PROBLEMA DO CARTEIRO CHINÊS

PROJETO E ANÁLISE DE ALGORITMOS

Discente: Lauryane Santos Siqueira
Professor: Leonardo Nogueira Matos

PROCC
Programa de Pós-Graduação em Ciência da Computação/UFS

UNIVERSIDADE FEDERAL DE SERGIPE

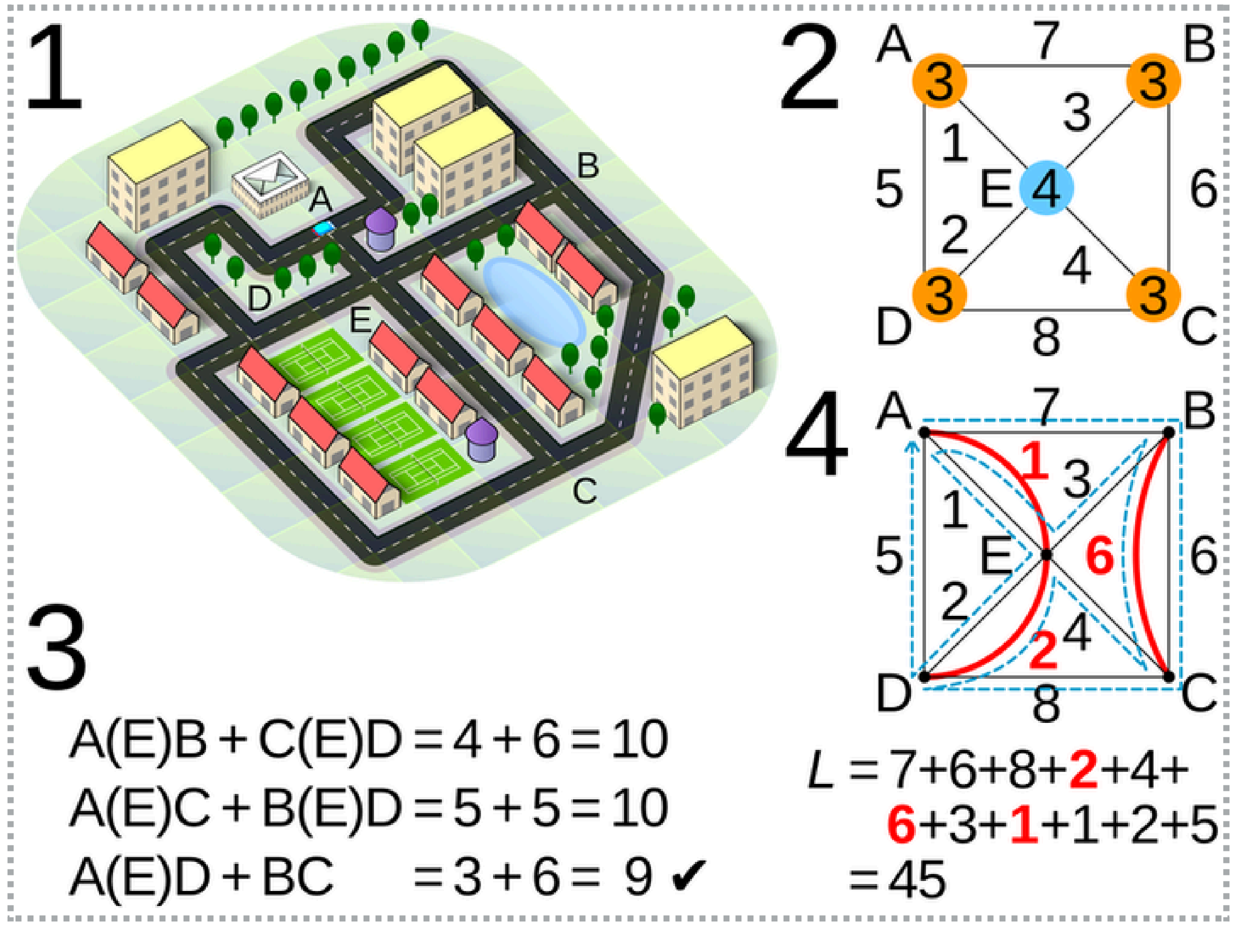
O QUE É O PROBLEMA?

