# Estudo: teoria dos grafos

## Introdução

As arvores possuem uma grande flexibilidade e diferentes formas de aplicação, mas, por natureza, possuem suas limitações, ou seja, só podem representar relações do tipo hierárquico, como relações entre pai e filhos. Outras relações são somente representadas indiretamente, como a de um irmão.

Podemos corrigir esse problema ao seguir a Teoria dos Grafos, que é composta por uma coleção de nós (vértices) e as conexões podem ser feita entre eles.

Os grafos possuem uma estrutura de dados bem versátil ao ser comparado com a estrutura de uma árvore, assim, pode representar um grande número e diferentes situações e eventos de diversos domínios. Sua versatilidade distingue de uma abstração mais próxima a abstração e analogia do cérebro humano. Assim, problemas mais complexo podem ser solucionados seguindo essa teoria.

Esse princípio teórico cresceu bastante nos últimos 200 anos, onde foi adotado por matemáticos, cientistas da computação e também nas pesquisas bioquímicas como uma ferramenta de auxílio para solução de milhares de problemas.

## Tipos de grafos

São empregados estruturas chamadas de grafos, onde G(V,E), onde V é um conjunto não vazio de objetos denominadas de vértices e Eé um conjunto de pares não ordenadas de V.

Drozdek (2013, p. 391) apresenta algumas características dos Grafos modelando com base em sua categoria. As mais restritivas: *Simple Graph* (do inglês, Grafo Simples) e o *Directed Graph* (do inglês, Grafo Direcionado), e também, categorias de grafos que permitem que dois grafos tenham mais de uma borda, como o os *multgraph* (do inglês, Multigrafos). Segue uma interpretação geométrica a seguir.

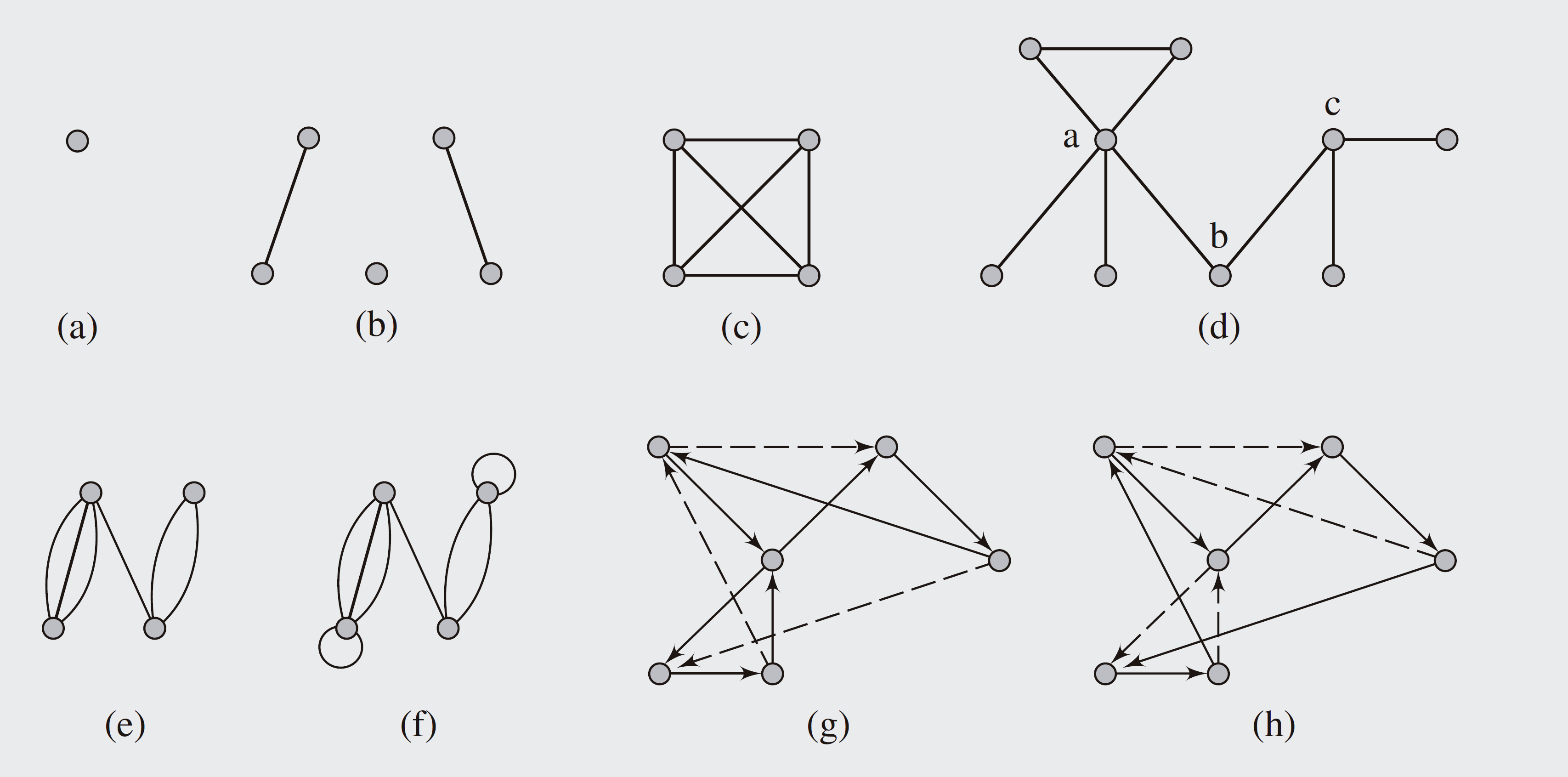


Figura 1: Interpretação geométrica dos grafos

|  |  |
| --- | --- |
| A – D | *Simple Graphs* |
| C | *Complete Graph* |
| E | *Multgraph* |
| F | *Pseudograph* |
| G | *Circuit in diagraph* |
| H | *Cycle in diagraph* |

* 1. **Grafo simples**

É um grafo não direcionado, sem laços e existe no máximo uma aresta entre quaisquer dois vértices (sem arestas paralelas). Para um grafo simples, o número de vizinhos de um vértice é igual à sua valência.

* 1. **Grafo completo**

É o grafo simples em que, para cada vértice do grafo, existe uma aresta conectando este vértice a cada um dos demais. Ou seja, todos os vértices do grafo possuem mesmo grau. O grafo completo de *n* vértices é frequentemente denotado por *Kn*. Ele tem *n*(*n*-1)/2 arestas (correspondendo a todas as possíveis escolhas de pares de vértices).

