20-11-2025

REPORTE
Equipo E-A-B-M-O-D-E-L
Actividad Voraz

Análisis de Algoritmos

Equipo: **Códigos:**

Eduardo Ramos Ochoa	304489918
Andres Santiago Aguirre Macias	220293594
Cesar Emmanuel Gómez Martínez	214843698

Profesor:

JORGE ERNESTO LOPEZ ARCE DELGADO

Modulo:

D01

Centro Universitario de Ciencias Exactas e Ingenierías

INTRODUCCIÓN

En el presente trabajo, explora la implementación y funcionamiento de los algoritmos voraces de Prim, Kruskal y Dijkstra, ahora bien, tal cual se ha tratado a lo largo de las sesiones en el aula, sabemos que la característica más importante de los algoritmos voraces es, la de "elegir la opción que parece más optima en un momento determinado" a fin de obtener la solución a un problema, si bien es cierto, no en todas las ocasiones, este enfoque resulta ser el mejor, al menos desde una perspectiva global, así entonces, los algoritmos voraces que se implementan en este trabajo, llevan a cabo selecciones a nivel local, con la intención de arribar a la solución óptima.

En los casos que ahora nos ocupan, esto es, tratándose de los algoritmos de Prim, Kruskal y Dijkstra, los dos primeros llevarán a cabo el desarrollo del "árbol de expansión menor" o "MST" por sus siglas en ingles, tomando como base, un grafo creado por nuestro equipo, y, por lo que toca al algoritmo de Dijkstra, este, al ejecutarse con el grafo antes referido, realizará el trazado de la ruta con menor peso con base en; el nodo inicial que se elija y, el nodo final que se fije.

OBJETIVOS:

- **GENERALES**

I- Como primer objetivo GENERAL, se llevará a cabo la creación de un grafo que cumpla con las indicaciones otorgadas por el profesor, esto es, concebir un grafo; no dirigido, ponderado y, con al menos 6 nodos y 9 aristas.

II- Nuestro segundo objetivo GENERAL consiste en, la implementación mediante programación en el lenguaje Python, de los algoritmos voraces de; Prim, Kruskal y Dijkstra, con el fin de llevar a cabo la comparación de comportamientos de los mismos.

- **PARTICULARES**

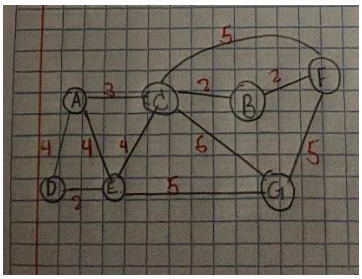
I- De forma particular, se realizará una breve comparación entre los algoritmos de Prim y Kruskal, misma que se enfocará en; a) diferencias en el proceso de c/u, b)las diferencias que se obtengan en los resultados arrojados y c) las complejidades aproximadas de estos.

II- Toda vez que se ha optado por implementar el algoritmo de Dijkstra, se mostrará la tabla de distancias mínimas obtenidas desde un nodo origen determinado, así como el camino más corto resultante hacia un nodo destino seleccionado dentro del propio grafo

DESARROLLO:

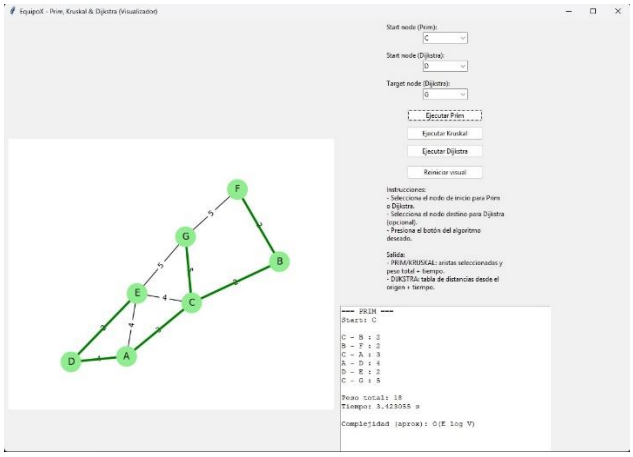
En primer lugar y, como paso necesario para la correcta compilación de nuestro programa, se empleó **Python en su versión 3.10+,** así como en, la creación de una interfaz gráfica mediante el uso de la **biblioteca Tkinter**, en dicha GUI, al ejecutar el programa, se obtiene, un menú interactivo que permite al usuario, seleccionar; ¿Qué algoritmo desea ejecutar?, así también, ¿Qué nodo del grafo implementado, desea que sirva como inicio y, cual nodo funja a modo de destino?, una vez realizada la selección del algoritmo a implementar, gracias a la interfaz gráfica, se puede observar en tiempo real, la ejecución de cada uno de los algoritmos, esto es, la realización del MST para el caso de PRIM y KRUSKAL, así como, el trazado de la ruta menor en el caso de DIJKSTRA, finalmente, se visualizan tanto los resultados en el grafo, esto último, mediante el uso de las bibliotecas **NetworkX y Matplotlib**, que facilitan la representación y manipulación de grafos y su visualización respectivamente.