- Proposto pelo matemático Kwan Mei-Ko, na década de 1960.
- O Problema do Carteiro Chinês é um problema clássico de teoria dos grafos.
- **Objetivo**: encontrar o menor percurso fechado que percorra todas as arestas do grafo pelo menos uma vez.
- Inspirado na situação de um carteiro que precisa passar por todas as ruas de um bairro gastando o mínimo de tempo/distância.
- **Aplicações**: entrega postal e roteirização de carteiros; coleta de lixo; varrição/limpeza de vias; inspeção e manutenção de redes (iluminação, elétrica, hidrantes); serviços urbanos de manutenção; rotas de distribuição que dependem de ruas específicas.


DEFINIÇÃO FORMAL

- Seja ***G* = (*V*,*E*)** um grafo conexo e ponderado, onde:
  - ***V*** = conjunto de vértices (interseções, pontos).
  - ***E*** = conjunto de arestas (ruas, trechos).
  - Cada aresta ***e* ∈ *E*** tem um peso ***w*(*e*)** (distância, tempo, custo).

- O Problema do Carteiro Chinês consiste em encontrar um circuito fechado ***C*** em ***G*** tal que:
  - Toda aresta ***e* ∈ *E*** seja percorrida ao menos uma vez em ***C***.
  - O custo total ***W*(*C*) = Σ*w*(*e*)** seja mínimo.



Fonte: Wikipedia

COMO A SOLUÇÃO DEPENDE DO GRAFO DE ENTRADA

- O Ponto Central: A resolução do Problema do Carteiro Chinês depende fundamentalmente das características do grafo.
- A Grande Questão: A pergunta crucial é: o grafo possui um caminho que passa por cada rua exatamente uma vez?

- Os Dois Cenários:

1. Se o grafo for um **Grafo Euleriano**, a solução é direta e ótima. Você pode encontrar um caminho que percorre todas as arestas sem repetição.
2. Se o grafo for **Não-Euleriano**, a solução é mais complexa. É necessário criar arestas virtuais ou "duplicar" as arestas existentes para transformar o problema no cenário ideal, com o menor custo possível.

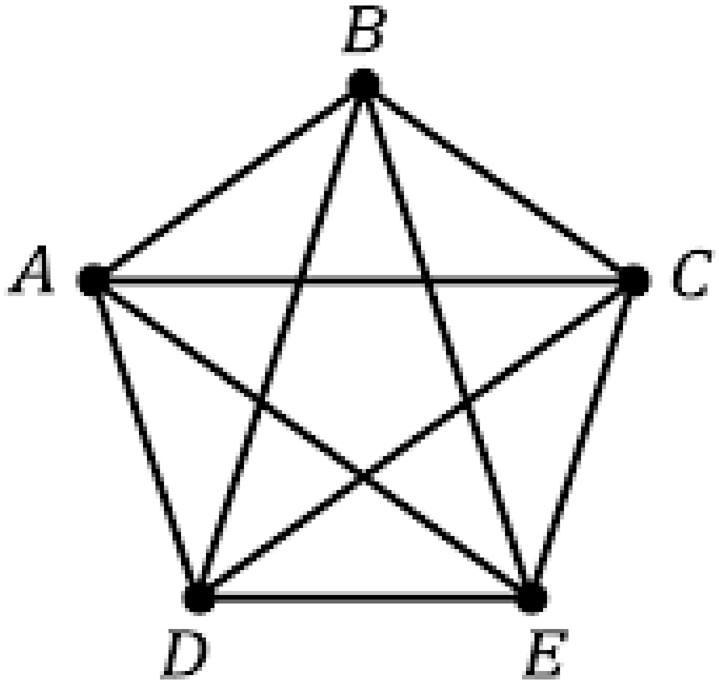
GRAFO EULERIANO

Um grafo é considerado euleriano se for conexo (todas as partes estão conectadas) e todos os seus vértices tiverem um grau par.

Grau par significa que um número par de arestas se encontram em cada vértice. Isso é a chave para a solução, pois garante que, ao entrar em um vértice por uma aresta, você sempre terá uma outra aresta para sair.

- **A Solução**: Existe um circuito euleriano, um caminho que percorre cada aresta exatamente uma vez e retorna ao início.
- **Conclusão**: A solução é ótima e trivial. O custo é simplesmente a soma do peso de todas as arestas.
- **Algoritmo**: Fleury é um exemplo de algoritmo para encontrar o circuito.

Grafo Euleriano



Fonte: EVULPO, 2022. Grafos de Euler: Identificação e aplicação.

