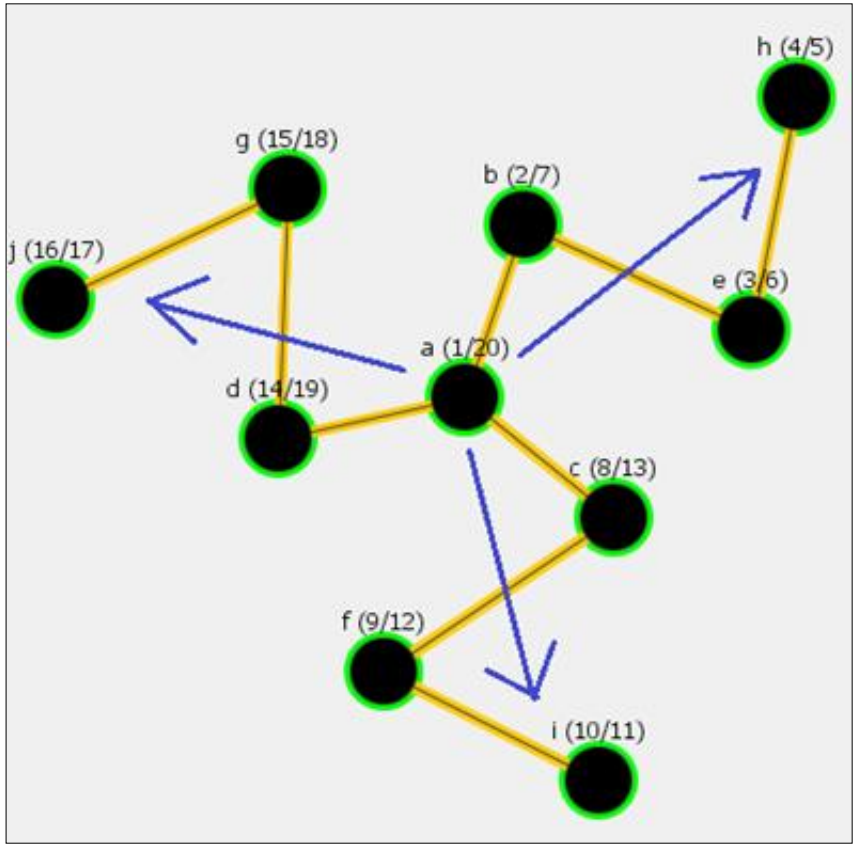


Fundamentação teórica

- Busca em profundidade (*Deep First Search*)
  - Utiliza uma pilha como estrutura auxiliar
  - Percorre um caminho ou ramo até esgotarem suas opções e então retorna para o último vértice com saídas disponíveis

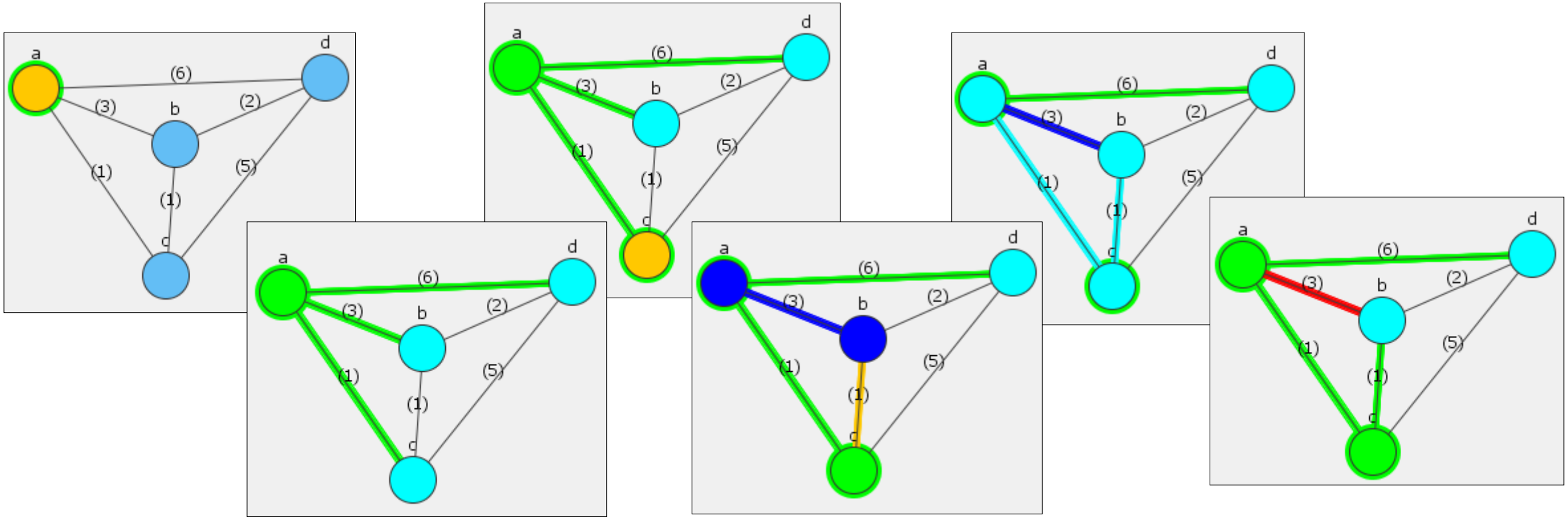
```
PROCESSANDO VERTICE >>>> a
LISTA DE VERTICES:
>>> b
>>> c
>>> d
EMPILHAMENTO > [a]

VISITANDO VERTICE > b

PROCESSANDO VERTICE >>>> b
LISTA DE VERTICES:
>>> e
EMPILHAMENTO > [a, b]

VISITANDO VERTICE > e
```

```
PROCESSANDO VERTICE >>>> e
LISTA DE VERTICES:
>>> h
EMPILHAMENTO > [a, b, e]

VISITANDO VERTICE > h

PROCESSANDO VERTICE >>>> h
EMPILHAMENTO > [a, b, e, h]

DESEMPILHA VERTICE > h
EMPILHAMENTO > [a, b, e]

DESEMPILHA VERTICE > e
EMPILHAMENTO > [a, b]

DESEMPILHA VERTICE > b
EMPILHAMENTO > [a]
```



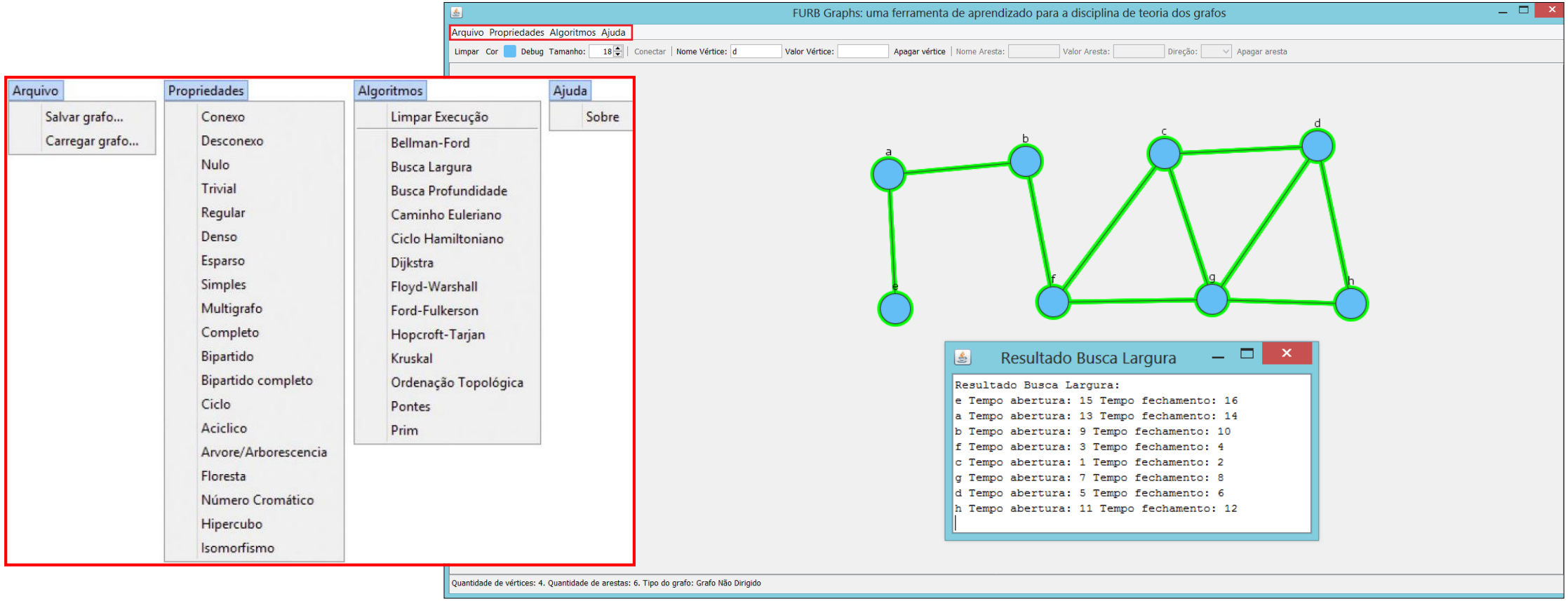
Fundamentação teórica

- Algoritmo de Dijkstra
  - Identifica o caminho mínimo entre o vértice de origem e todos os demais vértices do grafo



Fundamentação teórica

- FURB Graphs: uma aplicação para teoria dos grafos, por Borba (2014)



