6 Árvores

Este capítulo aborda algumas questões relativas a árvores, considerando que uma árvore é um tipo particular de grafo. Iniciamos com a caracterização de árvores dentro da Teoria dos Grafos, em seguida apresentamos o conceito de árvores geradoras e um exemplo de aplicação na engenharia estrutural para solucionar o problema de rigidez de treliças planas. Por fim, este capítulo traz o conceito de grafo valorado, juntamente com o problema das árvores geradoras de custo mínimo e dois algoritmos para resolvê-lo.

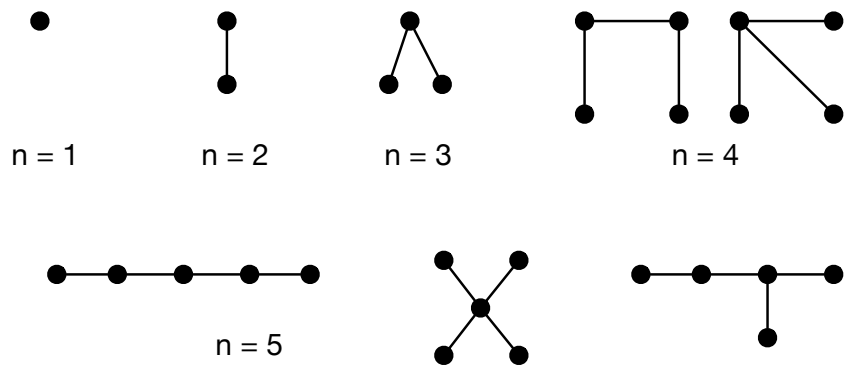
6.1 Caracterização de árvores

Árvores são importantes para o entendimento estrutural de grafos e para o design e análise de redes. Embora a disciplina de Estrutura de Dados trate de árvores como estruturas para armazenamento e recuperação eficiente de informação, entre outras aplicações, no presente capítulo as árvores são abordadas sob a ótica da Teoria dos Grafos.

Definição 6.1 (Árvore)**.** *Uma árvore é um grafo sem ciclos.*

Exemplo 6.1. *A Figura 6.1 mostra todas as árvores não rotuladas com até 5 vértices.*

Figura 6.1 – Árvores não rotuladas com até 5 vértices



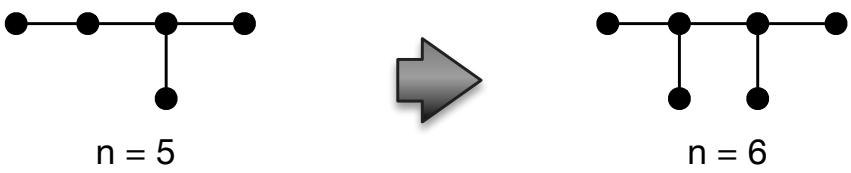
Fonte: o autor.

Árvores possuem algumas propriedades matemáticas interessantes, que auxiliam na sua caracterização como um tipo especial de Grafo. Mostraremos essas propriedades por meio das seguintes proposições:

Proposição 1: qualquer árvore com n vértices tem $n - 1$ arestas.

Qualquer árvore não rotulada com n vértices pode ser obtida a partir de uma árvore com $n - 1$ vértices adicionando-se uma aresta conectando o novo vértice a um vértice preexistente. Por exemplo, a Figura 6.2 mostra o exemplo da obtenção de uma árvore não rotulada de 6 vértices a partir de uma com 5 vértices.

Figura 6.2 – Construção de nova árvore não rotulada



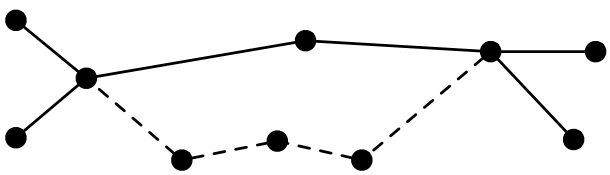
Fonte: o autor.

Iniciando por uma árvore com apenas um vértice, podemos construir qualquer árvore não rotulada adicionando sucessivamente uma nova aresta e um novo vértice. Em cada estágio, o número de vértices excede o número de arestas em 1 unidade, provando a veracidade da proposição 1.

Proposição 2: qualquer par de vértices em uma árvore está conectado por exatamente um caminho.

No processo de criação de árvores de n vértices a partir de árvores com $n - 1$ vértices, nenhum ciclo é criado porque a aresta adicionada conecta um vértice preexistente com o vértice novo. Em cada estágio o grafo permanece conexo, portanto, existe pelo menos um caminho conectando quaisquer dois vértices. Porém, eles não podem ser conectados por mais de um caminho porque qualquer par de caminhos configuraria um ciclo, conforme ilustrado pelas linhas tracejadas na Figura 6.3.

Figura 6.3 – Formação de um ciclo

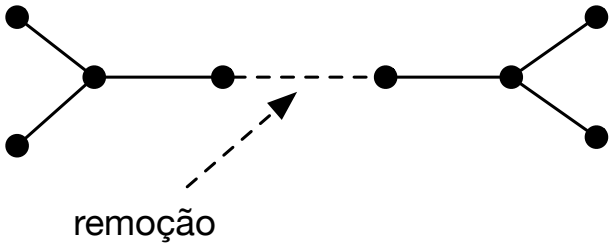


Fonte: o autor.

Proposição 3: a remoção de qualquer aresta de uma árvore desconecta a árvore.

A explicação para esta proposição é que quaisquer dois vértices **adjacentes** são conectados por exatamente um caminho — a aresta que os conecta — então a remoção desta aresta faz com que não haja mais caminho entre o par de vértices, conforme mostrado na Figura 6.4.

Figura 6.4 – Remoção de aresta de uma árvore

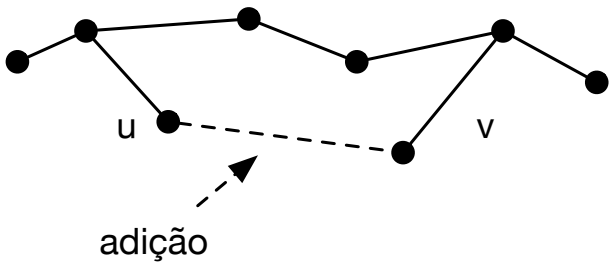


Fonte: o autor.

Proposição 4: a adição de uma aresta uma árvore cria um ciclo.

Considerando que dois vértices u e v de uma árvore já estão conectados por um caminho, a adição de uma nova aresta (u, v) produz um ciclo, o ciclo composto pelo caminho e pela aresta (u, v), conforme ilustrado na Figura 6.5.

Figura 6.5 – Adição de uma nova aresta em uma árvore



Fonte: o autor.

Estas proposições podem ser usadas como definições de árvores. O teorema 6.1 reúne as proposições em seis definições diferentes, sendo todas elas equivalentes. Qualquer uma das afirmações do teorema pode ser usada como definição de árvore e as outras cinco podem então ser deduzidas a partir dela.