* 1. **Multigrafos**

É um grafo que permite múltiplas arestas ligando os mesmos vértices (arestas paralelas).

* 1. **Pseudografos**

É um grafo que contém arestas paralelas e laços.

* 1. **Curcuit in diagraph**
  2. **Cycle in diagraph**

Outros grafos compreendidos na matemática:

* 1. **Grafo nulo**

É o grafo cujo conjunto de vértices é vázio.

* 1. **Grafo vazio**

É o grafo cujo conjunto de arestas é vazio.

* 1. **Grafo trivial**

É o grafo que possui apenas um vértice e nenhuma aresta.

* 1. **Grafo regular**

É um grafo em que todos os vértices tem o mesmo grau.

* 1. **Grafo conexo**

Um grafo é *conexo* se for possível estabelecer um caminho de qualquer vértice para qualquer outro vértice de um grafo. Se for sempre possível estabelecer um caminho de qualquer vértice para qualquer outro vértice mesmo depois de remover *k*-1 vértices, então diz-se que o grafo está ***k*-conexo**. Note que um grafo está *k*-conexo se, e somente se, contém *k* caminhos independentes entre qualquer par de vértices. O grafo de exemplo acima é conexo (e portanto 1-conexo), mas não é 2-conexo. Em um grafo genérico *G*, o **corte** associado a um conjunto *X*de vértices é o conjunto de todas as arestas que têm uma ponta em *X* e outra em *V(G) - X*, onde *V(G)* é o conjunto de todos os vértices pertencentes ao grafo *G*.

* 1. **Ponto de articulação**

É um vértice cuja remoção desliga um grafo. Uma ponte é uma aresta cuja remoção desliga um grafo. Um **componente biconectado** é um conjunto máximo de arestas tal que qualquer par de arestas do conjunto fazem parte de um ciclo simples comum. O contorno de um grafo é o comprimento do ciclo simples mais curto no grafo. O contorno de um grafo acíclico é, por definição, infinito.

* 1. **Árvore**

É um grafo simples acíclico e conexo. Às vezes, um vértice da árvore é distinto e chamado de *raiz*. As árvores são muito usadas como estrutura de dados.

* 1. **Floresta**

É um conjunto de árvores; equivalentemente a uma floresta, em algum grafo acíclico.

* 1. **Subgrafo**

É um grafo cujo conjunto dos vértices é um subconjunto do conjunto de vértices *G*, cujo conjunto de arestas é um subconjunto do conjunto de arestas de *G*, e cuja função *w* é uma restrição da função de *G*

* 1. **Subgrafo gerador**

É aquele obtido pela remoção de uma ou mais arestas de um outro grafo, dizemos então que este novo grafo obtido é gerador do primeiro.

* 1. **Subgrafo induzido**

É obtido pela remoção de vértices e consequente das arestas relacionadas com ele de um outro grafo, dizemos que este novo grafo é um grafo induzido do original.

* 1. **Grafo parcial**

É um subgrafo com o mesmo conjunto de vértices que *G*. Uma **árvore parcial** é um grafo parcial que é árvore. Todo grafo tem pelo menos uma árvore parcial.

* 1. **Clique**

É um subgrafo que também é um grafo completo. No grafo do exemplo acima, os vértices 1, 2 e 5 formam um clique.

* 1. **Conjunto independente**

Em um grafo é um conjunto de vértices não adjacentes entre si. No exemplo acima, os vértices 1, 3 e 6 formam um conjunto independente e 3, 5 e 6 são outro conjunto independente.

* 1. **Grafo planar**

É aquele que pode ser representado em um plano sem qualquer intersecção entre arestas. O grafo do exemplo é planar; o grafo completo de *n* vértices, para *n*> 4, não é planar.

* 1. **Grafo bipartido**

É o grafo cujos vértices podem ser divididos em dois conjuntos, nos quais não há arestas entre vértices de um mesmo conjunto. Para um grafo ser bipartido ele não pode conter circuitos de comprimento ímpar.

**Se um grafo G é bipartido, todo o circuito de G possui comprimento par.**Sejam V1 e V2 os dois conjuntos em que, de acordo com a definição de grafo bipartido, se particiona V(G). Toda a aresta de G conecta um vértice em V1 com outro em V2. Assim sendo, se X for um vértice de V1, para “voltar” a esse vértice terá de se ir a V2 e voltar a V1 um número indeterminado de vezes, e de cada vez serão percorridas duas arestas, uma de um vértice em V1 para um vértice em V2 e outra de um vértice em V2 para um vértice em V1. Logo, o número de arestas a percorrer será par, ou seja, o comprimento do circuito é par.

**Se todo o circuito de um grafo G possui comprimento par, então o grafo é bipartido.** Seja G um grafo em que todo o circuito tem comprimento par, e seja X um vértice de G. Denotemos por V1 o conjunto formado por X e por todos os vértices cuja distância a X é par. Seja V2 = V(G)\V1 (isto é, o conjunto formado pelos vértices de G que não pertencem a V1). Pretende mostrar-se que não existe qualquer aresta que conecte vértices de V1 ou vértices de V2. Suponhamos a existência de tal aresta, isto é, suponhamos a existência de dois vértices em V1 (ou V2), digamos Xi e Xj, conectados por uma aresta. Ora existe já um caminho de comprimento par entre Xi e Xj, já que existem caminhos, ambos de comprimento par (ou ímpar, no caso de Xi e Xj pertencerem a V2), entre Xi e X e entre X e Xj. Se a esse caminho juntarmos a aresta {Xi;Xj} obtemos um circuito de comprimento ímpar o que contraria a hipótese de apenas existirem circuitos de comprimento par.

* 1. **Grafo bipartido complexo**

É o grafo bipartido, cujo qualquer vértice do primeiro conjunto é adjacente a todos vértices do segundo conjunto

* 1. **Grafo k-partido**

É um grafo cujos vértices podem ser particionados em *k* conjuntos disjuntos, nos quais não há arestas entre vértices de um mesmo conjunto. Um grafo 2-partido é o mesmo que grafo bipartido.

