

# Atividade Prática 06

## Buscas em Grafos

---

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana  
Curso de Engenharia de Computação  
Disciplina de Estrutura de Dados 1 - EDCO3A  
Prof. Dr. Rafael Gomes Mantovani

---

### Instruções:

- Leia todas as instruções corretamente para poder desenvolver sua atividade/programa;
- Evite plágio (será verificado por meio de ferramentas automatizadas). Faça seu programa com os seus nomes de variáveis e lógica de solução. Plágios identificados anularão as atividades entregues de todos os envolvidos.
- Adicione comentários nos códigos explicando seu raciocínio e sua tomada de decisão. Porém, não exagere nos comentários, pois a própria estrutura do programa deve ser auto-explicativa.
- Salve sua atividade em um arquivo único, com todas as funções e procedimentos desenvolvidos. É esse **arquivo único** que deverá ser enviado ao professor.

## 1 Descrição da atividade

Implemente um programa que apresente o resultado de buscas (em largura e profundidade) em um grafo qualquer.

## 2 Entradas do programa

O programa deve receber como parâmetro um arquivo de entrada contendo diferentes linhas, em ordem:

1. a representação a ser usada para codificar o grafo. O valor 'M' designa implementação do grafo por meio de matriz de adjacência, e 'L' como lista de adjacência;
2. o método de percurso a ser empregado ('B' - BFS ou 'D' - DFS). Nossos exemplos de caso de teste serão todos com valor 'B' (*Breath-first search*), que é a busca em largura;
3. as arestas que compõem o grafo;

- o vértice inicial da busca ( $s$ );

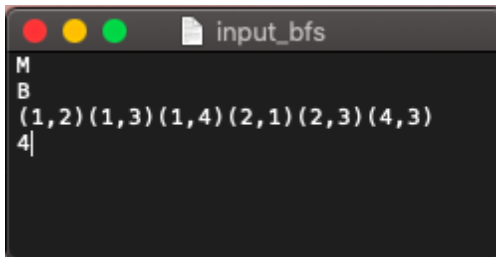
Os vértices serão todos numéricos (**iniciando em 1 e com valor máximo de 20**) e as arestas serão representadas por pares ordenados  $(x,y)$ , onde  $x$  é a origem e  $y$  o destino. **Caso o grafo seja não orientado, serão fornecidas no arquivo de entrada tanto a aresta  $(x,y)$  quanto a  $(y,x)$**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

## 2.1 Entrada

A Figura 1a mostra um exemplo de arquivo 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, para execução do algoritmo BFS, é apresentado o vértice inicial.

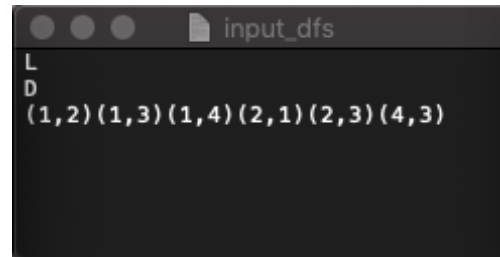
## 2.2 Saída

Como saída de 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 no por 'd'.



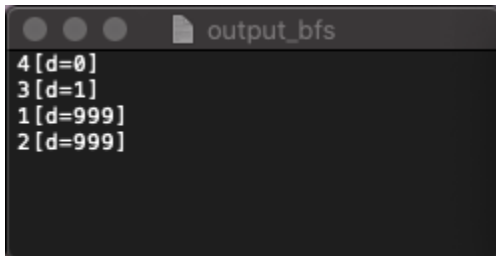
```
M
B
(1,2) (1,3) (1,4) (2,1) (2,3) (4,3)
4|
```

(a) Exemplo de arquivo de entrada para BFS.



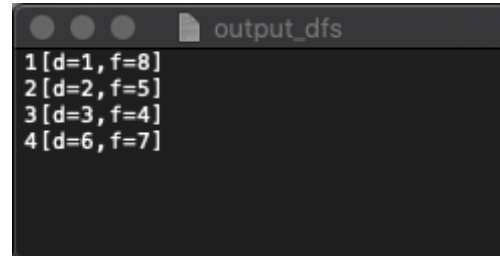
```
L
D
(1,2) (1,3) (1,4) (2,1) (2,3) (4,3)
```

(b) Exemplo de arquivo de entrada para DFS.



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

(c) Exemplo de arquivo de saída para BFS.



```
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.

## 3 Algumas observações importantes

- Na busca em largura (BFS) sempre será passado o vértice inicial ( $s$ ). Inicialize  $d[s] = 0$  e considere o valor 999 como infinito.

- Convencione a ordem de visitação dos adjacentes de um vértice de forma crescente de acordo com o seu número.
- a quantidade de vértices deve ser descoberta durante a aplicação.

## 4 Orientações gerais

Além da funcionalidade desejada, implementar também o controle de erros, para lidar com exceções que possam ocorrer, como por exemplo:

- problemas nas aberturas dos arquivos de entrada e saída;
- arquivo de entrada vazio (sem informação);
- arquivo de entrada fora do padrão esperado;
- etc.

Opcionalmente, para acompanhamento do desenvolvimento, pode-se criar um repositório individual no `github`.

### 4.1 Critério de correção

A nota na atividade será contabilizada levando-se em consideração alguns critério:

1. pontualidade na entrega;
2. não existir plágio;
3. completude da implementação (tudo foi feito);
4. o código compila e executa;
5. uso de `argc` e `argv` para controle dos arquivos de teste;
6. implementar o parser para entrada dos dados via arquivo texto;
7. implementação correta das estruturas necessárias;
8. legibilidade do código (identação, comentários nos blocos mais críticos);
9. implementação dos controles de erros (arquivos de entrada inválidos, e erros no programa principal);
10. controle de memória: chamar o destrutor e desalocar a memória de tudo se usar estruturas dinâmicas, fechar os arquivos, etc;
11. executar corretamente os casos de teste.

## 4.2 Dados para envio da atividade

Ao elaborar seu programa, crie um único arquivo fonte (.c) seguindo o padrão de nome especificado:

`ED1-<ANO>-<SEMESTRE>-AT06-BFS-DFS-<NOME>.c`

Exemplo:

`ED1-2022-1-AT06-BFS-DFS-RafaelMantovani.c`

A entrega da atividade será via Moodle: o link será disponibilizado na página da disciplina.

## 5 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.