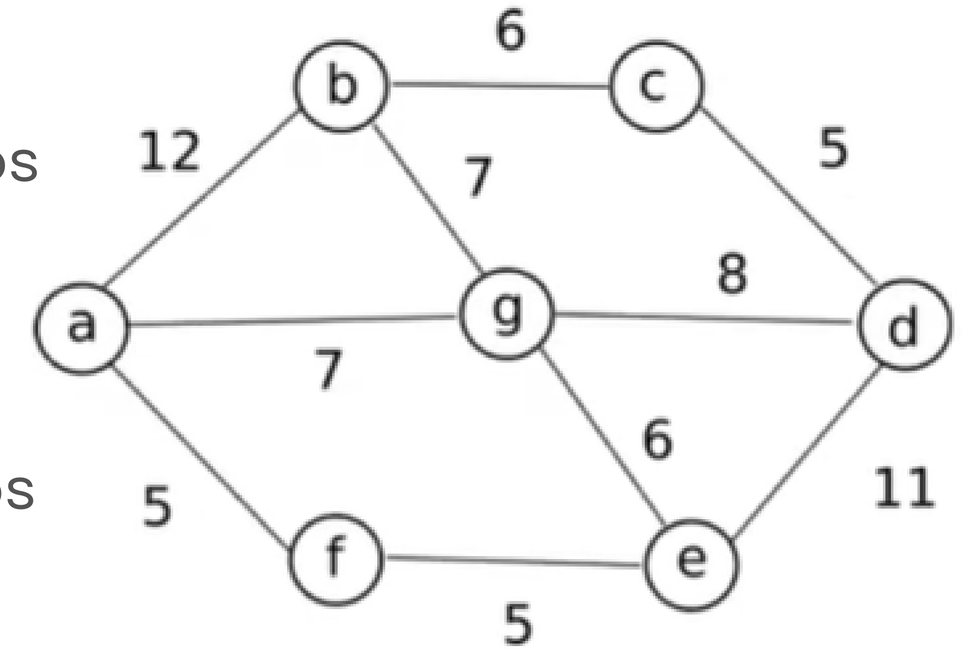
GRAFO NÃO-EULERIANO

O grafo possui um ou mais vértices com grau ímpar. Isso significa que, se você entrar em um desses vértices por uma aresta, não poderá sair dele usando uma aresta diferente sem repetir uma já percorrida.

- **A Lógica da Solução:** O objetivo é "reparar" o grafo, tornando-o euleriano. Isso é feito duplicando o conjunto de arestas que conecta os vértices de grau ímpar com o menor custo total possível.
- **O Processo:**

1. **Identificar os Vértices Ímpares**: O primeiro passo é encontrar todos os vértices com grau ímpar.
2. **Calcular o Pareamento de Menor Custo**: Para encontrar a solução ótima, usamos algoritmos de caminho mais curto (ex.: Dijkstra ou Floyd-Warshall) e um algoritmo de pareamento para encontrar a combinação de pares de vértices ímpares que tenham o menor custo total.
3. **A Solução Final**: O custo total da rota é a soma do peso de todas as arestas originais mais o custo do pareamento (das arestas duplicadas).