* 1. **Emparelhamento de grafos**

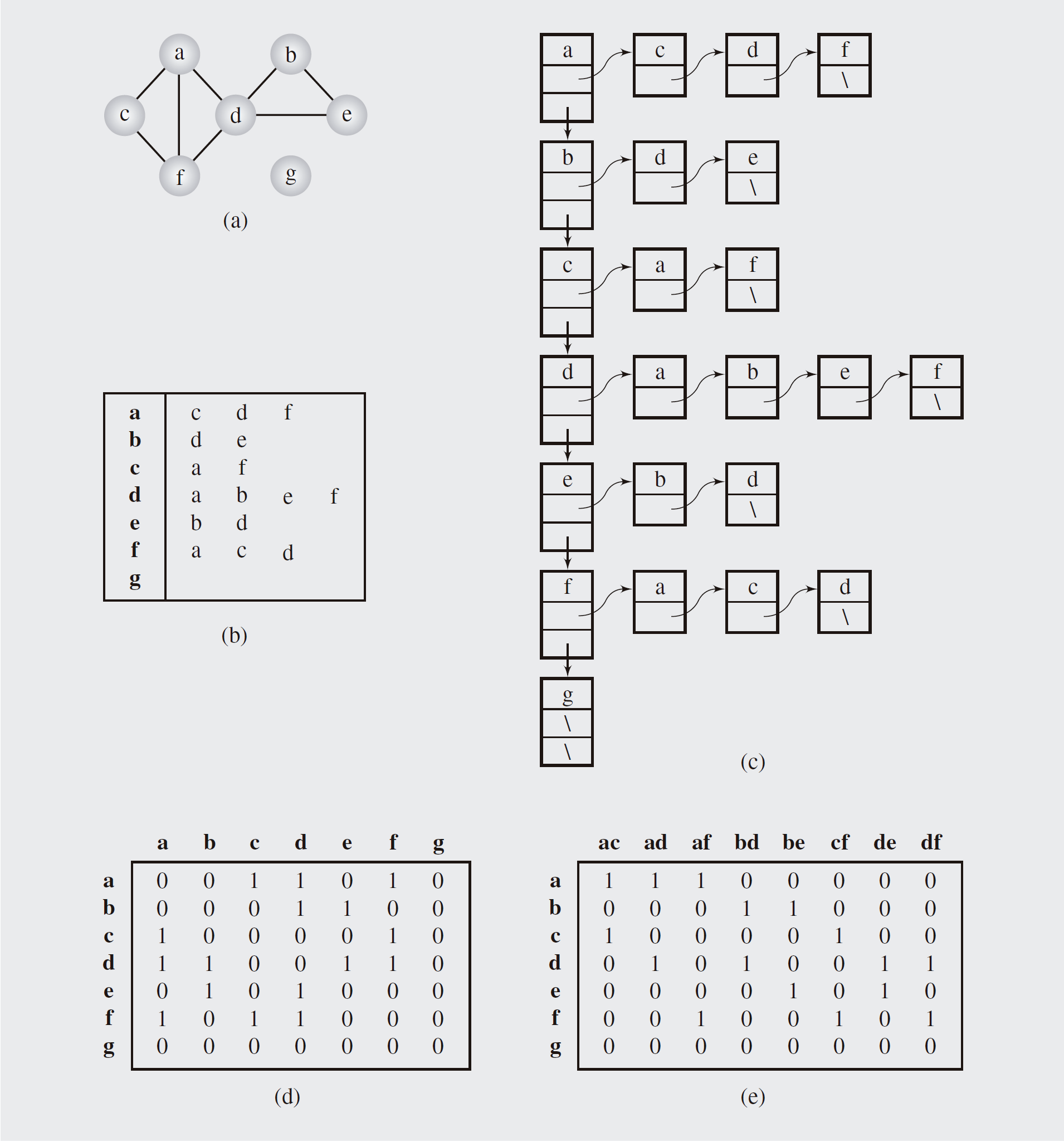
Consiste em partir o grafo em conjuntos de vértices a qual não compartilham nenhuma aresta entre eles.

* 1. **Teorema das 4 cores**

É baseado no problema das cores necessárias para se colorir um mapa sem que os países vizinhos compartilhem da mesma cor. Transformando o mapa em um grafo pode-se provar que pode-se representar qualquer mapa (um grafo planar) com apenas 4 cores (4 partições).

## Algumas representações dos grafos

Um gráfico representado como (b – c) uma lista de adjacências, (d) uma adjacência matriz, e (e) uma matriz de incidência.



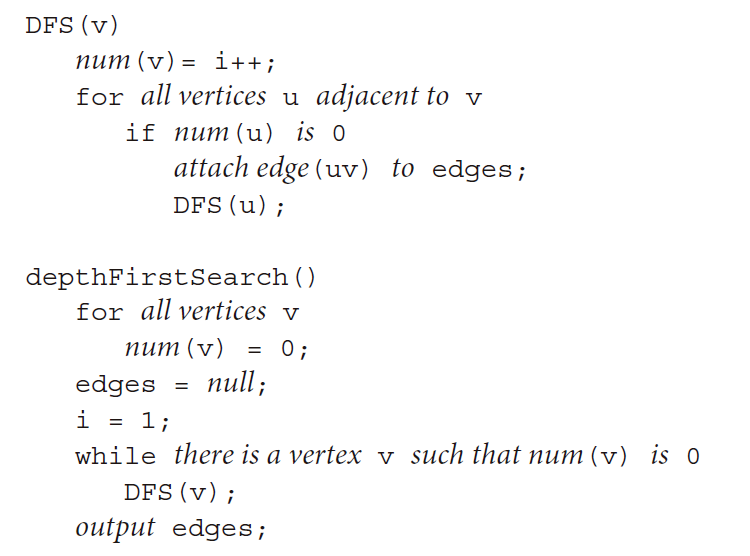
## Tranversal em grafo

Assim como é ocorrido nas árvores, percorrer um grafo significa em visitar uma vértice pelo menos uma vez. Porém, os algoritmos de passagens simples usados nas árvores não podem ser utilizados aqui, pois os grafos podem incluir ciclos. Ocorreria um loop infinito!