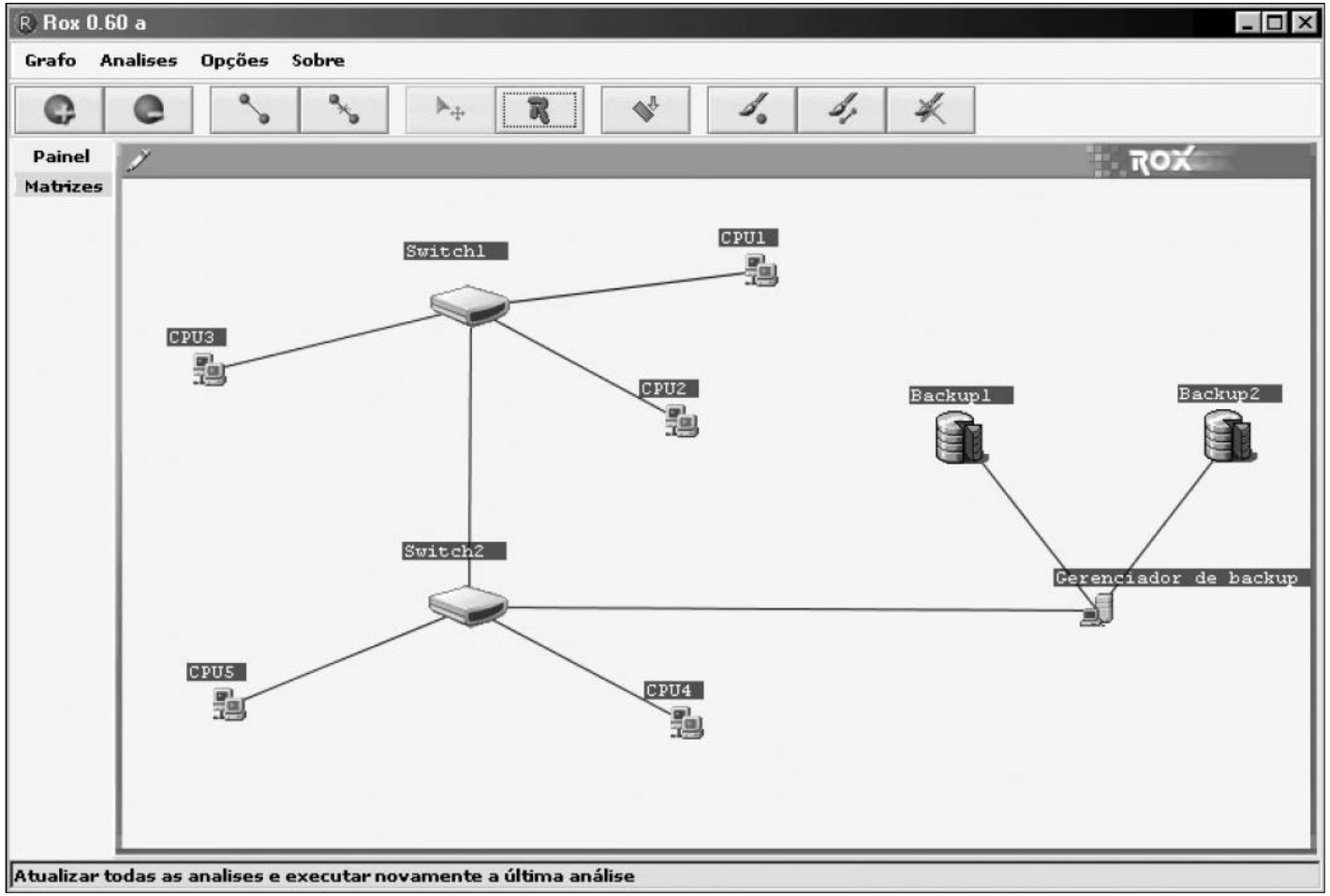
Trabalhos correlatos

Foram pesquisados três trabalhos correlatos a este, na área de ferramenta para ensino de grafos.

- Rox (SANGIORGI, 2004)
- TBC – Grafos/Web (SANTOS; COSTA, 2007)
- A-Graph (LOZADA, 2014)

Trabalhos correlatos

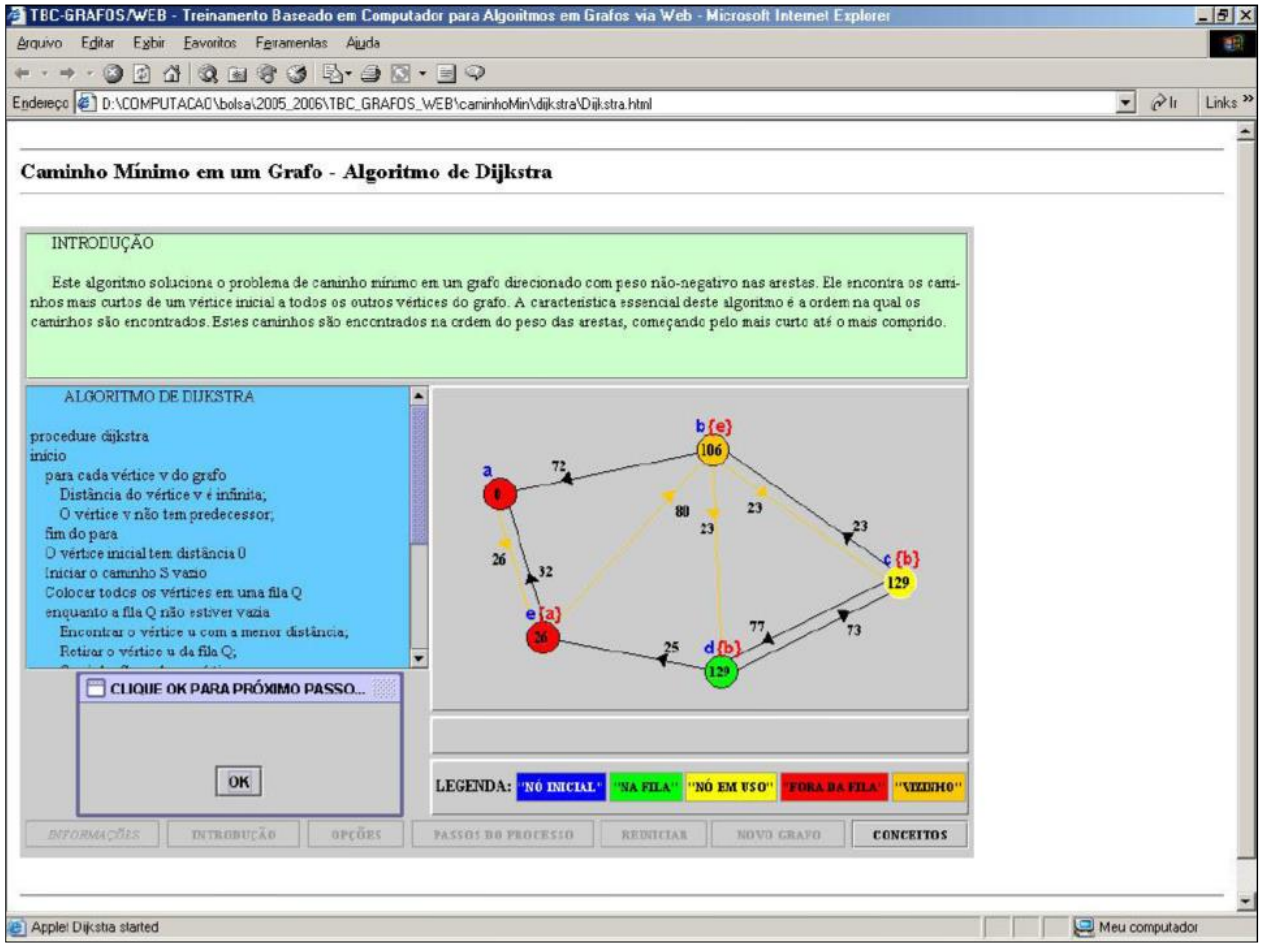
- Rox



Diferencial: Permite que os vértices do grafo sejam substituídos por imagens. Permite que o usuário escreva o próprio algoritmo e o execute na ferramenta.

Trabalhos correlatos

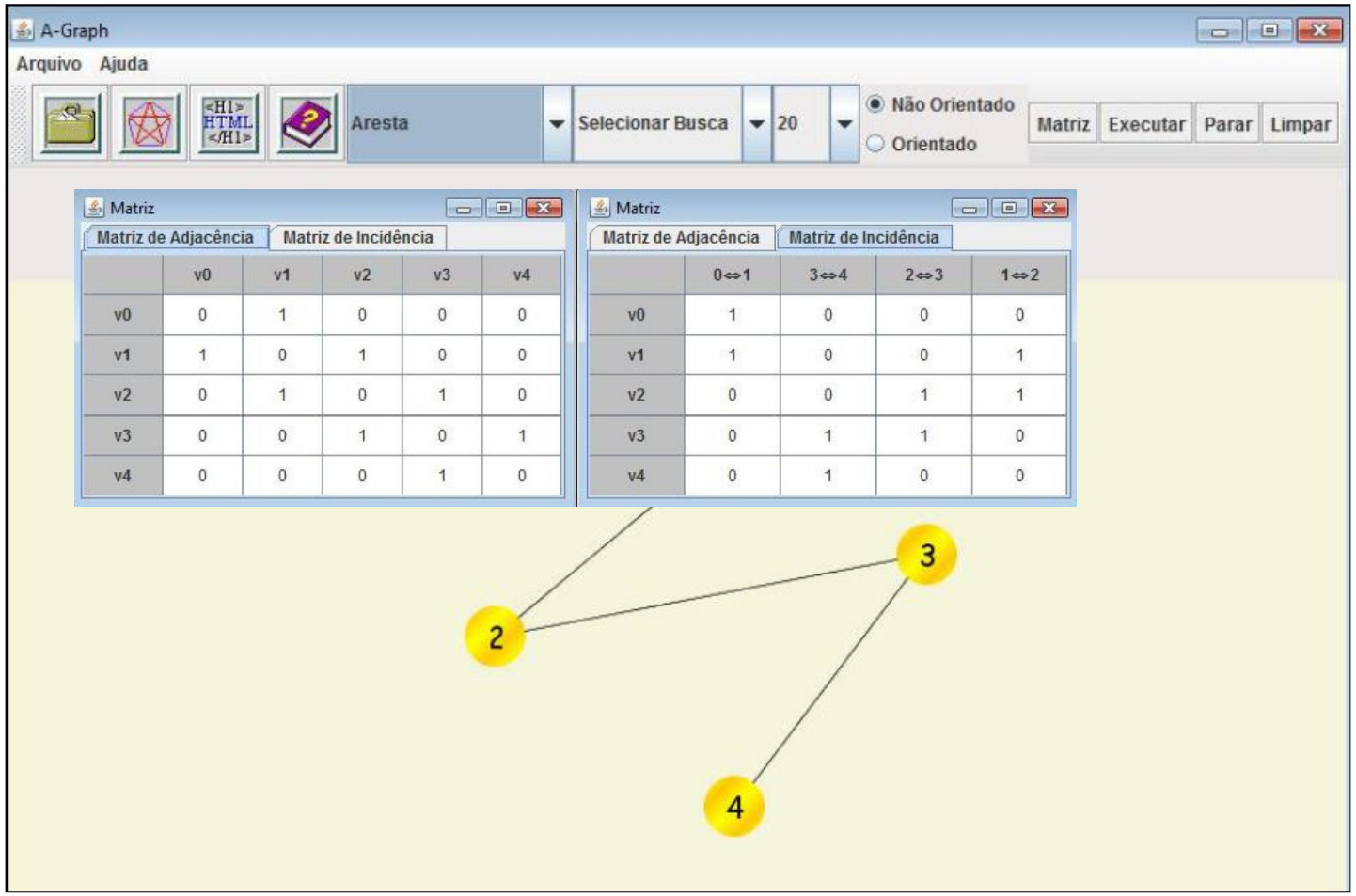
- TBC – Grafos/Web



Diferencial: Permite que o usuário avance passos do algoritmo. Fornece informações didáticas e apresenta o código fonte do algoritmo.