Teorema 6.1 (Definições equivalentes de árvores)**.** *Seja T um grafo com n vértices. Então as seguintes afirmações são equivalentes:*

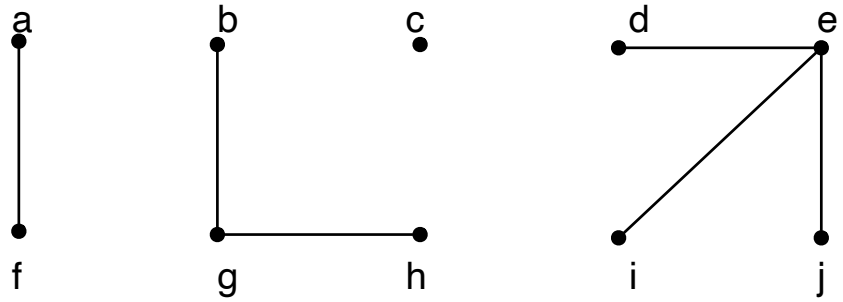
1. *T é uma árvore*[1]*.*
2. *T não tem ciclos e tem $n-1$ arestas.*
3. *T é conexo e tem $n-1$ arestas.*
4. *T é conexo e a remoção de qualquer uma de suas arestas o desconecta*[2]*.*
5. *Quaisquer dois vértices de T estão conectados por exatamente um caminho.*
6. *T não contém ciclos. E para qualquer nova aresta e, o grafo $(T+e)$ tem exatamente um ciclo*[3]*.*

Complementando as definições anteriores, temos o conceito de **floresta**:

Definição 6.2 (Floresta)**.** *Floresta é um grafo no qual qualquer par de vértices está conectado* ***no máximo*** *por um caminho. Uma floresta pode ser composta de uniões disjuntas de árvores.*

Exemplo 6.2. *A Figura 6.6 mostra um exemplo de floresta: o grafo $G=(V,E)$, sendo $V=\{a,b,c,d,e,f,g,h,i,j\}$ e $E=\{(a,f),(b,g),(h,g),(d,e),(j,e),(e,i)\}$*

Figura 6.6 – Exemplo de Floresta



Fonte: o autor.

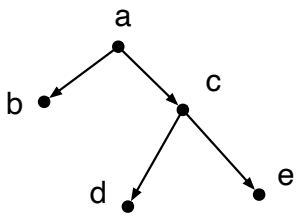
Ao considerarmos grafos dirigidos, a definição de árvore é significativamente diferente, como veremos a seguir.

Definição 6.3 (Árvore dirigida)**.** *Uma árvore dirigida T é um digrafo acíclico onde o grau de entrada de cada vértice é 1, exceto o do vértice raiz, que tem grau de entrada zero. A raiz de uma árvore dirigida T é um vértice r tal que qualquer outro vértice de T pode ser alcançado a partir de r.*

[1] Ou seja, T é um grafo conexo sem ciclos.
[2] Em outras palavras, todas as arestas de T são pontes.
[3] A adição da aresta e cria um ciclo em T.

Exemplo 6.3. *A Figura 6.7 mostra uma árvore dirigida, com o vértice a sendo sua raiz.*

Figura 6.7 – Árvore dirigida



Fonte: o autor.

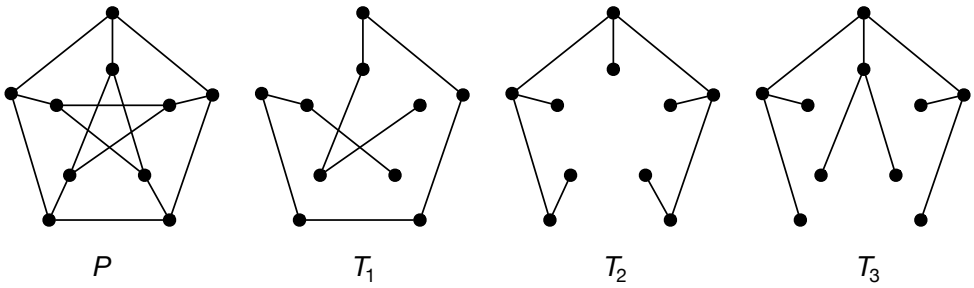
6.2 Árvores geradoras

As árvores geradoras, do inglês *Spanning Trees*[4] são amplamente utilizadas na engenharia e na tecnologia da informação para otimização e redução de custos de redes. Um exemplo deste conceito em tecnologia é o protocolo de redes *Spanning Tree Protocol*, que impede a formação de loops quando switches ou pontes são interligadas por vários caminhos diferentes, construindo redes lógicas livres de loops em redes Ethernet (STEVENS, 1993).

Definição 6.4 (Árvore geradora)**.** *Seja G um grafo conexo. Uma árvore geradora T em G é um subgrafo de G que inclui todos os seus vértices e também é uma árvore.*

Exemplo 6.4. *A Figura 6.8 mostra o grafo de Petersen P, com três de suas árvores geradoras: T_1, T_2 e T_3. Se o grafo for rotulado, ele tem 2000 árvores geradoras.*

Figura 6.8 – Árvores geradoras do grafo de Petersen



Fonte: o autor.

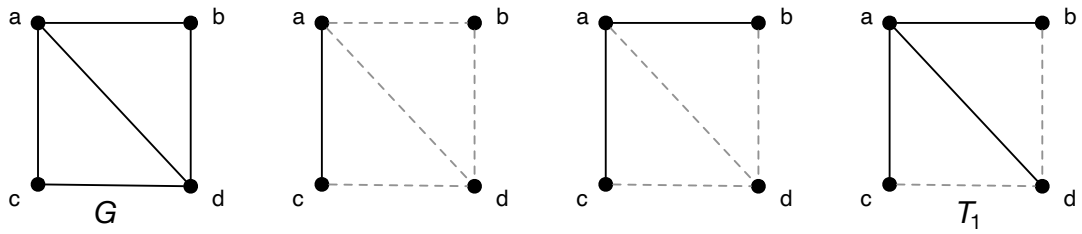
Dado um grafo conexo G, intuitivamente podemos utilizar dois métodos diferentes para encontrar uma árvore geradora T em G, o método por construção e o método por

[4] Não há um consenso sobre a tradução deste termo para o português. Alguns autores traduzem como "árvores de abrangência" ou "árvores de extensão". Nós preferimos usar o termo "árvores geradoras".

remoção. O conhecimento desses métodos é útil para a compreensão dos algoritmos para o problema do conector mínimo, que veremos a seguir.

No método construtivo, selecionamos arestas do grafo, uma de cada vez, tomando o cuidado para não formar ciclos. Repetimos o procedimento até que todos os vértices sejam incluídos. Por exemplo, no grafo G da Figura 6.9, selecionamos as arestas (a, c), (a, c) e (a, c), obtendo a árvore geradora T_1.

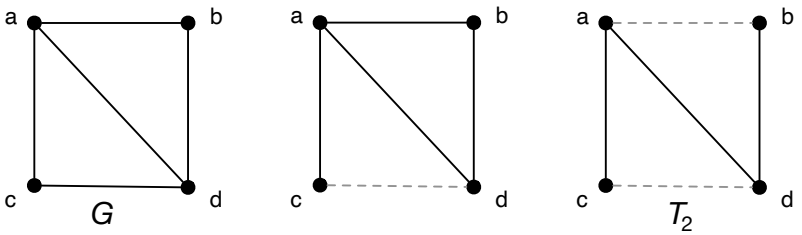
Figura 6.9 – Árvores geradora pelo método construtivo



Fonte: o autor.

No método por remoção, escolhemos um ciclo e removemos qualquer uma de suas arestas. Repetimos este procedimento até que não restem mais ciclos. Por exemplo, no grafo G da Figura 6.10, removemos a aresta (c, d), destruindo o ciclo acd, em seguida removemos a aresta (a, b) destruindo o ciclo abd e obtendo a árvore geradora T_2.

Figura 6.10 – Árvores geradora por redução



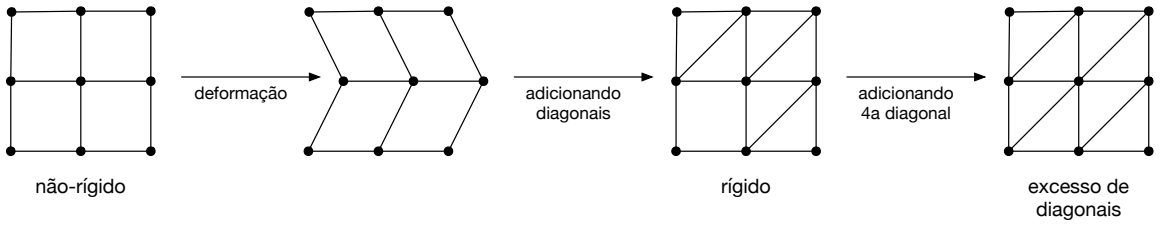
Fonte: o autor.

