

# Ciclos eulerianos: uma aproximação usando algoritmo de Fleury

Leandro L. A. Vieira,<sup>1</sup> Mateus P. Silva<sup>2</sup>

<sup>1</sup>Ciência da Computação – Universidade Federal de Viçosa (UFV-caf)  
– Florestal – MG – Brasil

(leandrolazaro<sup>1</sup>, mateus.p.silva<sup>2</sup>)@ufv.br

**Resumo.** *Este trabalho trata-se da implementação do algoritmo de Fleury para a solução do ciclo euleriano de um grafo.*

## 1. Introdução

A teoria dos grafos é um dos principais métodos de modelagem de algoritmos graças à sua simplicidade, o que a faz tangenciar a universalidade na solução dos mais variados problemas. Ademais, os avanços tecnológicos fazem com que o tamanho dos dados cresçam exponencialmente todos os anos, tornando obrigatório o estudo de formas mais eficientes de operações computacionais, especialmente as com grafos.

Um grafo euleriano apresenta um ciclo em que é possível iniciar um percurso num vértice arbitrário, passar por cada uma das arestas apenas uma vez, e retornar ao vértice inicial. Esse ciclo é chamado Ciclo Euleriano. É condição necessária que todos os vértices tenham grau par para que haja ciclo(s) euleriano(s).

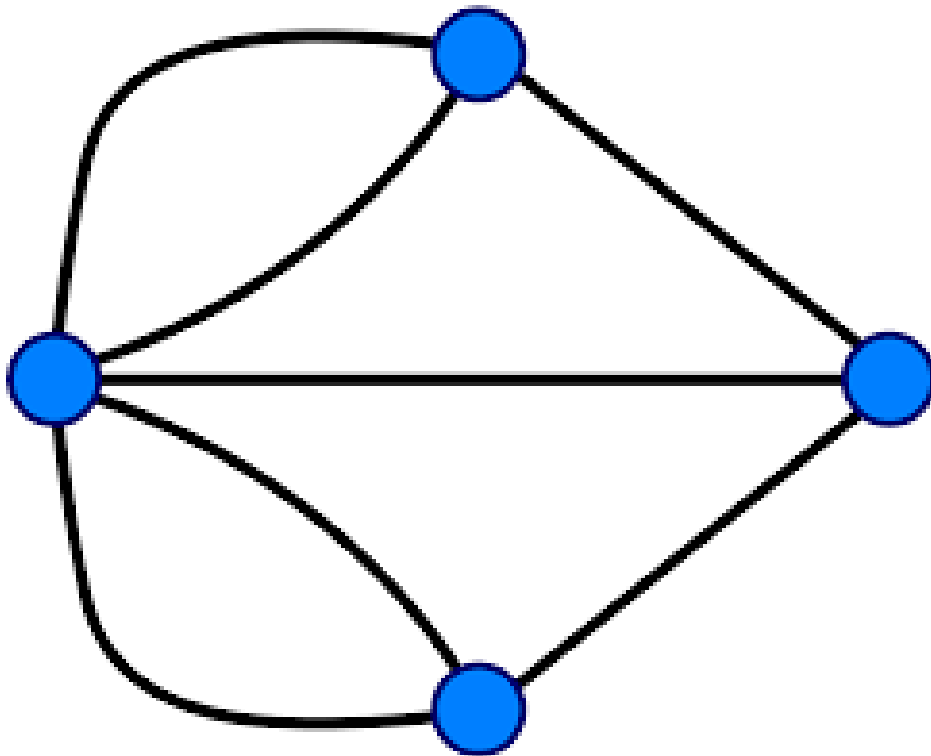


Figura 1. Grafo com ciclo euleriano

## 2. Como executar

Digite “make” para compilar e “make run” para abrir um menu autoexplicativo.

## 3. Implementação

Utilizamos a linguagem C++ no paradigma orientado a objetos pelo bom desempenho e quantidade de estruturas de dados prontas, como pilas e filas. Criamos uma única classe que representa um grafo utilizando listas de adjacência. Além disso, há também um inteiro que conta quantos vértices há no grafo.

```
1  class Graph
2  {
3      private:
4          unsigned int V;
5          list<int> *adj;
6      public:
7          Graph(string filename);
8          ~Graph();
9
10         void addEdge(int u, int v);
11         void rmvEdge(int u, int v);
12         void printEulerTour();
13         void printEulerUtil(int s);
14         int DFSCount(int v, bool visited[]);
15         bool isValidNextEdge(int u, int v);
16         bool isEulerian();
17     [... ]
18 }
```

Há funções para adicionar vértice, remover, exibir um caminho euleriano, caminho euleriano até um dado vértice, contar o grau de um vértice, verificar se o próximo vértice é valido (ou seja, não é ponte) e verificar se o grafo é euleriano.

### 3.1. O algoritmo de Fleury

O algoritmo funciona iterando todas as arestas do vértice, e, caso ela não seja uma ponte, ela é removida, e o algoritmo é chamado recursivamente para o outro vértice conectado a essa aresta.

```
1 void Graph::printEulerUtil(int u)
2 {
3     list<int>::iterator i;
4     for (i = adj[u].begin(); i != adj[u].end(); ++i)
5     {
6         int v = *i;
7
8
9         if (v != -1 && isValidNextEdge(u, v))
10        {
11            cout << (v+1) << " ";
12            rmvEdge(u, v);
13            printEulerUtil(v);
```

```

14     }
15 }
16 }

```

### 3.2. Criador de grafos aleatórios eulerianos

Também foi implementado um gerador muito simples de grafos eulerianos como arquivos de entrada para o algoritmo. Ele cria todas as arestas como arestas paralelas, garantindo a condição necessária para ser euleriano.

