327

Análise O desempenho do algoritmo de Dijkstra depende da forma como a fila de prioridades é implementada. Se a fila de prioridades é implementada como um heap (veja Seção 4.1.5), podemos usar, a um custo O(|V|), o procedimento Constroi do Programa 4.10 para a inicialização de A no Programa 7.22. O corpo do anel while é executado |V| vezes e, desde que o procedimento Refaz tem custo $O(\log |V|)$, o tempo total para executar a operação retira o item com menor peso é $O(|V|\log |V|)$. O while mais interno para percorrer a lista de adjacentes é executado O(|A|) vezes ao todo, uma vez que a soma dos comprimentos de todas as listas de adjacência é 2|A|. A operação DiminuiChave é executada sobre o heap A na posição Pos[v], a um custo $O(\log |V|)$. Logo, o tempo total para executar algoritmo de Dijkstra é $O(|V|\log |V| + |A|\log |V|) = O(|A|\log |V|)$.

Por que o Algoritmo de Dijkstra Funciona

Pelo fato de o algoritmo de Dijkstra sempre escolher o vértice mais leve (ou o mais perto) em V-S para adicionar ao conjunto solução S, o algoritmo usa uma estratégia gulosa. Apesar de estratégias gulosas nem sempre levarem a resultados ótimos, o algorimo de Dijkstra sempre obtém os caminhos mais curtos. Isso é verdade porque cada vez que um vértice é adicionado ao conjunto S, temos que $p[u] = \delta(Raiz, u)$.

7.10 O Tipo Abstrato de Dados Hipergrafo

Um hipergrafo ou r-grafo é um grafo não direcionado $G_r = (V, A)$ no qual cada aresta $a \in A$ conecta r vértices, sendo r a ordem do hipergrafo. Os grafos estudados até agora são 2-grafos (ou hipergrafos de ordem 2). Hipergrafos são utilizados na Seção 5.5.4 sobre hashing perfeito.

A Figura 7.21 apresenta um 3-grafo contendo os vértices $\{0,1,2,3,4,5\}$, as arestas $\{(1,2,4),(1,3,5),(0,2,5)\}$ e os pesos 7, 8 e 9, respectivamente.

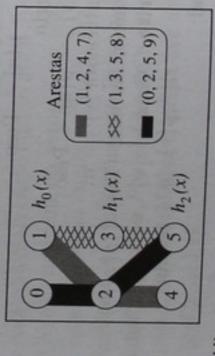


Figura 7.21 Hipergrafo de grau r=3 contendo 6 vértices e 3 arestas.

As operações de um tipo abstrato de dados hipergrafo são praticamente as mesmas definidas para o tipo abstrato de dados grafo, a saber:

- Criar um hipergrafo vazio. A operação retorna um hipergrafo contendo |V | vértices e nenhuma aresta.
- 2. Inserir uma aresta no hipergrafo. Recebe a aresta $(V_1, V_2, ..., V_r)$ e seu peso para serem inseridos no hipergrafo.
- 3. Verificar se existe determinada aresta no hipergrafo. A operação retorna true se a aresta (V_1, V_2, \ldots, V_r) estiver presente, senão retorna false.
- 4. Obter a lista de arestas incidentes em determinado vértice. Essa operação aparece na maioria dos algoritmos que utilizam um hipergrafo e, pela sua importância, será tratada separadamente logo a seguir.
- 5. Retirar uma aresta do hipergrafo. Retira a aresta $(V_1, V_2, ..., V_r)$ do hipergrafo e a retorna.
- 6. Imprimir um hipergrafo.
- 7. Obter a aresta de menor peso de um hipergrafo. A operação retira a aresta de menor peso dentre as arestas do hipergrafo e a retorna.

Uma operação que aparece com frequência é a de obter a lista de arestas incidentes em determinado vértice. Para implementar esse operador de forma independente da representação escolhida para a aplicação em pauta, precisamos de três operações sobre hipergrafos, a saber:

- Verificar se a lista de arestas incidentes em um vértice v está vazia. A operação retorna true se a lista estiver vazia, senão retorna false.
- 2. Obter a primeira aresta incidente a um vértice v, caso exista.
- 3. Obter a próxima aresta incidente a um vértice v, caso exista.

A forma mais adequada para representar um hipergrafo é por meio de estruturas de dados em que para cada vértice v do grafo é mantida uma lista das arestas que incidem sobre o vértice v, o que implica a representação explícita de cada aresta do hipergrafo. Essa é uma estrutura orientada a arestas e não a vértices como as representações apresentadas nas Seções 7.2.1, 7.2.2 e 7.2.3.

Existem duas representações usuais para hipergrafos: as matrizes de incidência e as listas de incidência. A Seção 7.10.1 apresenta a implementação de matrizes de incidência usando arranjos. A Seção 7.10.2 apresenta a implementação de listas de incidência usando arranjos.

7.10.1 Implementação por meio de Matrizes de Incidência