Ciclos eulerianos e o problema do carteiro chinês

Algoritmos em Grafos

Marco A L Barbosa

Conteúdo

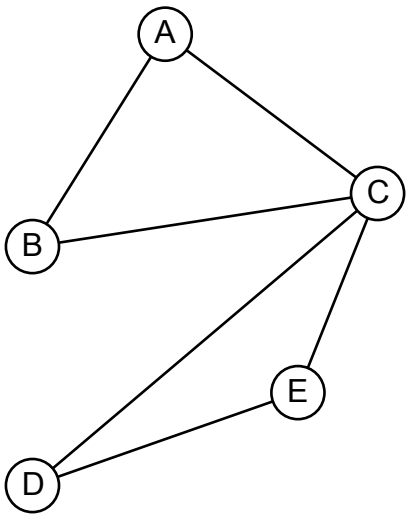
O estudo utilizando apenas este material **não é suficiente** para o entendimento do conteúdo. Recomendamos a leitura das referências no final deste material e a resolução (por parte do aluno) de todos os exercícios indicados.

Introdução

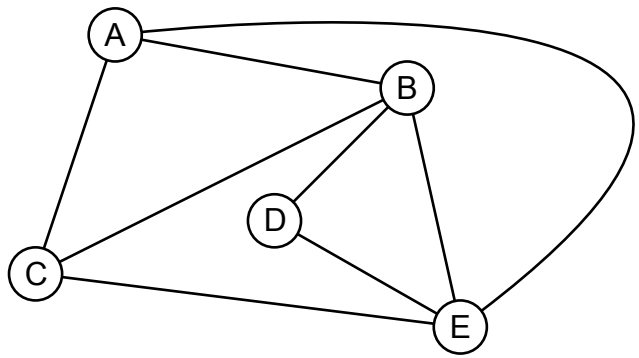
Introdução

- Um **ciclo euleriano** (caminho euleriano) é um ciclo (caminho) que usa cada aresta do grafo exatamente uma vez
- Um grafo que contém um ciclo euleriano é chamado de **grafo euleriano**
- Um grafo que contém um caminho euleriano, mas não contém um ciclo euleriano é chamado de **grafo semi-euleriano**

Exemplos



Ciclo Euleriano: A, C, D, E, C, B, A



Caminho Euleriano: A, B, D, E, B, C, E, A, C

Propriedades

Propriedades

- Lema 1
  - Dado um grafo não orientado conexo $G = (V, E)$ com todos os vértices de grau par, então qualquer par de vértices $u, v \in G$ faz parte de um ciclo sem arestas repetidas.

Propriedades

- Lema 1
  - Dado um grafo não orientado conexo $G = (V, E)$ com todos os vértices de grau par, então qualquer par de vértices $u, v \in G$ faz parte de um ciclo sem arestas repetidas.
- Prova (por contradição)
  - Suponha que exista um par de vértices $u, v \in G$ que não admita um ciclo em comum. Como o grafo é conexo, então existe um caminho p tal que $u \overset{p}{\rightsquigarrow} v$. Isto implica que deve existir uma aresta (x, y) no caminho p cuja a remoção torna o grafo desconexo, caso contrário existiria um outro caminho alternativo $u \overset{p'}{\rightsquigarrow} v$ disjunto de p. A remoção da aresta (x, y) gera duas componentes, sendo que x e y pertencem a componentes distintas. Desta forma, x e y são os únicos vértices de grau ímpar na sua componente, mas isto é uma contradição, pois o número de vértices de grau ímpar em um (sub)grafo deve ser par.

Propriedades

- Teorema 1
  - Um grafo não orientado conexo G é um grafo euleriano se e somente se todo vértice de G tem grau par.

Propriedades

- Teorema 1
  - Um grafo não orientado conexo G é um grafo euleriano se e somente se todo vértice de G tem grau par.
- Prova (ida)
  - Seja $G = (V, E)$ um grafo euleriano e seja p um ciclo euleriano de G. Cada ocorrência de um vértice $v \in V$ em p, implica uma aresta que chega em v e uma aresta que sai de v. Como todas as arestas de E fazem parte de p, o número de arestas incidentes em cada vértice é par.

