Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0503 - Algoritmos e Estruturas de Dados II

# Relatório Exercício 05

**Grafos, Floyd-Warshall e Excentricidade**

Aluno: Cauê Pereira Cermak 8936864
Professor: Leonardo Tórtoro Pereira

Julho
2022

# Conteúdo

# 1 Introdução

1. Proposta Esse exercício consiste na na solução logística de uma empresa chamada Braba Log, que está gastando demais com combustível e precisa encontrar a melhor cidade para estabelecer um centro de distribuição de mercadorias. Além disso ela também quer saber qual é a cidade mais periférica da malha logistica e qual é a cidade mais distante da cidade mais periférica. Foi fornecida a malha logística da empresa, onde cada cidade está ligada por uma rodovia(aresta), a cidade está representada no mapa por coordenadas geográficas(x,y).

A implementação desenvolvidq seguiu a seguinte ordem ler o arquivo que contém os vértices e arestas(1), representá-las através de um vetor strings, ou seja uma matriz de char. Após armazenar os vertices, foi feita a leitura des arestas, esta representação foi feita através de uma matriz de ordem igual ao número de vértices previamente informado. Foi aplicado o algorítmo de *floyd warshall* sobre essa matriz, depois calculada a distância euclediana entre os nós e representada em um vetor(excentricidade). Tendo o vetor excentricidade em mãos era ,essencialmente , encontrar o maior(mais periférico) e menor(mais central) valor.

O código foi desenvolvido em linguagem C, devido a maior familiaridade com tal linguagem em relação a lingugagem Java. Tal desenvolvimento foi trabalhoso, devido eu estar desenvolvendo sozinho e devido a dificuldades conceituais.

2.Tive bastante dificuldade de entender o conceito de distância euclediana e como implementá-la, não encontrei materiais na internet.

# 2 Desenvolvimento

- Entendimento do problema[6][4][5]
- Estudo de uma representação viável dos dados
- Desenvolvimento das funções que tratam os valores inseridos por stdin
- Desenvolvimento das funções mais complexas
- Fazer relatório no Latex[3][1][2][7]

## 2.1 Ordem da implementação

1. Ler número de vértices
2. Criar vetor de vértices, representado por strings(matriz)
3. Ler e montar matriz de arestas
4. Chamar função que executa o algoritmo Floyd-Warshall
5. Criar o vetor excentricidade a partir da matriz retornada pela função "floywarshall"
6. Por fim, encontrar o vértice central, periférico e o mais distante do periférico