6.2.1 Estudo de caso: rigidez de treliças planas

Treliças planas são estruturas muito utilizadas na construção de telhados, galpões, pontes e torres de comunicação. Compostas por barras metálicas ou de madeira que funcionam por tração ou compressão, elas tem como característica principal a falta de rigidez para rotação nas conexões entre as barras. Isso faz com que a estrutura possa ser facilmente deformada no plano, o que não é desejável. Para adicionar rigidez à estrutura, utilizam-se barras diagonais. A Figura 6.11 mostra um exemplo de treliça não rígida com

2 linhas e duas colunas. Uma força lateral facilmente a deformaria. Ao adicionarmos 3 diagonais, tornamos a estrutura rígida. Note que não seria necessário adicionar uma quarta diagonal, visto que estaríamos "gastando" material desnecessariamente em uma estrutura já rígida.

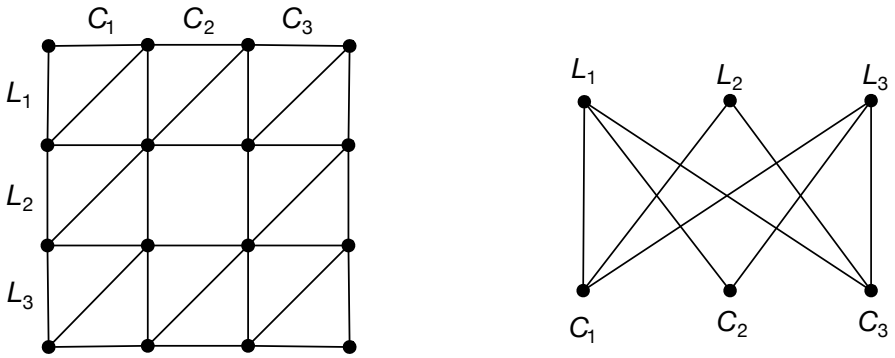
Figura 6.11 – Adicionando rigidez a uma treliça plana



Fonte: o autor.

Podemos utilizar grafos para analisar a questão de rigidez, bem como falta ou excesso de diagonais em treliças planas. Para modelar estes grafos, vamos considerar as linhas e colunas de retângulos formados nos espaços entre as barras horizontais e verticais. Enumerando as linhas de retângulo sequencialmente de cima pra baixo, temos $l_1, l_2, \ldots$. Analogamente, enumeramos as colunas $c_1, c_2, \ldots$ da esquerda para a direita. O retângulo na linha i e coluna j é $r(i, j)$. Na modelagem do problema, criamos um grafo bipartido $G = (V_1 \cup V_2, E)$, em que $V_1 = \{l_1, l_2, \ldots\}$, $V_2 = \{c_1, c_2, \ldots\}$ e $E = \{(i, j) |$ existe uma barra diagonal no retângulo $r(i, j)\}$. A Figura 6.12a mostra uma estrutura de treliça com 3 linhas e 3 colunas. Na Figura 6.12b, podemos ver o grafo bipartido resultante da modelagem do problema. Note que os vértices l_2 e c_2 não são adjacentes porque não existe barra diagonal no retângulo $r(2, 2)$ da estrutura.

Figura 6.12 – Modelagem de treliças



(a) Estrutura 3×3 (b) Grafo bipartido resultante

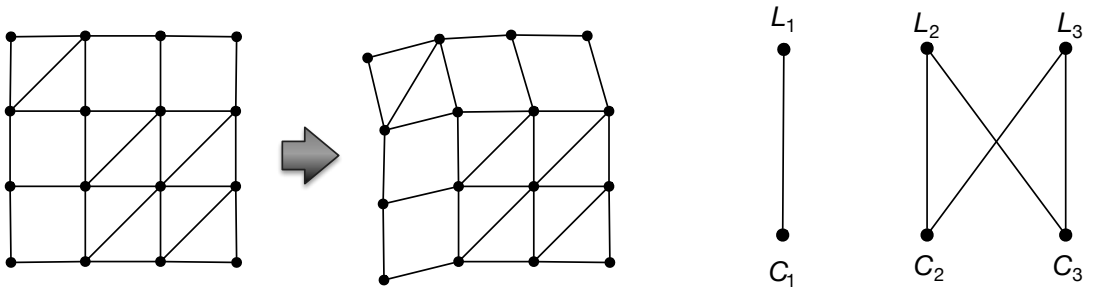
Fonte: o autor.

Para a estrutura de treliça ser rígida, as seguintes condições precisam ser atendidas: (i) a linha l_i deve permanecer paralela à linha l_j, para todas as linhas l_i e l_j; (ii) a coluna

c_i deve permanecer paralela à coluna c_j, para todas as colunas c_i e c_j; e (iii) a linha l_i deve permanecer perpendicular à coluna c_j, para todas as linhas l_i e colunas c_j.

Treliças rígidas sempre resultam em grafos bipartidos conexos, como o da Figura 6.12. Por outro lado, se o grafo for não-conexo, significa que a treliça modelada não tem rigidez, podendo ser facilmente deformada como a estrutura mostrada na Figura 6.13.

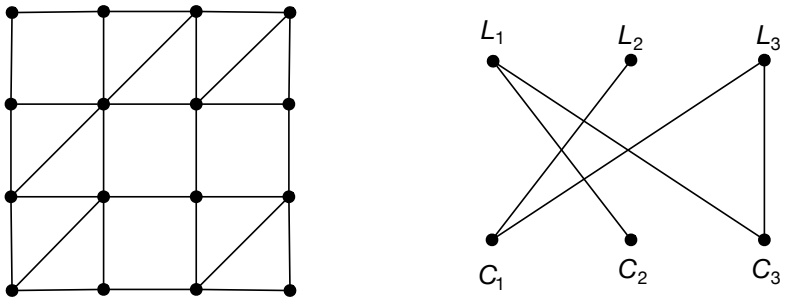
Figura 6.13 – Treliça sem rigidez



Fonte: o autor.

Além disso, se o grafo resultante da modelagem da estrutura for uma árvore geradora, a estrutura está numa configuração rígida utilizando o mínimo de barras diagonais possível para manter a rigidez, conforme ilustrado pelo exemplo da Figura 6.14.

Figura 6.14 – Treliça rígida com o mínimo de barras diagonais



Fonte: o autor.

6.3 Grafos valorados

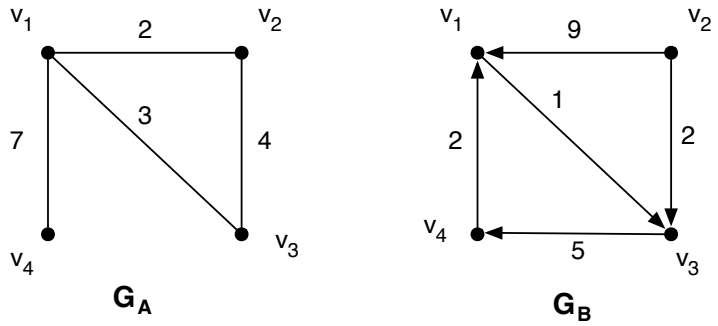
A seguir, veremos o problema das árvores geradoras de custo mínimo. Mas antes precisamos conhecer um novo tipo de grafo, chamado Grafo Valorado.

