Programación II

TPO. Caminos más cortos en grafos: Dijkstra

1. El problema de los caminos más cortos

Este problema puede plantearse de dos formas diferentes:

1. Dado un grafo dirigido con pesos G (o sea, un objeto de tipo GrafoTDA) y un nodo v tal que $v \in G.Vertices()$, encontrar el mínimo costo para ir desde v hasta cada uno de los restantes nodos de G.

2. Dado un grafo dirigido con pesos G (o sea, un objeto de tipo GrafoTDA) y un nodo v tal que $v \in G.Vertices()$, encontrar el camino de mínimo costo para ir desde v hasta cada uno de los restantes nodos de G.

Observe que en el primer caso nos limitamos a encontrar el mínimo costo sin especificar por dónde se debe ir. Un ejemplo servirá para arrojar luz sobre el tema. La figura 1 muestra una solución de la primera versión del problema aplicada al grafo de la izquierda con nodo de referencia **1**.

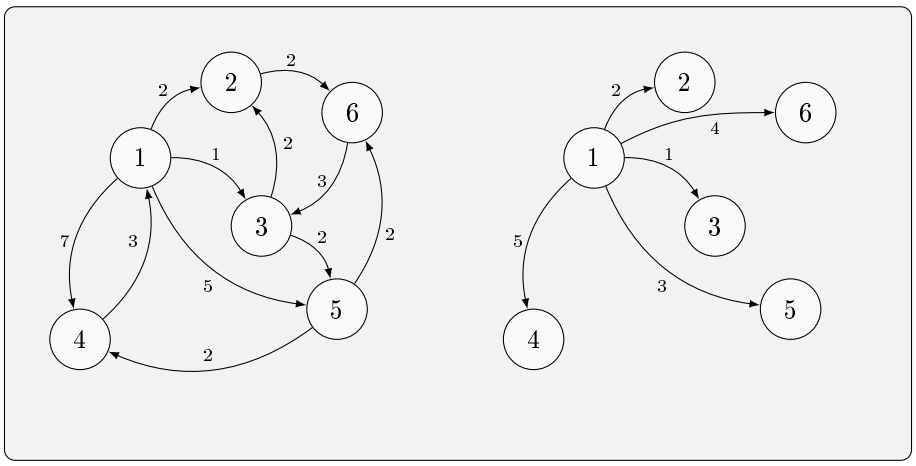

Figura 1: Costos mínimos desde el vértice **1**.

Puede observarse que aparecen aristas que no existen en el grafo original. Por ejemplo, la arista entre los vértices **1** y **4** con costo **5**, que resume el camino **1–3–5–4** o la arista que va de **1** a **6** con costo **4**, que resume el camino **1–3–6**. De la misma manera, la arista de costo **3** que va de **1** a **6** resume el camino **1–3–6**.

En la figura 2 se muestra la solución para la segunda versión del problema, siempre con vértice de referencia **1**.

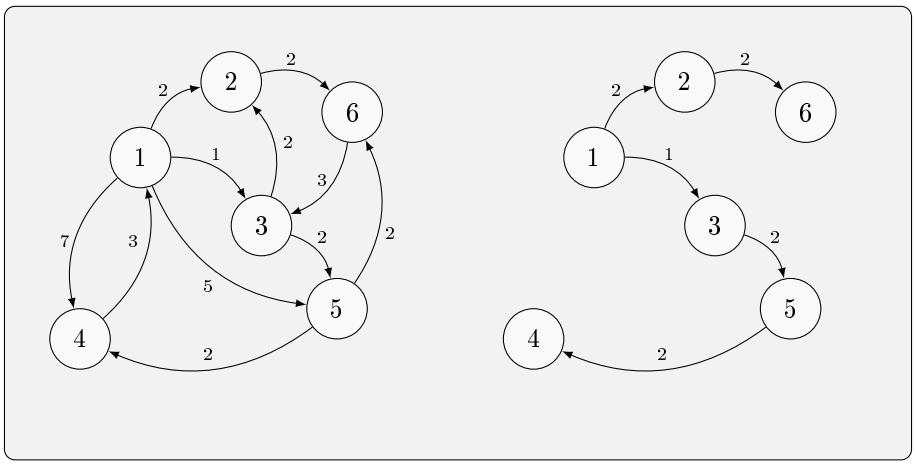

Figura 2: Caminos de costo mínimo desde el vértice **1**.