Trabalhos correlatos

- A-Graph



Diferencial: Apresenta as matrizes de adjacência e de incidência do grafo.

Características dos trabalhos correlatos

Características / trabalhos correlatos	Lozada (2014)	Santos e Costa (2007)	Sangiorgi (2004)
permite a criação de grafos livremente	Sim	Sim	Sim
suporta dígrafos	Sim	Sim	Sim
permite trocar as imagens dos vértices	Sim	Não	Sim
executa busca em grafos (BFS, DFS)	Sim	Sim	Não
executa árvore geradora mínima (Kruskal e Prim)	Não	Sim	Não
executa caminho mínimo entre vértices (Dijkstra e Bellman-Ford)	Sim	Sim	Não
analisa ciclos Hamiltonianos	Não	Não	Sim
analisa grafos bipartidos	Não	Não	Sim
apresenta informativos referente aos algoritmos	Não	Sim	Não
apresenta material teórico quanto aos algoritmos	Não	Sim	Não
apresenta recursos de apoio ao aprendizado dos algoritmos (legendas, código fonte)	Sim	Sim	Sim
permite carregar um código fonte de terceiro para utilizar na execução	Não	Não	Sim
possui plataforma de ampla disponibilidade (web)	Não	Sim	Não

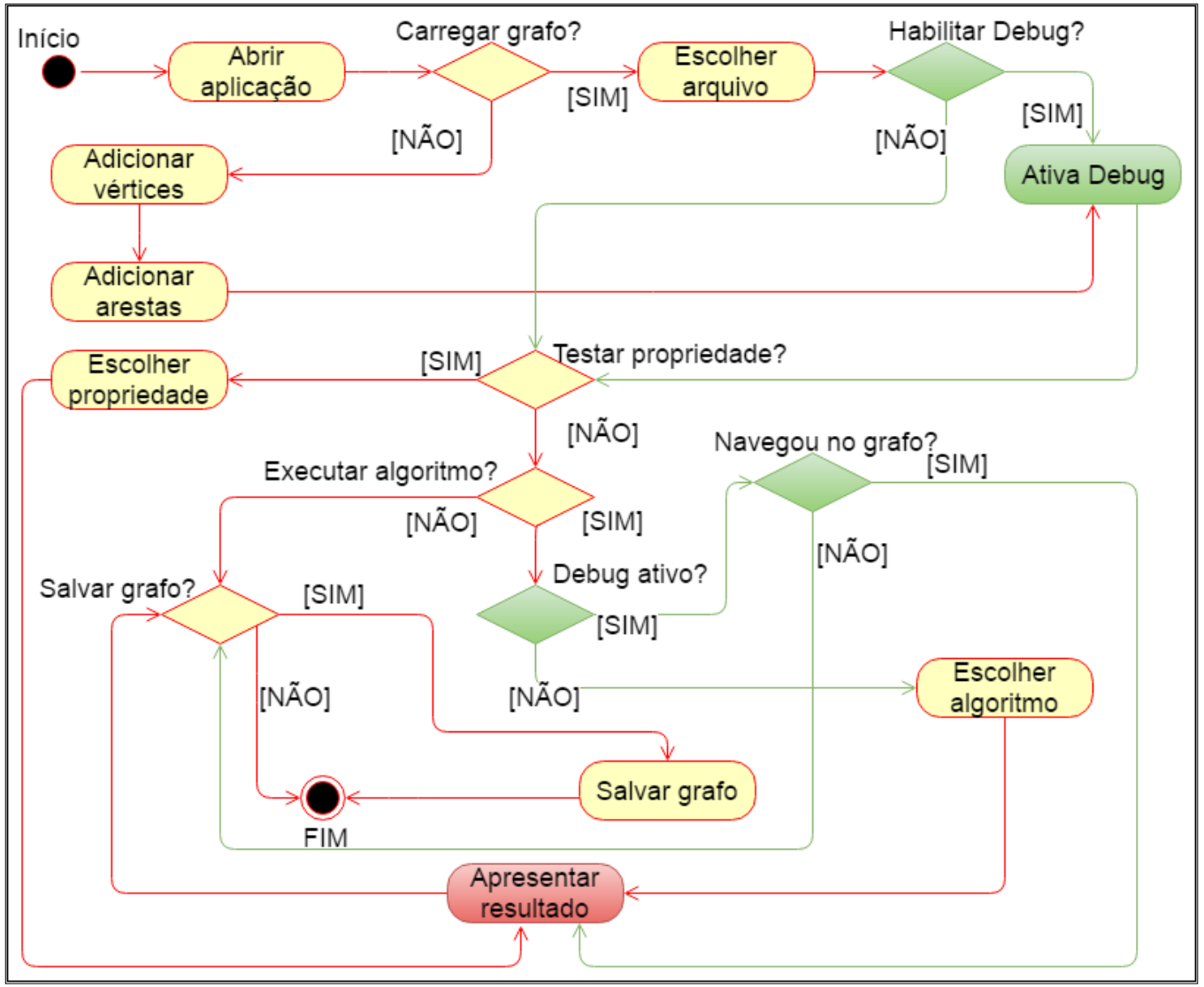
Requisitos

A seguir são listados os requisitos funcionais da ferramenta:

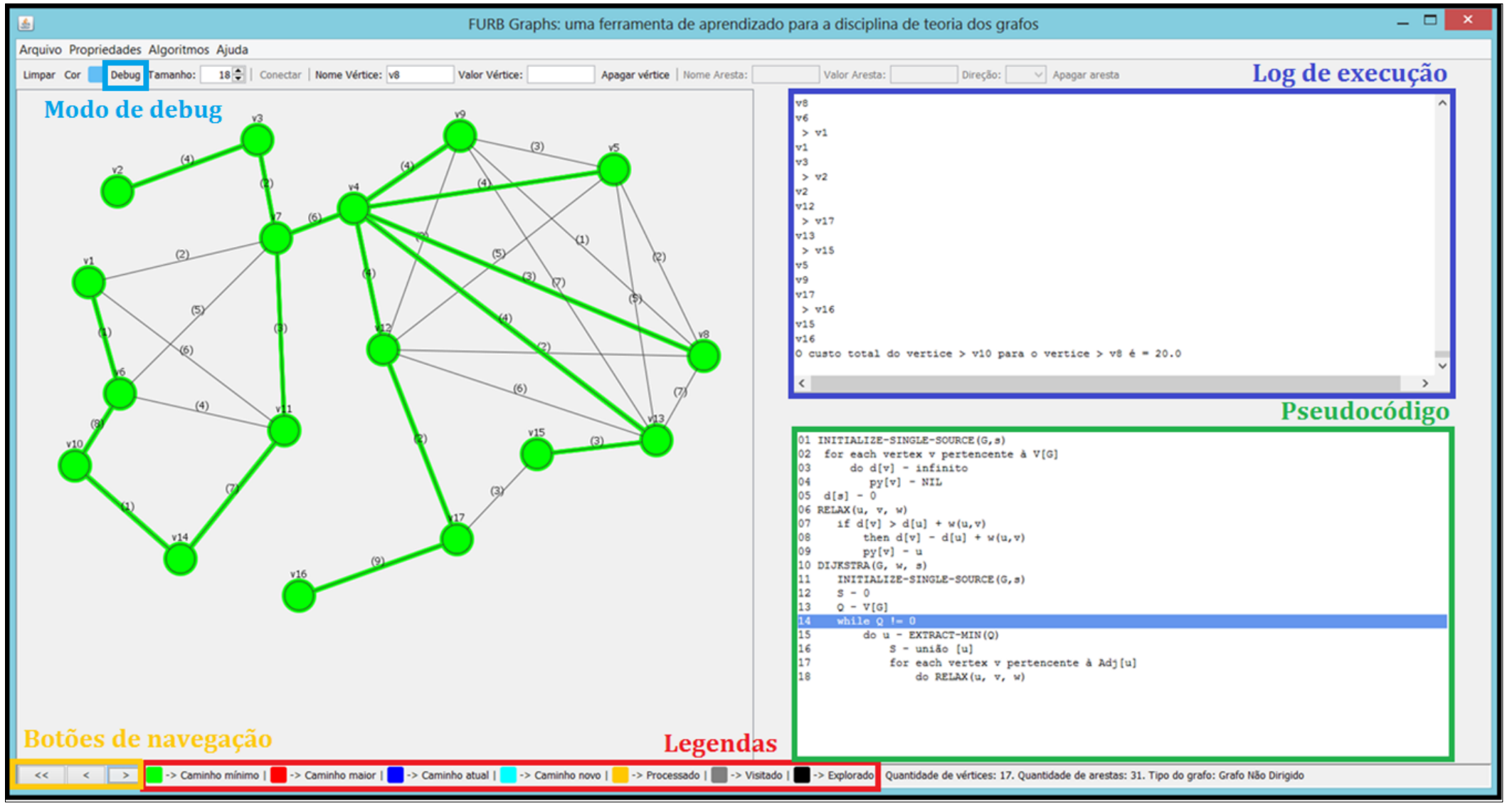
- Permitir que o usuário visualize os atributos dos vértices e das arestas de um grafo através da interface gráfica adaptada do trabalho de Borba (2014)

- Permitir que o usuário controle o passo a passo (avance e retroceda) da execução dos algoritmos BFS, DFS e Dijkstra