Definição 6.5 (Grafo valorado)**.** *Um grafo valorado é um grafo (dirigido ou não dirigido) no qual cada uma de suas arestas está associada a um valor numérico chamado peso ou custo da aresta.*

O custo de uma aresta representa algum dado significativo do problema modelado pelo grafo. Este dado pode ser distância, custo monetário, tempo, ou qualquer outro parâmetro relevante que pode ser expresso como um valor numérico. Dependendo do problema, podemos, inclusive, ter custos negativos. Dependendo do contexto, o custo de uma aresta (v_i, v_j) pode ser denotado por $c_{i,j}$ (de "custo"), $d_{i,j}$ (de "distância") ou $w_{i,j}$ (da palavra *"weight"*). Um custo de valor infinito entre v_i e v_j representa ausência de aresta ou de caminho entre estes dois vértices.

A Figura 6.15 ilustra dois exemplos de grafos valorados. No grafo G_a, por exemplo, o custo da aresta (v_1, v_2) é igual a 2, ou $c_{1,2} = d_{1,2} = w_{1,2} = 2$.

Figura 6.15 – Exemplos de grafos valorados



Fonte: o autor.

Definição 6.6 (Matriz de custos)**.** *Dado um grafo valorado $G = (V, E)$ com n vértices, sua matriz de custos é uma matriz W de tamanho $n \times n$, com seus elementos definidos da seguinte forma:*

$$w_{i,j} = \begin{cases} 0 & se\ v_i = v_j, \\ \infty & se\ (v_i, v_j) \notin E, \\ custo & se\ (v_i, v_j) \in E. \end{cases}$$

Exemplo 6.5. *Para os grafos G_a e G_b, da Figura 6.15, teríamos, respectivamente as seguintes matrizes de custo W_a e W_b:*

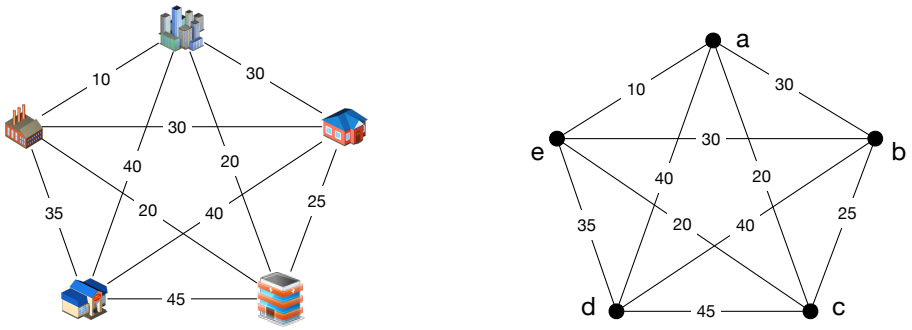
$$W_a = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 7 \\ 2 & 0 & 4 & \infty \\ 3 & 4 & 0 & \infty \\ 7 & \infty & \infty & 0 \end{bmatrix} \end{array} \qquad W_b = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & \infty & e_3 & \infty \\ 9 & 0 & 2 & \infty \\ \infty & \infty & 0 & 5 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

6.4 Problema de árvore geradora de custo mínimo

No desenho de circuitos eletrônicos, uma série de componentes é interconectada por fios localizados em uma placa de apoio. Por diversas razões (técnicas, econômicas etc.), é desejável que se use a menor quantidade possível de fio no circuito. Um problema semelhante ocorre, por exemplo em redes de computadores e projetos de redes de fornecimento de água tratada, eletricidade ou estradas.

Para ilustrar o problema, a Figura 6.16a mostra 5 localidades mutuamente interligadas por estradas não pavimentadas. Os valores associados às linhas correspondem à distância de cada trecho de estrada. Queremos pavimentar estradas usando a quantidade mínima de asfalto de forma que todas as localidades estejam interligadas indiretamente por estradas asfaltadas. Portanto, precisamos definir quais estradas serão pavimentadas e quais não serão.

Figura 6.16 – Problema de pavimentação de estradas



(a) Mapa das 5 cidades (b) Grafo valorado resultante

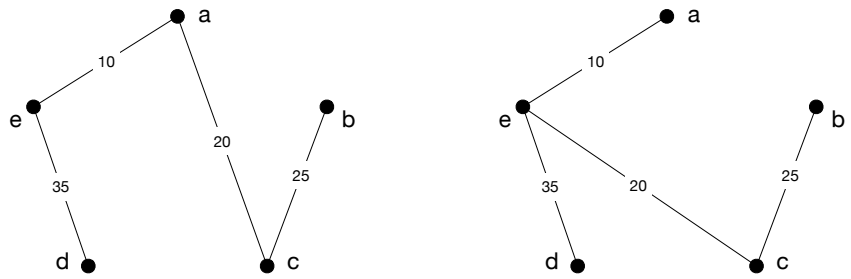
Fonte: o autor.

Para buscar a solução do problema, vamos modelar o mapa como um grafo valorado, mostrado na Figura 6.16b tendo as cidades como vértices e os trechos de estradas como arestas valoradas. O custo de cada aresta corresponde à extensão de cada trecho de estrada.

Sabendo que a pavimentação das estradas deve ser feita utilizando a quantidade mínima de asfalto e que todas as cidades devem estar "conectadas" indiretamente por estradas pavimentadas, a escolha de quais estradas pavimentar leva a um desenho de

árvore geradora do grafo. De todas as árvores geradoras, queremos encontrar aquela cuja soma dos custos de suas arestas seja o menor possível, correspondendo ao projeto mais "barato" para a pavimentação da malha rodoviária. A Figura 6.17 mostra duas soluções, ambas com custo total no valor de 90.

Figura 6.17 – Duas soluções do problema de pavimentação de estradas



Fonte: o autor.

Este problema é chamado de Problema da Árvore Geradora de Custo Mínimo ou Problema do Mínimo Conector. Existem dois algoritmos para encontrar árvores geradoras de custo mínimo em grafos valorados: o algoritmo de Prim e o algoritmo de Kruskal. Antes de conhecer os algoritmos, vamos definir o conceito de árvore geradora de custo mínimo.

Definição 6.7 (Árvore geradora de custo mínimo)**.** *Seja T uma árvore geradora com custo total mínimo em um grafo conexo valorado G. Então, T é uma árvore geradora de custo mínimo ou um conector mínimo de G.*

6.4.1 Algoritmo de Prim

Este algoritmo foi inicialmente desenvolvido pelo matemático tcheco Vojtěch Jarník em 1930 (JARNÍK, 1931), e depois redescoberto e publicado por Robert Prim em 1957 (PRIM, 1957). Trata-se de um algoritmo guloso e sua ideia básica é iniciar a formação da árvore geradora de custo mínimo a partir de um vértice raiz arbitrário. Em cada iteração do algoritmo, a árvore cresce por meio da adição de um novo vértice. Dentre todos os vértices candidatos a se conectarem na árvore, é escolhido aquele que adiciona o menor custo possível naquele momento.

