Grafo Euleriano

Gabriel Santos Barbosa e Savio Lopes Rabelo

Grafos

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Resumo

Neste trabalho mostramos como verificar se um grafo é Euleriano e, após isso, como apresentar um circuito Euleriano para tal grafo. Utilizamos o algoritmo de Hierholzer para realizar essa tarefa. Os métodos empregados foram implementados na linguagem de programação Python.

1. Introdução

O problema de encontrar ciclos Eulerianos foi discutido pela primeira vez no século XVIII. Em uma antiga cidade chamada Königsberg, existiam sete pontes, e as pessoas se perguntavam se era possível sair e voltar para o mesmo local atravessando cada ponte exatamente uma vez. Uma forma de representar esse problema matematicamente, é através do uso de grafos, em que a estrutura correspondente é chamada de Circuito Euleriano, onde as pontes representam as arestas e os vértices representam os locais de partida. Dessa forma, o problema de encontrar um ciclo Euleriano pode ser interpretado como partir e voltar para o mesmo vértice cruzando cada aresta uma única vez. Esse conceito foi introduzido por Leonard Euler em 1736.

Grafos que possuem um circuito Euleriano são chamados Grafos Eulerianos. Uma das principais condições para um grafo ser Euleriano é que todos os vértices precisam ser de grau par. Entretanto, essa condição não é suficiente, pois também é necessário que o grafo seja conexo [1]. Euler provou que uma condição necessária para a existência de circuitos Eulerianos é de que todos os vértices tenham grau par, e afirmou, sem prova de que grafos conexos com todos os vértices pares tem um circuito Euleriano. A primeira prova completa desta última afirmação foi publicada em 1873 por Carl Hierholzer [2].

Há, ainda, grafos com os chamados caminhos Eulerianos. Um caminho Euleriano é um caminho no grafo que visita todas as arestas exatamente uma

vez. Se houver 2 vértices de grau ímpar em um determinado grafo G, existirá um caminho Euleriano em G. Nesse caso, ao se acrescentar uma aresta ligando estes dois vértices, o novo grafo passa a ser um circuito Euleriano.

Pode-se assim enunciar um corolário do Teorema de Euler para Grafos como sendo: *Um grafo G conexo possui caminho euleriano se e somente se ele tem exatamente zero ou dois vértices de grau ímpar.*

2. Problemática

Forneça um algoritmo que verifique se um grafo não direcionado é ou não Euleriano. Caso seja, o algoritmo deve fornecer um *tour* Euleriano. A entrada é um arquivo de texto que contém uma aresta por linha. Cada vértice é representado por uma *string*. Um exemplo de conteúdo de um arquivo de entrada válido é:

```
vermelho azul
vermelho verde
azul verde
azul amarela
amarela vermelho
```

3. Implementação

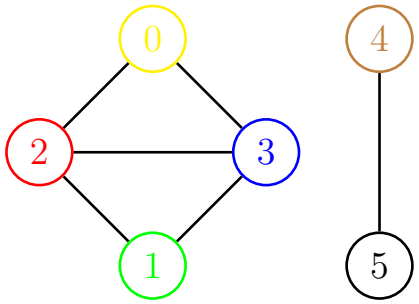
Para implementar o problema apresentado na seção 2, foi utilizada a linguagem de programação Python. O primeiro passo para resolver esse problema consiste em representar o arquivo de entrada do algoritmo como um grafo. O método utilizado para representar o grafo foi o de matriz de adjacência.

No exemplo abaixo, é ilustrado uma entrada de um arquivo de texto. Com o intuito de facilitar a manipulação de dados, os valores dos vértices, representados pelas cores no arquivo, foram substituídos por números inteiros variando entre zero e o número máximo de vértices do grafo. A Figura 1 mostra como o grafo do Exemplo 1 ficaria após essa substituição.

