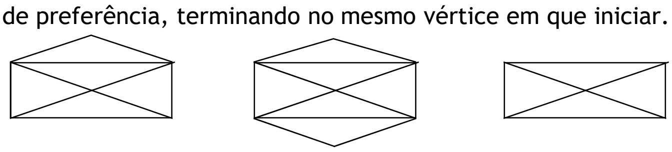
Algoritmos em Grafos: Circuitos de Euler e Problema do Carteiro Chinês

R. Rossetti, A. P. Rocha, L. Ferreira, J. P. Fernandes, F. Ramos, G. Leão

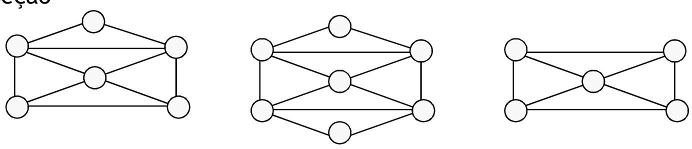
FEUP, MIEIC, CAL

Circuitos de Euler

- Puzzle: desenhar as figuras abaixo sem levantar o lápis e sem repetir arestas; de preferência, terminando no mesmo vértice em que iniciar.



- Reformulação como problema em Teoria de Grafos: colocar um vértice em cada interseção

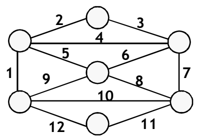


- **Caminho de Euler**: caminho que visita cada aresta exatamente uma vez
- Problema resolvido por **Leonhard Euler** em 1736 e que marca o início da Teoria dos Grafos
- **Circuito de Euler:** caminho de Euler que começa e acaba no mesmo vértice

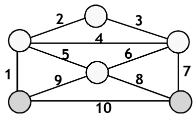
Condições necessárias e suficientes (1/2)

- Um grafo *não dirigido* contém um *circuito* de Euler sse
  (1) é conexo e
  (2) cada vértice tem grau (nº de arestas incidentes) par.
- Um grafo *não dirigido* contém um *caminho* de Euler sse
  (1) é conexo e
  (2) todos menos dois vértices têm grau par (estes dois vértices serão os vértices de início e fim do caminho).

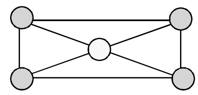
Circuito de Euler



Caminho de Euler



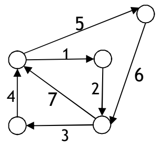
Sem caminho ou circuito de Euler



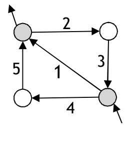
Condições necessárias e suficientes (2/2)

- Um grafo *dirigido* contém um *circuito* de Euler *sse*
  (1) é (fortemente) conexo e
  (2) cada vértice tem o mesmo grau de entrada e de saída.
- Um grafo *dirigido* contém um *caminho* de Euler *sse*
  (1) é (fortemente) conexo e
  (2) todos menos dois vértices têm o mesmo grau de entrada e de saída, e os dois vértices têm graus de entrada e de saída que diferem de 1.

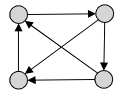
Com circuito de Euler



Com caminho de Euler



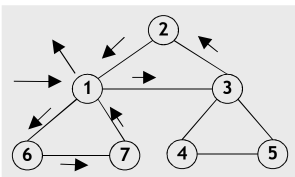
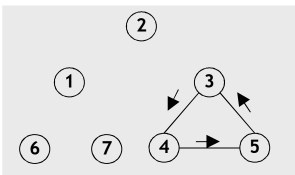
Sem circuito ou caminho de Euler



Método baseado em pesquisa em profundidade para encontrar um circuito de Euler

1. Escolher um vértice qualquer e efetuar uma pesquisa em profundidade a partir desse vértice
   - Visitar vértice: se tiver arestas incidentes não visitadas, escolher uma dessas arestas, marcá-la como visitada, e visitar vértice adjacente
   - Se o grafo satisfizer as condições necessárias e suficientes, esta pesquisa termina necessariamente no vértice de partida, formando um circuito, embora não necessariamente de Euler
2. Enquanto existirem arestas por visitar
   - 2.1 Procurar o primeiro vértice no caminho (circuito) obtido até ao momento que possua uma aresta não percorrida
   - 2.2 Lançar uma sub-pesquisa em profundidade a partir desse vértice (sem voltar a percorrer arestas já percorridas)
   - 2.3 Inserir o resultado (circuito) no caminho principal

Exemplo em grafo não dirigido

	Arestas por visitar	Caminho desta iteração	Caminho acumulado
1ª iter.		**1-3*-2-1-6-7-1** Com arestas por visitar	**1-3*-2-1-6-7-1**
2ª iter.		**3-4-5-3**	**1-3-4-5-3-2-1-6-7-1** **(Circuito de Euler)**

