Diseño y Análisis de Algoritmos. Problema 2: El Zoológico

Jesús Santos Capote y Kenny Villalobos Morales

Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba

1. Definición del Problema

Se tiene un grafo no dirigido de n vértices, unidos por arcos ponderados. Se desea computar para cada par de vértices s y t cuántos arcos pertenecen a algún camino de costo mínimo entre s y t. La representación computacional del grafo de entrada será una matriz de costos.

2. Primera Aproximación

2.1. Idea del Algoritmo

Se le realiza al algoritmo de Dijkstra la siguiente modificación: Sea $d[u]$ el costo de llegar desde origen hasta el vértice u. En cada nodo, en vez de guardar un padre, se guarda una lista de padres, de forma tal que cuando se haga $relax(u, v)$, si el costo computado hasta el momento de llegar a v desde el origen $d[v]$, es igual al costo de ir del origen a u $d[u]$ más el costo del arco (u, v) entonces añadimos a la lista de padres de v el vértice u. Mediante estas listas de padres podemos reconstruir todos los caminos de costo mínimo que llegan a u. Se tendrá una matriz *solution* donde en la posición (i, j) almacenará el conteo de la cantidad de caminos de costo mínimo entre el nodo i y el nodo j. Por cada vértice del grafo se efectúa el algoritmo de Dijkstra con la modificación anteriormente explicada. Cada vez que se termine una ejecución de Dijkstra para el origen de turno, llamemosle u, para cada par (u, v) se recorren todos los caminos de costo mínimo encontrados y se cuentan todas los arcos que participan en dichos caminos, los arcos se contaran solo una vez, pues a medida que se recorren los caminos se irá marcando en una matriz booleana los arcos ya contados. Por cada arco no marcado que se visite se aumenta en 1 el valor de $solution[u, v]$. Luego de esto se tendrán computados la cantidad de aristas que participan en caminos de costo mínimo para los pares donde origen de los caminos sea u, pares (u, v), donde v es un vértice del grafo distinto de u. Repitiendo este proceso para cada uno de los vértices del grafo en *solution* estará almacenada la respuesta deseada.

2.2. Correctitud

La modificación realizada a la operación relax que efectúa Dijkstra no afecta la correctitud del algoritmo, pues no se modifica su comportamiento, solo

agregamos información a los nodos. Luego podemos afirmar que se calculan correctamente los caminos de costo mínimo desde el orígen de turno hacia el resto de los nodos. Luego, el algoritmo calcula todos las caminos de costo mínimo entre cada par de nodos y cuenta los arcos que intervienen en los caminos una sola vez.

2.3. Complejidad Temporal

La modificación hecha al algoritmo de Dijkstra no afecta su complejidad temporal, solo se agregan operaciones de complejidad constante a la operación relax. Dijkstra tiene complejidad $O(|E|+|V|log|V|)$. Luego reconstruir todos los caminos de costo mínimo entre un par de vértices es $O(V)$. En cada iteración del algoritmo se ejecuta una vez el algoritmo de Dijkstra y luego por cada uno de los $|V|-1$ pares a analizar en la iteración se reconstruyen sus caminos de costo mínimo, luego pueden existir como máximo $|E|$ camino de costo mínimo diferentes de un origen a un destino, luego el costo de contar todos los caminos de costo mínimo entre los $|V|-1$ pares es $O((|V|-1)|E||V|) = O(|V|^2|E|) = O(|V|^4)$. Luego el coste de una iteración es $O(|E|+|V|log|V|+|V|^4)$. El algoritmo realiza $|V|$ iteraciones. Por tanto la complejidad temporal del algoritmo es $O(|V|*(|E|+|V|log|V|+|V|^4))$, que es $O(|V|^5)$