Especificação – Diagrama de atividades



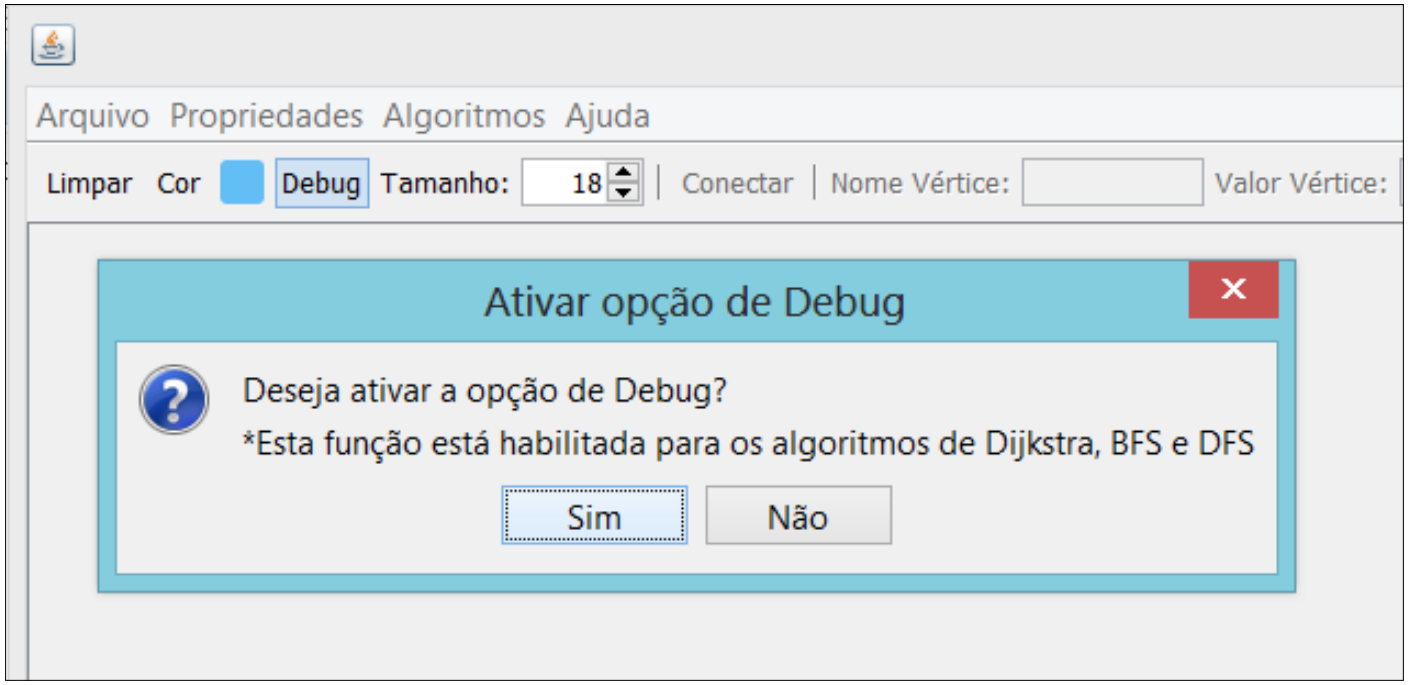
Desenvolvimento da ferramenta



Desenvolvimento da ferramenta

Modo de Debug

- Permite a navegação “passo a passo” dos algoritmos BFS, DFS e Dijkstra



Desenvolvimento da ferramenta

Botões de navegação

- Permite que o usuário interaja com os algoritmos (BFS, DFS e Dijkstra), avançando um passo na execução

- Permite que o usuário interaja com os algoritmos (BFS e DFS), retrocedendo um passo na execução

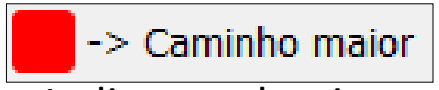
- Permite que o usuário interaja com os algoritmos (BFS, DFS e Dijkstra), retrocedendo todos os passos até o início da execução

Desenvolvimento da ferramenta

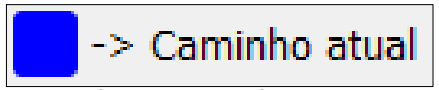
Legendas

Foram criadas legendas para instruir os usuários quanto ao estado dos vértices e arestas durante a execução.

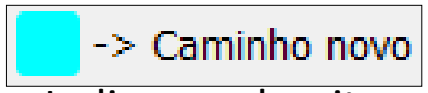
- Indica no algoritmo de Dijkstra qual é o **caminho mínimo** percorrido no grafo durante a navegação.



- Indica no algoritmo de Dijkstra qual é o **caminho maior** após uma comparação entre dois caminhos.



- Indica no algoritmo de Dijkstra qual é o **caminho atual** até determinado vértice que está sendo comparado.



- Indica no algoritmo de Dijkstra qual é o **caminho novo** até determinado vértice que será comparado

Desenvolvimento da ferramenta

- Indica para todos os algoritmos qual vértice ou aresta foi **processado**.

- Indica nos algoritmos BFS e DFS qual vértice foi **visitado**.

- Indica nos algoritmos BFS e DFS qual vértice foi totalmente **explorado**.

Desenvolvimento da ferramenta

Log de execução

- Filas
- Pilhas
- Árvores
- Estados e atributos dos vértices
- Comparações

```
Vetor de roteamento - <c <f <b <a>>, d , g <h>>>

VISITANDO VERTICE > e
FILA >> [a, e]


VERTICE >>> a JÁ FOI EXPLORADO!
REMOVENDO VERTICE >>> a
FILA >> [e]

PROCESSANDO VERTICE >>> e

Vetor de roteamento - <c <f <b <a <e>>>, d , g <h>>>

VISITANDO VERTICE > e
FILA >> [e]
```

```
PROCESSANDO VERTICE >>>> f
EMPILHAMENTO > [c, a, b, g, h, d, e, f]

DESEMPILHA VERTICE > f
EMPILHAMENTO > [c, a, b, g, h, d, e]

DESEMPILHA VERTICE > e
EMPILHAMENTO > [c, a, b, g, h, d]

DESEMPILHA VERTICE > d
EMPILHAMENTO > [c, a, b, g, h]

DESEMPILHA VERTICE > h
EMPILHAMENTO > [c, a, b, g]

DESEMPILHA VERTICE > g
EMPILHAMENTO > [c, a, b]

DESEMPILHA VERTICE > b
EMPILHAMENTO > [c, a]
```

```
#####################################
#>>>>>>>>>>>> COMPARAÇÃO <<<<<<<<<<<<#
Caminho atual que será comparado:  > v10 > v14 > v11 > v7 > v4 > v9
O vertice v9 já tem um pai > v4
O caminho  > v10 > v14 > v11 > v7 > v4 > v5 > v9 (Custo = 24.0)
é menor que o caminho  > v10 > v14 > v11 > v7 > v4 > v9 (Custo = 21.0)??
> Não

#################################
```

Desenvolvimento da ferramenta

Pseudocódigo

- Discriminação das linhas e ou blocos de código que se referem ao passo executado do algoritmo

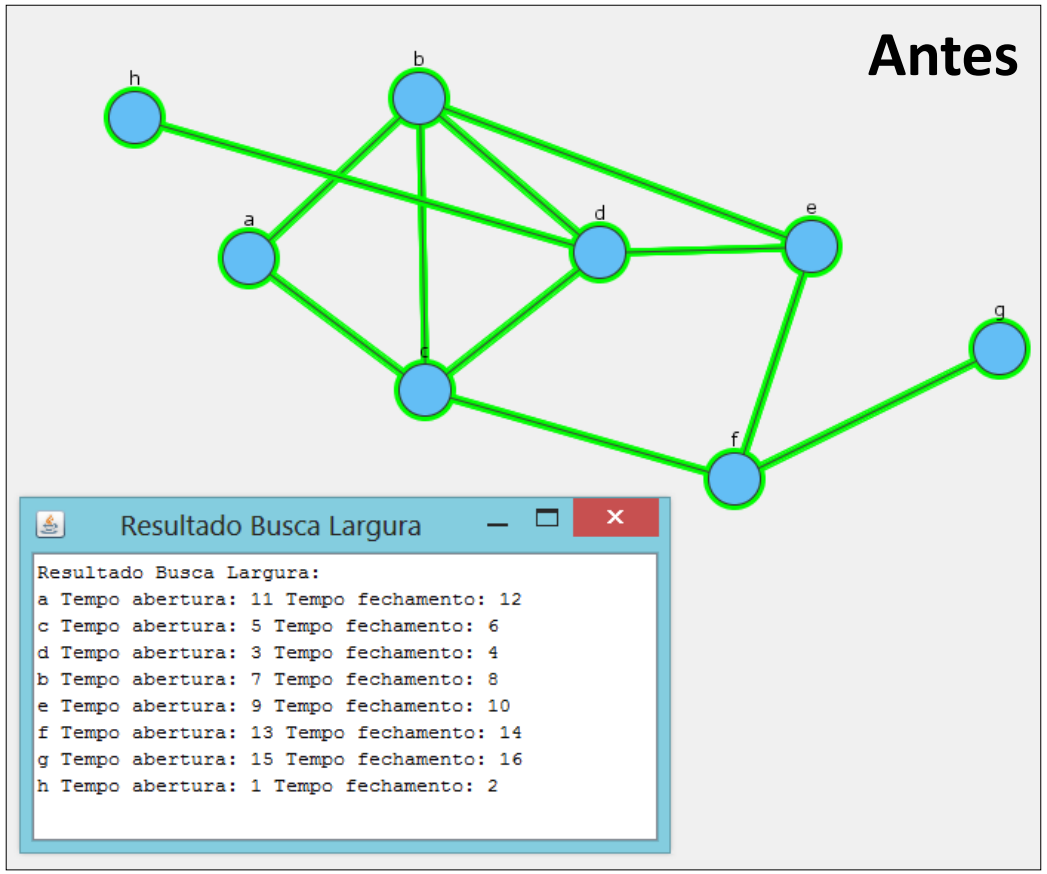
```
01 dfs(G)
02    para cada vértice u - V[G]
03        cor[u] - BRANCO
04    tempo - 0
05    para cada vértice u pertencente à V[G]
06        se cor[u] = BRANCO
07            DFS-VISIT(u)
08 DFS-VISIT(u)
09    cor[u] - CINZA
10    tempo - tempo + 1
11    d[u] - tempo
12    para cada vértice v pertencente à Adj(u)
13        se cor [v] = BRANCO
14            DFS-VISIT(v)
15    cor[u] - PRETO
16    f[u] - tempo - (tempo + 1)
```

```
01 INITIALIZE-SINGLE-SOURCE(G,s)
02  for each vertex v pertencente à V[G]
03      do d[v] - infinito
04         py[v] - NIL
05  d[s] - 0
06 RELAX(u, v, w)
07    if d[v] > d[u] + w(u,v)
08       then d[v] - d[u] + w(u,v)
09       py[v] - u
10 DIJKSTRA(G, w, s)
11    INITIALIZE-SINGLE-SOURCE(G,s)
12    S - 0
13    Q - V[G]
14    while Q != 0
15        do u - EXTRACT-MIN(Q)
16            S - união [u]
17            for each vertex v pertencente à Adj[u]
18                do RELAX(u, v, w)
```