Estruturas de dados e eficiência temporal

- Tempo de execução: O(|E| + |V|)
  - Cada vértice e aresta é percorrido uma única vez
  - Cada vez que se percorre um adjacente, avança-se o apontador de adjacentes (para não voltar a percorrer as mesmas arestas)
  - Usam-se listas ligadas para efetuar inserções em tempo constante

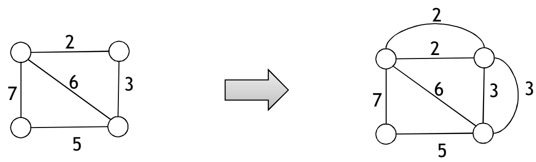
Problema do carteiro chinês (*Chinese postman problem*)

- Dado um grafo pesado conexo G=(V,E), encontrar um caminho fechado (i.e., com início e fim no mesmo vértice) de peso mínimo que atravesse cada aresta de *G* pelo menos uma vez.
  - A um caminho assim chama-se *percurso ótimo do carteiro Chinês*.
  - A um caminho fechado (não necessariamente de peso mínimo) que atravesse cada aresta pelo menos uma vez chama-se *percurso do carteiro*.
- Problema estudado pela primeira vez por Mei-Ku Kuan em 1962, relacionado com a distribuição de correspondência ao longo de um conjunto de ruas, partindo e terminando numa estação de correios.
- Resolúvel em tempo polinomial para grafos dirigidos ou não dirigidos, mas infelizmente o problema é NP-completo (tempo exponencial) quando se combinam arestas dirigidas com arestas não dirigidas (grafos mistos)
  - Exemplo: percurso do camião do lixo, quando algumas ruas têm sentidos únicos

Abordagem

- Se o grafo G for Euleriano, a solução é trivial, pois qualquer circuito de Euler é um percurso ótimo do carteiro Chinês.
  - Cada aresta é percorrida exatamente uma vez.
- Se o grafo G não for Euleriano, pode-se construir um grafo Euleriano G* duplicando algumas arestas de G, selecionadas por forma a conseguir um grafo Euleriano com peso total mínimo.

Não Euleriano (peso 23)

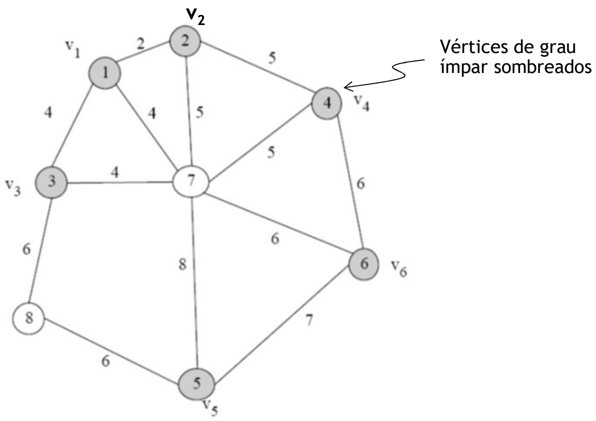


Euleriano! (peso 28)

Método para grafos não dirigidos (1/4)

- Passo 1: Achar todos os vértices de grau ímpar em *G*. Seja *k* o nº (par!) destes vértices. Se *k*=0, fazer *G*=G* e saltar para o passo 6.

G:



Método para grafos não dirigidos (2/4)

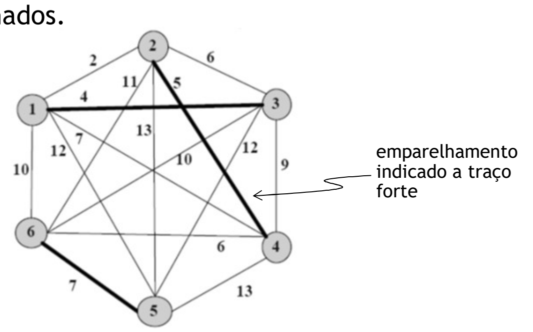
- Passo 2: Achar os caminhos mais curtos e distâncias mínimas entre todos os pares de vértices de grau ímpar em *G*.

$d(v_i,v_j)$	v1	v2	v3	v4	v5	v6
v1	-	2	4	7	12	10
v2		-	6	5	13	11
v3			-	9	12	10
v4				-	13	6
v5					-	7
v6						-

Método para grafos não dirigidos (3/4)

- Passo 3: Construir um grafo completo *G'* com os vértices de grau ímpar de *G* ligados entre si por arestas de peso igual à distância mínima calculada no passo 2.
- Passo 4: Encontrar um emparelhamento perfeito (envolvendo todos os vértices) de peso mínimo em *G'*. Isto corresponde a emparelhar os vértices de grau ímpar de *G*, minimizando a soma das distâncias entre vértices emparelhados.