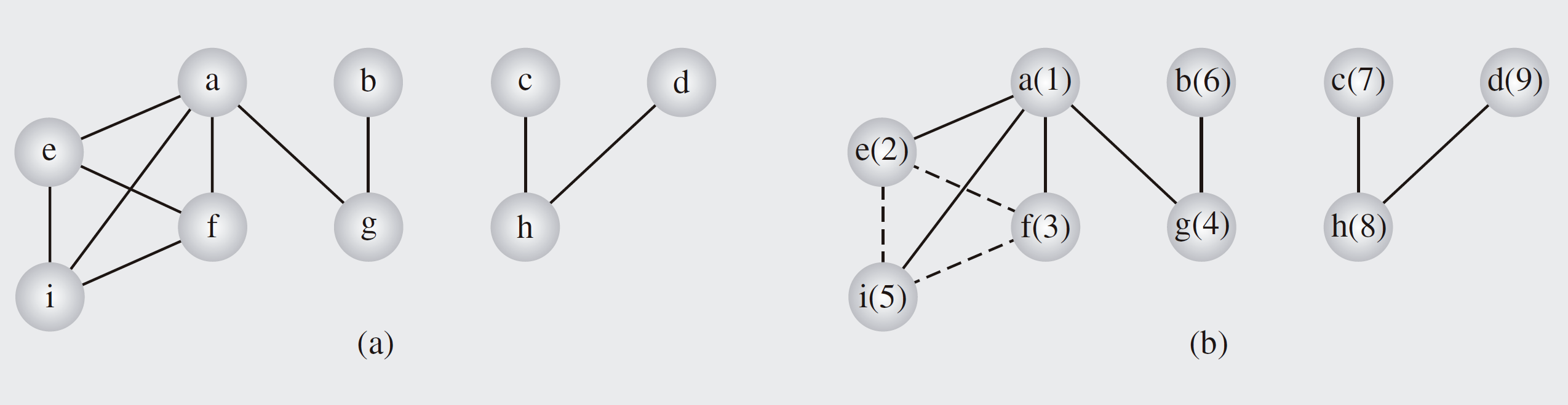
Para evitar essa anomalia, cada vértice visitada pode ser marcada para evitar a revisitação, assim, o grafos podem ser isolados e algumas partes dele é deixado de fora de algum tipo de processo de os métodos de passagem da árvore não modificados forem aplicados.

Um algoritmo desenvolvido por John Hopcroft e Robert Tarjan, conhecido como *depth-first* pode realizar esse tipo de pesquisa. Neste algoritmo, cada vértice v é visitado e, em seguida, cada vértice não visitado adjacente a v é visitado. Se um vértice v não tem adjacente vértices ou todos os seus vértices adjacentes foram visitados, nós retroceder para o antecessor de v. O percurso é concluído se este processo de visita e de o primeiro vértice onde o percurso começou. Se ainda houver alguns vértices não visitados em o gráfico, o percurso continua reiniciando para um dos vértices não visitados.

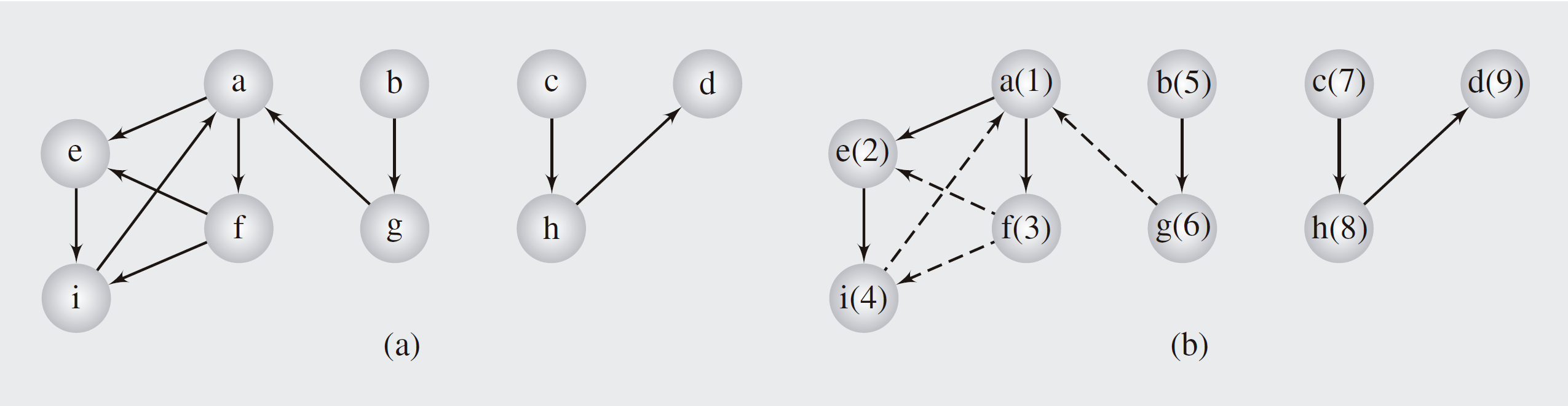
Embora não seja necessário para o resultado adequado deste método, o algoritmo atribui um número exclusivo para cada vértice acessado para que os vértices agora são renumerados. Isso vai ser útil em aplicações posteriores deste algoritmo.



Um exemplo de aplicação do algoritmo breadthFirstSearch () para um grafo:



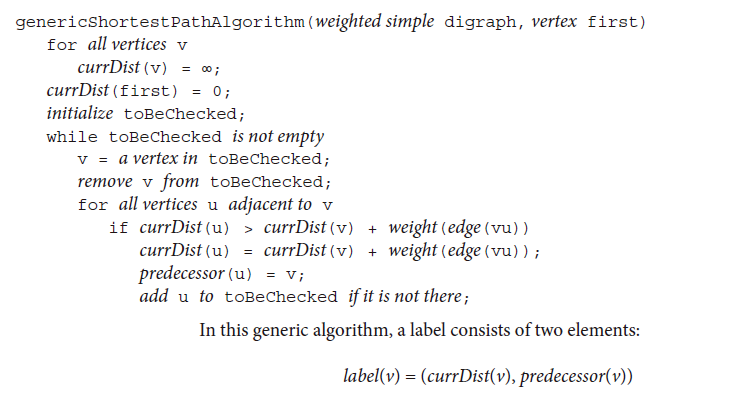
O algoritmo breadthFirstSearch () aplicado a um dígrafo:

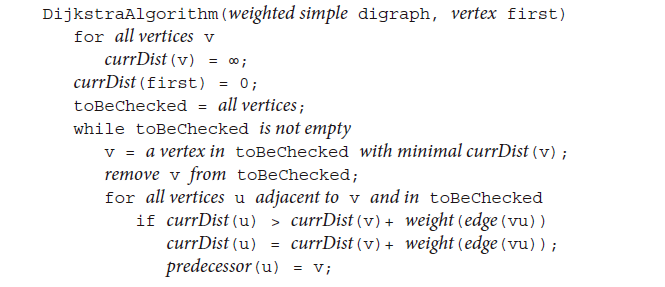


## Caminhos mais curtos

Um grande problema ainda é persistido na teoria dos grafos, e encontrar o caminho mais curto foi um grande desafio, mas há diferentes formas de aplicações e cada uma delas possuem os seus benefícios.

