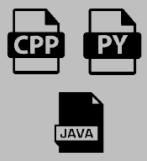
Taller en Sala Nro. 7
Algoritmos Voraces en Grafos

En la vida real, el algoritmo de Dijkstra se utiliza actualmente para definir protocolos de enrutamiento de redes como el OSPF. Más información en http://bit.ly/2uhRJw8

 **Trabajo en Parejas**	 **Hoy, plazo máximo de entrega**	 **Docente entrega código suelto en GitHub**
 **Sí .cpp, .py o .java**	 **No .zip, .txt, html o .doc**	 **Alumnos entregan código suelto por GitHub**

Ejercicio a resolver

1. Implementen el algoritmo de Dijkstra, el cual dado un grafo con pesos *no* negativos y un vértice inicial *v*, halla el camino más corto desde *v* hasta todos los demás vértices en el grafo (o determina que no hay camino entre ellos).

1.1 Implementen el algoritmo como tal, utilizando una tabla de distancias mínimas hasta cada uno de los vértices del grafo, y una tabla de padres para que pueda reconstruir el camino desde *v* hasta cada uno de los demás vértices.

```
public static Pair<int[], int[]> dijkstra(Digraph g, int v) {
   // complete...
  }
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Tenga en cuenta lo siguiente:

☑ En caso de que no haya forma de llegar a un vértice *s* desde *v*, la distancia hasta este en la tabla debe ser -1

☑ El par que retorna debe contener el arreglo de distancias y el arreglo de padres en las posiciones *first* y *second,* respectivamente.

☑ Utilicen valores centinela para denotar distancias infinitas y nodos sin padre. Para este caso en específico, es conveniente considerar como infinita distancia el valor máximo que puede almacenar un entero: `Integer.MAX_VALUE`.

☑ Le recomendamos que implemente el algoritmo utilizando una cola de prioridad, y meta en esta los nodos en forma de pares de la siguiente manera: `{peso, indice}`.

Si decide implementarlo de esta forma, puede utilizar la clase *Comparador* que le proveemos para que la cola de prioridad vaya soltando los nodos según los pesos (tienen mayor prioridad los que tienen menor peso). Para esto debe crear la cola de prioridad en su método de la siguiente manera:

```
PriorityQueue<Pair<Integer, Integer>> pq = new
PriorityQueue<>(new Comparador());
```

1.2 Implementen un método que dada la tabla de padres generada por el método que implementó en el numeral **1.1,** reconstruya el camino más corto desde un vértice *inicio* hasta un vértice *fin*.

```
public static ArrayList<Integer> obtenerCamino(int inicio,
int fin, Pair<int[], int[]> par) {
 // complete...
}
```

Tenga en cuenta lo siguiente:

☑ Asuma que el par que recibe es el retornado luego de llamar el Dijkstra desde el vértice *inicio*.

☑ En caso de que *inicio* = *fin*, retorne *null*.

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

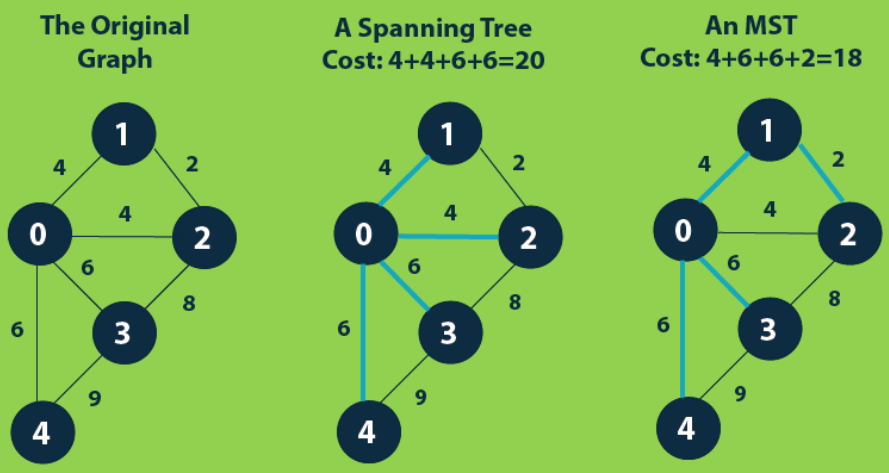
☑ En caso de que no exista camino entre *inicio* y *fin*, retorne *null*.

Nótese que este es el mismo problema del *Taller Nro. 4*, sólo que esta vez utilizamos un algoritmo mucho más eficiente y correcto. De hecho, este es el algoritmo para grafos en general con la menor complejidad asintótica, siempre y cuando el grafo *no* tenga aristas con peso negativo.

2. Dado un grafo conectado, no-dirigido y con peso, encuentren el costo total del subconjunto de aristas de costo mínimo que conservan el grafo conectado utilizando el algoritmo de Prim.

```
public static int prim(Digraph g) {
   // complete...
}
```

Este problema se conoce como el de hallar el Árbol de mínimo costo (*Minimum Spanning Tree*, *MST*), y difiere del problema de los talleres 4 y 6 (que por cierto aún no hemos solucionado de manera exacta) en que en este caso no es un recorrido lo que se busca (mucho menos uno cerrado), sino un árbol.

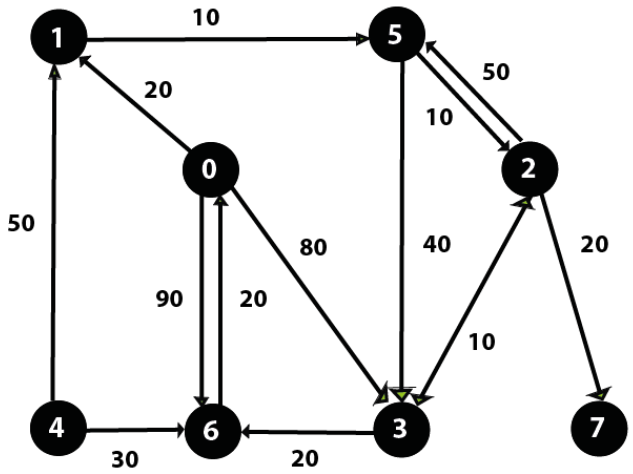


Nuevamente les recomendados utilizar una cola de prioridad para la implementación (se utilizaría exactamente igual que en el Dijkstra

Ayudas para resolver el Ejercicio

Ayudas para resolver el Ejercicio 1.2

Como un ejemplo, consideren el siguiente grafo:



En este grafo, luego de correr el Dijkstra desde 0 la tabla de distancias es [0, 20, 40, 50, -1, 30, 70, 60], y la de padres [-1, 0, 5, 2, -1, 1, 3, 2].

Esto nos permite determinar que, por ejemplo, el camino más corto desde 0 hasta 6 tiene un costo total de 70, y que el camino como tal es 0, 1, 5, 2, 3, 6.

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 **Ext.** 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agende una cita con él a través de **http://bit.ly/2gzVg10** , en la pestaña *Semana. Si no da clic en esta pestaña, parecerá que toda la agenda estará ocupada.*