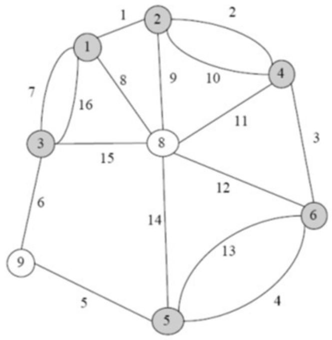
G':



Método para grafos não dirigidos (4/4)

- Passo 5: Para cada par *(u, v)* no emparelhamento perfeito obtido, adicionar pseudo-arestas (arestas paralelas duplicadas) a *G* ao longo de um caminho mais curto entre *u* e *v*. Seja G* o grafo resultante.
- Passo 6: Achar um circuito de Euler em G*. Este circuito é um percurso óptimo do carteiro Chinês.

G*:



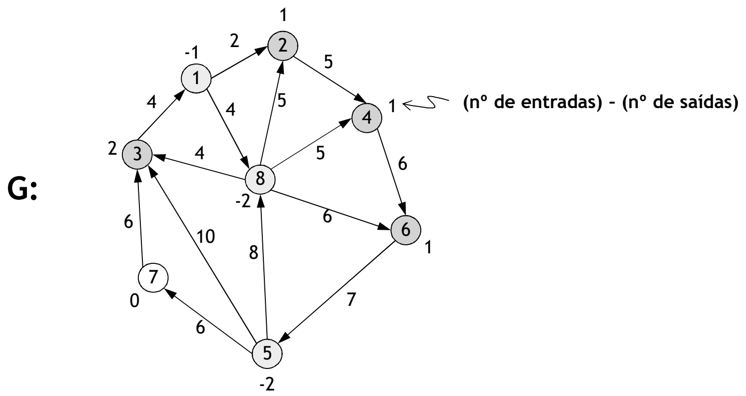
Nº de sequência de visita no circuito Euleriano (pesos omitidos)

* Realização do passo 4 - Emparelhamento perfeito de peso mínimo

- O problema de encontrar um emparelhamento perfeito de peso mínimo pode ser reduzido ao problema de encontrar um emparelhamento de peso máximo num grafo genérico por uma simples mudança de pesos
  - Basta substituir cada peso w_{ij} por $M+1-w_{ij}$, *em* que *M* é o peso da aresta mais pesada
  - Sendo o grafo completo e com número par de vértices, um emparelhamento de peso máximo é necessariamente perfeito
- Um emparelhamento de peso máximo num grafo genérico pode ser encontrado em tempo polinomial (ver referências).

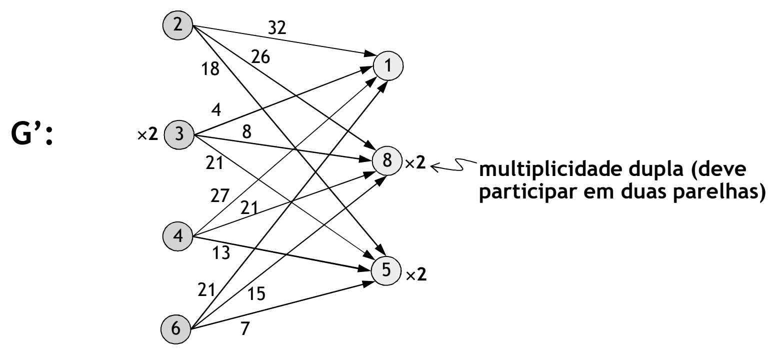
Método para grafos dirigidos (1/4)

- Passo 1: No grafo *G* dado, identificar os vértices com nºs diferentes de arestas a entrar e a sair



Método para grafos dirigidos (2/4)