2. El algoritmo de Dijkstra

El algoritmo de Dijkstra es un algoritmo *greedy*. Estos algoritmos construyen la solució n iterativamente seleccionando en cada paso un componente de la solución entre un conjunto de candidatos. El criterio de selección es local. En nuestro caso, los candidatos son los vértices que van siendo seleccionados. El criterio es la menor distancia acumulada desde el vértice de referencia. En cada paso, se verifica si e lnuevo vértice ofrece un camino mejor que los que ya se tenían. Esto se muestra en la figura 3

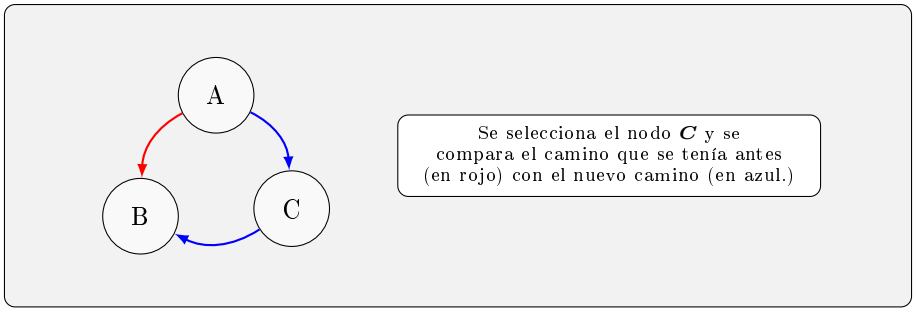

Figura 3: El proceso del algoritmo de Dijkstra.

Por ejemplo, en el caso de las figuras 1 y 2 tendríamos lo siguiente. En la columna de la derecha, los candidatos están ordenados por el costo para llegar hasta ellos desde el nodo de referencia, que es **1**. Estos costos se encuentran entre corchetes y pueden variar cuando se encuentra un camino mejor pasando por el nodo seleccionado. En el caso de que no haya ningún camino directo desde la referencia, el costo se coloca en infinito.

paso	selección	candidatos
0	$\{1\}$	$\{3[1], 2[2], 5[5], 4[7], 6[\infty]\}$
1	$\{1, 3\}$	$\{2[2], 5[3], 4[7], 6[\infty]\}$
2	$\{1, 2, 3\}$	$\{5[3], 6[4], 4[7]\}$
3	$\{1, 2, 3, 5\}$	$\{6[4], 4[5]\}$
4	$\{1, 2, 3, 5, 6\}$	$\{4[5]\}$
5	$\{1, 2, 3, 4, 5, 6\}$	$\emptyset$

3. Entregables

1. Un método externo que reciba un objeto de tipo **GrafoTDA** y un nodo del grafo y devuelva los grafos resultantes del algoritmo de Dijkstra (ambas formas.)

2. Un cálculo detelladado de la complejidad del algoritmo para este caso.

3. Una propuesta de optimización de la implementación estática de GrafoTDA que vimos en clase para mejorar la complejidad total del algoritmo.

Referencias

[1] Brassard, Gilles; Bratley, Paul: *Fundamentals of Algorithmics*, Prentice-Hall, 1996.

[2] Cormen, Thomas; Leiserson, Charles; Rivest, Ronald: *Introduction à l'algorithmique*, Dunod, 2004 (en francés.)

[3] Cormen, Thomas; Leiserson, Charles; Rivest, Ronald: *Introduction to Algorithms*, MIT Press, 2002.

[4] Goodrich, Michael; Tamassia, Roberto: *Algorithm Design and Applications*, John Wiley & Sons, 2015.

[5] Kleinberg, Jon; Tardos, éva: *Algorithm Design*, Pearson / Addison-Wesley, 2006.

[6] Sedgewick, Robert; Wayne, Kevin: *Algorithms*, Addison-Wesley, 2011.

[7] Skiena, Steven: *The Algorithm Design Manual*, Springer, 2012.