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. La modalidad de Dijkstra utilizada tiene complejidad $O(|V|^2)$. 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 $O(|V|^5)$

3. Segunda Solución

3.1. Idea del algoritmo

Realizamos el algoritmo de Floyd-Warshall a partir de la matriz de costos inicial (cost) para obtener la matriz de las distancias de costo mínimo entre todo par de vértices (i,j), llamemos a esta matriz fw. Luego, para cada par de vértices (v_i, v_j), con $i \neq j$, hallamos todos los vértices v_k, tal que $fw[i,j] = fw[i,k]+fw[k,j]$, estos vértices pertencen a algún camino de costo mínimo de i a j, al conjunto de estos vértices para cada par (i,j) llamémoslo $CCM_(i,j)$. Luego, para cada par de vértices (v_i, v_j), con $i \neq j$, hallamos el número de vértices v_p que cumplen que: $fw[i,p] + dist[p,j] = fw[i,j]$, esta es la cantidad de aristas que llegan a j, pertenecientes a algún camino de costo mínimo proveniente de i, llamemos a este número $e_(i,j)$. Luego hacemos un recorrido por cada par de vértices (v_i, v_j) la cantidad de aristas que pertenecen a algún camino de costo mínimo de i a j es igual a la suma de las aristas que llegan a cada vértice k perteneciente a $CCM_(i,j)$ en caminos de costo mínimo provenientes de i:

$S_(v_i, v_j) = \sum e_(i,k), \forall k \in CCM_(i,j)$

3.2. Correctitud

Primero demostremos que si una arista pertenece a algún camino de costo mínimo entre i y j, el algoritmo la cuenta:

Si una arista (x,y) pertence a algún camino de costo mínimo entre i y j, entonces: y es un vértice que pertence a algún camino de costo mínimo entre i y

j y se cumple que: el costo mínimo de un camino de i a y sumado con el costo de un camino de costo mínimo de y a j es igual al costo de algún camino de costo mínimo de i a j.

Demostremos el enunciado anterior:

El algoritmo de Floyd-Warshall calcula correctamente el costo del camino de costo mínimo entre todo par de vértices (i, j). Sea fw la matriz que devuelve el algoritmo de Floyd-Warshall. Todos los caminos de costo mínimo tienen el mismo costo por definición de camino de costo mínimo. Sea P algún camino de costo mínimo entre i y j en el cual participe el vértice y. Los caminos de costo mínimo poseen subestructura óptima, es decir, para cualquier par de vértices $(v, w) \in P$, el subcamino de P que va de v a w es un camino de costo mínimo entre v y w. Por tanto el subcamino de P de i a y es un camino de costo mínimo, luego el costo de dicho subcamino está almacenado en $fw[i, y]$. Análogamente se cumple para el subcamino de P que va de y a j y su valor está almacenado en $fw[y, j]$. Por tanto, $fw[i, y]$ es el costo del camino de costo mínimo que va de i a y, $f[y, j]$ es el costo del camino de costo mínimo que va de y a j. Luego como $f[i, y]$ es mínimo y $f[y, j]$ es mínimo, su suma es mínima, es decir, $fw[i, y] + fw[y, j]$ es mínima. El costo del camino de costo mínimo entre i y j es $fw[i, j]$ y coincide con el costo de P. Luego en P se empieza en i, se pasa por y y luego se va a j, por tanto el costo de P es $fw[i, y] + fw[y, j]$ y como P es de costo mínimo entonces $fw[i, j] = fw[i, y] + fw[y, j]$, con $y \in P$. Luego queda demostrado que si y es un vértice que pertence a algún camino de costo mínimo entre i y j se cumple que: el costo mínimo de un camino de i a y sumado con el costo de un camino de costo mínimo de y a j es igual al costo de algún camino de costo mínimo de i a j.

Luego, si (x, y) pertenece a algún camino de costo mínimo entre i y j, x, y pertenece a algún camino de costo mínimo entre i y y, pues todo subcamino de un camino de costo mínimo es de costo mínimo. Finalmente si (x, y) pertenece a algún camino de costo mínimo entre i y y, entonces se cumple que la suma del costo de la arista (x, y) con el costo de un camino de costo mínimo de i a x debe ser igual al costo de un camino de costo mínimo de i a y.

Por lo cual: $y \in CCM_(i, j) \wedge (x, y) \in e_(i, y)$, y por lo tanto el algoritmo cuenta a (x, y)

Ahora demostremos que si el algoritmo cuenta una arista, esta pertence a algún camino de costo mínimo ente i y j:

3.3. Complejidad Temporal

El algoritmo genera la matriz de los costos caminos de costo mínimo para todos los pares de vértices (v_i, v_j) utilizando el algoritmo de Floyd-Warshall, este tiene una complejidad temporal $O(|V|^3)$, luego por cada elemento de la matriz, o sea, par de vértices, (excepto los pares (i, j), con $i = j$) el algoritmo recorre todos los vértices del grafo (excepto i), para reconocer cuales vértices pertenecen a caminos de costo mínimo entre i y j y cuántas aristas de la forma (k, j) pertenecen a caminos de costo mínimo de i a j, esta comprobación sucede en tiempo constante c, por lo cual este procedimiento tiene una complejidad

temporal $O(|V|^3)$. Luego, para cada par de vértices (i, j) se recorren los vértices previamente calculados que pertenecen a caminos de costo mínimo de i a j (a lo sumo $|V|-1$) y se suma el número de aristas que llegan a cada uno de esos vértices pertenecientes a caminos de costo mínimo provenientes de i, este procedimiento tiene una complejidad temporal $O(|V|^3)$. Por tanto la complejidad temporal del algoritmo es $O(3 * |V|^3)$, que es $O(|V|^3)$.