

# Atividade Prática 05

## “Percursos em Grafos”

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana  
Curso de Engenharia de Computação  
Disciplina de Estrutura de Dados - ED62A  
Prof. Dr. Rafael Gomes Mantovani

### 1 Descrição

Implemente um programa que apresente o resultado de uma busca em um grafo. O programa deve receber como parâmetro um arquivo de entrada contendo:

- i) o número de vértices de um grafo;
- ii) as arestas do grafo;
- iii) o método de percurso a ser empregado ('B' - BFS ou 'D' - DFS); e
- iv) a representação a ser usada ('M' - matriz de adjacência ou 'L' - lista de adjacência).

Os vértices serão todos numéricos (**no máximo 20, iniciando em 1**) e as arestas serão representadas por pares ordenados  $(x,y)$ , onde  $x$  é a origem e  $y$  o destino. **Caso o grafo seja não orientado, serão fornecidas no arquivo de entrada tanto a aresta  $(x,y)$  quanto a  $(y,x)$** , portanto não há necessidade de tratamento especial quanto à orientação na construção do grafo.

#### 1.1 Entrada

As Figuras 1a (BFS) e 1b (DFS) mostram exemplos de arquivos de entrada. A primeira linha indica o tipo de estrutura usada para codificar os grafos (M ou L). A segunda linha indica qual algoritmo será executado (B ou D). Na terceira linha são fornecidas todas as arestas que compõem o grafo. E na quarta linha, caso o algoritmo a ser executado seja BFS, é apresentado o vértice inicial.

#### 1.2 Saída

Caso o arquivo de entrada especifique a execução de uma BFS, o programa deve apresentar em um arquivo de saída (Fig 1c): a sequência de vértices visitados e o correspondente tempo de descoberta. Lembre-se que o tempo de descoberta, nesse caso, é denotado por 'd'. Caso a opção de execução seja DFS, o programa deve apresentar no arquivo de saída (Fig 1d) a sequência de vértices visitados e os correspondentes tempos de abertura e fechamento, na ordem em que eles são percorridos.

```
input_bfs
M
B
(1,2)(1,3)(1,4)(2,1)(2,3)(4,3)
4|
```

(a) Exemplo de arquivo de entrada para BFS.

```
input_dfs
L
D
(1,2)(1,3)(1,4)(2,1)(2,3)(4,3)
```

(b) Exemplo de arquivo de entrada para DFS.

```
output_bfs
4[d=0]
3[d=1]
1[d=999]
2[d=999]
```

(c) Exemplo de arquivo de saída para BFS.

```
output_dfs
1[d=1, f=8]
2[d=2, f=5]
3[d=3, f=4]
4[d=6, f=7]
```

(d) Exemplo de arquivo de saída para DFS.

Figura 1: Valores de entrada e correspondentes arquivos de saída gerado pelo programa.

## 2 Observações

- Na busca em largura (BFS) sempre será passado o vértice inicial ('s'). Inicialize  $d[s] = 0$  e considere o valor 999 como infinito.
- Na busca em profundidade (DFS), implemente a função externa (DFS) que chama a função de busca (DFS-Visit) para gerar uma floresta de busca em profundidade, se for o caso, iniciando também no vértice 1.
- Convencione a ordem de visitação dos adjacentes de um vértice de forma crescente de acordo com o seu número.

Além disso, a manipulação do programa é controlada usando-se os argumentos **argc** e **argv** da função **main**. Para executar o programa por linha de comando, deve-se obedecer o seguinte padrão:

**<nome do programa> <arquivo de entrada> <arquivo de saída>**

Exemplo:

**percursoGrafos entrada.txt saida.txt**

onde:

- “percursoGrafos” é o nome do arquivo da aplicação (avls.c);
- “entrada.txt” é o arquivo de entrada;
- “saida.txt” é o arquivo de saída.

### 3 Orientações gerais

- Implementar também o controle de erros, para lidar com exceções que possam ocorrer (arquivo não aberto, erro de leitura, padrão incorreto, etc);
- Para acompanhamento do desenvolvimento, criar um repositório individual com o código desenvolvido no github Classroom, por meio do link: <https://classroom.github.com/a/uVCT8cCH>. Os repositórios serão privados, com acesso apenas do professor e do aluno.
- Entrega do programa final: via Moodle. O aluno deve submeter o fonte no link da atividade disponibilizado na página da disciplina no Moodle.
- **Data de entrega: 14/12/2019.**
- Os códigos desenvolvidos por cada aluno serão também verificados por ferramentas de plágio. Códigos iguais/similares terão nota zero.

### 4 Links úteis

Arquivos em C:

- <https://www.inf.pucrs.br/~pinho/LaproI/Arquivos/Arquivos.htm>
- <https://www.geeksforgeeks.org/basics-file-handling-c/>
- <https://www.programiz.com/c-programming/c-file-input-output>

Argumentos de Linha de comando (argc e argv):

- [https://www.tutorialspoint.com/cprogramming/c\\_command\\_line\\_arguments.htm](https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm)
- <http://linguagemc.com.br/argumentos-em-linha-de-comando/>
- [http://www.univasf.edu.br/~marcelo.linder/arquivos\\_pc/aulas/aula19.pdf](http://www.univasf.edu.br/~marcelo.linder/arquivos_pc/aulas/aula19.pdf)
- [http://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-31\\_Argumentos\\_linha\\_comando.html](http://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-31_Argumentos_linha_comando.html)
- <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node145.html>

### Referências

- [1] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.
- [2] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.
- [3] Adam Drozdek. Estrutura De Dados E Algoritmos Em C++. Cengage, 2010.