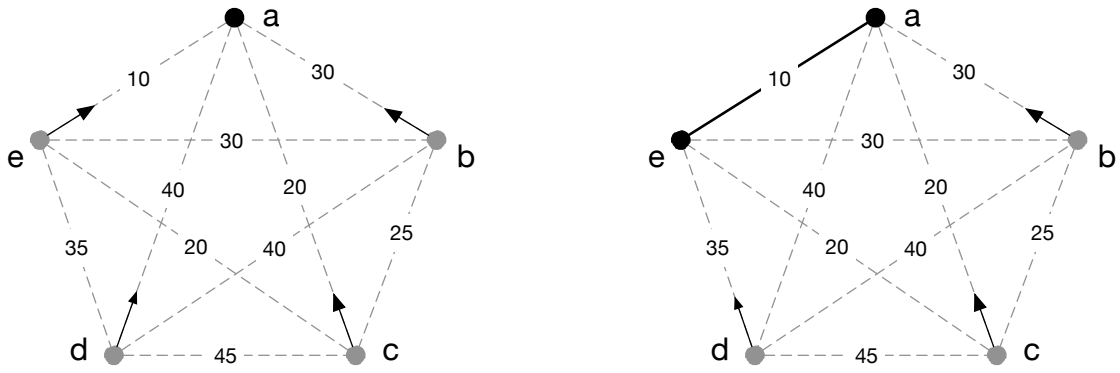
Informalmente, o algoritmo pode ser descrito pelos seguintes passos:

1. Inicie a árvore a partir de um vértice arbitrário;
2. Faça a árvore crescer com uma aresta. Dentre as arestas que conectam a árvore a vértices que ainda não fazem parte da árvore, escolha a aresta de menor custo;
3. Repita o passo anterior até que todos os vértices sejam conectados à árvore.

O pseudocódigo do algoritmo algoritmo 6.1 é descrito em Cormen et al. (2001), e apresenta uma forma interessante de implementar o processo de agregação de novos vértices à arvore. Ao invés da árvore manter controle de suas opções de crescimento "de dentro para fora", a lógica é invertida e cada vértice ainda não conectado à árvore mantém o custo mínio atual para se conectar e uma referência para o vértice da árvore no qual ele se conectaria com este custo mínimo. Estes dados são mantidos no algoritmo como predecessor (ρ) e "chave" de cada vértice. O algoritmo também usa uma fila de prioridades para escolher em cada iteração o vértice com o menor custo para se conectar à árvore naquele momento.

Por exemplo, para o grafo da Figura 6.16b, se escolhermos arbitrariamente o vértice ***a*** para iniciar a construção da árvore, temos os seguintes valores de predecessor e chave: $\rho(b) = a$ e $chave(b) = 30$; $\rho(c) = a$ e $chave(c) = 20$; $\rho(d) = a$ e $chave(d) = 35$; $\rho(e) = a$ e $chave(e) = 10$. A Figura 6.18a mostra as referências para predecessores como setas pequenas. Na próxima iteração do algoritmo, o vértice ***e*** é escolhido para se conectar à árvore por ter o menor custo de "conexão". Neste momento, os demais vértices que ainda não fazem parte da árvore verificam se existe opção mais barata para se conectar à árvore considerando o vértice recém adicionado (vértice ***e***). Note que na Figura 6.18b, o vértice d tem seus atributos atualizados para $\rho(d) = e$ e $chave(d) = 35$.

Figura 6.18 – Desenvolvimento do algoritmo de Prim



(a) Início em a (b) Primeira iteração

Fonte: o autor.

O pseudocódigo do algoritmo de Prim tem os seguintes elementos:

- G: um grafo valorado
- W: matriz de custos do grafo
- r: vértice raiz da árvore, escolhido arbitrariamente;
- $\rho(v)$: predecessor do vértice v na árvore. Representa também referência para o vértice v se conectar à árvore com custo mínimo;

- $chave(v)$: custo mínimo atual para o vértice v se conectar à árvore; item Q: é uma fila de prioridades (mínima) de vértices, tendo como chave o custo mínimo atual para o vértice se conectar à árvore. A operação REMOVE-MINIMO(Q) retira da fila e retorna o vértice com menor valor atual de chave
- $adj(u)$: é uma iteração com todos os vértices adjacentes ao vértice u (veja seção 2.2);
- $w_{u,v}$: custo da aresta (u, v).

Algoritmo: PRIM(G, W, r)

```
// Inicialização dos atributos dos vértices
para cada v ∈ V faça
    ρ(v) ← nil;
    chave(v) ← ∞;
// Inicialização do vértice raiz
chave(r) ← 0;
// Inicialização da fila de prioridades
Q ← V;
// Laço principal
enquanto Q ≠ ∅ faça
    u ← REMOVE_MINIMO(Q);
    para cada v ∈ Adj(u) faça
        se (v ∈ Q) e (w_{u,v} < chave(v)) então
            ρ(v) ← u;
            chave(v) ← w_{u,v};
```

Algoritmo 6.1: Algoritmo de Prim

O algoritmo de Prim encontra uma solução ótima, isto é, ele encontra uma árvore geradora de custo mínimo. Seu desempenho, todavia, depende fundamentalmente da implementação da fila de prioridades. Caso a fila seja implementada com *binary heaps* a complexidade do algoritmo de Prim é $O(E\ log\ V)$.

6.4.2 Algoritmo de Kruskal

Anteriormente vimos o método construtivo para formação de árvores geradoras. Kruskal (1956) se inspirou neste método para desenvolver um algoritmo para determinação de árvores geradoras de custo mínimo. O algoritmo de Kruskal encontra uma **floresta** geradora de custo mínimo em grafos valorados. Se o grafo for conexo, então o algoritmo encontra uma **árvore** geradora de custo mínimo.

Basicamente, trata-se de um algoritmo guloso que adiciona uma nova aresta de custo mínimo em cada passo, tomando o cuidado de não formar ciclos durante este processo. Em resumo, o algoritmo pode ser descrito pelos seguintes passos:

1. Crie uma floresta F na qual cada vértice do grafo individualmente é uma árvore;
2. Crie um conjunto S com todas as arestas do grafo;
3. Enquanto S não estiver vazio e F ainda puder crescer:
   a) remova de S uma aresta com custo mínimo;
   b) se a aresta removida de S conectar duas árvores diferentes, então adiciona a aresta à floresta F, unindo as duas árvores para formar uma árvore única.

O aspecto mais complicado do algoritmo é evitar a formação de ciclos, que pode ocorrer pela inclusão em F de uma aresta que conecta dois vértices que já fazem parte de uma mesma árvore. Uma maneira extremamente elegante, simples e eficiente para resolver esta questão é apresentada por Cormen et al. (2001), que sugerem o uso de estruturas de dados de conjuntos disjuntos de vértices para manter o controle da formação da floresta e evitar a formação de ciclos (veja seção 4.5 sobre estrutura de dados para conjuntos disjuntos de vértices).

Algoritmo: KRUSKAL(G, W)

```
// Inicialização da árvore
T ← ∅;
// Inicialização dos conjuntos
para cada v ∈ V faça
    CriaConjunto(v);
// Ordenação das arestas
Ordene as arestas de E em ordem crescente de custo;
// Laço principal
para cada (u, v) ∈ E, em ordem crescente de custo faça
    se BuscaConjunto(u) = BuscaConjunto(v) então
        Uniao(u, v);
        T ← T ∪ (u, v);
```

Algoritmo 6.2: Algoritmo de Kruskal

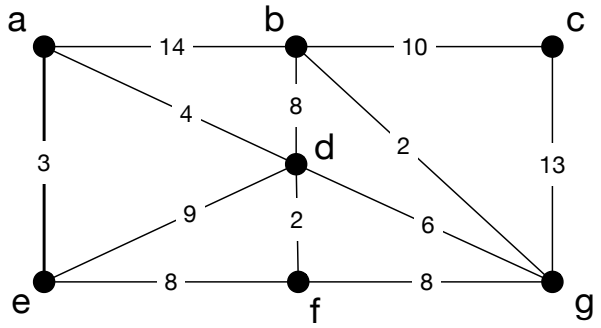
O pseudocódigo mostrado no algoritmo 6.2 implementa o método de Kruskal utilizando a estrutura de conjuntos disjuntos. Os principais elementos do pseudocódigo são os seguintes:

- G e W: representam um grafo valorado, sendo W o conjunto de informações sobre os custos das arestas (em uma matriz de custos, por exemplo);
- T: é a árvore geradora de custo mínimo (pode ser armazenada em uma estrutura de dados de grafo não dirigido);
- $CriaConjunto(v)$, $BuscaConjunto(v)$ e $Uniao(u, v)$: são operações sobre a estrutura de dados de conjuntos disjuntos, conforme descrito na seção 4.5.

