Ciencias de la Computación I

Optimización y Emparejamiento

Eduardo Contrera Schneider

Universidad de la Frontera

21 de noviembre de 2016

El camino más Corto

Consideremos un grafo dirigido conexo sin lazos $G = (V, E)$. A cada arista $e = (a, b)$ de este grafo le asignamos un número real positivo llamado el **peso** de e, que denotamos con $p(e)$ o $p(a, b)$, Si $x, y \in V$ pero $(x, y) \notin E$, definimos $p(x, y) = \infty$. Un grafo con asignaciones de peso en sus aristas se dice que es un **grafo ponderado**.
Para cualesquiera dos vértices $a, b \in V$ escribimos $d(a, b)$ como la distancia más corta de a a b. Si no existe tal camino de a a b, definimos $d(a, b) = \infty$. Para determinar la distancia más corta de un vértice fijo v_0 a los demás, así como un camino simple dirigido más corta para hacia cada uno de estos vértices, aplicamos el algoritmo de Dijkstra.

Algoritmo de Dijkstra

Algoritmo del camino más corto

1. Hacemos el contador $i = 0$ y $S_0 = \{v_0\}$. Etiquetamos v_0 con $(0, -)$ y cada $v \neq v_0$ con $(\infty, -)$.
   Si $n = 1$, entonces $V = \{v_0\}$ y el problema está resuelto.
   Si $n > 1$, continuamos con el paso 2.
2. Para cada $v \in \bar{S}_i$, reemplazmos (cuando sea posible), la etiqueta de v por la nueva etiqueta final $(L(v), y)$, donde

$$L(v) = \min_{u \in S_i} \{L(v), L(u) + p(u, v)\}$$

y y es un vértice en S_i que produce el $L(v)$ mínimo.

Algoritmo del camino más corto

3. Si cada vértice de $\bar{S}_i$, para algún $0 \leq i \leq n-2$, tiene la etiqueta $(\infty, -)$, entonces el grafo contiene la información que estamos buscando. Si no, existe al menos un vértice $v \in \bar{S}_i$ que no está etiquetado como $(\infty, -)$, y hacemos lo siguiente:
    1. Seleccionamos un vértice v_{i+1}, tal que $L(v_{i+1})$ sea mínimo (para todo v de este tipo). Puede haber varios de estos vértices, en cuyo caso podemos elegir culquiera de los posibles candidatos. El vértice v_{i+1} es un elemento de $\bar{S}_i$ que es el más cercano a v_0.
    2. Asignamos $S_i \cup \{v_{i+1}\}$ a S_{i+1}
    3. Incrementamos el contador i en 1. Si $i = n-1$, el grafo etiquetado contiene la información deseada. Si $i < n-1$, regresamos al paso 2.

Árbol Recubridor Minimal

Sea $G = (V, E)$ un grafo no dirigido conexo sin lazos tal que $|V| = n$ y cada arista e tiene asignado un número real positivo $p(e)$.

Algoritmo de Kruskal

1. Hacemos el contador $i = 1$ y seleccionamos un arista e_1 en G tal que $p(e_1)$ sea lo más pequeño posible.
2. Para $1 \leq i \leq n-2$, si hemos seleccionado las aristas $e_1, e_2, ..., e_i$, entonces seleccionamos la arista e_{i+1} de las aristas restantes en G de modo que
   - $p(e_{i+1})$ sea lo más pequeño posible.
   - El subgrafo de G determinado por las aristas $e_1, e_2, ..., e_i, e_{i+1}$ (y los vértices incidentes) no contenga ciclos.
3. Reemplazamos i con $i + 1$. Si $i = n - 1$, el subgrafo de G determinado por las aristas $e_1, e_2, ..., e_{n-1}$ es conexo con vértices y $n - 1$ aristas, y es un árbol recubridor óptimo para G. Si $i < n - 1$, regresamos al paso 2.

Algoritmo de Prim

Algoritmo de Prim

1. Hacemos el contador $i = 1$ y colocamos un vértice arbitrario $v_1 \in V$ en el conjunto P. Definimos $N = V - \{v_1\}$ y $T = \emptyset$.
2. Para $1 \leq i \leq n-1$, donde $|V| = n$, sean $P = \{v_1, v_2, ..., v_i\}$, $T = \{e_1, e_2, ..., e_{i-1}\}$ y $N = V \setminus P$. Añadimos a T la arista más corta (la arista de peso minimal) de G que conecta un vértice $x \in P$ con un vértice $y (= v_{i+1})$ en N. Colocamos y en P y lo eliminamos de N.
3. Incrementamos el contador en 1.

   Si $i = n$, el subgrafo de G determinado por las aristas $e_1, e_2, ..., e_{n-1}$ es conexo, con n vértices y $n-1$ aristas y es un árbol óptimo para G.

   Si $i < n$, regresamos al paso 2.

Complejidad Computacional

Analizar los tiempos de corrida de los algoritmos de Prim y de Kruskal es interesante, debido a que dependiendo de las condiciones del problema, un algoritmo va a ser más eficiente que el otro.

Tiempo de Corrida

1. El algoritmo de Kruskal corre en tiempo $O(|E|log(|E|))$.
2. El algoritmo de Prim corre en tiempo $O(|V|^2)$.

No es difícil establecer qué algoritmo usar en cada caso.