Ao determinar o caminho mais curto do vértice v para o vértice u, informações sobre as distâncias entre os vértices intermediários w têm de ser gravadas Esta informação pode ser registrado como um rótulo associado a esses vértices, onde o rótulo é apenas a distância de v a w ou a distância, juntamente com o antecessor de w neste caminho. Os métodos de encontrar o caminho mais curto dependem desses rótulos. Dependendo em quantas vezes esses rótulos são atualizados, os métodos resolvendo o problema de caminho mais curto são divididos em duas classes: métodos de configuração de rótulo e *labelcorrecting*.

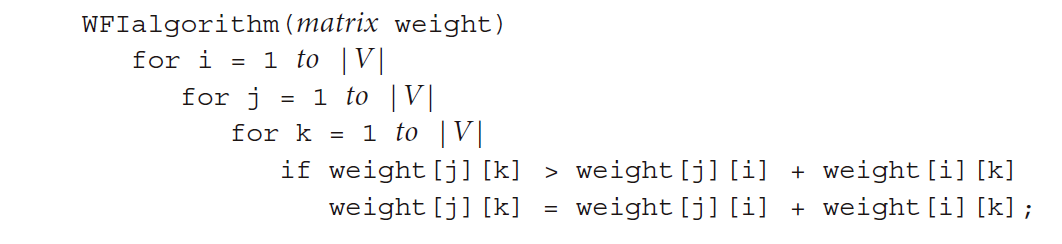
**



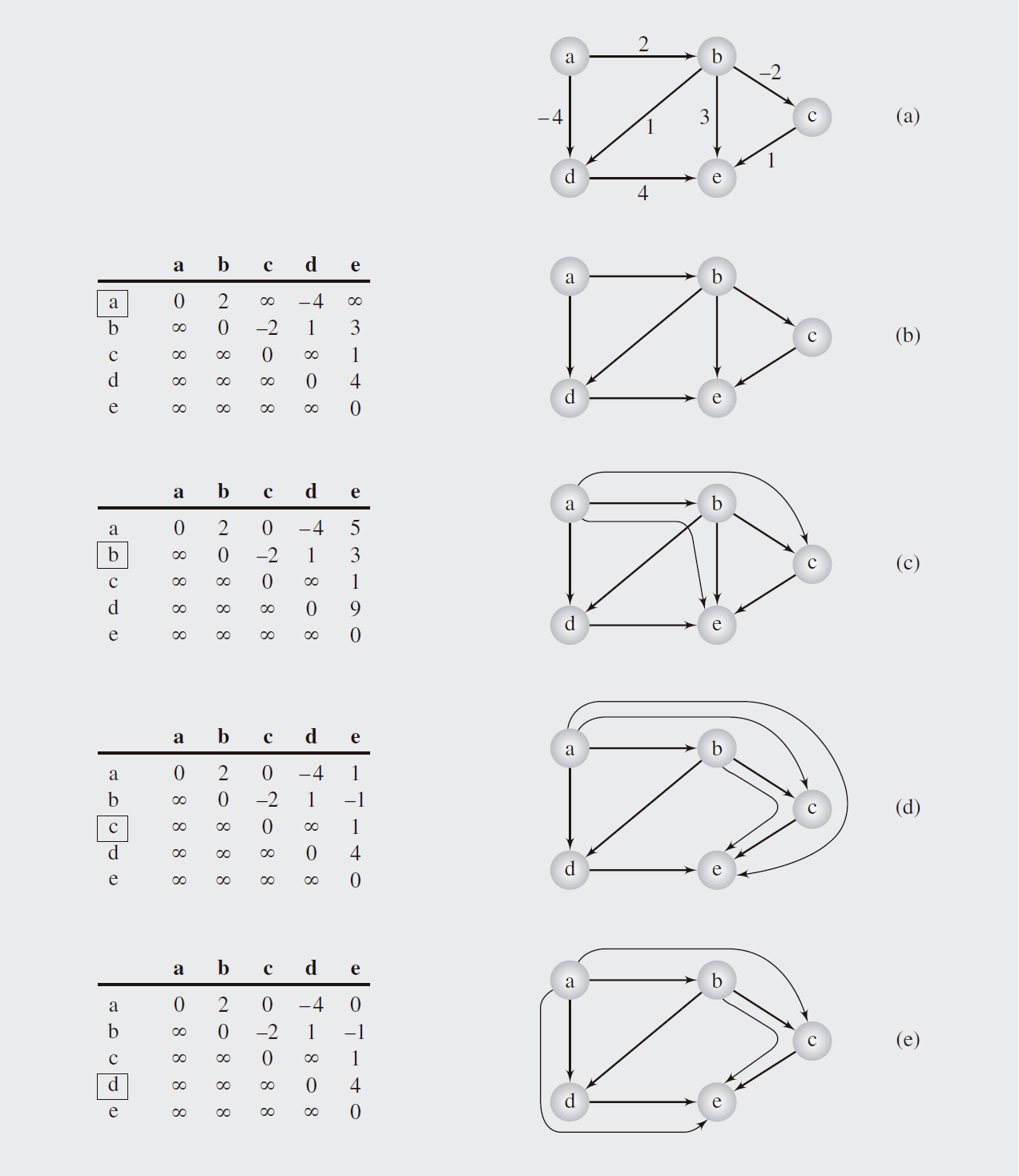
* 1. **Problema do caminho mais curto**

Embora a tarefa de encontrar todos os caminhos mais curtos de qualquer vértice para qualquer outro vértice parece ser mais complicado do que a tarefa de lidar com uma única fonte, um método desenhado por Stephen Warshall e implementado por Robert W. Floyd e P. Z. Ungerman faz isso de uma forma surpreendentemente simples, desde uma matriz adjacente.