## 2.2 Representação Visual do Código

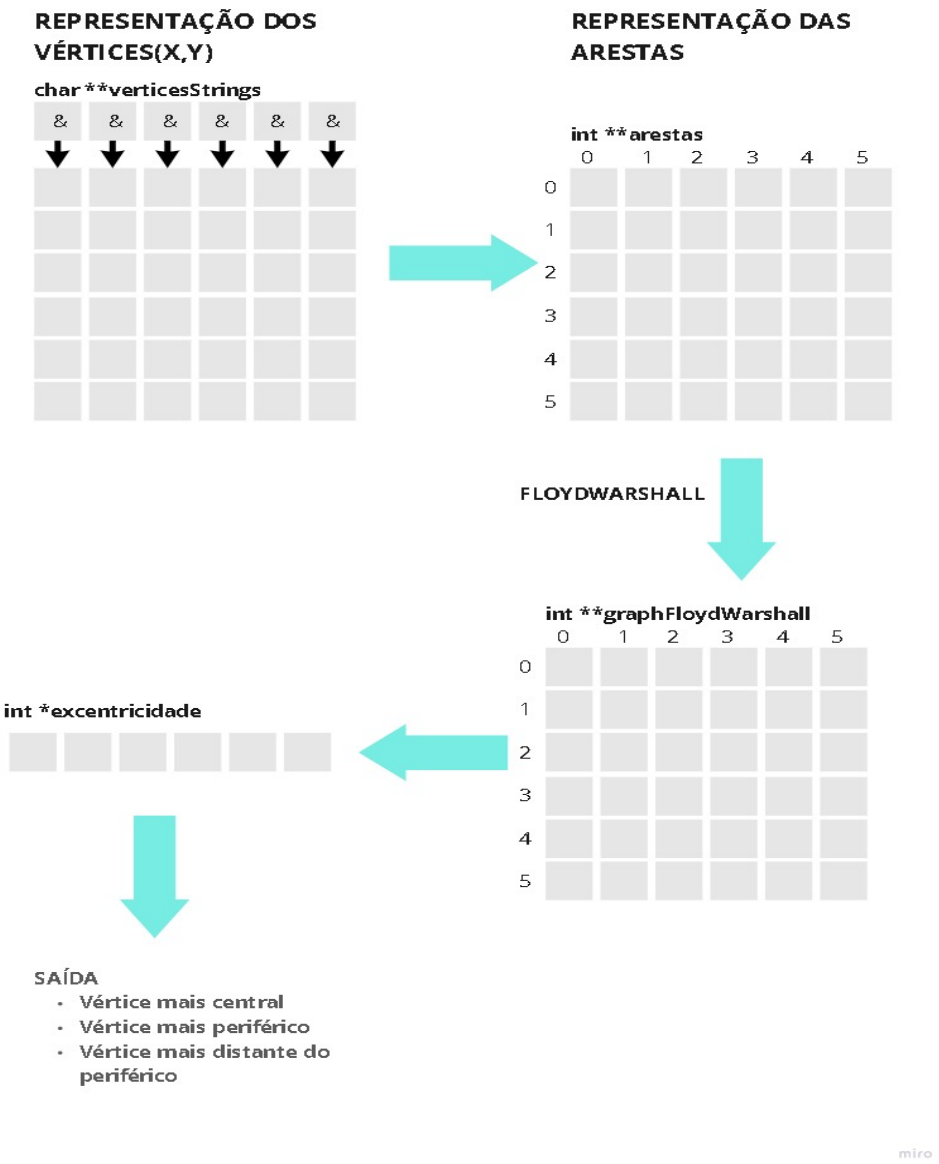


Figura 1: Representação visual do código

## 2.3 Estruturação da main()

```c
int main() {

  //ler nV -> numberOfVertices
  int numberOfVertices;
  numberOfVertices = read_i_line();

  //cria vetor de vertices/strings(matrix)
  char **vertices;
  vertices = create_vetor_vertices(numberOfVertices);//O(n)

  //ler e montar matriz de aresta
  int **graph;
  graph = read_edges(numberOfVertices, vertices);
  printf("\n_____\nArestas\n");
  print_matrix(graph,numberOfVertices);//O(n²), n = numberOfVertices

  //chamando funcao que executa algoritimo floydwarshall
  printf("\nFloydWarshall\n");
  int **graph_floyd = floydWarshall_(graph,numberOfVertices);//O(n³+ 2n²), n = numberOfVertices
  print_matrix(graph_floyd, numberOfVertices);//O(n²), n = numberOfVertices

  //cria-se o vetor excentricidade
  int *vetorExcentricidade ;
  vetorExcentricidade = criaVetorExcentricidade(graph_floyd, numberOfVertices);//O(n²), n = numberOfVertices
  printVetorVertices(vertices, numberOfVertices);//O(n), n = numberOfVertices

  //encontrando vertice central, periferico e vertice mais distante do periferico
  encontraVerticeCentralPeriferico_DistPeriferico(vetorExcentricidade, graph_floyd, vertices, numberOfVertices);//O(2n),

  return 0;
}
```

Figura 2: Representação visual do código

## 2.4 Complexidade

As principais funções e suas respectivas complexidades são:

```c
create_vetor_vertices(numberOfVertices);//O(n)
print_matrix(graph,numberOfVertices);//O(n²), n = numberOfVertices
floydWarshall_(graph,numberOfVertices);//O(2n²), n = numberOfVertices
print_matrix(graph_floyd, numberOfVertices);//O(n²), n = numberOfVertices
criaVetorExcentricidade(graph_floyd, numberOfVertices);//O(n²), n =
numberOfVertices
printVetorVertices(vertices, numberOfVertices);//O(n), n = numberOfVertices
encontraVerticeCentralPeriferico_DistPeriferico(vetorExcentricidade,
graph_floyd, vertices, numberOfVertices);//O(2n), n = numberOfVertices
```

Figura 3: Principais funções.