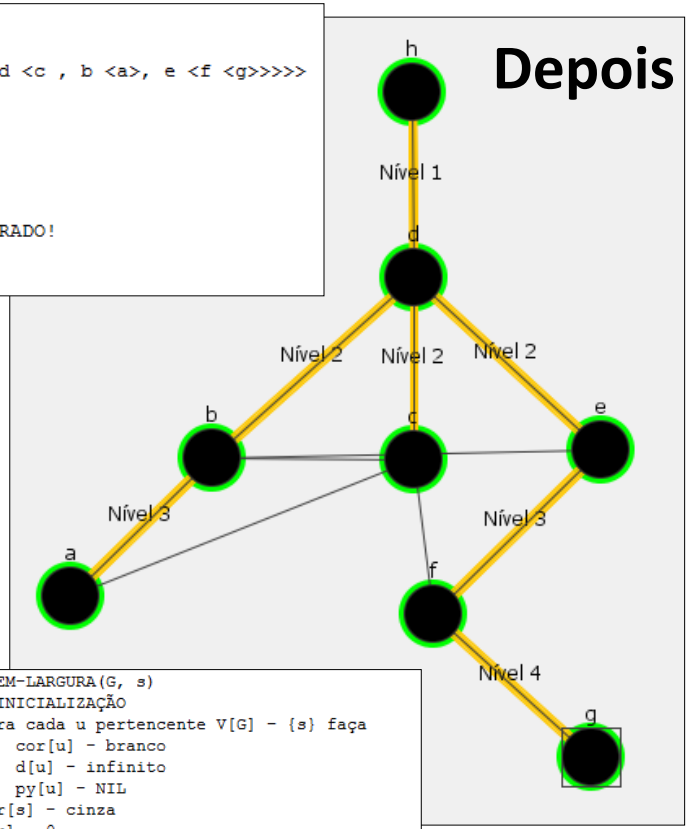
```
01 BUSCA-EM-LARGURA(G, s)
02     > INICIALIZAÇÃO
03     para cada u pertencente V[G] - {s} faça
04         cor[u] - branco
05         d[u] - infinito
06         py[u] - NIL
07     cor[s] - cinza
08     d[u] - 0
09     py[u] - NIL
10     Q - 0
11     ENQUEUE(Q, s)
12         enquanto Q diferente 0 faça
13             u - DEQUEUE(Q)
14             para cada v pertencente Adj[u] faça
15                 se cor[v] == branco então
16                     cor[v] - cinza
17                     d[v] - d[u] + 1
18                     py[v] - u
19                     ENQUEUE(Q, v)
20             cor[u] - preto
```

Desenvolvimento da ferramenta

Navegação no algoritmo BFS



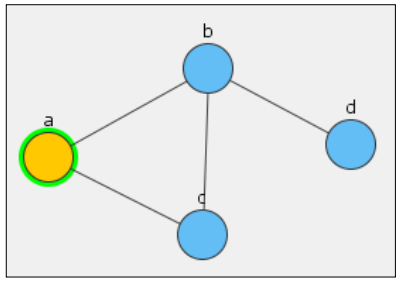

```
PROCESSANDO VERTICE >>> g

Vetor de roteamento - <h <d <c , b <a>, e <f <g>>>>>

VISITANDO VERTICE > g
FILA >> [g]


VERTICE >>> g JÁ FOI EXPLORADO!
REMOVENDO VERTICE >>> g
FILA >> []
```



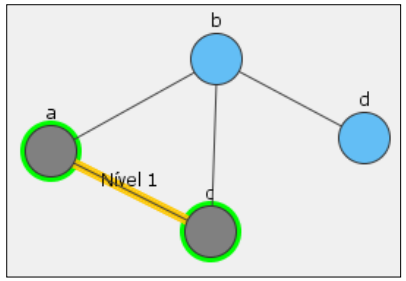

```
01 BUSCA-EM-LARGURA(G, s)
02     > INICIALIZAÇÃO
03     para cada u pertencente V[G] - {s} faça
04         cor[u] - branco
05         d[u] - infinito
06         py[u] - NIL
07     cor[s] - cinza
08     d[u] - 0
09     py[u] - NIL
10     Q - 0
11     ENQUEUE(Q, s)
12         enquanto Q diferente 0 faça
13             u - DEQUEUE(Q)
14             para cada v pertencente Adj[u] faça
15                 se cor[v] == branco então
16                     cor[v] - cinza
17                     d[v] - d[u] + 1
18                     py[v] - u
19                     ENQUEUE(Q, v)
20             cor[u] - preto
```

Desenvolvimento da ferramenta

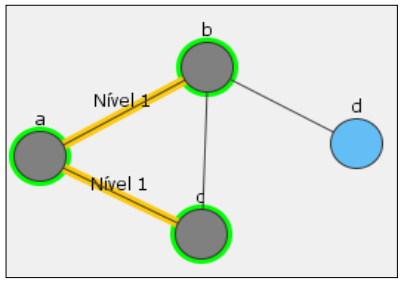
Funcionamento da navegação no algoritmo BFS



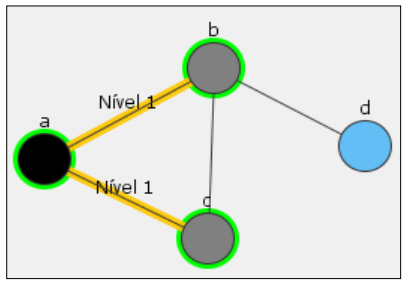
Identifica o vértice de origem selecionado.
Preenche uma lista com os vértices adjacentes à visitar.
*Colore o vértice e arestas.
**Destaca a linha no pseudocódigo.



Identifica que existem vértices à visitar, logo, remove o primeiro da fila e visita este vértice.
Vincula o vértice A como pai do vértice C.
Informa o nível do vértice na árvore.
Adiciona o vértice C na lista de vértices à processar.



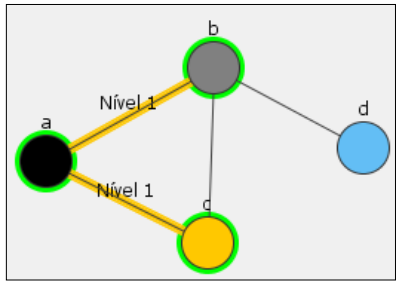
Identifica que existem vértices à visitar, logo, remove o primeiro da fila e visita este vértice.
Vincula o vértice A como pai do vértice B.
Informa o nível do vértice na árvore.
Adicionar o vértice B na lista de vértices à processar.



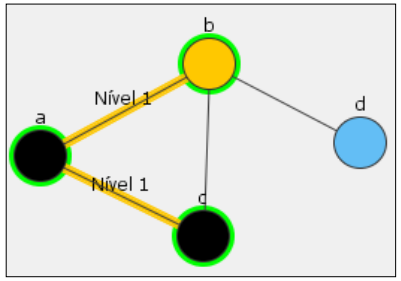
Identifica que não existem mais vértices à visitar, logo, o vértice A é marcado como vértice explorado.

*Ao final dos processamentos são atribuídas as cores aos vértices e arestas.
**Ao final dos processamentos é destacada a linha e ou bloco do pseudocódigo referente a execução.

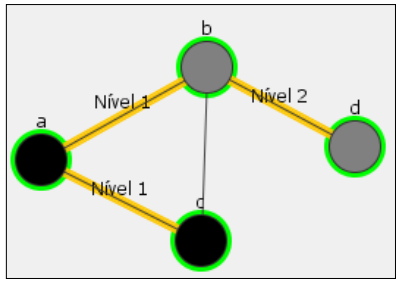
Desenvolvimento da ferramenta



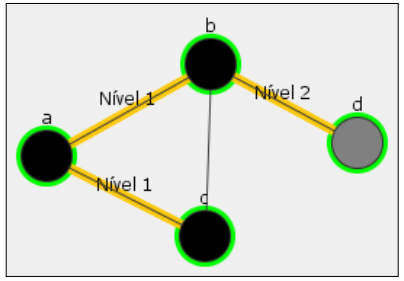
Identifica que existem vértices à processar.
Se houverem vizinhos adjacentes que não foram visitados ainda, adiciona-os a lista de vértices à visitar.



Identifica que o vértice C já foi visitado, explorado e não possui vizinhos à visitar.
Identifica que existem vértices a processar. se houverem vizinhos adjacentes que não foram visitados ainda, adiciona-os a lista de vértices a visitar.



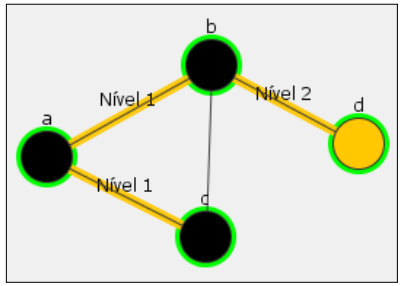
Identifica que existem vértices à visitar, logo, remove o primeiro da fila e visita este vértice.
Informa o nível do vértice na árvore.
Vincula o vértice B como pai do vértice D.
Adiciona o vértice D na lista dos vértices à processar.



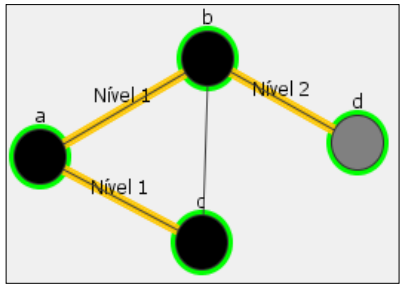
Identifica que não existem mais vértices à visitar, logo, o vértice B é marcado como vértice já explorado.