O gráfico pode inclua pesos negativos. O algoritmo é o seguinte:



Uma execução de WFIalgorithm():



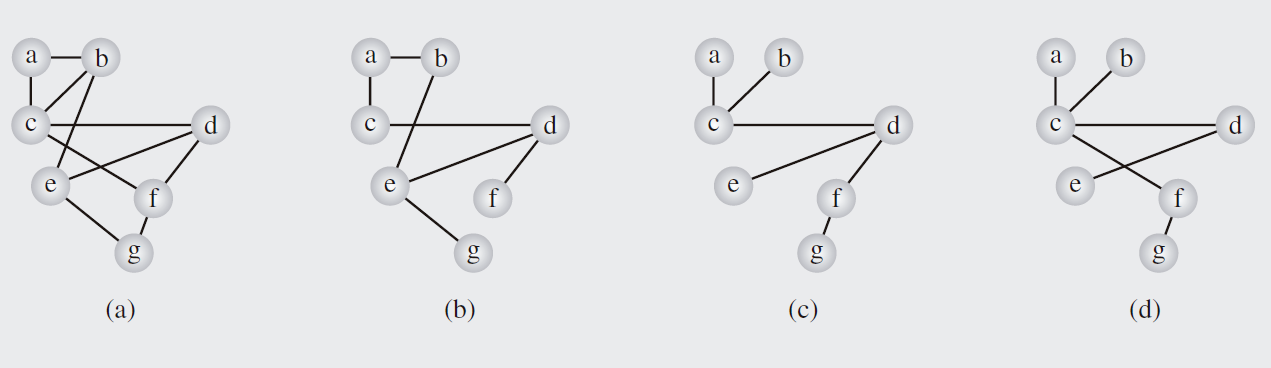
## Detecção de cliclos

Muitos algoritmos dependem de detectar ciclos em gráficos. Acabamos de ver que, como um efeito colateral, WFIalgorithm () permite a detecção de ciclos em gráficos. No entanto, é um algoritmo cúbico, que em muitas situações é muito ineficiente. Portanto, outros métodos de detecção de ciclos têm de ser explorados.

## Árvores de Abrangências

Considere o problema da figura abaixo, que representa a conexão de várias cidades. Se a situação económica obrigar esta companhia a encerrar o maior número de as conexões possíveis, quais delas devem ser mantidas para se certificar de que ele ainda está possível chegar a qualquer cidade de qualquer outra cidade, se apenas indiretamente? Uma possibilidade é a gráfico na figura

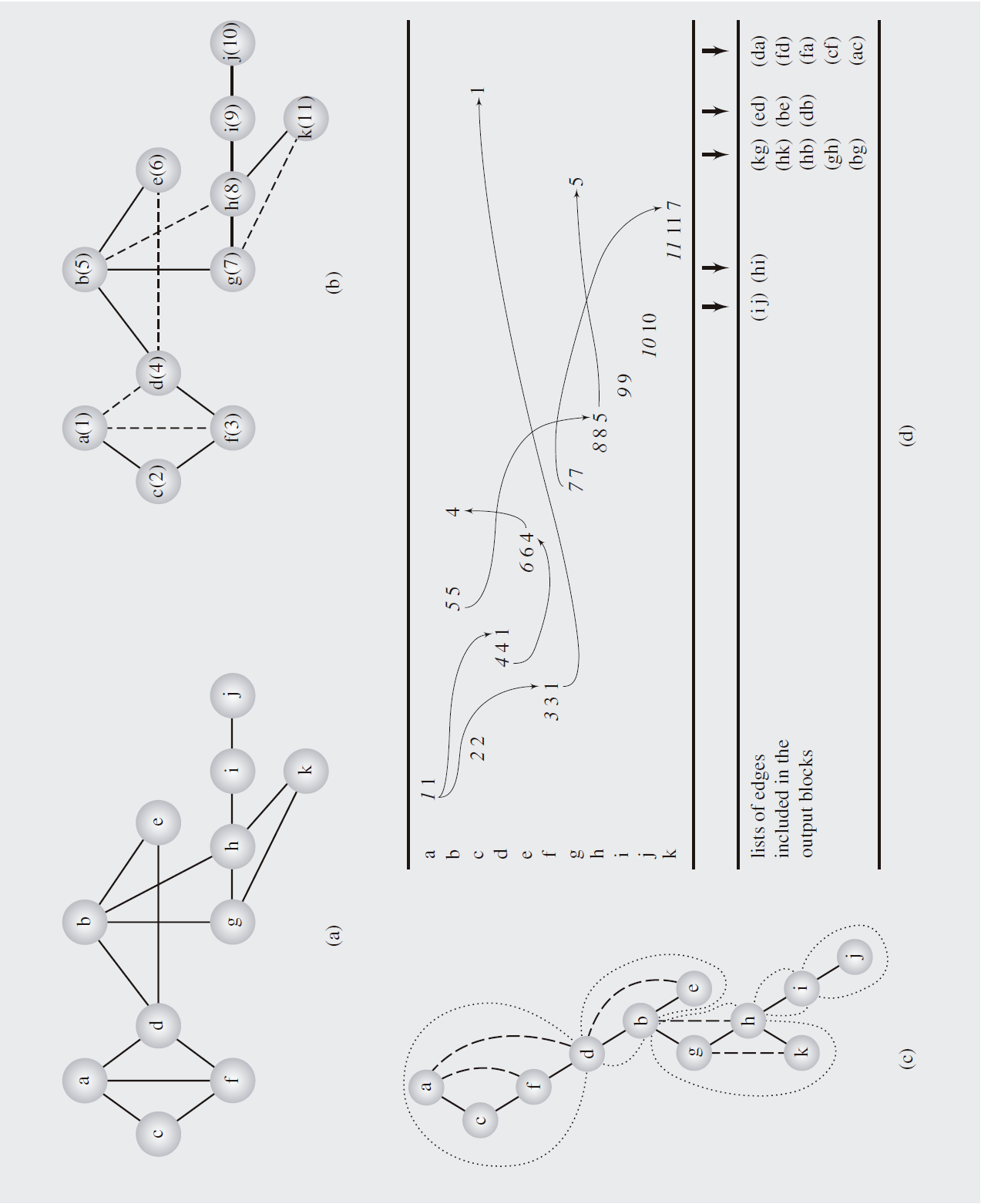
Um gráfico representando (a) as conexões aéreas entre sete cidades e (b – d) três possíveis conjuntos de conexões:



A solução para este problema não é ideal em que as distâncias entre as cidades não foram tidas em conta. Porque existem conexões de seis bordas alternativas entre as cidades, a companhia aérea usa o custo dessas conexões para escolher o melhor, garantindo o custo ideal. Isso pode ser alcançado por ter distâncias mais curtas para os seis Conexões. Este problema pode agora ser formulado como encontrar uma árvore de abrangência mínima, que é uma árvore de abrangência em que a soma dos pesos de suas bordas é mínima. O problema anterior de encontrar uma árvore de abrangência em um gráfico simples é um caso de mínimo problema da árvore de abrangência em que os pesos para cada borda são considerados iguais um. Portanto, cada árvore de abrangência é uma árvore mínima em um gráfico simples.