## 4. Resultados

### 4.1. Grafo da proposta

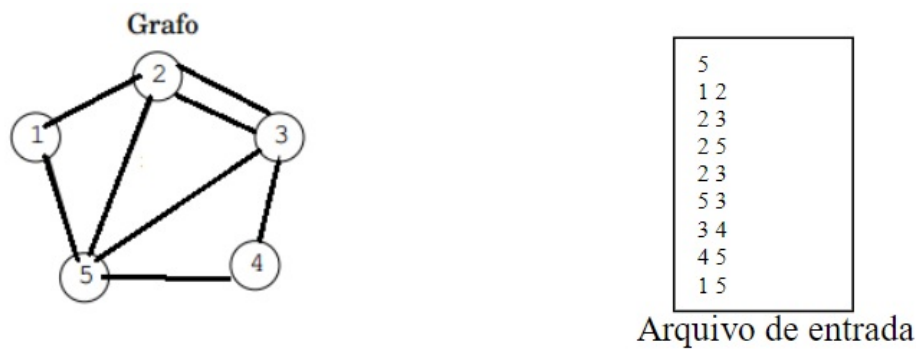


Figura 2. Grafo pedido

Utilizando o grafo da proposta, foi obtido o seguinte ciclo:

```

matelementalista@DESKTOP-0KBOGII: /mnt/e/Documents/git/fleury_simple-tp2-grafos-ufv
matelementalista@DESKTOP-0KBOGII:/mnt/e/Documents/git/fleury_simple-tp2-grafos-ufv$ make run
./fleury.o
-----
TP2 de Grafos - Ciclo de Euler
-----
1 - Abrir arquivo e mostrar ciclo
2 - Gerar grafo euleriano aleatorio em arquivo
9 - Sair
-----
1
Digite o nome do arquivo. Ele deve estar na pasta input:
proposta.txt
1 2 3 5 2 3 4 5 1
Digite 0 para mostrar o menu novamente

```

Figura 3. Ciclo obtido na execução do programa

Trata-se de um ciclo válido.

### 4.2. Grafo euleriano com 100 vértices e 150 arestas

Trata-se do arquivo max.txt.

<b>100</b>						
94	56	64	76	60	84	34
56	94	44	69	84	60	23
56	80	69	44	2	3	82
80	56	71	99	3	2	73
15	54	99	71	52	24	17
54	15	1	86	24	52	74
17	18	86	1	54	4	17
18	17	5	69	4	54	21
84	1	69	5	61	10	37
1	84	55	88	10	61	50
33	59	88	55	73	15	42
59	33	69	87	15	73	50
81	59	87	69	97	93	24
59	81	98	1	93	97	10
54	39	1	98	2	94	24
39	54	98	51	94	2	10
6	98	51	98	94	51	24
98	6	39	3	51	94	48
41	70	3	39	96	32	51
70	41	48	80	32	96	47
61	55	80	48	53	95	77
55	61	24	8	95	53	47
72	22	8	24	63	28	41
22	72	86	47	28	63	30
87	10	47	86	2	100	41
10	87	82	72	100	2	34
6	37	72	82	27	35	92
37	6	56	87	35	27	34
76	64	87	56	23	34	80
21	34	34	21	35	99	39
						82
						57
						12
						22
						86
						55
						86
						7
						65
						7
						24
						45
						77
						48
						11
						86
						11
						3
						15
						33
						15
						99
						35

Tabela 1. Quantidade de vértices em negrito. Arestas demarcadas como adjacências.

```

mateamentalista@DESKTOP-0KBOGI: /mnt/e/Documents/git/fleury_simple-tp2-grafos-ufv
mateamentalista@DESKTOP-0KBOGI:/mnt/e/Documents/git/fleury_simple-tp2-grafos-ufv$ make run
./fleury.o
-----
TP2 de Grafos - Ciclo de Euler
-----
1 - Abrir arquivo e mostrar ciclo
2 - Gerar grafo euleriano aleatorio em arquivo
9 - Sair
-----
1
Digite o nome do arquivo. Ele deve estar na pasta input:
max.txt
1 84 60 84 17 18 17 74 17 84 1 86 47 77 48 80 56 94 2 3 39 54 15 73 82 72 22 12 22 72 82 57 82 73 15 33 59 81 59 20 5
9 33 80 39 35 27 35 64 76 64 35 99 71 99 35 39 3 11 86 55 61 10 87 69 44 69 5 69 87 56 80 33 15 54 4 54 39 80 48 51 9
8 6 37 21 2 100 2 94 51 48 77 47 86 1 98 51 94 56 87 10 24 8 24 52 24 45 24 10 61 55 88 55 86 11 3 2 21 34 23 93 97 9
3 23 19 23 34 92 34 21 37 6 98 1
Digite 0 para mostrar o menu novamente

```

Figura 4. Ciclo obtido na execução de max.txt

### **4.3. Considerações finais**

Através deste trabalho, foi possível implementar o algoritmo de Fleury que cria um ciclo euleriano. Foi criado o algoritmo de forma o mais simples possível, sem nenhum diferencial, porém de forma eficiente.

### **4.4. Referências**

Boa parte do código foi baseada nesta implementação:  
<https://www.geeksforgeeks.org/fleury-s-algorithm-for-printing-eulerian-path/>