*Ao final dos processamentos são atribuídas as cores aos vértices e arestas.
**Ao final dos processamentos é destacada a linha e ou bloco do pseudocódigo referente a execução.

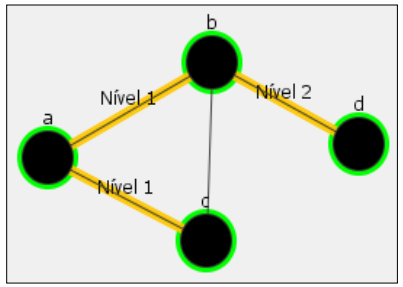
Desenvolvimento da ferramenta



Identifica que existem vértices à processar.
Se houverem vizinhos adjacentes que não foram visitados ainda, adiciona-os a lista de vértices à visitar.



Identifica que não existem mais vértices à visitar.
Identifica que não existem mais vértices à processar.
Retorna o vértice para o estado de visitado.

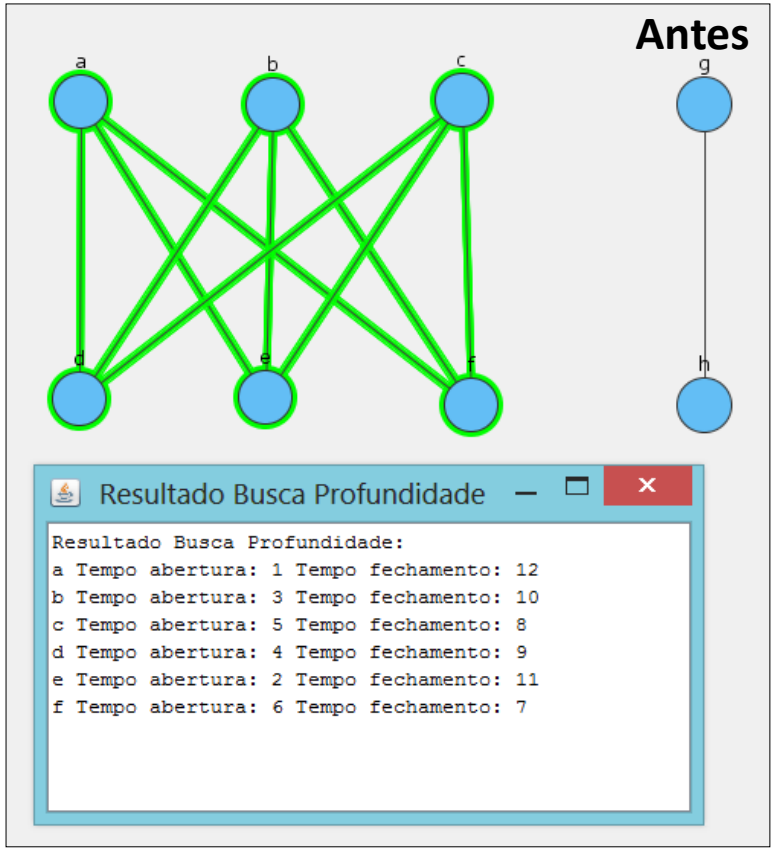


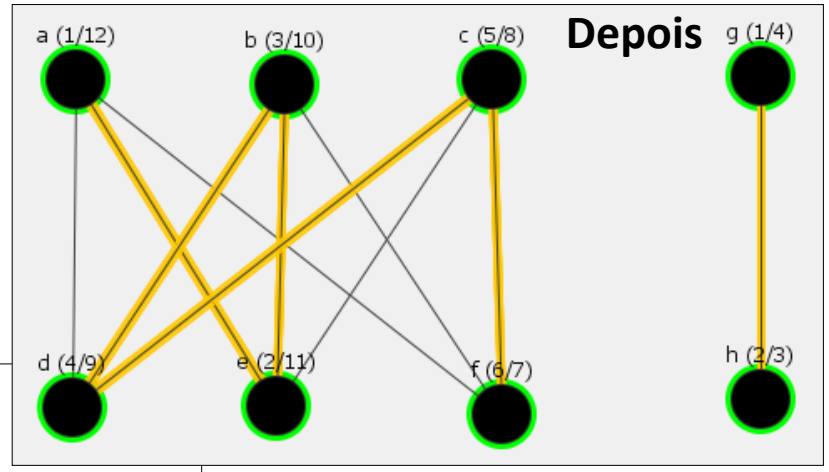
Identifica que todos os vértices adjacentes já foram visitados.
Marca o vértice D como vértice já explorado.

*Ao final dos processamentos são atribuídas as cores aos vértices e arestas.
**Ao final dos processamentos é destacada a linha e ou bloco do pseudocódigo referente a execução.

Desenvolvimento da ferramenta

Navegação no algoritmo DFS



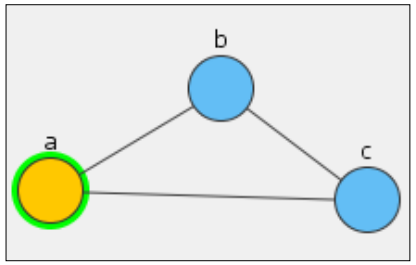



```
01 dfs(G)
02    para cada vértice u - V[G]
03        cor[u] - BRANCO
04    tempo - 0
05    para cada vértice u pertencente à V[G]
06        se cor[u] = BRANCO
07            DFS-VISIT(u)
08 DFS-VISIT(u)
09    cor[u] - CINZA
10    tempo - tempo + 1
11    d[u] - tempo
12    para cada vértice v pertencente à Adj(u)
13        se cor [v] = BRANCO
14            DFS-VISIT(v)
15    cor[u] - PRETO
16    f[u] - tempo - (tempo + 1)
```

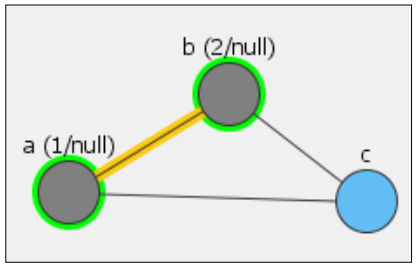
```
PROCESSANDO VERTICE >>>> g
LISTA DE VERTICES:
>>> h
EMPILHAMENTO > [g]

VISITANDO VERTICE > h

PROCESSANDO VERTICE >>>> h
EMPILHAMENTO > [g, h]

DESEMPILHA VERTICE > h
EMPILHAMENTO > [g]

DESEMPILHA VERTICE > g
EMPILHAMENTO > []
```

Desenvolvimento da ferramenta

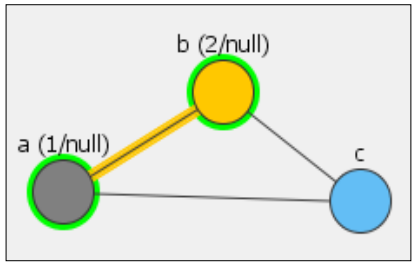
Funcionamento da navegação no algoritmo DFS



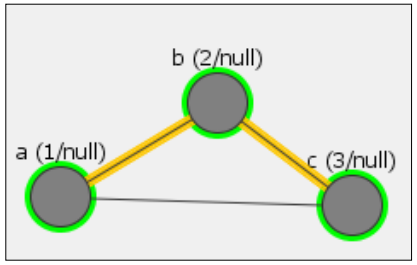
Identifica o vértice de origem selecionado.
Preenche uma pilha com os vértices à visitar.
*Colore o vértice e arestas.
**Destaca a linha no pseudocódigo.



Identifica que existem vértices à visitar, logo, remove o vértice da pilha e visita este vértice.
Vincula o vértice A como pai do vértice B.
Adiciona o tempo de abertura da visitação ao vértice de origem (A) como 1 e ao vértice B como (tempo de abertura de A + 1).
Adiciona o vértice B à pilha de vértices à processar.



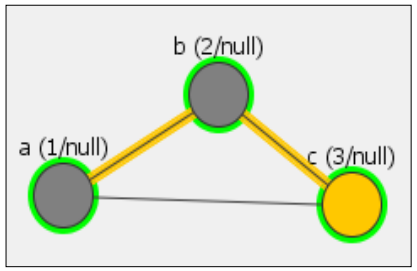
Identifica que existem vértices a processar.
Se houverem vizinhos adjacentes que não foram visitados ainda, adiciona-os a lista de vértices à visitar.



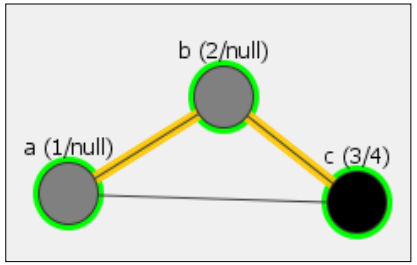
Identifica que existem vértices à visitar, logo, remove o vértice da pilha e visita este vértice.
Vincula o vértice B como pai do vértice C.
Adiciona o tempo de abertura da visitação ao vértice C como (tempo de abertura de B + 1).
Adiciona o vértice C a pilha de vértices à processar.

*Ao final dos processamentos são atribuídas as cores aos vértices e arestas.
**Ao final dos processamentos é destacada a linha e ou bloco do pseudocódigo referente a execução.

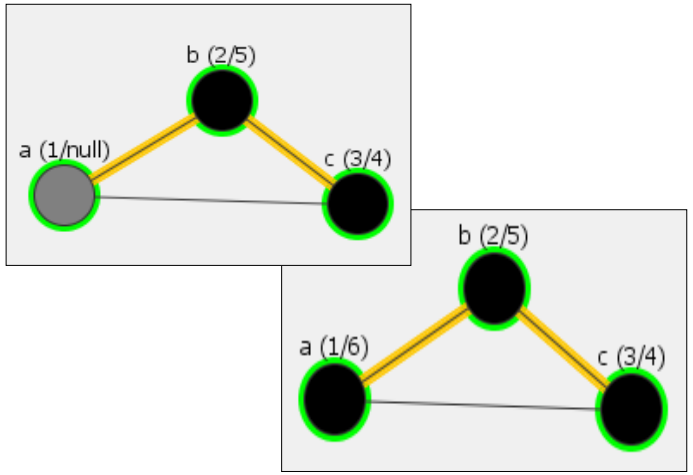
Desenvolvimento da ferramenta



Identifica que existem vértices a processar.
Se houverem vizinhos adjacentes que não foram visitados ainda, adiciona-os a pilha de vértices à visitar.



