Práctica MARP
Algorítmo de dijkstra con montículo sesgado

Juan Chozas Sumbera

2 de enero de 2020

UNIVERSIDAD COMPLUTENSE
MADRID

1. Implementación

He elegido Java como lenguaje para variar, ya que el año pasado hice las dos prácticas en C++. Para representar grafos he elegido la representación mediante listas de adyacencia, que contienen de pares de enteros para almacenar el vértice adyacente y el coste de la arista.

Para el montículo sesgado se almacena un entero (tamaño), un nodo (raíz), y una tabla (clave: entero, valor: nodo). La inclusión de la operación *decrecerClave* trae la necesidad de almacenar y mantener la tabla que almacena todos los nodos del montículo. Un nodo almacena un puntero a su nodo padre, además de a sus dos hijos. Para una implementación eficiente de *decrecerClave*, uso el puntero al padre del nodo objetivo para *cortar* el nodo objetivo del padre (en caso de que la clave decrecida sea menor que la del padre).

2. Ejecución

Incluyo también un script llamado *dijkstra* que puede usarse para compilar y ejecutar el programa. Admite varios argumentos, de los cuales solo es obligatorio *-n NUM*, usado para generar un grafo de *NUM* nodos. Se puede omitir este argumento en caso de usar la opción *-t 1* o *-t 2*, que ordena la ejecución de uno de los casos de prueba. Los grafos usados para probar el algoritmo se crearon de forma aleatoria, y se puede elegir la semilla con la opción *-s SEM*. La opción *-h* proporciona más información acerca de las opciones.