Logo resultando em $n^3 + 5n^2 + 4n + cte$, sendo assim o código tendo uma complexidade de O(n3), onde n é o número de vértices.

## 2.5 Desenvolvimento

Salvamos os vértices em um vetor de strings ou, também, pode ser visto como uma matriz.

O algoritmo de Floyd-Warshall tem como objetivo calcular o caminho mais curto entre todos os pares de vértices em um grafo, retornando uma matriz com tais distâncias entre os vértices.

O algoritmo que cria o vetor excentricidade é bem simples, passando por cada uma das linhas da matriz retornada pelo algoritmo Floyd-Warshall e salvando para cada vértice - índice do vetor - o caminho mais longo, retornando tal vetor.

Por último, temos a função que percorre o vetor excentricidade e encontra o vértice mais central(com menor valor), o vértice mais periférico(com maior valor) e o vértice mais distante do vértice periférico. Esta função guarda os índices de tais vértices, busca-se no vetor onde estão armazenados os vértices(x,y), imprimindo-os.

# 3 Resultados

O resultado saiu como esperado. Sinceramente, tive que fazer algumas pequenas manipulações, pois usando o caso de teste dado no enunciado do exercício, gera-se um vetor excentricidade com mais de um vértice central e mais de um vértice periférico.

## 3.1 Entradas do exercício

1. Número de nós
2. Nós, composto pelas coordenadas do plano cartesiano "x,y"
3. Número de arestas
4. Arestas, ligando os nos "X1,Y1":"X2,Y2"

10
0,0
3,7
10,4
2,9
4,3
7,1
8,5
9,10
5,8
6,2
40
0,0:4,3
0,0:6,2
0,0:7,1
3,7:2,9
3,7:5,8
3,7:4,3
10,4:8,5
10,4:7,1
10,4:6,2
10,4:9,10
2,9:3,7
2,9:5,8
2,9:4,3
4,3:0,0
4,3:3,7
4,3:2,9
4,3:6,2
4,3:7,1
7,1:0,0
7,1:10,4
7,1:4,3
7,1:6,2
7,1:8,5
8,5:10,4
8,5:7,1
8,5:6,2
8,5:9,10
8,5:5,8
9,10:5,8

9,10:8,5
9,10:10,4
5,8:3,7
5,8:2,9
5,8:9,10
5,8:8,5
6,2:0,0
6,2:10,4
6,2:4,3
6,2:7,1
6,2:8,5