Identifica que não existem mais vértices à processar.
Identifica que não existem mais vértices à visitar.
Marca o vértice C como vértice já explorado.
Atribui um tempo de fechamento ao vértice que é Igual ao último tempo de fechamento + 1, ou então o tempo de abertura do próprio vértice + 1.
Desempilha o vértice da pilha de processamento.

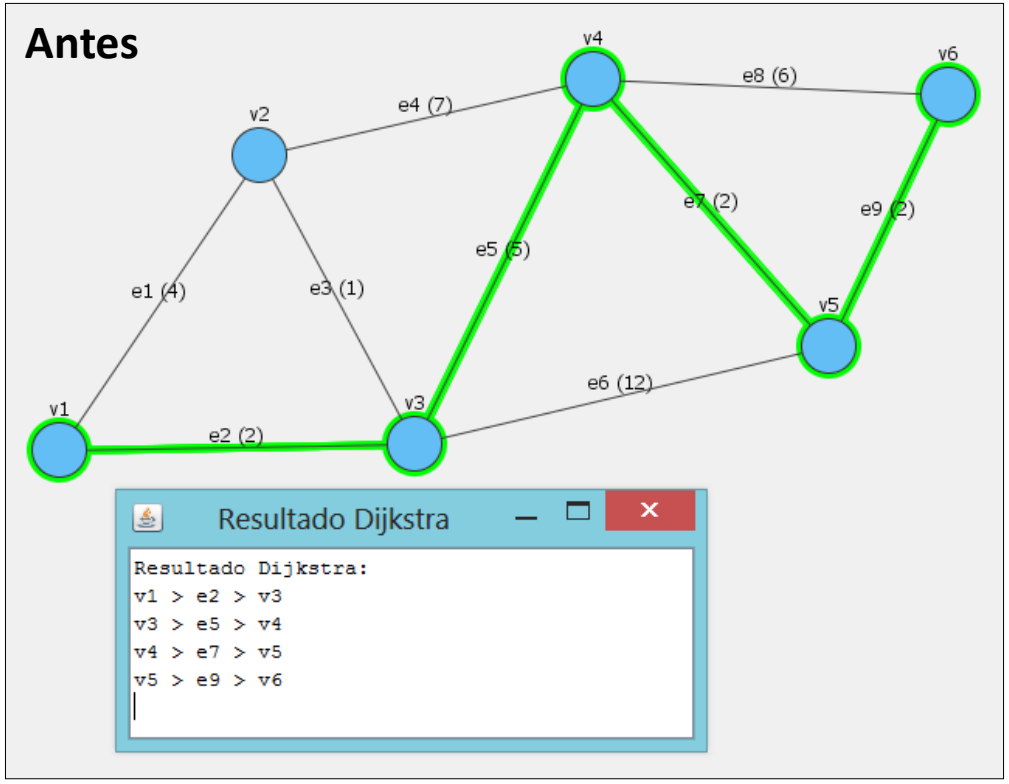


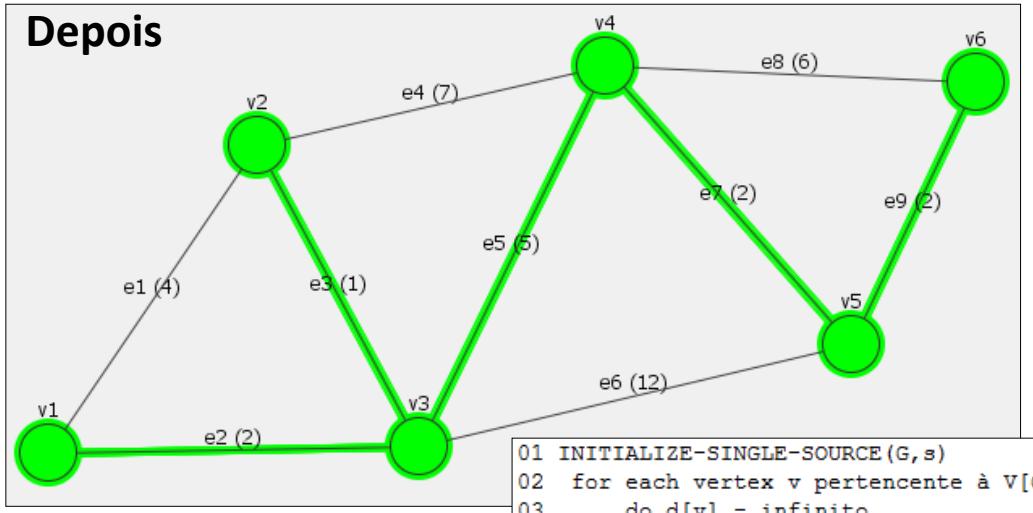
Identifica que não existem mais vértices à processar.
Identifica que não existem mais vértices à visitar.
Marca o vértice C como vértice já explorado.
Atribui um tempo de fechamento ao vértice que é Igual ao último tempo de fechamento + 1, ou então o tempo de abertura do próprio vértice + 1.
Desempilha o vértice da pilha de processamento.

*Ao final dos processamentos são atribuídas as cores aos vértices e arestas.
**Ao final dos processamentos é destacada a linha e ou bloco do pseudocódigo referente a execução.

Desenvolvimento da ferramenta

Navegação no algoritmo de Dijkstra



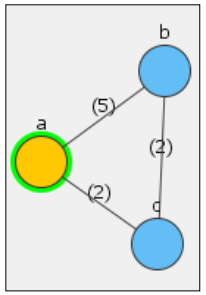


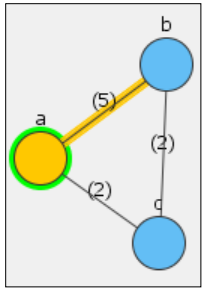

```
O menor caminho encontrado por vértice:
v1
 > v3
v3
 > v2
 > v4
v4
 > v5
v5
 > v6
v6
v2
O custo total do vertice > v1 para o vertice > v6 é = 11.0
```

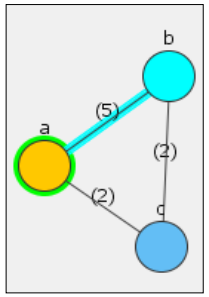
```
01 INITIALIZE-SINGLE-SOURCE(G,s)
02  for each vertex v pertencente à V[G]
03      do d[v] - infinito
04         py[v] - NIL
05  d[s] - 0
06 RELAX(u, v, w)
07    if d[v] > d[u] + w(u,v)
08       then d[v] - d[u] + w(u,v)
09       py[v] - u
10 DIJKSTRA(G, w, s)
11    INITIALIZE-SINGLE-SOURCE(G,s)
12    S - 0
13    Q - V[G]
14    while Q != 0
15       do u - EXTRACT-MIN(Q)
16           S - união [u]
17           for each vertex v pertencente à Adj[u]
18               do RELAX(u, v, w)
```

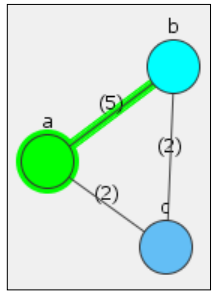
Desenvolvimento da ferramenta

Funcionamento da navegação no algoritmo de Dijkstra

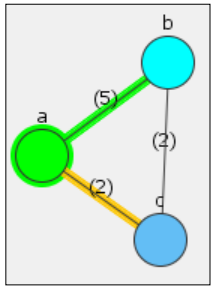


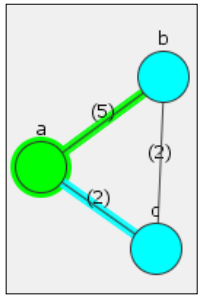


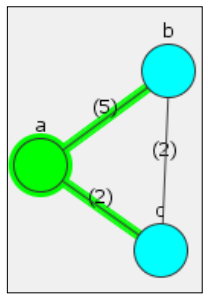




Identifica o vértice de origem.
Preenche uma lista com os vértices adjacentes e inicia a visita ao primeiro encontrado.
Identifica que o vértice B não possui um vértice pai, logo, atribui o vértice A como vértice pai.
Identifica que o caminho mínimo de A para B até o momento é através da aresta (A,B) com custo 5.
Adiciona o vértice B a lista de vértices já visitados.

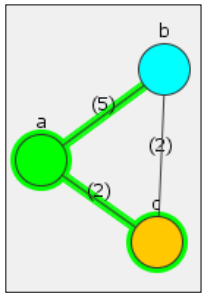


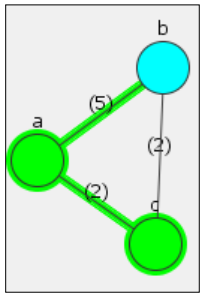


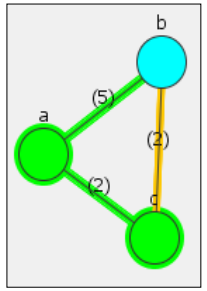


Visita o próximo vértice da lista dos vértices adjacentes.
Identifica que o vértice C não possui um vértice pai, logo, atribui o vértice A como vértice pai.
Identifica que o caminho mínimo de A para C até o momento é através da aresta (A,C) com custo 2.
Adicionar o vértice C a lista de vértices já visitados.

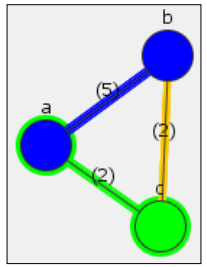
Desenvolvimento da ferramenta

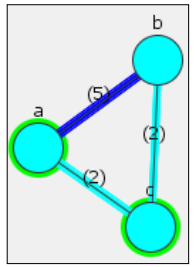


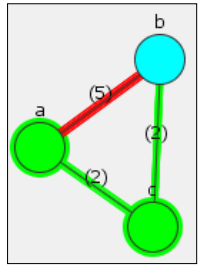




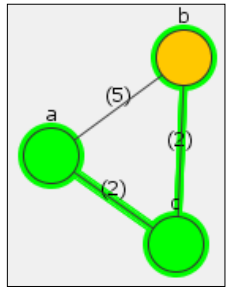
Identifica que não existem mais vértices adjacentes à visitar.
Marca o vértice C como processado. Percorre a lista de vértices já visitados e seleciona o vértice de menor custo, neste caso, vértice C com custo 2.
Identifica os vértices adjacentes não processados. Verifica o caminho de C para B.
Identifica que já existe um caminho para B através de A.
Preenche uma lista com os vértices do caminho atual (A,B).
Preenche uma lista com os vértices do caminho novo (A,C,B).
Executa a comparação entre o caminho (A,B) e (A,C,B) para identificar qual é o menor caminho.