O tempo de execução do algoritmo para um grafo $G = (V, E)$ depende do algoritmo utilizado para a ordenação das arestas e da implementação da estrutura de dados de conjuntos disjuntos. A inicialização da árvore leva tempo $O(1)$. Assumindo que estamos utilizando a implementação apresentada na seção 4.5, a inicialização dos conjuntos disjuntos leva tempo $O(V)$ para ser realizada. A ordenação de arestas pode ser feita em tempo $O(E\ log\ E)$. O laço principal executa $O(E)$ operações $BuscaConjunto$ e $Uniao$, resultando em um tempo $O(E + \alpha(V))$, em que $\alpha(V)$ é uma função de decresce assintoticamente devido ao mecanismo de achatamento das árvores que representam os conjuntos disjuntos de vértices, fazendo com que $\alpha(V) = O(log\ V) = O(log\ E)$. Portanto podemos considerar que o algoritmo é $O(E\ log\ V)$.

Como exemplo de execução do algoritmo de Kruskal, vamos utilizá-lo para obter uma árvore geradora de custo mínimo do grafo da Figura 6.19.

Figura 6.19 – Grafo valorado para o algoritmo de Kruskal



Fonte: o autor.

Ordenando as arestas por ordem crescente de custo, podemos obter a seguinte sequência, mostrada no Quadro 3.

Quadro 3 – Sequência de arestas ordenadas

v_i	d	b	a	a	g	e	f	b	e	b	c	a
v_j	f	g	e	d	d	f	g	d	d	c	g	b
custo	2	2	3	4	6	8	8	8	9	10	13	14

Fonte: o autor.

A Figura 6.20 mostra passo a passo a formação da árvore geradora de custo mínimo juntamente com a representação da floresta por meio dos conjuntos disjuntos de vértices.

No Quadro 4, podemos acompanhar passo a passo a evolução da floresta formada pelos conjuntos disjuntos de vértices, seguindo a sequência de arestas ordenadas, mostrada no Quadro 3. Note que os dados apresentados no Quadro 4 estão relacionados com as representações gráficas da Figura 6.20. A primeira linha do quadro mostra o estado dos conjuntos logo após sua inicialização. Neste momento, temos uma floresta formada por 7 vértices individuais, conforme mostrado na Figura 6.20a. As arestas ainda não incluídas na árvore geradora de custo mínimo são mostradas em linhas cinza tracejadas.

Logo em seguida, na primeira iteração do laço principal, o algoritmo tenta incluir a aresta (d, f) na árvore geradora de custo mínimo. Como os vértices d e f não fazem parte da mesma árvore, a inclusão da aresta não vai criar um ciclo. Portanto, a aresta é incluída na árvore, e os conjuntos que contém os vértices d e f são unidos por meio da operação $uniao(d, f)$.

Quadro 4 – Formação dos conjuntos disjuntos

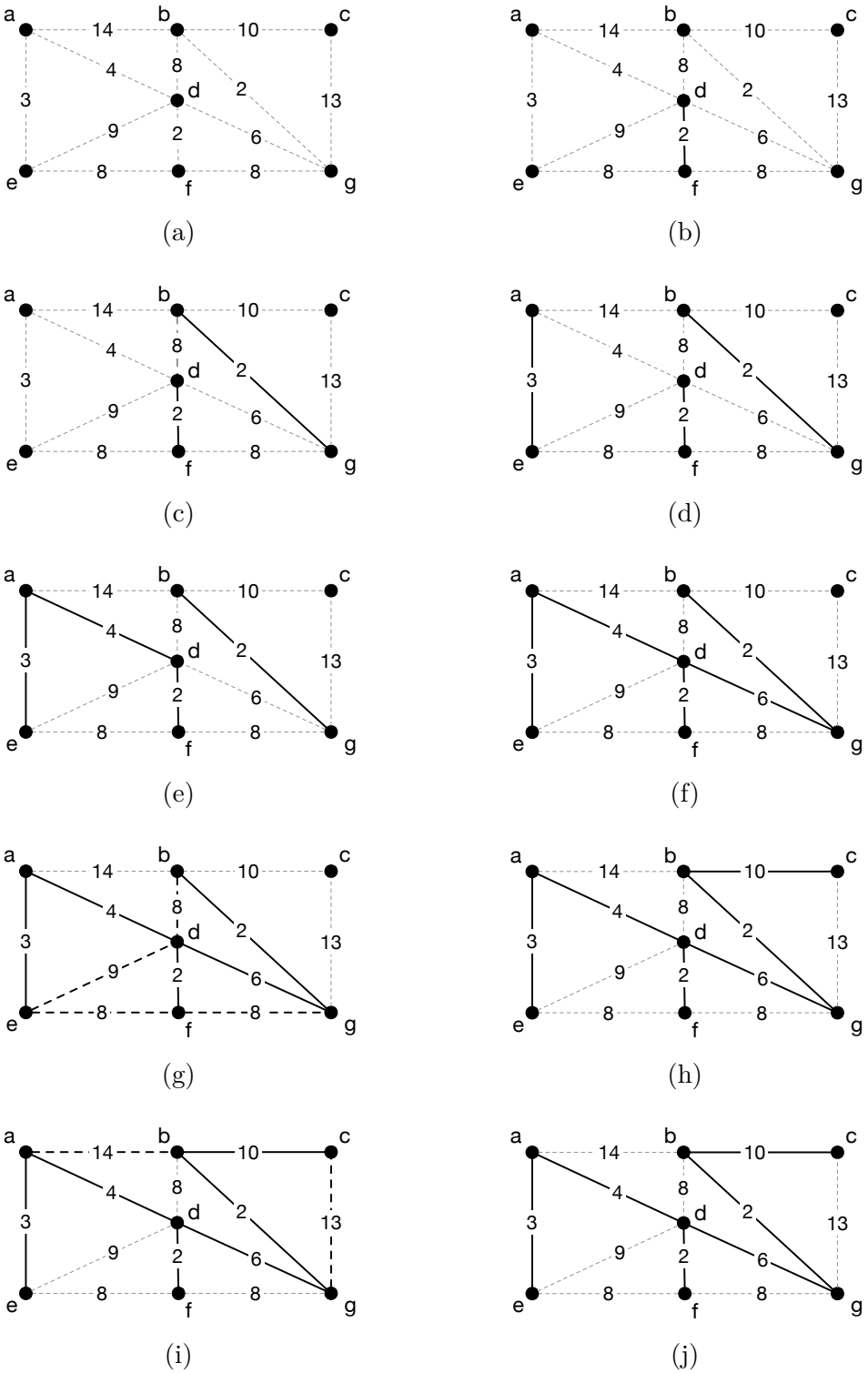
aresta	operação	Coleção de conjuntos disjuntos						
inicio	—	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$	$\{g\}$
(d, f)	$uniao(d, f)$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d, f\}$	$\{e\}$		$\{g\}$
(b, g)	$uniao(b, g)$	$\{a\}$	$\{b, g\}$	$\{c\}$	$\{d, f\}$	$\{e\}$		
(a, e)	$uniao(a, e)$		$\{b, g\}$	$\{c\}$	$\{d, f\}$	$\{e, a\}$		
(a, d)	$uniao(a, d)$		$\{b, g\}$	$\{c\}$	$\{d, f, e, a\}$			
(g, d)	$uniao(g, d)$			$\{c\}$	$\{d, f, e, a, b, g\}$			
(e, f)	—			$\{c\}$	$\{d, f, e, a, b, g\}$			
(f, g)	—			$\{c\}$	$\{d, f, e, a, b, g\}$			
(b, d)	—			$\{c\}$	$\{d, f, e, a, b, g\}$			
(e, d)	—			$\{c\}$	$\{d, f, e, a, b, g\}$			
(b, c)	$uniao(b, c)$				$\{d, f, e, a, b, g, c\}$			
(c, g)	—				$\{d, f, e, a, b, g, c\}$			
(a, b)	—				$\{d, f, e, a, b, g, c\}$			

Fonte: o autor.

