Proyecto Final
Estructura de Datos y Algoritmos 2

Juan Camilo Villa Correa Alejandro Ríos Muñoz

2023–06–01

1. Código - Dataset

En la carpeta adjunta se encuentran los archivos de todo el programa. El programa consiste en 4 algoritmos de grafos, para la búsqueda del camino más corto o de navegación sobre estos. Cada algoritmo tiene su implementación única, trabajada en clase y adaptada para el funcionamiento general del programa.

El dataset utilizado se construyó a partir del mapa de la Universidad EAFIT, e identificando cada uno de los puntos principales del campus. Contando con alrededor de 70 sitios distintos dentro de todo el campus universitario. Se realizó la construcción con el apoyo de múltiples recursos personales y hallados en internet para una ágil implementación. En los archivos adjuntos se encuentra el dataset utilizado para ser convertido en grafo.

Es importante indicar, que se crearon 2 grafos distintos para la implementación general del grafo. Primeramente, un grafo con pesos, que fue utilizado con los algoritmos del camino más coroto y de navegación, y posteriormente, un grafo sin pesos, para el algoritmo de búsqueda. En ambos casos, los grafos no son direccionados.

2. Resultados

Se realizó la implementación de los 4 algoritmos indicados (Dijkstra, Bellman-Ford, Floyd-Warshall y BFS). Los tiempos obtenidos en promedio fueron los siguientes, luego de realizar múltiples ejecuciones. Ver figura 1 .

- Tiempo Dijkstra: 0.001486159 s.
- Tiempo Floyd-Warshall: 0.011874088 s.
- Tiempo Bellman-Ford: 0.008140587 s.
- Tiempo BFS: 0.003513938 s.

Notoriamente, los dos algoritmos de mayor eficiencia en complejidad de tiempo en ejecución son Dijkstra y BFS. Ambos algoritmos cumplen papeles diferentes, pero con una eficiencia muy efectiva. Dijkstra es el algoritmo más eficiente de los cuatro analizados, mientras que el algoritmo de Floyd-Marshall es el menos eficiente en complejidad de tiempo.

Hay que tener en cuenta, que estos resultados pueden varias dependiendo del grafo y de múltiples situaciones dentro de la búsqueda que se termine realizando.

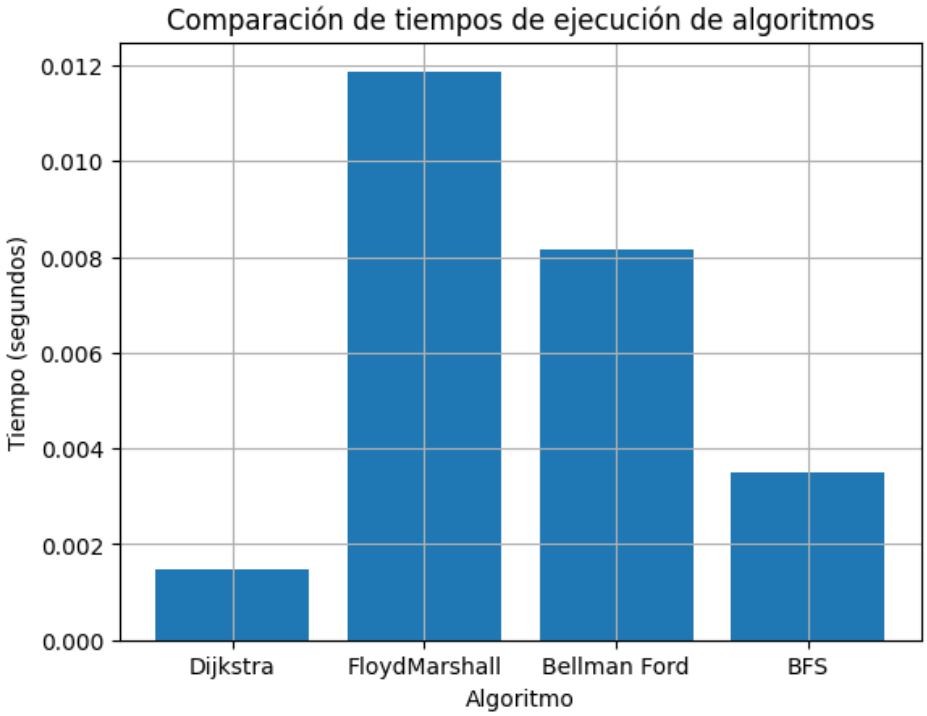

Figura 1: Resultados algoritmos.

3. Tiempos de ejecución

Al analizar los resultados obtenidos al ejecutar el programa, se puede destacar la coherencia entre la teoría vista en clase y los tiempos de ejecución observados. En particular, se observa que el algoritmo de Floyd-Marshall es el que requiere más tiempo para ejecutarse. Esta observación se alinea con la idea fundamental detrás del algoritmo de Floyd-Marshall, que consiste en dividir el problema en subproblemas más pequeños y resolverlos paso a paso. En otras palabras, el algoritmo calcula el camino más corto entre todos los pares de nodos, dividiendo así el problema original en una serie de subproblemas intermedios. Por lo tanto, es comprensible que el algoritmo de Floyd-Marshall presente un mayor tiempo de ejecución en comparación con otros algoritmos.

Por otro lado, los algoritmos de Dijkstra, BFS (Breadth-First Search) y Bellman-Ford también se evaluaron en términos de tiempo de ejecución. Estos algoritmos también se comportaron de manera acorde a lo esperado. El algoritmo de Dijkstra es conocido por su eficiencia en encontrar el camino más corto desde un origen hasta todos los demás nodos en un grafo con pesos no negativos. Mientras tanto, el algoritmo BFS se destaca por su capacidad de explorar un grafo en anchura, expandiendo gradualmente los nodos vecinos desde un origen dado. Por otro lado, el algoritmo de Bellman-Ford es útil para encontrar el camino más corto desde un origen hasta cualquier otro nodo en un grafo, incluso en presencia de aristas con pesos negativos.

En resumen, los resultados obtenidos en este estudio confirman la validez de la teoría y el conocimiento adquirido en clase. Los algoritmos de Dijkstra, BFS y Bellman-Ford demuestran su eficiencia en la búsqueda de caminos más cortos, y el algoritmo de Floyd-Marshall se destaca por su enfoque en la resolución de subproblemas. Este análisis y comprensión de los tiempos de ejecución de los algoritmos es crucial para seleccionar la mejor opción en diferentes escenarios y optimizar el rendimiento de las aplicaciones que requieren la resolución de problemas de rutas o caminos en grafos.