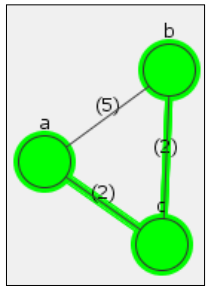






Percorre a lista do caminho atual e colore de azul escuro.
Percorre a lista do caminho novo e colore de azul escuro.
Identifica qual é o menor caminho entre os dois.
Diferencia o menor de verde e o maior de vermelho.





Identifica que não existem mais vértices adjacentes à visitar.
Marca o vértice B como processado.

Experimentos

Os experimentos foram realizados em computadores com sistema operacional Windows 8, 8.1 e 10. Foi testado o Java nas versões 7 e 8. Sete usuários participaram dos experimentos.

Foram realizados três experimentos:

- Fundamentais
- BFS
- DFS e Dijkstra

Experimentos

Para a realização dos experimentos foi elaborado um questionário para identificar o perfil dos usuários.

Sexo	100 % masculino
Idade	57,1 % tem entre 20 e 25 anos 28,6 % tem entre 25 e 30 anos 14,3 % tem entre 30 e 35 anos
Atua na área de computação?	85,7 % sim 14,3 % não
Nível de escolaridade	14,3 % ensino médio completo 71,4 % ensino superior incompleto 14,3 % ensino superior completo
Compreende o que é um grafo?	71,4 % sim 28,6 % não
Já estudou alguma disciplina envolvendo grafos?	42,9 % sim 57,1 % não

Experimentos

Experimentos fundamentais

Tarefas/Respostas	**Sim**	**Não**
Carregue ou desenhe um novo grafo com o mínimo de 6 vértices e 9 arestas, onde todos os vértices estejam interligados	100 %	
Se o grafo foi carregado, pule para a questão 4. Caso contrário, utilize o mouse clicando para criar vértices na tela, selecione cada vértice e atribua um nome a cada um	57,1%	42,9 %
Selecione cada par de vértice, clique com o botão direito e selecione conectar. Selecione cada par de vértice que possuir uma aresta e atribua um valor numérico aleatoriamente para cada	57,1%	42,9%
Utilize o mouse para clicar no botão “debug” no menu de ações da ferramenta e confirme a ativação do modo de debug	100 %	

Experimentos

Experimentos com o algoritmo BFS

Tarefas / Respostas	**Sim**	**Não**
Utilize o mouse no menu “Algoritmos” e selecione a opção Busca Largura. Siga as instruções apresentadas na tela	100%	
Utilize o mouse no botão de navegação para avançar cada passo do algoritmo. Analisando a coloração dos vértices e arestas em relação com o log de execução e o pseudocódigo em cada passo	100%	
Utilize o mouse nos botões de navegação para retroceder um passo ou para retroceder todos os passos. Após avance e retroceda os passos conforme desejar	100%	
Avance todos os passos do algoritmo e, utilizando o mouse para arrastar os vértices, organize-os se orientando através dos níveis indicados nas arestas, nível 1, nível 2.. seguindo a ordem	100%	

Experimentos

Experimentos com os algoritmos de DFS e Dijkstra

Tarefas / Respostas	**Sim**	**Não**
Utilize o mouse no menu “Algoritmos” e selecione a opção Busca Profundidade. Siga as instruções apresentadas na tela	100%	
Utilize o mouse no botão de navegação para avançar cada passo do algoritmo. Analisando a coloração dos vértices e arestas em relação com o log de execução e o pseudocódigo em cada passo	100%	
Utilize o mouse nos botões de navegação para retroceder um passo ou para retroceder todos os passos. Após avance e retroceda os passos conforme desejar	100%	
Utilize o mouse no menu “Algoritmos” e selecione a opção Dijkstra. Siga as instruções apresentadas na tela	100%	
Utilize o mouse para avançar os passos do algoritmo através do botão de navegação indicado na tela, verifique as legendas, logs e o pseudocódigo a cada passo	100%	

Experimentos

Foram realizadas duas avaliações com os sete usuários após os experimentos.

- Avaliação de usabilidade da ferramenta
- Avaliação de compreensão da navegação

Experimentos

Avaliação de usabilidade da ferramenta

Perguntas / Critérios de avaliação	Concordo totalmente	Concordo parcialmente	Foi imparcial	Discordo totalmente
1. De modo geral, você acredita que a ferramenta foi intuitiva e de fácil utilização?	71,4%	28,6%		
2. Para você, a interface gráfica e a disposição dos campos facilitaram a utilização da ferramenta?	28,6%	71,4%		
3. Você consegue se lembrar facilmente de como fazer as operações na ferramenta?	100%			
4. Você achou importante a utilização das legendas na ferramenta?	71,4%	14,3%	14,3%	
5. O modelo de navegação através dos botões foi prático e interativo?	57,1%	42,9%		
6. Você precisaria de ajuda para operar a ferramenta?	14,3%	28,6%		57,1%
7. A ferramenta em algum momento apresentou algum comportamento inesperado?	14,3%	42,9%	14,3%	28,6%
8. Seria adequado recomendar esta ferramenta para outras pessoas que estudam grafos?	100%			
9. A representação gráfica do passo a passo contribui para o interesse em grafos?	100%			

Experimentos

Avaliação de compreensão da navegação

Perguntas / Critérios de avaliação	Concordo totalmente	Concordo parcialmente	Foi imparcial	Discordo totalmente
1. Na sua opinião, as cores utilizadas na representação gráfica contribuíram para a compreensão dos algoritmos?	85,7%	14,3%		
2. O log de execução foi claro e detalhado o suficiente para acompanhar os passos dos algoritmos?	71,4%	28,6%		
3. O pseudocódigo ajudou para a compreensão dos algoritmos?	28,6%	14,3%	57,1%	
4. Você teve dificuldades para interpretar cada passo avançado nos algoritmos?	14,3%			85,7%
5. A funcionalidade de retroceder um passo do algoritmo se fez útil para compreensão?	100%			
6. A funcionalidade de retroceder todos os passos do algoritmo se fez útil para compreensão?	100%			
7. A combinação das legendas, logs, pseudocódigo e representação gráfica juntos fizeram sentido para você?	85,7%	14,3%		

Experimentos

Perguntas / Critérios de avaliação	Concordo totalmente	Concordo parcialmente	Foi imparcial	Discordo totalmente
8. A compreensão geral dos algoritmos melhora depois de utilizar a ferramenta?	100%			
9. Para o estudo de grafos, seria útil utilizar esta ferramenta?	100%			
10. No algoritmo de busca em largura, você compreendeu o vetor de roteamento em relação com a árvore gerada?	71,4%	28,6%		
11. No algoritmo de busca em profundidade, o empilhamento e desempilhamento do algoritmo de busca em profundidade ficou visível e compreensível?	100%			
12. No algoritmo de Dijkstra, as comparações dos caminhos foram compreendidas?	42,9%	57,1%		

Comparativo em relação aos trabalhos correlatos

Características / trabalhos	Lozada (2014)	Santos e Costa (2007)	Sangiorgi (2004)	Borba (2014)	Trabalho proposto
permite a criação de grafos livremente	Sim	Sim	Sim	Sim	Sim
suporta dígrafos	Sim	Sim	Sim	Sim	Sim
permite trocar as imagens dos vértices	Sim	Não	Sim	Não	Não
executa busca em grafos (BFS, DFS)	Sim	Sim	Não	Sim	Sim
executa árvore geradora mínima (Kruskal e Prim)	Não	Sim	Não	Sim	Sim
executa caminho mínimo entre vértices (Dijkstra e Bellman-Ford)	Sim	Sim	Não	Sim	Sim
analisa ciclos Hamiltonianos	Não	Não	Sim	Sim	Sim
analisa grafos bipartidos	Não	Não	Sim	Sim	Sim
apresenta informativos referente aos algoritmos	Não	Sim	Não	Não	Sim
apresenta material teórico quanto aos algoritmos	Não	Sim	Não	Não	Sim
apresenta recursos de apoio ao aprendizado dos algoritmos (legendas, código fonte)	Sim	Sim	Sim	Não	Sim
permite carregar um código fonte de terceiro para utilizar na execução	Não	Não	Sim	Não	Não
possui plataforma de ampla disponibilidade (web)	Não	Sim	Não	Não	Não
permite o acompanhamento passo a passo dos algoritmos	Não	Sim	Não	Não	Sim
permite o acompanhamento do log de execução passo a passo dos algoritmos	Não	Não	Não	Não	Sim

Conclusões

- Muitos usuários possuem conhecimento sobre o que é um grafo, porém, nunca o estudaram de forma adequada.
- A ferramenta se mostrou capaz de transmitir a mecânica dos algoritmos de forma visual e intuitiva, pois, os usuários que não possuíam conhecimento sobre a disciplina de teoria dos grafos conseguiram compreender completamente os algoritmos.
- Mesmo sem visualizar o pseudocódigo e as legendas, os usuários conseguiram compreender bem o funcionamento dos algoritmos.

Limitações

- O processo de navegação do algoritmo de Dijkstra ficou limitado a funcionalidade de avançar passo a passo e de retroceder todos os passos. O motivo se da devido a metodologia utilizada no início das implementações, onde, foi adotada uma estratégia que se mostrou mais complexa do que o esperado para implementar o retrocesso de passos do algoritmo. Então foi dado continuação a implementação dos demais algoritmos (BFS e DFS) já adotando a estratégia mais adequada. Sendo assim, não houve tempo hábil para finalizar esta funcionalidade, devido este erro no início das implementações.

Extensões

- Implementação da funcionalidade de retroceder passo a passo a navegação do algoritmo de Dijkstra.
- Implementação de atalhos para se navegar na execução dos algoritmos.
- Implementação da navegação completa do algoritmo de Prim.
- Implementação da navegação completa do algoritmo de Kruskal.

Extensões

- Implementação da matriz de adjacência e de incidência para apresentar na tela durante a navegação dos algoritmos
- Alteração na interface gráfica para permitir a ocultação do *log* e do pseudocódigo.
- Efetuar melhorias de desempenho nos algoritmos de navegação.
- Efetuar melhorias de usabilidade.

Demonstração

Obrigado!