## 3.2 Saída esperada

1. Vértice mais central
2. Vértice mais periférico
3. Vértice mais distante do vertice mais periférico

6,2
0,0
9,10

## 3.3 Saída do Programa

Fiz um resultado mais detalhado para analisar o que foi descrito no começo deste tópico, a respeito de ter mais de um vértice central e mais de um vértice periférico, isso pode ser visto através do vetor com os valores de excentricidade de cada vértice.

```
Arestas

I - I - I - I - 1 - 1 - I - I - I - 1 -
I - I - I - 1 - 1 - I - I - I - 1 - I -
I - I - I - I - I - 1 - 1 - 1 - I - 1 -
I - 1 - I - I - 1 - I - I - I - 1 - I -
1 - 1 - I - 1 - I - 1 - I - I - I - 1 -
1 - I - 1 - I - 1 - I - 1 - I - I - 1 -
I - I - 1 - I - I - 1 - I - 1 - 1 - 1 -
I - I - 1 - I - I - I - 1 - I - 1 - I -
I - 1 - I - 1 - I - I - 1 - 1 - I - I -
1 - I - 1 - I - 1 - 1 - 1 - I - I - I -

FloydWarshall

2 - 2 - 2 - 2 - 1 - 1 - 2 - 3 - 3 - 1 -
2 - 2 - 3 - 1 - 1 - 2 - 2 - 2 - 1 - 2 -
2 - 3 - 2 - 3 - 2 - 1 - 1 - 1 - 2 - 1 -
2 - 1 - 3 - 2 - 1 - 2 - 2 - 2 - 1 - 2 -
1 - 1 - 2 - 1 - 2 - 1 - 2 - 3 - 2 - 1 -
1 - 2 - 1 - 2 - 1 - 2 - 1 - 2 - 2 - 1 -
2 - 2 - 1 - 2 - 2 - 1 - 2 - 1 - 1 - 1 -
3 - 2 - 1 - 2 - 3 - 2 - 1 - 2 - 1 - 2 -
3 - 1 - 2 - 1 - 2 - 2 - 1 - 1 - 2 - 2 -
1 - 2 - 1 - 2 - 1 - 1 - 1 - 2 - 2 - 2 -

Vetor excentricidade:
3 -3 -3 -3 -3 -2 -2 -3 -3 -2 -

Vertices:
0,0 - 3,7 - 10,4 - 2,9 - 4,3 - 7,1 - 8,5 - 9,10 - 5,8 - 6,2 -

maior excentricidade: 3, indice:0
menor excentricidade: 2, indice:9
```

```
------------------
```

RESULTADO:

```
vertice Periferico = 0,0
vertice Central = 6,2
Mais distante do vertice periferico = 9,10

------------------
```

# 4 Conclusão

Em relação ao Latex, ao contrário da última entrega, foi bem mais fácil escrever o relatório, estou até começando a gostar.
Tem sido agradável observar como estou tendo familiaridade com códigos, estruturas, formas de pensar as estruturas a cada trabalho que passa. No começo, sinto que me sentia rígido pensando e implementando os códigos, agora me sinto mais flexível, pensando sempre em soluções mais fáceis de fazer o mesmo problema.
Poderia ter criado uma melhor estruturação das bibliotecas, uma coerência maior de nome de variáveis e funções, porém devido eu estar com muitas matérias e muitas demandas não consegui passar esse "pente-fino". O fato de ter desenvolvido em C, fez com que não conseguisse reutilizar os códigos feitos pelo professor em aula, isso também é um ponto positivo, pois consegui desenvolver tudo.
Vejo que no início da matéria tinha uma dificuldade imensa em lidar com coisas que hoje em dia são simples e triviais, por exemplo, manipulação de arquivos, desenvolvimento de algoritmos. É nítido ver como minha mente está conseguindo abstrair mais os problemas, pensar em uma solução e percorrê-la. Estou conseguindo entender o objetivo da matéria e não mais temer os algoritmos de estruturas de dados; são apenas algoritmos e eles "independem" do tipo de dado manipulado.
Pouco falei do algoritmo Floyd-Warshall, mas como disse no parágrafo interior, isso não tá sendo muito relevante pra mim, estou aprendendo coisas que julgo bem mais interessantes do que apenas algoritmos, apesar de estar aprendendo isso também.
Agradeço ao professor Leornardo Tórtoro Pereira pela forma que ministra as aulas e cativa os alunos com sua nítida paixão pela programação.

# Referências


[1] Colaborativo. Comandos básicos de latex. https://pt.wikibooks.org/wiki/Latex.

[2] Colaborativo. Comandos básicos de latex. https://pt.wikibooks.org/wiki/Latex.

[3] Andrade Doherty. Uma introdução ao latex. 2000.

[4] University of San Francisco. Visualization: Floyd-warshall all-pairs shortest path. https://www.cs.usfca.edu/ galles/visualization/Floyd.html.

[5] National University of Singapure. visualising data structures and algorithms through animation. https://visualgo.net/en.

[6] Leonardo Tórtoro Pereira. Grafos - floyd-warshall e vértice mais central.

[7] tex.stackecvhange. Comunidade latex online. https://tex.stackexchange.com/.