

FACULDADE DE COMPUTAÇÃO

LUCAS ROSSI RABELO

ANTÔNIO CARLOS NETO

PAULO RENATO CAMPOS BARBOSA

**Universidade Federal de Uberlândia – UFU**

TRABALHO SOBRE GRAFOS, REFERENTE À DISCIPLINA DE ALGORITMOS E ESTRUTURAS DE DADOS DOIS.

Uberlândia

2017

LUCAS ROSSI RABELO – 11611BCC044

ANTÔNIO CARLOS NETO – 11611BCC054

PAULO RENATO CAMPOS BARBOSA – 11611BCC006

**Universidade Federal de Uberlândia – UFU**

ANÁLISE DE IMPLEMENTAÇÃO

Uberlândia

2017

* Introdução:

Neste trabalho, iremos comparar o desempenho dos métodos de percorrimento de grafos de acordo com duas formas de implementação distintas, a matriz de adjacências e a lista de adjacências. Efetuaremos a análise e a comparação em três grafos distintos, cada qual implementado de uma forma em linguagem C.

Utilizaremos a biblioteca em C “time.h” para efetuar a comparação do tempo de execução de cada método de percorrimento nas duas implementações.

A partir dos dados da análise, concluiremos qual o método de percorrimento mais optimizado para qual forma de implementação.

* Materiais e métodos:

Utilizamos para a implementação do código, a linguagem de programação procedimental C, através da interface de desenvolvimento “Code-Blocks” e das bibliotecas stdio.h, stdlib.h e time.h, esta ultima foi utilizada para obter o tempo de execução das funções de percorrimento.

Os grafos utilizados foram:



Grafo 1: Grafo de 15 Vértices e 20 Arestas

Grafo 2 : Grafo de 6 vértices e 8 arestas



Grafo 3 : Grafo de 9 vértices e 13 arestas

Ambos os grafos foram implementados em um arquivo de texto, com nomes dj2.txt, dj.txt e dj3.txt respectivamente, de modo que o programa possa buscar e trabalhar com o grafo do arquivo.

* Análises.

Os pesos de cada aresta foram representados da seguinte forma “[ Custo | Delay | Capacidade | Tráfego atual do link], organizados do lado da aresta correspondente.

* Análise dos métodos de percorrimento para o Grafo 1 em listas e matriz de adjacência.

Foram usados os seguintes pesos no Grafo 1:



Representação do Grafo 1 no arquivo de texto:

0 -> 1 = [ 1 | 4 | 6 | 3 ]  
0 -> 2 = [ 5 | 4 | 8 | 9 ]  
0 -> 3 = [ 10 | 11 | 1 | 7 ]  
1 -> 4 = [ 3 | 12 | 2 | 4 ]  
2 -> 5 = [ 1 | 1 | 1 | 4 ]  
2 -> 6 = [ 7 | 1 | 2 | 1 ]  
3 -> 7 = [ 3 | 6 | 9 | 13 ]  
3 -> 8 = [ 1 | 10 | 9 | 8 ]   
4 -> 9 = [ 8 | 4 | 6 | 6 ]  
5 -> 9 = [ 5 | 5 | 14 | 15 ]  
5 -> 10 = [ 2 | 4 | 6 | 9 ]  
6 -> 10 = [ 15 | 5 | 15 | 6 ]  
7 -> 10 = [ 2 | 4 | 8 | 16 ]  
8 -> 10 = [ 12 | 6 | 3 | 1 ]  
9 -> 11 = [ 9 | 6 | 9 | 1 ]  
9 -> 12 = [ 3 | 2 | 1 | 6 ]  
10 -> 12 = [ 21 | 2 | 1 | 1 ]  
11 -> 13 = [ 4 | 3 | 2 | 4 ]  
12 -> 14 = [ 10 | 12 | 1 | 1 ]  
13 -> 14 = [ 1 | 4 | 5 | 6 ]

* Implementação das funções em Matriz de adjacências.
* Algoritmo de Dijkstra

Nesse tópico testaremos as funções de percorrimento no Grafo 1, pela implementação de Matriz de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de Dijkstra:

* Vértice emissor: 0
* Quantidade de receptores: 14
* Receptores: Vértices 1 ,2,3,4,5,6,7,8,9,10,11,12,13,14
* Valor da métrica: 32,00
* Tempo gasto com execução: 41 ms

Para a métrica Delay e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo:

* Valor da métrica: 24,00
* Tempo gasto com execução: 49ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 99,00
* Tempo gasto com execução: 2042ms
* Busca em profundidade

Para efetuar o algoritmo de busca em profundidade, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da Métrica: 44,00
* Tempo gasto com execução: 52ms

Para métrica Delay e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da Métrica: 63,00
* Tempo gasto com execução: 54ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Profundidade, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: Valor da Métrica: 171,00
* Tempo gasto com execução: 1477ms
* Busca em Largura

Para efetuar o algoritmo de busca em largura, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em largura, obtemos os seguintes valores:

* Valor da Métrica: 36,00
* Tempo gasto com execução: 1938ms

Para a métrica Delay e o algoritmo de busca em largura, obtemos os seguintes valores;

* Valor da Métrica: 25,00
* Tempo gasto com execução: 828ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Largura, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 104,00
* Tempo gasto com execução: 2420ms
* Implementação das funções em Lista de Adjacências
* Algoritmo de Dijkstra

Nesse tópico testaremos as funções de percorrimento no Grafo 1, pela implementação de Lista de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de Dijkstra:

* Vértice emissor: 0
* Quantidade de receptores: 14
* Receptores: Vértices 1 ,2,3,4,5,6,7,8,9,10,11,12,13,14
* Valor da métrica: 32,00
* Tempo gasto com execução: 53 ms

Para a métrica Delay e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo:

* Valor da métrica: 24,00
* Tempo gasto com execução: 52ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 99,00
* Tempo gasto com execução: 3995ms
* Busca em profundidade

Para efetuar o algoritmo de busca em profundidade, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da Métrica: 48,00
* Tempo gasto com execução: 52ms

Para métrica Delay e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da Métrica: 60,00
* Tempo gasto com execução: 55ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Profundidade, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: Valor da Métrica: 202,00
* Tempo gasto com execução: 1500ms
* Busca em Largura

Para efetuar o algoritmo de busca em largura, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em largura, obtemos os seguintes valores:

* Valor da Métrica: 52,00
* Tempo gasto com execução: 796ms

Para a métrica Delay e o algoritmo de busca em largura, obtemos os seguintes valores;

* Valor da Métrica: 32,00
* Tempo gasto com execução: 740ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Largura, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 175,00
* Tempo gasto com execução: 3354ms
* Análise dos métodos de percorrimento para o Grafo 2 em listas e matriz de adjacência.

Foram usados os seguintes pesos no Grafo 2:



Representação do Grafo 2 no arquivo de Texto:

0 -> 1 = [3|2|1|1]  
0 -> 2 = [1|8|0|2]  
1 -> 3 = [8|5|2|3]  
1 -> 4 = [2|3|3|5]  
1 -> 2 = [1|7|2|3]  
2 -> 4 = [5|5|0|7]  
3 -> 4 = [2|3|8|5]  
3 -> 5 = [1|1|3|0]

* Implementação das Funções em Matriz de Adjacências
* Algoritmo de Dijkstra

Nesse tópico testaremos as funções de percorrimento no Grafo 2, pela implementação de Matriz de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de Dijkstra:

* Vértice Emissor: 0
* Quantidade de Receptores: 5
* Receptores: Vértices 1, 2, 3, 4 e 5
* Valor da Métrica: 10,00
* Tempo gasto com execução: 2 ms

Para a métrica Delay e o algoritmo de Dijkstra, utilizaremos os mesmos valores que utilizamos para Custo:

* Valor da Métrica: 17,00
* Tempo gasto com execução: 2ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 90,00
* Tempo gasto com execução: 2267ms
* Busca em profundidade

Para efetuar o algoritmo de busca em profundidade, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da Métrica: 14,00
* Tempo gasto com execução: 2ms

Para métrica Delay e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da Métrica: 19,00
* Tempo gasto com execução: 2ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Profundidade, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: Valor da Métrica: 95,00
* Tempo gasto com execução: 2149ms
* Busca em Largura