O algoritmo continua a adicionar sucessivamente arestas à árvore até chegar na aresta (e, f). Neste ponto os vértices e e f já fazem parte da mesma árvore na floresta. Por conseguinte, a inclusão desta aresta formaria um ciclo e ela não é incluída na árvore. O mesmo acontece com as arestas (f, g), (b, d), e (e, d). A Figura 6.20g mostra estas quatro arestas em linhas pretas tracejadas: claramente podemos observar que a inclusão de qualquer uma delas na floresta formaria um ciclo.

Na iteração subsequente, a aresta (b, c) é adicionada à árvore geradora de custo mínimo, e depois as arestas (a, b) e (c, g) são ignoradas por formarem ciclos (mostradas em linhas pretas pontilhadas na Figura 6.20i. Finalmente, o algoritmo encerra encontrando a árvore geradora de custo mínimo mostrada na Figura 6.20j, com custo total igual a 27.

Figura 6.20 – Exemplo de execução do algoritmo de Kruskal



Fonte: o autor.

6.5 Exercícios

1. Sabendo que uma árvore é um tipo particular de grafo, escreva pelo menos cinco definições diferentes para árvores.

2. Desenhe um exemplo de uma árvore com sete vértices e...

   a) exatamente dois vértices de grau 1;
   b) exatamente quatro vértices de grau 1;
   c) exatamente seis vértices de grau 1.

3. Utilize o "lema do aperto de mão" (*handshaking lemma*) para provar que qualquer árvore com n vértices, em que $n \geq 2$, tem pelo menos dois vértices de grau 1.

4. Uma **floresta** é um grafo (não necessariamente conexo), em que cada uma de suas componentes conexas é uma árvore.

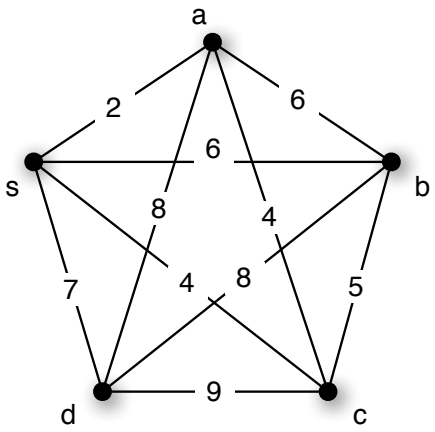
   a) Seja G uma floresta com n vértices e k componentes conexas. Quantas arestas existem em G?
   b) Construa (e desenhe) uma floresta com 12 vértices e 9 arestas.
   c) É verdade que qualquer floresta com k componentes tem pelo menos $2k$ vértices de grau 1?

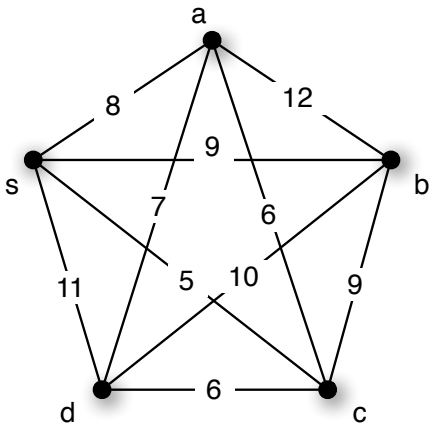
5. Para cada um dos itens abaixo, tente desenhar um grafo que atenda a descrição dada, ou então explique por que tal grafo não existe.

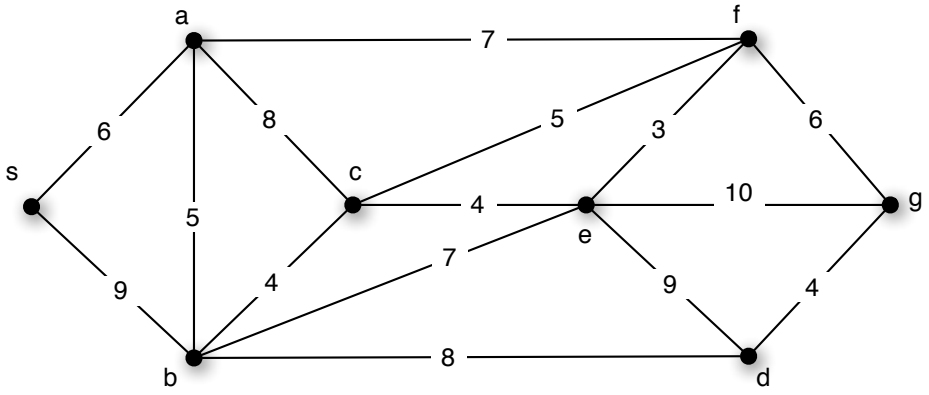
   a) Um grafo simples com 6 vértices, 2 componentes conexas e 11 arestas.
   b) Um grafo simples com 7 vértices, 3 componentes conexas e 10 arestas.
   c) Um grafo simples com 8 vértices, 2 componentes conexas, 9 arestas e exatamente 3 ciclos.
   d) Um grafo simples com 8 vértices, 2 componentes conexas, 10 arestas e exatamente 3 ciclos.
   e) Um grafo simples com 9 vértices, 2 componentes conexas, 10 arestas e exatamente 2 ciclos.
   f) Um grafo simples conexo com 9 vértices, 12 arestas e exatamente 2 ciclos.
   g) Um grafo simples conexo com 9 vértices, 12 arestas e exatamente 3 ciclos.
   h) Uma floresta com 10 vértices, 2 componentes conexas e 9 arestas.
   i) Uma floresta com 10 vértices, 3 componentes conexas e 9 arestas.
   j) Um grafo simples conexo com 11 vértices, 14 arestas e 5 ciclos disjuntos (ciclos que não compartilham arestas).

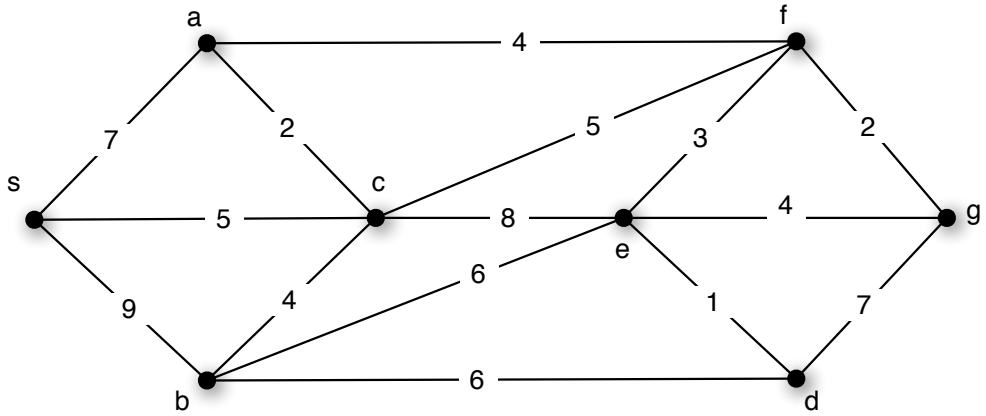
6. Prove que a seguinte afirmação é verdadeira ou falsa: "Qualquer grafo simples conexo com n vértices e n arestas deve conter exatamente 1 ciclo".

