TRABAJO PRÁCTICO OBLIGATORIO
Programación III

Participantes:

1086926 - Kavcic, Luciano Ezequiel
1084711 - Rey, Federico Sebastián

Profesor:

1112094 - Rodriguez, Guillermo Horacio

Curso de Verano | Año 2021

UADE

ENUNCIADO

Usted va a recibir la Clase Grafo implementada en Java y deberá implementar 3 (tres) de los siguientes algoritmos:

-Algoritmo Breadth-First Search (BFS)

-Algoritmo Depth-First Search (DFS)

-Algoritmo de Prim

-Algoritmo de Kruskal

-Algoritmo de Dijsktra

-Algoritmo de Floyd

ALGORITMOS ELEGIDOS PARA LA IMPLEMENTACIÓN

1) DFS (Depth-First-Search): Algoritmo de búsqueda en profundidad.

   Parte desde un nodo fuente explorando recursivamente sus sucesores y desde el último nodo descubierto explora en profundidad cada arista. Cuando todas las aristas desde ese nodo han sido exploradas, la búsqueda retrocede al nodo desde la cual ese nodo fue descubierto. El proceso continúa hasta que todos los nodos alcanzables desde el nodo fuente original han sido descubiertos.
   Si restan nodos sin descubrir, uno de ellos es seleccionado como nuevo nodo fuente y la búsqueda se repite.

2) BFS (Breadth-First- Search): Algoritmo de búsqueda en anchura.

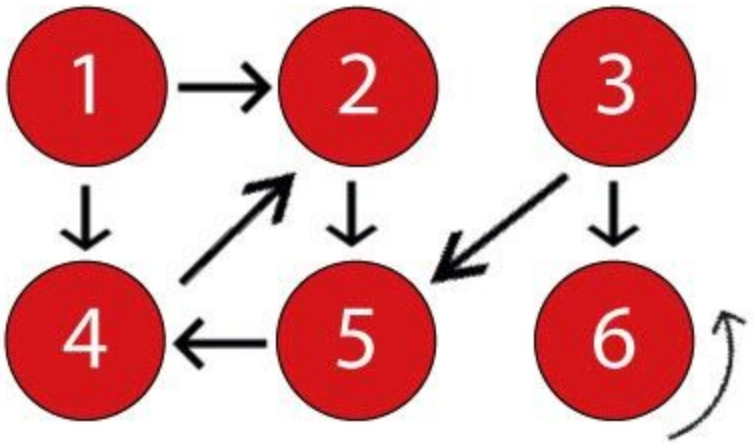
   Los nodos se recorren en el orden primero en ancho o por niveles. Para cada nodo alcanzable desde el nodo origen, BFS obtiene el camino más corto desde el origen hasta ese nodo (No existe camino desde el origen hasta ese nodo dado que tenga menos aristas).

3) Prim

   Es un algoritmo basado en la propiedad del árbol de recubrimiento de costo mínimo. Utiliza la técnica Greedy incorporando una arista en cada paso.
   Comienza con un solo árbol formado por un nodo cualquiera del grafo, en cada paso selecciona la arista de mínimo costo que conecta al árbol un nodo que no está en él. Termina cuando todos los nodos han sido incorporados al árbol de recubrimiento.

DFS (Depth-First-Search)

Grafo de Entrada



Salida

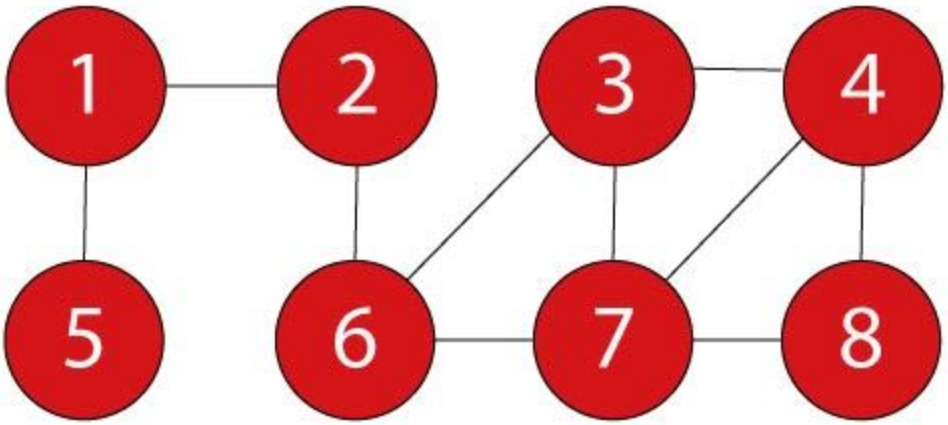
```
DFS
Origen: 1
1 -> 4 -> 2 -> 5
```

Dado un nodo de **origen** (en este caso el **nodo 1** y que puede ser hardcodeado para modificarlo fácilmente en la **línea 48 del archivo DFS.java**), vemos el recorrido que hace a partir de allí.
Al ser un **grafo dirigido**, se pudo haber encaminado por el nodo 2 o el nodo 4, de forma azarosa se encaminó por el **nodo 4**, para luego ir al **nodo 2** y finalmente al **nodo 5**.
Podemos observar que los **nodos 3 y 6** quedan excluidos del recorrido porque desde el nodo origen 1, se hace imposible dirigirse a ellos dos.

