18. Considere uma ampliação de uma rede de distribuição de água (Almeida, 2010). Foram definidas as distâncias entre os pontos de armazenamento e quais pontos de armazenamento serão conectados fisicamente. De modo a suportar situações de contingência, deverá haver rotas entre todos os pontos de armazenamento e as rotas serão compostas de dois tubos, de modo que o fluxo de água ocorra em ambas as direções. Por motivos de eficiência de distribuição, deseja-se minimizar a distância percorrida pela água distribuída.

a) Modele o problema utilizando grafos.

b) Descreva um algoritmo que determine quais pontos devem ser conectados fisicamente.

c) Qual a complexidade do algoritmo?

19. Marcelo e Alice estão em duas cidades diferentes. Marcelo deseja enviar certa quantia de dinheiro para Alice utilizando uma cadeia de agentes certificados (Almeida, 2010). Cada agente tem uma lista de pares de cidades onde atua, cobrando um valor fixo para realizar o transporte, que depende do agente e das cidades de origem e destino. Logo, a quantia de dinheiro diminui na medida em que é transportada pela cadeia de agentes. Marcelo precisa decidir quais agentes e o respectivo trajeto que ele deverá contratar visando maximizar a quantia de dinheiro que efetivamente chegará até Alice. A fim de manter o controle, Marcelo decide que qualquer solução válida deverá garantir que o dinheiro a ser transferido esteja nas mãos de apenas um agente em qualquer instante, embora múltiplos agentes possam ser utilizados ao longo do trajeto.

a) Modele o problema utilizando grafos.

b) Apresente um algoritmo que resolva o problema. Ele é ótimo?

20. Apresente uma implementação eficiente para o **algoritmo de Kruskal** (vide Seção 7.8.3). O algoritmo de Kruskal obtém uma árvore geradora mínima de um grafo $G = (V, A)$ adicionando à floresta, a cada passo, uma 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.

Para obter uma implementação eficiente do algoritmo de Kruskal é necessário:

a) utilizar uma fila de prioridades para obter as arestas em ordem crescente de pesos, mesmo porque pode não ser necessário utilizar todas as arestas do grafo;

b) testar se uma dada aresta adicionada ao conjunto solução S forma um ciclo.

No segundo passo, a maneira mais eficiente de verificar se uma dada aresta forma um ciclo é mediante a utilização de estruturas dinâmicas para tratar **conjuntos disjuntos**. Nesse caso, os elementos de um conjunto são representados por um objeto. Tendo em vista que x denota um objeto, considere as seguintes operações:

a) CriaConjunto (x): cria um novo conjunto cujo único membro é x, o qual passa a ser seu representante. Para que os conjuntos sejam disjuntos, é necessário que x não pertença a outro conjunto.

b) União (x, y): une os conjuntos dinâmicos que contêm x e y, digamos C_x e C_y, em um novo conjunto que é a união desses dois conjuntos. O representante do novo conjunto pode ser x ou y. Uma vez que os conjuntos na coleção devem ser disjuntos, os conjuntos C_x e C_y são destruídos.

c) EncontreConjunto (x): retorna um apontador para o representante do conjunto (único) contendo x.

Um primeiro refinamento do algoritmo de Kruskal pode ser visto no Programa 7.28.

Programa 7.28 *Primeiro refinamento do algoritmo de Kruskal*

```
   procedure Kruskal;
1  S := ∅;
2  for v := 0 to Grafo.NumVertices−1 do CriaConjunto (v);
3  Ordena as arestas de A pelo peso;
4  for cada (u, v) de A tomadas em ordem ascendente de peso do
5    if EncontreConjunto (u) <> EncontreConjunto (v)
     then begin
6          S := S + {(u, v)};
7          Uniao (u, v);
           end;
```

A implementação das operações Uniao e EncontraConjunto deve ser realizada de forma eficiente. Esse problema é conhecido na literatura como **União-EncontraConjunto** (do inglês ***Union-find***). O tempo de execução do Programa 7.28 é como se segue. A inicialização do conjunto S tem custo $O(1)$ e ordenar as arestas na linha 3 custa $O(|A| \log |A|)$. A linha 2 realiza $|V|$ operações CriaConjunto, e o anel envolvendo as linhas 4-7 realiza $O(|A|)$ operações EncontreConjunto e Uniao, a um custo $O((|V| + |A|)\alpha(|V|))$ onde $\alpha(|V|)$ é uma função que cresce tão lentamente que $\alpha(|V|) < 4$ (Cormen, Leiserson, Rivest e Stein, 2001, p. 453.) O limite inferior para construir uma estrutura dinâmica envolvendo m operações EncontreConjunto e Uniao e n operações CriaConjunto é $m\alpha(n)$. Como G é conectado, temos que $|A| \geq |V| - 1$, e assim as operações sobre conjuntos disjuntos custam $O(|A|\alpha(|V|)$. Como $\alpha(|V|) = O(\log |A|) = O(\log |V|)$, o tempo total do algoritmo de Kruskal é $O(|A| \log |A|)$. Como $|A| < |V|^2$, então $\log |A| = O(\log |V|)$, e o custo do algoritmo de Kruskal é também $O(|A| \log |V|)$.

21. Apresente uma implementação das operações sobre o **tipo abstrato de dados hipergrafo** da Seção 7.10.2 por meio de **listas de incidência** usando apontadores.