Exemplo 1: **Arquivo de entrada**

```
vermelho azul
vermelho verde
azul verde
azul amarela
amarela vermelho
marrom preto
```

Figura 1: **Grafo do Exemplo 1**



Com o grafo criado, o próximo passo é identificar se ele é um grado Euleriano. Como citado anteriormente, um grafo Euleriano tem que possuir todos os vértices com grau par e precisa ser conexo. Para verificar a conectividade do grafo, foi utilizado o algoritmo de busca em profundidade (DFS). Esse algoritmo marca os vértices visitados com um determinado valor; se no final da busca, a quantidade de vértices visitados for diferente da quantidade de vértices total do grafo, então conclui-se que o grafo é desconexo. Já para a questão da análise da paridade dos vértices, foi feita uma soma, por linha, de cada valor da matriz de adjacência; visto que os valores das linhas (colunas) representam os respectivos vértices, e as posições correspondentes às arestas são atribuídas com o número 1, a soma dos valores da linha fornece o grau daquele vértice. Após a realização desses dois testes, é possível identificar se um grafo é Euleriano.

Agora que sabemos se o grafo é Euleriano ou não, podemos nos preocupar em encontrar um circuito Euleriano. Existem basicamente dois algoritmos, bastante conhecidos na literatura, que realizam essa tarefa: algoritmo de Fleury e o algoritmo de Hierholzer [3]. O método empregado nesse trabalho foi o algoritmo de Hierholzer. Esse algoritmo funciona da seguinte forma:

- **1. Inicialização**: durante a execução do algoritmo, ele checa se uma aresta já foi visitada. Portanto, marcamos toda aresta como "não-visitada".

- **2. Encontrar o nó inicial**: para o primeiro *subtour* o algoritmo precisa de um nó inicial. Para grafos Eulerianos pode-se tomar um nó arbitrário. Após escolher o nó inicial, marcamos ele como visitado.

- **3. Encontre um vizinho "não-visitado"**: para escolher a próxima aresta do nosso *subtour* atual, o algoritmo examina todas as arestas do nó atual (ativo). Dentre essas arestas, o algoritmo escolhe uma aleatoriamente. Adicionamos essa aresta para o *subtour* e a marcamos como visitada. O nó da outra aresta agora se torna o novo nó atual (ativo).

- **4. Verifique a existência de um ciclo**: para verificar se um ciclo foi formado, comparamos o nó atual com o nó inicial do *subtour*. Se nenhum ciclo foi formado, repetimos o passo 3.

- **5. Atualize o ciclo completo**: o *subtour* foi completado e será integrado no *tour*. Se o *subtour* for um ciclo, o algoritmo substitui o no inicial/final do *subtour* no *tour* por todos os nós do *subtour*.

- **6. Verifique se existem arestas faltando**: depois de cada integração do *subtour*, o algoritmo checa se o *tour* forma um circuito Euleriano completo. Em caso afirmativo, o algoritmo está terminado e returna o *tour* resultando. Caso contrário, pelo menos um outro *subtour* existe. Fazemos uma busca nele por retornar aos passos anteriores, a partir do passo 3.

A seguir, apresentamos o pseudocódigo para o algoritmo de Hierholzer.

Procedure 1 Algoritmo de Hierholzer

Entrada: Um grafo Euleriano
Saída: Um tour

noInicial ← no arbitrario
subtour ← ∅
while a tour ser um caminho ou ciclo Euleriano **do**
    noInicial ← no que está no tour e que não foi visitado
    subtour ← {noInicial}
    noAtual ← noInicial
    do
        {noAtual,u} ← pegue a aresta que não foi visitada partindo do noAtual
        subtour ← subtour ∪ {u}
        noAtual ← u
    while noInicial ≠ noAtual
    tour ← tour ∪ subtour
end while

Referências

[1] UFSC, Euleriano, 2017. Disponível em http://www.inf.ufsc.br/grafos/temas/euleriano/euleriano.html. Acesso em 03 de outubro de 2017.

[2] N. Biggs, E. K. Lloyd, R. J. Wilson, Graph Theory, 1736-1936, Oxford University Press, 1976.

[3] M. J. Becker, Hierholzer's algorithm, 2015.