IMPLEMENTADO EN DFS.java

BFS (Breadth-First- Search)

Grafo de Entrada



Salida

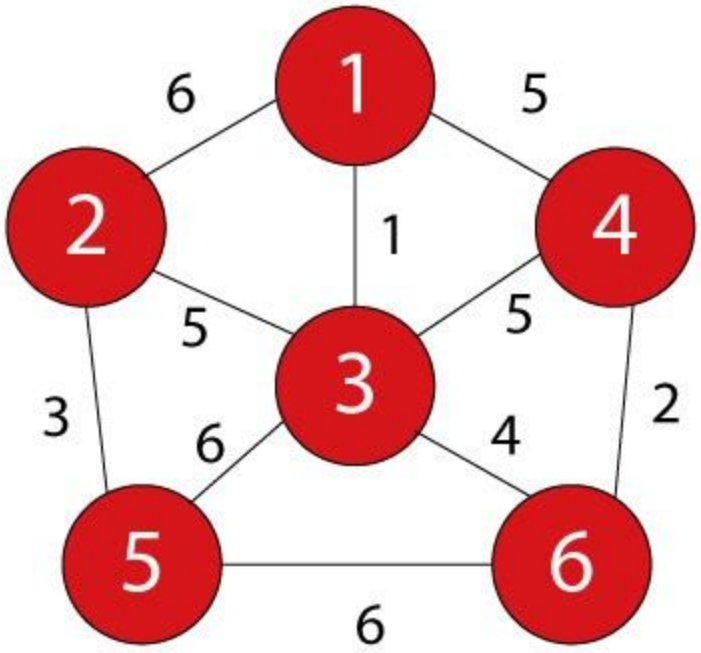
```
Origen: 2
Nodo: 6 Padre: 2 |
Nodo: 1 Padre: 2 |
Nodo: 7 Padre: 6 |
Nodo: 3 Padre: 6 |
Nodo: 8 Padre: 7 |
Nodo: 4 Padre: 7 |
Nodo: 5 Padre: 1 |
```

Dado un nodo de **origen** (en este caso el **nodo 2** y que puede ser hardcodeado para modificarlo fácilmente en la **línea 81 del archivo BFS.java**), vemos el recorrido que hace a partir de allí.
Siempre buscando el camino más corto para recorrer, irá al **nodo 6** y al **nodo 1.** Luego del **nodo 6** irá al **nodo 7** y al **nodo 3,** para más tarde partir del **nodo 7** hacia el **nodo 8** y el **nodo 4** y finalizar volviendo hacia el **nodo 1** donde le faltó visitar desde allí al **nodo 5.**

IMPLEMENTADO EN BFS.java

Prim

Grafo de Entrada



Salida

```
Prim
Origen: 1
 Nodo: 5 -
 Nodo: 2 - Peso: 3 -> Nodo: 5
 Nodo: 4 -
 Nodo: 6 - Peso: 2 -> Nodo: 4
 Nodo: 3 - Peso: 5 -> Nodo: 2   Peso: 4 -> Nodo: 6
 Nodo: 1 - Peso: 1 -> Nodo: 3
```

Dado un nodo de **origen** (en este caso el **nodo 1** y que puede ser hardcodeado para modificarlo fácilmente en la **línea 98 del archivo Prim.java**), vemos el recorrido que hace a partir de allí, aunque de alguna manera en este caso en la pantalla ya esté la “película terminada”.
Arrancando de abajo hacia arriba por el orden de impresión, vemos que partiendo del origen ya mencionado buscará la arista de menor costo y esta será la de **peso 1** que la une con el **nodo 3**, de allí otra vez irá por el camino menos costoso y este será el de la arista de **peso 4** que la

une con el **nodo 6** (podemos verlo en la extrema derecha de la penúltima línea) y de allí se dirige al **nodo 4** con **peso 2**. Luego de esto no le queda más remedio que volver al **nodo 3** y ver que desde allí el camino menos costoso es el de **peso 5** al **nodo 2** (Y es por esto que en la penúltima línea vemos dos pesos y tres nodos, ya que el nodo 3 se ha unido tanto al nodo 2 como al nodo 6. Con esto me refería a que teníamos impresa la “película terminada”), para finalmente ir del **nodo 2** al **nodo 5** con un **peso 3.**

IMPLEMENTADO EN Prim.java