Para efetuar o algoritmo de busca em largura, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em largura, obtemos os seguintes valores:

* Valor da Métrica: 8,00
* Tempo gasto com execução: 1089ms

Para a métrica Delay e o algoritmo de busca em largura, obtemos os seguintes valores;

* Valor da Métrica: 9,00
* Tempo gasto com execução: 1352ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Largura, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 86,00
* Tempo gasto com execução: 5345ms
* Implementação das Funções em Lista de Adjacências
* Algoritmo de Dijkstra

Nesse tópico testaremos as funções de percorrimento no Grafo 2, pela implementação de Lista de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de Dijkstra:

* Valor da métrica: 10,00
* Tempo gasto em execução: 2ms

Para a métrica Delay e o algoritmo de Dijkstra, utilizaremos os mesmos valores que utilizamos para Custo:

* Valor da métrica: 17,00
* Tempo gasto em execução: 2ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da métrica: 45,00
* Tempo gasto em execução: 1563ms
* Busca em profundidade

Para efetuar o algoritmo de busca em profundidade, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da métrica: 11,00
* Tempo gasto em execução: 2ms

Para métrica Delay e o algoritmo de busca em profundidade, obtemos os seguintes valores:

* Valor da métrica: 19,00
* Tempo gasto em execução: 2ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Profundidade, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da métrica: 12,38
* Tempo gasto em execução: 1271ms
* Busca em Largura

Para efetuar o algoritmo de busca em largura, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em largura, obtemos os seguintes valores:

* Valor da Métrica: 12,00
* Tempo gasto com execução: 1381ms

Para a métrica Delay e o algoritmo de busca em largura, obtemos os seguintes valores;

* Valor da Métrica: 13,00
* Tempo gasto com execução: 1344ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Largura, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 89,00
* Tempo gasto com execução: 3482ms
* Análise dos métodos de percorrimento para o Grafo 3 em listas e matriz de adjacência.

Foram usados os seguintes pesos no Grafo 3



Representação do Grafo 3 no arquivo de texto:

0 -> 1 = [ 5 | 0 | 3 | 6 ]  
0 -> 3 = [ 5 | 5 | 7 | 2 ]

1 -> 2 = [ 5 | 7 | 8 | 4 ]

1 -> 3 = [ 2 | 6 | 5 | 3 ]

1 -> 4 = [ 0 | 2 | 2 | 5 ]

3 -> 2 = [ 7 | 4 | 9 | 8 ]

3 -> 7 = [ 4 | 9 | 8 | 3 ]

3 -> 8 = [ 5 | 3 | 0 | 1 ]

4 -> 5 = [ 2 | 4 | 8 | 8 ]

2 -> 5 = [ 1 | 4 | 1 | 7 ]

2 -> 8 = [ 8 | 2 | 2 | 3 ]

6 -> 8 = [ 9 | 1 | 5 | 9 ]

6 -> 7 = [ 6 | 8 | 2 | 8 ]

* Implementação das Funções em Matriz de Adjacências
* Algoritmo de Dijkstra

Nesse tópico testaremos as funções de percorrimento no Grafo 3, pela implementação de Matriz de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de Dijkstra:

* Vértice Emissor: 0
* Numero de Receptores : 8
* Receptores: Vértices 1,2,3,4,5,6,7 e 8
* Valor da Métrica: 25,00
* Tempo gasto com a execução: 7ms

Para a métrica Delay e o algoritmo de Dijkstra, utilizaremos os mesmos dados que utilizamos para o Custo:

* Valor da Métrica: 14,00
* Tempo gasto com a execução: 7ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 13,29
* Tempo gasto com a execução: 1804ms
* Busca em Profundidade

Nesse tópico testaremos as funções de percorrimento no Grafo 3, pela implementação de Matriz de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de busca em profundidade, ainda seguindo os mesmos parâmetros de antes:

* Valor da Métrica: 38,00
* Tempo gasto com a execução: 12ms

Para a métrica Delay e o algoritmo de busca em profundidade, utilizaremos os mesmos dados que utilizamos para o Custo:

* Valor da Métrica: 29,00
* Tempo gasto com a execução: 8ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em profundidade, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 15,57
* Tempo gasto com a execução: 1132ms
* Busca em Largura

Para efetuar o algoritmo de busca em largura, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em largura, obtemos os seguintes valores:

* Valor da Métrica: 31,00
* Tempo gasto com execução: 1912ms

Para a métrica Delay e o algoritmo de busca em largura, obtemos os seguintes valores;

* Valor da Métrica: 22,00
* Tempo gasto com execução: 1412ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Largura, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 32,33
* Tempo gasto com execução: 2857ms
* Implementação das Funções em Lista de Adjacências
* Algoritmo de Dijkstra

Nesse tópico testaremos as funções de percorrimento no Grafo 3, pela implementação de Matriz de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de Dijkstra:

* Valor da Métrica: 25,00
* Tempo gasto em execução: 11ms

Para a métrica Delay e o algoritmo de Dijkstra, utilizaremos os mesmos dados que utilizamos para o Custo:

* Valor da Métrica: 14,00
* Tempo gasto em execução: 8ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 13,29
* Tempo gasto em execução: 1755ms
* Busca em Profundidade

Nesse tópico testaremos as funções de percorrimento no Grafo 3, pela implementação de Lista de adjacências.

Primeiro utilizaremos a métrica Custo e utilizaremos o algoritmo de busca em profundidade, ainda seguindo os mesmos parâmetros de antes:

* Valor da Métrica: 36,00
* Tempo em execução: 7ms

Para a métrica Delay e o algoritmo de busca em profundidade, utilizaremos os mesmos dados que utilizamos para o Custo:

* Valor da Métrica: 20,00
* Tempo em execução: 7ms

Para a métrica Utilização máxima do enlace e o algoritmo de Dijkstra, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 15,14
* Tempo em execução: 2128ms
* Busca em Largura

Para efetuar o algoritmo de busca em largura, utilizaremos os mesmos parâmetros utilizados para o algoritmo de Dijkstra.

Para métrica Custo e o algoritmo de busca em largura, obtemos os seguintes valores:

* Valor da Métrica: 36,00
* Tempo gasto com execução: 886ms

Para a métrica Delay e o algoritmo de busca em largura, obtemos os seguintes valores;

* Valor da Métrica: 14,00
* Tempo gasto com execução: 740ms

Para a métrica Utilização máxima do enlace e o algoritmo de Busca em Largura, utilizaremos os mesmos dados inseridos para o Custo e um tamanho de mensagem de 80:

* Valor da Métrica: 32,33
* Tempo gasto com execução: 2126ms
* Conclusão:

No grafo 1, percebe-se que tanto a matriz de adjacências quanto a lista de adjacências possuem um desempenho semelhante nas buscas por profundidade e no algoritmo de Dijkstra; porém, na busca em largura é possível notar uma diferença grande de desempenho entre as mesmas.  
O algoritmo de busca em largura foi mais eficiente na matriz de adjacências ao retornar a métrica Utilização Máxima do Enlace (UME) (2420ms contra 3354ms), e foi mais eficiente na lista de adjacências ao retornar a métrica Custo (796ms contra 1938ms).

Já no grafo 2, é possível notar diferença de desempenho em todos os algoritmos de busca; em Dijkstra aplicado à matriz, a busca pelo UME foi menos eficiente que na aplicação à lista (2267ms contra 1563ms).  
No algoritmo de busca por profundidade a situação se repete, novamente na busca pelo UME o algoritmo de busca se mostra mais eficiente quando aplicado à lista de adjacências.  
No algoritmo de busca em largura, a diferença nas métricas de Custo e Delay não é significante, enquanto que, novamente, na UME à uma grande diferença a favor da lista de adjacências (3482ms contra 5345ms).

No grafo 3, não há diferença significante no tempo de execução do algoritmo de Dijkstra para nenhuma métrica em nenhuma implementação; enquanto que no algoritmo de busca por profundidade a matriz possui um desempenho melhor ao buscar pelo UME (1132ms contra 2128ms).  
Na busca em largura, a lista de adjacências apresenta um desempenho significantemente melhor em todas as métricas.