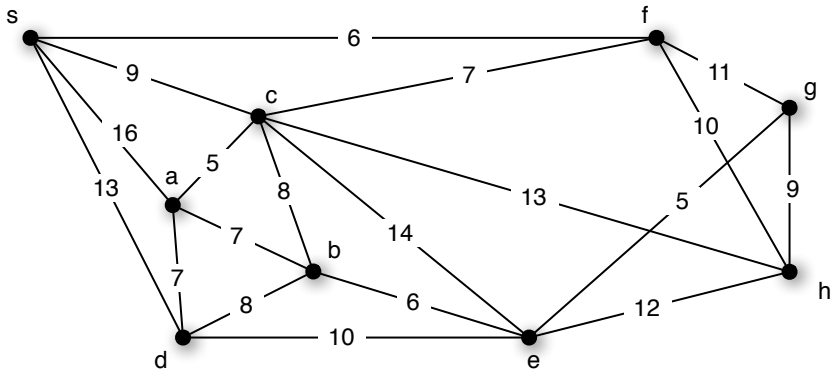
7. Dados os grafos abaixo, encontre suas respectivas árvores geradoras de custo mínimo utilizando o algoritmo de Prim. Mostre passo a passo a solução de acordo com os passos de execução do algoritmo, mostrando os valores de $chave(v)$ e $\rho(v)$ para todos os vértices. Utilize o vértice s como vértice inicial e resolva empates dando preferência para os vértices em ordem alfabética. Desenhe as árvores resultantes e calcule seus custos.

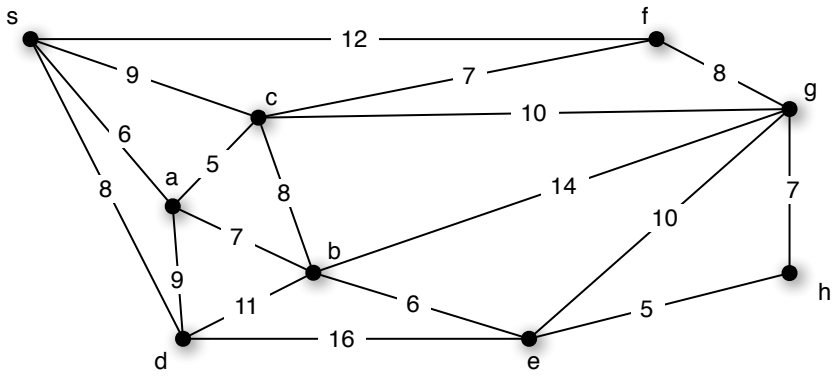












8. Encontre as respectivas árvores geradoras de custo mínimo dos grafos da questão anterior utilizando o algoritmo de Kruskal. Escreva a lista de arestas em ordem crescente de custo. Escreva a lista de conjuntos de vértices em cada passo do algoritmo, indicando a ocorrência de operações $UNION(u, v)$. Desenhe as árvores resultantes e calcule seus custos.

9. Adapte os algoritmos de Prim e Kruskal para encontrar árvores geradoras de custo **máximo** de grafos valorados.

10. Implemente o algoritmo de Prim.

11. Implemente o algoritmo de Kruskal.

12. (PosComp – 2002) Considere um grafo G satisfazendo as seguintes propriedades: (i) G é conexo e (ii) se removermos qualquer aresta de G, o grafo obtido é desconexo. Então é correto afirmar que o grafo G é:

    a) Um circuito (ciclo)
    b) Não bipartido
    c) Uma árvore
    d) Hamiltoniano
    e) Euleriano

13. (PosComp – 2008) Um grafo $G(V, E)$ é uma árvore se G é conexo e acíclico. Assinale a definição que **NÃO** pode ser usada para definir árvores.

    a) G é conexo e o número de arestas é mínimo.
    b) G é conexo e o número de vértices excede o número de arestas por uma unidade.
    c) G é acíclico e o número de vértices excede o número de arestas por uma unidade.
    d) G é acíclico e, para todo par de vértices v, w, que não são adjacentes em G, a adição da aresta (v, w) produz um grafo contendo exatamente um ciclo.
    e) G é acíclico e o número de arestas é mínimo.

14. (PosComp – 2008) Seja $G(V, E)$ um grafo tal que $|V| = n$ e $|E| = m$. Analise as seguintes sentenças:

    I. Se G é acíclico com no máximo $n - 1$ arestas, então G é uma árvore.
    II. Se G é um ciclo, então G tem n árvores geradoras distintas.
    III. Se G é conexo com no máximo $n - 1$ arestas, então G é uma árvore.
    IV. Se G é conexo e tem um ciclo, então para toda árvore geradora T de G, $E(G) - E(T) = \emptyset$.

    A análise permite concluir que (assinale a alternativa correta):

    a) apenas os itens I e III são verdadeiros.
    b) apenas os Itens II e III são verdadeiros.
    c) apenas o item I é falso.
    d) todos os itens são verdadeiros.
    e) apenas os itens II e IV são verdadeiros.

15. (PosComp – 2012) Sejam $G = (V, E)$ um grafo conexo não orientado com pesos distintos nas arestas e $e \in E$ uma aresta fixa, em que $|V| = n$ é o número de vértices e $|E| = m$ é o número de arestas de G, com $n \leq m$. Com relação à geração da árvore de custo mínimo de G, AGM_G, assinale a alternativa correta.

    a) Quando e tem o peso da aresta com o $(n-1)$-ésimo menor peso de G então e garantidamente estará numa AGM_G .
    b) Quando e tem o peso da aresta com o maior peso em G então e garantidamente não estará numa AGM_G.
    c) Quando e tem o peso maior ou igual ao da aresta com o n-ésimo menor peso em G então e pode estar numa AGM_G .
    d) Quando e tem o peso distinto do peso de qualquer outra aresta em G então pode existir mais de uma AGM_G.
    e) Quando e está num ciclo em G e tem o peso da aresta de maior peso neste ciclo então e garantidamente não estará numa AGM_G.

16. Deseja-se supervisionar as redes de comunicação de dados de um conjunto de empresas. Cada empresa tem sua própria rede, que é independente das redes das outras empresas e é constituída de ramos de fibra óptica. Cada ramo conecta duas filiais distintas (ponto-a-ponto) da empresa. Há, no máximo, um ramo de fibra interligando diretamente um mesmo par de filiais. A comunicação entre duas filiais pode ser feita diretamente por um ramo de fibra que as interliga, se este existir, ou, indiretamente, por meio de uma sequência de ramos e filiais. A rede de cada empresa permite a comunicação entre todas as suas filiais. A tabela 5 apresenta algumas informações acerca das redes dessas empresas.

Quadro 5 – Empresas

empresa	n.º de filiais	n.º de ramos de fibra entre filiais
E1	9	18
E2	10	45
E3	14	13
E4	8	24

Com relação à situação apresentada acima, é correto deduzir que (responda V ou F):

a) no caso da empresa E1, a falha de um ramo da rede certamente fará que, ao menos, uma filial não possa mais comunicar-se diretamente com todas as outras filiais da empresa;

b) na rede da empresa E2, a introdução de um novo ramo de rede certamente violará a informação de que há somente um par de fibras entre duas filiais;

c) no caso da empresa E3, a falha de um único ramo de rede certamente fará que, ao menos, uma filial não possa mais comunicar-se direta ou indiretamente com todas as outras filiais da empresa;

d) na rede da empresa E4, todas as filiais da empresa comunicam-se entre si diretamente.