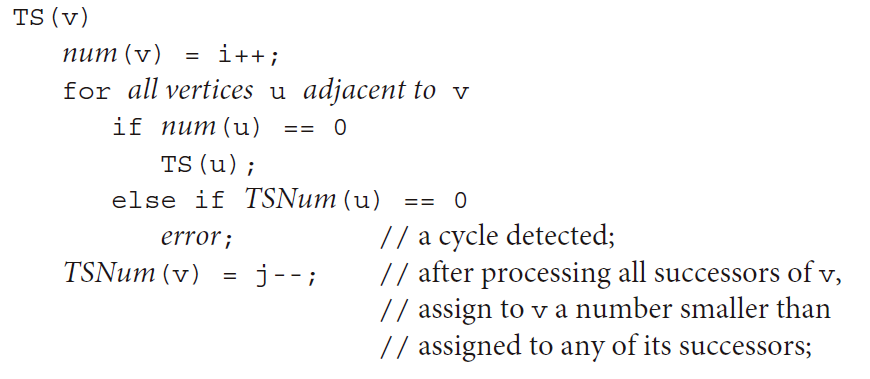
## Conectividade

Em muitos problemas, estamos interessados em encontrar um caminho no gráfico de um vértice para qualquer outro vértice. Para gráficos não direcionados, isso significa que não há partes separadas, ou subgrafos, do gráfico; para um digrafo, isso significa que existem alguns lugares no gráfico para o qual podemos obter a partir de algumas direções, mas não são necessariamente capazes de retornar aos pontos de partida.

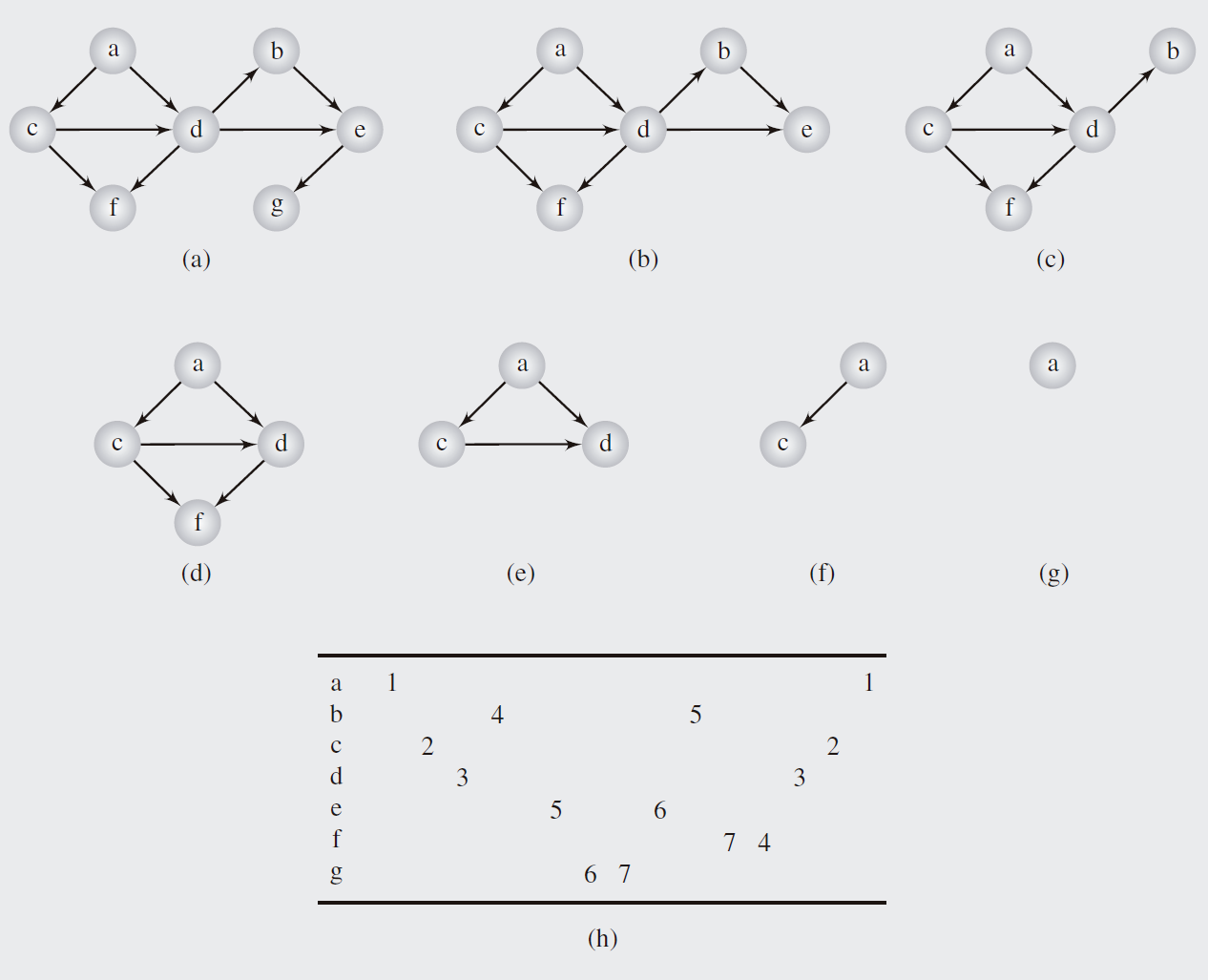


## Tipo topológico

Em muitas situações, há um conjunto de tarefas a serem executadas. Para alguns pares de tarefas, importa qual tarefa é executada primeiro, enquanto que para outros pares, a ordem de execução não é importante. Por exemplo, os alunos precisam levar em consideração quais cursos são pré-requisitos ou corequisitos para outros cursos ao fazer um cronograma para o próximo semestre para que o computador de programação II não pode ser tomada antes de programar o computador I, mas o primeiro pode ser levado junto com, digamos, ética ou introdução à Sociologia.