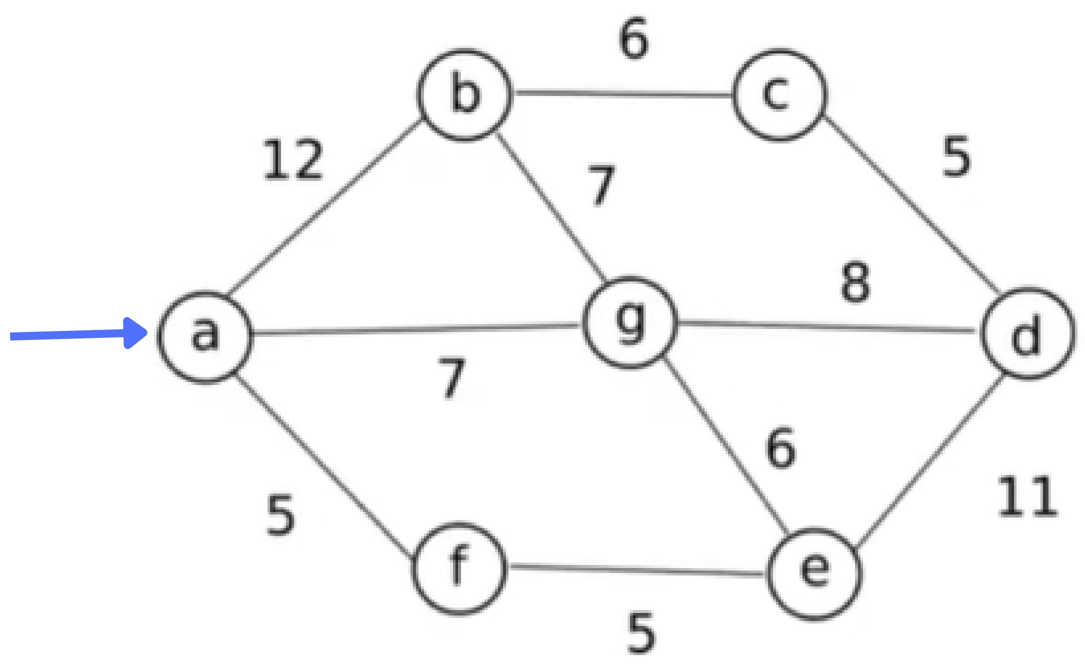
Grafo Não-Euleriano



Fonte: Prof. Alexandre Levada, 2021. O problema do carteiro chinês. [https://www.youtube.com/watch?v=RbKqfa6uHAE&t=13s]. YouTube.

INSTÂNCIA DO PROBLEMA

```
a;b;12
a;f;5
a;g;7
b;c;6
b;g;7
c;d;5
d;e;11
d;g;8
e;f;5
e;g;6
```



IMPLEMENTAÇÃO EM PYTHON

```python
1 usage
def chinese_postman(file_path, out_dir="out"):
    # 0. Preparação de entrada e saída.
    G = read_graph(file_path) # Retorna erro se o grafo não for conexo
    pos = nx.spring_layout(G, seed=42)

    os.makedirs(out_dir, exist_ok=True)
    output_path_G = os.path.join(out_dir, "original.png")
    draw_graph(G, pos, output_path_G, title="Grafo original")

    base_cost = total_weight(G)

    # 1. Verifica se já é euleriano
    if is_eulerian(G):
        route_pairs, node_seq = eulerian_route_pairs_and_nodes(G)
        added_cost = 0.0
        total = base_cost + added_cost

        print("Sequência de nós do caminho final:", " -> ".join(map(str, node_seq)))
        print("Caminho final (arestas):", route_pairs)
        print(f"Custo base: {base_cost}")
        print(f"Custo adicional: {added_cost}")
        print(f"Custo total: {total}")
        return
```

IMPLEMENTAÇÃO EM PYTHON

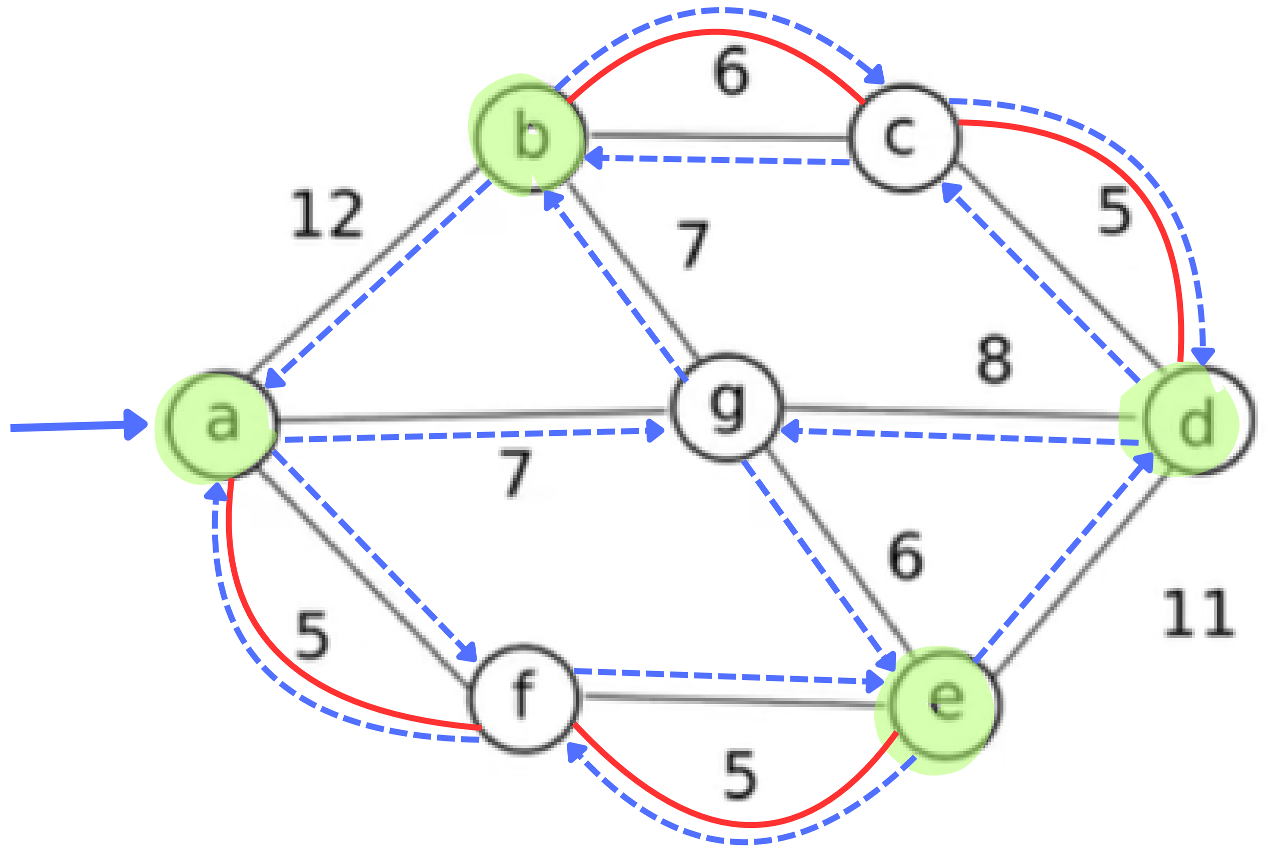
```
# 2. Vértices ímpares
odd = odd_vertices(G)
print("Vértices ímpares:", odd)

# 3. Emparelhamento
matching, cost = brute_force_minimum_matching(G, odd) # simples (por força bruta)
print("Emparelhamento escolhido:", matching)
print("Custo adicional:", cost)

# 4. Duplicação das arestas dos caminhos escolhidos
G_aug = G.copy()
for u, v in matching:
    path = nx.shortest_path(G, u, v, weight="weight")
    for a, b in zip(path, path[1:]):
        w = min_edge_weight(G, a, b)
        G_aug.add_edge(a, b, weight=w, duplicate=True)
output_path_G_aug = os.path.join(out_dir, "augmented.png")
draw_augmented_graph(G_aug, pos, output_path_G_aug, title="Após duplicação")
```

