UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS.

Práctica 3: Heurísticas voraces (Algoritmo de Dijkstra)

Análisis y diseño de algoritmos

Autor:
Gloria Leticia González Troncoso.

1 Introducción

Para esta práctica lo que se hará es una implementación de los algoritmos voraces para realizar el algoritmo de Dijkstra. A lo largo del departamental se dió una explicación de las distintas formas de un ordenamiento y acomodo de grafos con distintos métodos.

2 Desarrollo

En el desarrollo de este problema es ver como funcina el algoritmo de Dijkstra, este lo que hace es ver cual es el camino más corto entre los vértices y los pesos que hay de en las aristas. Siguiendo, se comparan los distintos pesos y los que tienen menor valor para seguír con el camino hasta cierto vértice al final.

Para empezar, en el código se usarán librerías como heapq, random y time. Para iniciar se usas una clase donde se agregarán las funciones para crear el grafo, el primero es para obtener el grafo e iniciarlo como vacio; después se generan vértices en arreglos, los cuales también se encuentran en vacio; en las aristas se define un inicio (de donde va a partir), un final (es a donde va dirigida la arista de un vértice a otro, así como un peso para cada una de ellas). En la arista mantiene la estructura como [inicio,fin] y este tendrá un peso asignado.

Para inicializar el algoritmo este tendrá las distancias en un valor infinito, donde para cada vértice se obtiene a donde van dirigidos entre los vértices.Los vértices y el camino empieza en '0'.

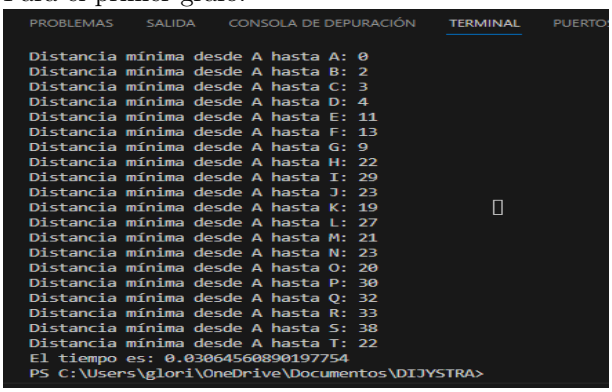
Usando ahora una organización de "cola", que es para darle prioridad a cada vértice y arista (con su peso), mientras la cola sea la distancia actual (variable) y vérticie actual, será igual a sacar un elemento donde ahora es el prioritario; ahora si la distancia es más grande que la distancia según el vértice actua. Para terminar con la creación del grafo, se toma un elemento sercano y el peso, donde será comparado con el grafo donde aparece usando el vértice.

3 Implementación

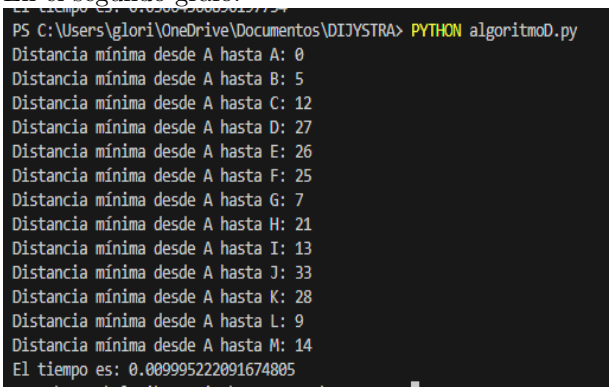
Para dar valores los pesos de cada arista y de vértice a vértice donde va dirigida la arista, se utilizó "grafo.agregar arista('A', 'B', 2)". Y para nombrar a los vértices se toma la funsión: "grafo.agregar vertice('A')". Así nombrando las aristas y los vértices requerdos.

Demostrando cada caso donde desde el vértice 'A' hasta cada uno de los demás, su camino más corto según los pesos que tenga cada arista.

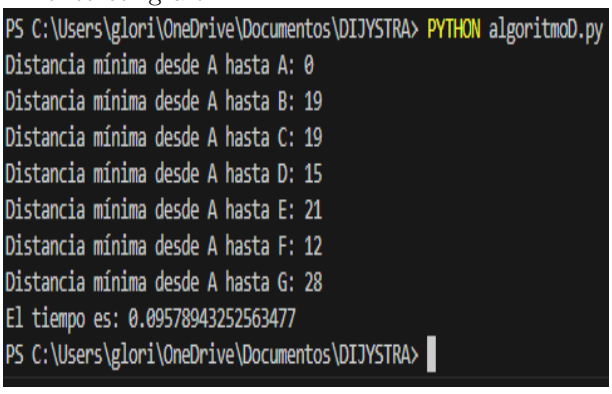
Para el primer grafo:



En el segundo grafo:



En el tercer grafo:



Para cada grafo, su tiempo de ejecución es distindo por la complejidad que manejan al estar analizando los vértices y las aristas. Mientras más vértices y

aristas adyascentes en cada vértice haya, el tiempo que tardará será mayor.

Su complejidad: En la práctica y desarrollo del algoritmo de Dijkstra, se dió s conocer que su complejidad depende de las estructuras de datos utilizadas. Si se utiliza una cola de prioridad basada en el mínimo de datos agregados, la complejidad temporal es O((número de vértices)+(número de aristas) (log —n.vértices—)).

4 Conclusión

Dentro del análisis de esta práctica y del mismo código, el algoritmo de Dijkstra utiliza grafos y su finalidad es encontrar el camino más rápido para llegar a una posición indicada. Al ser un algoritmo verás por su forma que encuentra cada camino para cada vértice tomando en cuenta el peso de cada arista.

5 Referencias

https://manualdelatex.com/tutoriales/figurasgoogle$_v ignette$
$https://www.codingame.com/playgrounds/7656/los-caminos-mas-cortos-con-el-algoritmo-de-dijkstra/el-algoritmo-de-dijkstra$