Propriedades

- Prova (volta)
  - Seja $G = (V, E)$ um grafo com todos os vértices de grau par. Na construção de um caminho em G sempre é possível chegar e sair de um vértice por arestas ainda não utilizadas. Ou seja, é possível construir um ciclo arbitrário C a partir de um vértice qualquer v (Lema 1). Se C contém todas as arestas de G, temos um ciclo euleriano. Senão, construímos um grafo G', tal que $G'.E = G.E -$ arestas de C. Em G' todos os vértices tem grau par, e pelo menos um vértice de C está em $G'.V$ e tem grau maior que 0 (senão o grafo não seria conexo). Recomeçamos este processo para o grafo G', começando com um vértice $v' \in C$ com grau maior que 0 e construímos um ciclo C'. Os ciclos C e C' podem ser unidos para formar um único ciclo. Continuando este processo até acabar as arestas do grafo, obteremos necessariamente um ciclo único que contém todas as arestas de G.

Algoritmo de Hierholzer

Algoritmo de Hierholzer (primeira versão)

```
hierholzer-1(G)
1 G' = (G.V, G.E)
2 v_0 = um vértice de G
3 C = caminho contendo apenas v_0
4 while G'.E ≠ ∅
5     u = vértice em C tal que d(u) > 0 em G
6     U = ciclo em G' que contém u
7     C = C substituindo u por U
8     G'.E = E - arestas de U
9 return C
```

Algoritmo de Hierholzer (primeira versão)

```
hierholzer-1(G)
1 G' = (G.V, G.E)
2 v_0 = um vértice de G
3 C = caminho contendo apenas v_0
4 while G'.E ≠ ∅
5     u = vértice em C tal que d(u) > 0 em G
6     U = ciclo em G' que contém u
7     C = C substituindo u por U
8     G'.E = E - arestas de U
9 return C
```

- Análise do tempo de execução
  - As operações da linhas 5 e 7 podem ser implementadas em tempo constante (usando lista duplamente encadeada), portanto o total destas linhas é $O(E)$
  - O tempo total das linhas 6 e 8 (usando análise agregada) é $O(E)$
  - Portanto, o tempo de execução do algoritmo é $O(E)$

Exemplo de execução do procedimento `hierholzer-1`

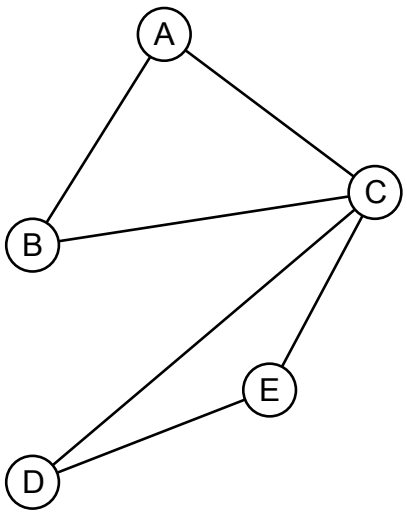
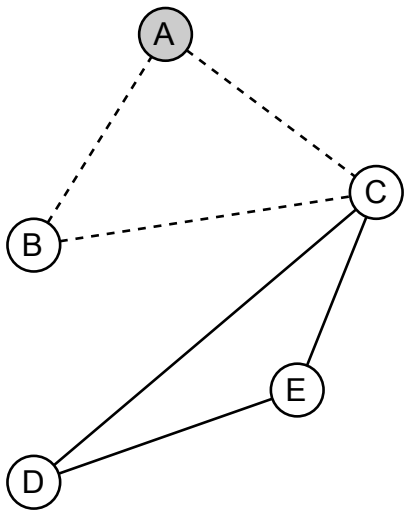
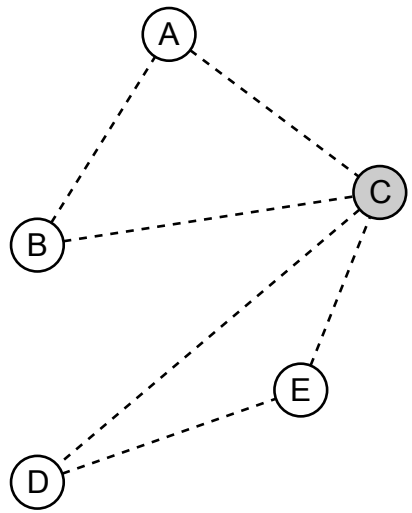


Figura 1



Vértice selecionado: A
Ciclo atual: A
Ciclo criado: A, B, C, A
Junção dos ciclos: A, B, C, A



Vértice selecionado: C
Ciclo atual: A, B, C, A
Ciclo criado: C, E, D, C
Junção dos ciclos:
A, B, C, E, D, C, A

Algoritmo de Hierholzer (primeira versão)