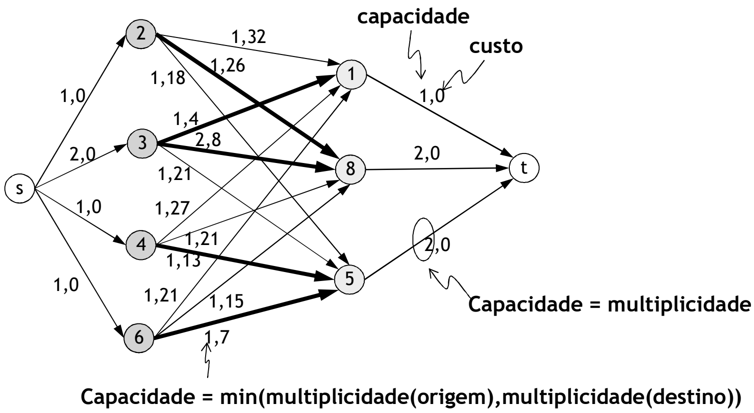
- Passo 2: Determinar os caminhos mais curtos de vértices que têm défice de saídas para vértices que têm défice de entradas e representar as distâncias respetivas num grafo bipartido *G'*.
  - Vértices são anotados com multiplicidade (nº de parelhas em que deve participar) igual ao défice absoluto



Método para grafos dirigidos (3/4)

- Passo 3: Formular problema de emparelhamento óptimo como problema de fluxo máximo de custo mínimo e resolver.

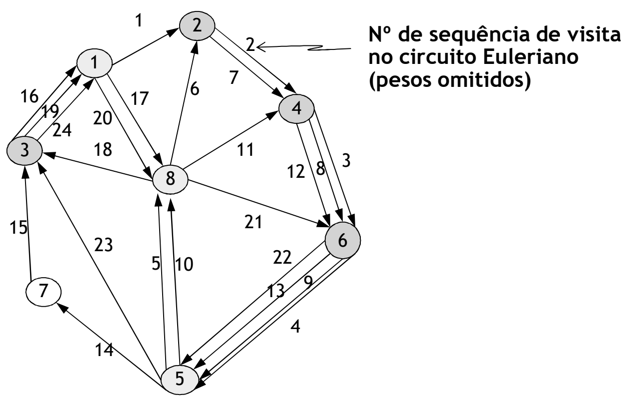
G'':



Método para grafos dirigidos (4/4)

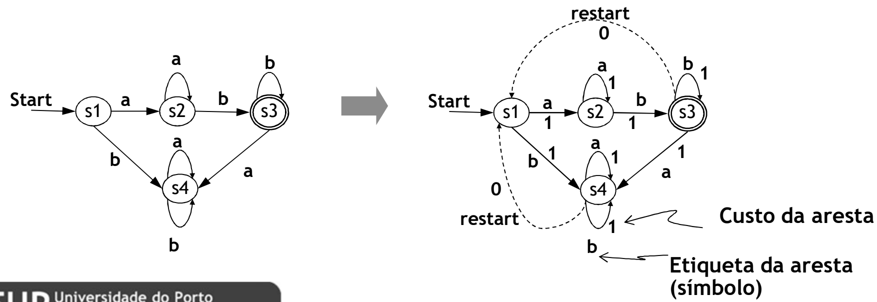
- Passo 4: Obter grafo Euleriano *G**, duplicando em *G* os caminhos mais curtos entre os vértices emparelhados no passo 3, e obter um circuito Euleriano.

G*:



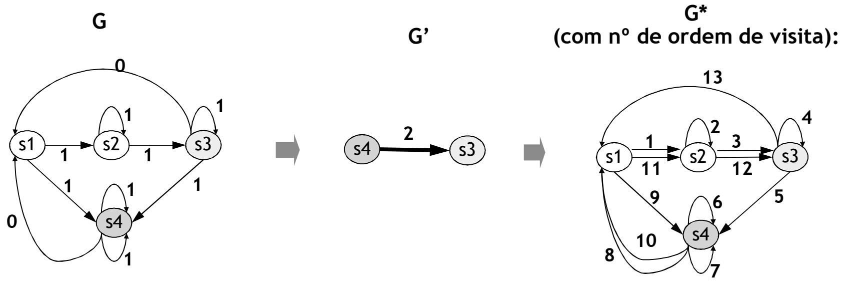
* Exemplo de aplicação (1/2)

- Achar um conjunto de sequências de teste completas (do estado inicial a um estado final) de comprimento total mínimo cobrindo todas as transições num autómato finito
  - Ligam-se os estados finais ao estado inicial e procura-se um percurso ótimo do carteiro
  - Nota: conceito de estado final faz mais sentido em máquinas de estados UML; no caso de autómatos finitos, podem-se considerar como tal estados de aceitação e estados absorventes (donde não é possível sair)



* Exemplo de aplicação (2/2)

- Resolução do problema do carteiro chinês dirigido:



- Solução final:
  - Caminho de Euler usando etiquetas:
    a-a-b-b-a-a-b-restart-b-restart-a-b-restart
  - Strings de teste: aabbaab, b, ab

Referências e mais informação

- "The Algorithm Design Manual", Steven S. Skiena, Springer-Verlag, 1998