IMPLEMENTAÇÃO EM PYTHON

```python

    # 5. Circuito euleriano
    route_pairs, node_seq = eulerian_route_pairs_and_nodes(G_aug) # obter circuito euleriano
    total = total_weight(G_aug)
    added_cost = total - base_cost

    print("Sequência de nós do caminho final:", " -> ".join(map(str, node_seq)))
    print("Caminho final (arestas):", route_pairs)
    print(f"Custo base: {base_cost}")
    print(f"Custo adicional: {added_cost}")
    print(f"Custo total: {total}")

if __name__ == "__main__":
    entrada = "graph3.input"
    chinese_postman(entrada)
```

SOLUÇÃO FINAL

```
Vértices ímpares: ['a', 'b', 'd', 'e']
Emparelhamento escolhido: [('a', 'e'), ('b', 'd')]
Custo adicional: 21.0
Sequência de nós do caminho final: a -> f -> e -> d -> c -> d -> g -> e -> f -> a -> g -> b -> c -> b -> a
('Caminho final '
 "(arestas):[('a', 'f'), "
 "('f', 'e'), ('e', 'd'), "
 "('d', 'c'), ('c', 'd'), "
 "('d', 'g'), ('g', 'e'), "
 "('e', 'f'), ('f', 'a'), "
 "('a', 'g'), ('g', 'b'), "
 "('b', 'c'), ('c', 'b'), "
 "('b', 'a')]")
Custo base: 72.0
Custo adicional: 21.0
Custo total: 93.0

Process finished with exit code 0
```



SOLUÇÃO FINAL

```
Vértices impares: ['a', 'b', 'd', 'e']
Emparelhamento escolhido: [('a', 'e'), ('b', 'd')]
Custo adicional: 21.0
Sequência de nós do caminho final: a -> f -> e -> d -> c -> d -> g -> e -> f -> a -> g -> b -> c -> b -> a
('Caminho final '
 "(arestas):[('a', 'f'), "
 "('f', 'e'), ('e', 'd'), "
 "('d', 'c'), ('c', 'd'), "
 "('d', 'g'), ('g', 'e'), "
 "('e', 'f'), ('f', 'a'), "
 "('a', 'g'), ('g', 'b'), "
 "('b', 'c'), ('c', 'b'), "
 "('b', 'a')]")
Custo base: 72.0
Custo adicional: 21.0
Custo total: 93.0

Process finished with exit code 0
```