- O procedimento `hierholzer-1` foi derivado diretamente da prova do Teorema 1, e por isto, podemos verificar facilmente que ele é correto. No entanto, a sua implementação é um pouco trabalhosa
- A seguir, apresentamos uma versão do algoritmo de Hierholzer que utiliza uma pilha para auxiliar na construção do ciclo, o que facilita a implementação. No entanto a prova de corretude não é tão simples (fica como exercício)

Algoritmo de Hierholzer (segunda versão)

```
hierholzer-2(G)
1 v_0 = um vértice de G
2 s = {v_0} // pilha
3 C = {} // lista vazia
4 while s ≠ ∅
5    u = s.pop() // desempilha
6    C.pre-add(u) // adiciona na frete da lista
7    remover todas as aresta e do início de u.adj
       tal que e.visitidada = true
8    while u.adj ≠ ∅
9        s.push(u) // empilha
10       (u, v) = u.adj.remove() // remove a primeira aresta
11       (v, u).visitada = true
12       u = v
13       remover todas as aresta e do início de u.adj
         tal que e.visitidada = true
14 return C
```

Algoritmo de Hierholzer (segunda versão)

- Funcionamento
  - s é o ciclo temporário
  - C é o ciclo definitivo
  - O laço da linha 8 constrói um ciclo que começa e termina com o vértice desempilhado na linha 5
  - No caso de grafos não orientado, a remoção da aresta (u, v) da lista de adjacências u na linha 9 não garante que ela não será mais visitada (ela ainda está na lista de v). As linhas 7, 11 e 13 garantem que a arestas já visitadas não serão visitadas novamente

Algoritmo de Hierholzer (segunda versão)

- Análise do tempo de execução
  - Usamos análise agregada
  - Cada aresta é removida da lista de adjacências duas vezes. A operação de remoção pode ser implementada em tempo constante, desta forma, o tempo total das operações de remoção é $\Theta(E)$
  - As operações `pop`, `push` e `pre-add` podem ser implementadas com tempo constante. Como o grafo de entrada é conexo e portanto $E > V - 1$, a quantidade de vezes que estas operações são realizadas é limitado por E, desta forma, o tempo total gasto com estas operações é $O(E)$
  - Portanto, o tempo de execução do algoritmo é $O(E)$

O problema do carteiro chinês

O problema do carteiro chinês

- Dado um grafo conexo com peso nas arestas, o **problema do carteiro chinês** consiste em encontrar um ciclo de peso mínimo que passe por cada aresta pelo menos uma vez
- Aplicações
  - Entrega de correspondência
  - Coleta de lixo
  - Nebulização no combate a dengue

O problema do carteiro chinês

- Grafo euleriano
  - Aplicar o algoritmo de Hierholzer

O problema do carteiro chinês

- Grafo euleriano
  - Aplicar o algoritmo de Hierholzer
- Grafo não euleriano
  - Transformar o grafo em euleriano adicionando arestas artificiais e aplicar o algoritmo de Hierholzer

O problema do carteiro chinês

- Grafo euleriano
  - Aplicar o algoritmo de Hierholzer
- Grafo não euleriano
  - Transformar o grafo em euleriano adicionando arestas artificiais e aplicar o algoritmo de Hierholzer
  - Se o grafo for semi-euleriano, adicionar uma aresta artificial que representa o caminho mínimo entre os dois vértices de grau ímpar (o caminho mínimo pode ser encontrado usando o algoritmo de Dijkstra)

O problema do carteiro chinês

- Grafo euleriano
  - Aplicar o algoritmo de Hierholzer
- Grafo não euleriano
  - Transformar o grafo em euleriano adicionando arestas artificiais e aplicar o algoritmo de Hierholzer
  - Se o grafo for semi-euleriano, adicionar uma aresta artificial que representa o caminho mínimo entre os dois vértices de grau ímpar (o caminho mínimo pode ser encontrado usando o algoritmo de Dijkstra)
  - Se o grafo tiver 4 ou mais vértices de grau ímpar
    - Montar um grafo completo com os vértices de grau ímpar, onde cada aresta representa o menor caminho entre o par de vértices (algoritmo de Floyd-Warshall)
    - Encontrar a melhor combinação de pares de vértices (emparelhamento perfeito, algoritmo de Edmonds de complexidade polinomial)

Referências

Referências

- Caminho euleriano. Wikipédia. https://en.wikipedia.org/wiki/Eulerian_path
- Problema do carteiro chinês. Wikipédia. https://en.wikipedia.org/wiki/Route_inspection_problem