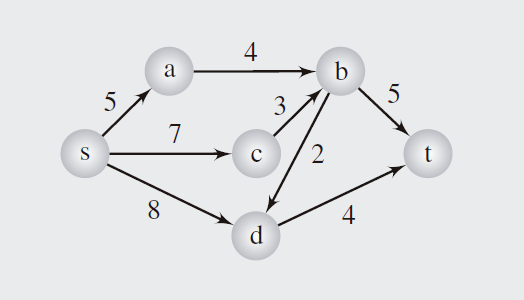
Exemplo de execução de Topological Sort:



## Redes

Um tipo importante de gráfico é uma rede. Uma rede pode ser exemplificada por uma rede de tubulações usadas para entregar a água de uma fonte a um destino. No entanto, a água não é simplesmente bombeado através de um tubo, mas através de muitas tubulações com estações manypumping no meio. As tubulações são do diâmetro diferente e as estações sãodo o poder diferente de modo que a quantidade de água que pode ser bombeada possa diferir o encanamento deum a outro. Por exemplo, a rede na Figura 8,19 tem oito tubos e seis estações de bombeamento. Os números mostrados nesta figura são as capacidades máximas de cada pipeline. Por exemplo, o tubo que vai a nordeste da fonte s, a pipesa, tem uma capacidade de 5 unidades (digamos, 5.000 galões por hora). O problema é maximizar a capacidade de toda a rede para que ele possa transferir a quantidade máxima de água.

Pode não ser óbvio como realizar esse objetivo. Observe que o pipe SA vindo da fonte vai para uma estação que tem apenas um tubo de partida, AB, ofcapacity 4. Isto significa que não podemos colocar 5 unidades através de pipe SA, porque pipe AB não pode transferi-lo. Também, a quantidade de água que vem à estação b tem que ser controlada também porque se ambas as tubulações entrantes, AB e CB, são usadas à capacidade cheia, a seguir a tubulação de entrada, BT, não pode processá-la tampouco. Está longe de ser óbvio, especialmente para grandes redes, o que as quantidades de água colocada através de cada tubo deve ser a utilização da rede maximamente. A análise computacional deste problema de rede particular foi iniciada por Lester R. Ford e D. Ray Fulkerson. Desde o seu trabalho, dezenas de algoritmos foram publicados para resolver este problema.



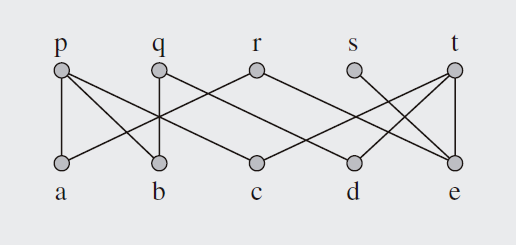
**Teorema:** Em qualquer rede, o fluxo máximo de s para t é igual ao mínimo capacidade de qualquer corte.

## Matching (do inglês, Correspondência)

Suponha que há cinco vagas de emprego a, b, c, d, e e e cinco candidatos p, q, r, s, e t com as qualificações apresentadas nesta tabela:

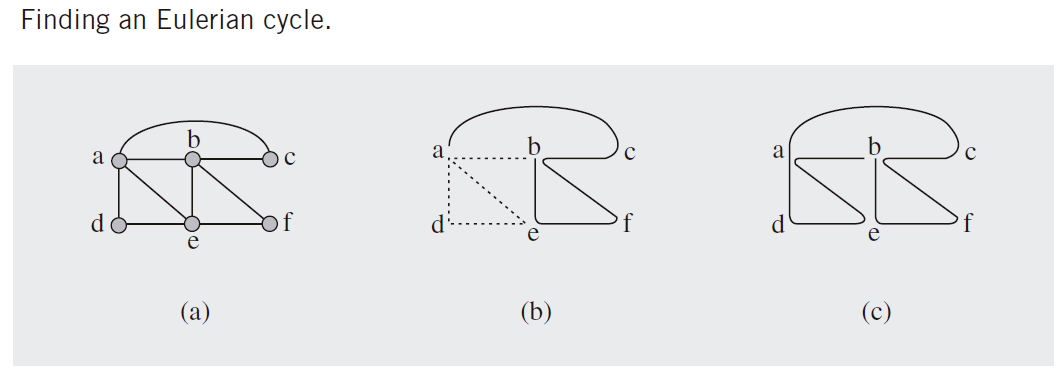
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Applicants: | p | q | r | s | t |
| Jobs: | abc | bd | ae | e | cde |

O problema é como encontrar um trabalhador para cada trabalho; ou seja, como combinar trabalhos com trabalhadores. Há muitos problemas deste tipo. O problema de correspondência de trabalho pode ser modelado com um gráfico bipartido. Um gráfico bipartido é aquele em que o conjunto de vértices v pode ser dividido em dois subconjuntos v1 e v2 de tal forma que, para cada aresta (VW), se vértice v está em um dos dois conjuntos v1 ou v2, então w está no outro conjunto. Neste exemplo, um conjunto de vértices, v1, representa os candidatos, o outro conjunto, v2, representa empregos e bordas representam trabalhos para os quais os candidatos são qualificados (Figura 8,26). A tarefa é encontrar um fósforo entre o trabalho e os candidatos de modo que um requerente seja combinado com um trabalho. Em um caso geral, pode não haver candidatos suficientes, ou pode haver nenhuma maneira de atribuir um requerente para cada abertura, mesmo se o número de requerentes exceder o número de aberturas. Por conseguinte, a tarefa agora é atribuir aos requerentes o maior número possível de postos de trabalho.



## Grafo de Euler e de Hamilton

Uma trilha Euleriana em um gráfico é um caminho que inclui todas as arestas do gráfico apenas uma vez. Um ciclo euleriano é um ciclo que é também uma trilha euleriana. Um gráfico que tem um euleriano ciclo é chamado um gráfico euleriano. Um teorema comprovado por Euler (pronuncia-se: oiler) diz que um grafo é euleriano se todo vértice do grafo é incidente a um número par de arestas. Além disso, um gráfico contém uma trilha euleriana se tiver exatamente dois vértices incidentes com um número ímpar de arestas.



## Colocaração de grafos

Às vezes, queremos encontrar um número mínimo de conjuntos de vértices não sobrepostos, onde cada conjunto inclui vértices que são independentes - isto é, eles não estão conectados por nenhuma borda. Por exemplo, há várias tarefas e várias pessoas executando essas tarefas. Se uma tarefa pode ser executada por uma pessoa ao mesmo tempo, as tarefas precisam ser agendadas para que seja possível realizá-las. Nós formamos um gráfico no qual as tarefas são representadas por vértices; duas tarefas são unidas por uma aresta se a mesma pessoa for necessária para realizá-las, ou seja, as duas tarefas não podem ser executadas por uma pessoa ao mesmo tempo. Agora tentamos construir um número mínimo de conjuntos de tarefas independentes. Como as tarefas de um conjunto podem ser executadas simultaneamente, o número de conjuntos indica o número de intervalos de tempo necessários para executar todas as tarefas.