Como ilustrado na Figura 7.19, o algoritmo considera as arestas do grafo ordenadas pelo peso. Considere C_1 e C_2 duas árvores conectadas por (u, v). Uma vez que (u, v) tem de ser uma aresta leve conectando C_1 com alguma outra árvore, então (u, v) é uma aresta segura para C_1. O algoritmo de Kruskal é um algoritmo guloso porque, a cada passo, ele adiciona à floresta uma aresta de menor peso. Em outras palavras, o algoritmo de Kruskal obtém uma árvore geradora mínima adicionando uma aresta de cada vez à floresta e, a cada passo, usa a aresta de menor peso que não forma um ciclo. O algoritmo inicia com uma floresta de $|V|$ árvores de um vértice: em $|V|$ passos, une duas árvores até que exista apenas uma árvore na floresta.

A implementação do algoritmo de Kruskal não é apresentada aqui. Mais detalhes podem ser obtidos no Exercício 7.20.

7.9 Caminhos mais Curtos

Esta seção trata do problema de encontrar o caminho mais curto entre dois vértices de um grafo direcionado ponderado $G = (V, A)$. Uma aplicação para este problema ocorre quando um motorista deseja obter o caminho mais curto entre Diamantina e Ouro Preto, duas cidades históricas de Minas Gerais. Dado um mapa do Estado de Minas Gerais contendo as distâncias entre cada par de interseções adjacentes, como obter o caminho mais curto entre as duas cidades? Nesse caso nós podemos modelar o mapa rodoviário como um grafo em que vértices representam interseções, arestas representam segmentos de estrada entre interseções, e o peso de cada aresta, a distância entre interseções.

O problema descrito no parágrafo anterior é equivalente a obter os caminhos mais curtos a partir de uma única origem. Dado um grafo direcionado ponderado $G = (V, A)$, o **peso** de um caminho $c = (v_0, v_1, \ldots, v_k)$ é a soma de todos os pesos das arestas do caminho:

$$p(c) = \sum_{i=1}^{k} p(v_{i-1}, v_i).$$

O caminho mais curto é definido por:

$$\delta(u, v) = \begin{cases} min\left\{p(c) : u \overset{c}{\leadsto} v\right\}, & \text{se existir um caminho de } u \text{ a } v, \\ \infty, & \text{caso contrário.} \end{cases}$$

Um **caminho mais curto** do vértice u ao vértice v é então definido como qualquer caminho c com peso $p(c) = \delta(u, v)$. O peso das arestas pode ser interpretado como outras métricas diferentes de distância, tais como tempo, custo, penalidade, perdas, ou qualquer quantidade acumulada através do caminho que se deseja minimizar.

O procedimento VisitaBfs do Programa 7.11 para realizar a busca em largura em um grafo $G = (V, A)$ obtém a distância do vértice origem $u \in V$ para cada vértice alcançável $v \in V$. De fato, a busca em largura obtém o **caminho mais curto** de u até v, onde os pesos das arestas são todos iguais (vide Seção 7.5).

Nesta seção, vamos tratar do problema de obter os **caminhos mais curtos a partir de uma origem**: dado um grafo ponderado $G = (V, A)$, desejamos obter o caminho mais curto a partir de um dado vértice origem $s \in V$ até cada $v \in V$. Muitos outros problemas podem ser resolvidos pelo algoritmo para o problema origem única, como as seguintes variações:

- **Caminhos mais curtos com destino único**: Encontrar um caminho mais curto para um vértice destino t a partir de cada $v \in V$. Este problema pode ser reduzido ao problema origem única invertendo a direção de cada aresta do grafo, o que pode ser realizado pelo Programa 7.14 para obter o grafo transposto G^T de um grafo G.
- **Caminhos mais curtos entre um par de vértices**: O algoritmo para resolver o problema origem única resolve também este problema, sendo a melhor opção conhecida para ele.
- **Caminhos mais curtos entre todos os pares de vértices**: Este problema pode ser resolvido pela aplicação do algoritmo origem única $|V|$ vezes, uma vez para cada vértice origem. Existem outras opções de algoritmos para o caso todos-os-pares que podem ser mais eficientes quando o grafo for denso, mas que não serão tratados aqui, pois fogem ao escopo deste livro (vide Aho, Hopcroft e Ullman, 1983; Cormen, Leiserson, Rivest e Stein, 2001).

Um caminho mais curto em um grafo $G = (V, A)$ não pode conter ciclo nenhum, uma vez que a remoção do ciclo do caminho produz um caminho com os mesmos vértices origem e destino e um caminho de menor peso. Assim, podemos assumir que caminhos mais curtos não possuem ciclos. Uma vez que qualquer caminho acíclico em G contém no máximo $|V|$ vértices, então o caminho também contém no máximo $|V| - 1$ arestas.

A representação de caminhos mais curtos em um grafo $G = (V, A)$ pode ser realizada pela variável Antecessor. Para cada vértice $v \in V$, o $Antecessor[v]$ é um outro vértice $u \in V$ ou nil (-1). O algoritmo para obter caminhos mais curtos atribui a Antecessor os rótulos de vértices de uma cadeia de antecessores com origem em um vértice v e que anda para trás ao longo de um caminho mais curto até o vértice origem s. Assim, dado um vértice v no qual $Antecessor[v] \neq nil$, o procedimento ImprimeCaminho do Programa 7.12 pode ser usado para imprimir o caminho mais curto de s até v.

Ao contrário do que ocorre durante a execução do algoritmo de busca em largura, durante a execução do algoritmo para obter caminhos mais curtos, os valores em $Antecessor[v]$ não necessariamente indicam caminhos mais curtos. Entretanto,