A efecto de cumplir con el primer objetivo general, hemos diseñado un grafo; ponderado, no dirigido, **con un total de 7 nodos y 10 aristas**, es decir, procurando cumplir con los requisitos de esta actividad. Cada arista cuenta con un peso asociado que condiciona las decisiones que toman los algoritmos implementados, a continuación, se muestra el diseño inicial o conceptual de grafo, cabe decir que, se tomó como inspiración para el resultado final.

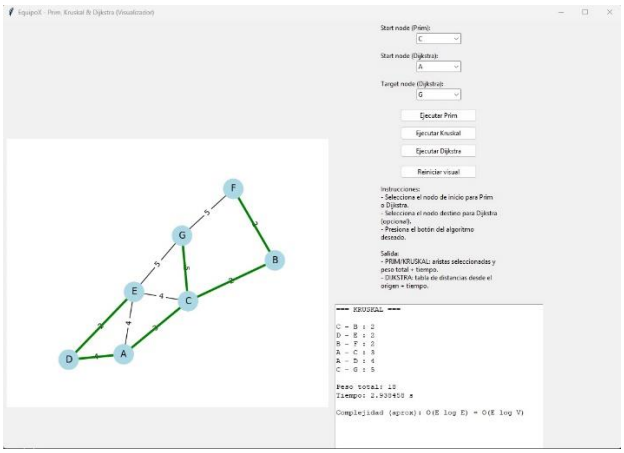


Grafo empleado a modo de inspiración, procurando que cumpliese con las indicaciones.

Una vez que diseñamos nuestro grafo para la actividad, se procedió con la implementación de los algoritmos voraces de Prim y Kruskal, encargados de construir un Árbol de Expansión Mínima (MST). En el caso de Prim, se hizo uso de un heap como estructura de prioridad para seleccionar de manera eficiente la arista de menor peso que conectara a un nodo recientemente incorporado. Por su parte, Kruskal requirió la implementación de una estructura Union-Find (unión y búsqueda), encargada de evitar ciclos al momento de seleccionar aristas.

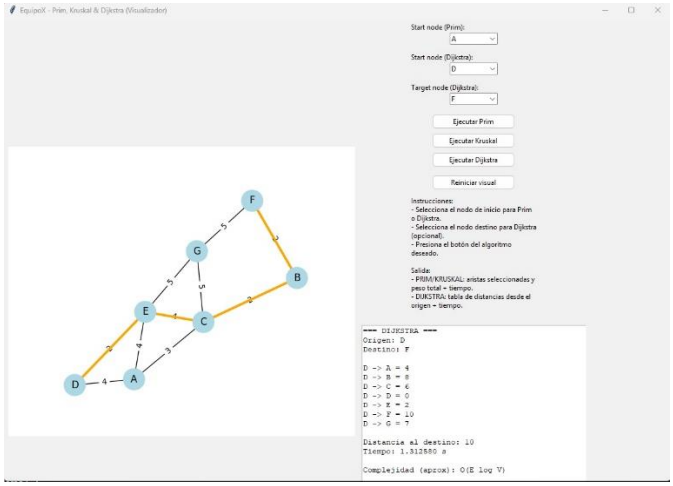


Ejecución / resultado del algoritmo de PRIM, mostrando el MST y los tiempos de ejecución



Ejecución / resultado del algoritmo de KRUKAL, mostrando el MST y los tiempos de ejecución

Finalmente, tal cual se ha comentado en líneas anteriores de este reporte, nuestro equipo optó por incluir el algoritmo de Dijkstra, si bien es verdad, este difiere de los anteriores por no realizar el MST, sino buscar la ruta dentro del grafo con menor peso, la estructura del grafo que diseñamos, nos permite y facilita igualmente su implementación.



Ejecución de algoritmo DIJKSTRA, mostrando la ruta con menor peso, con base en los datos ingresado a la GUI.

Durante la ejecución de los tres algoritmos, el sistema fue configurado para mostrar en tiempo real la construcción del MST o el avance de la búsqueda del camino más corto. Asimismo, se incorporó un panel lateral de texto dentro de la interfaz, encargado de imprimir:

COMPARACIÓN ENTRE ALGORITMOS PRIM Y KRUSKAL

Habiendo realizado varias pruebas entre los algoritmos de PRIM y KRUSKAL, esto es,

Criterio	Prim	Kruskal
Enfoque	Expansión desde un nodo inicial	Selección global de aristas
Estructura clave	Heap (cola de prioridad)	Union-Find
Selección de aristas	Local: siempre la más barata que conecte un nodo nuevo	Global: siempre la más barata sin generar ciclos
Mejor rendimiento	Grafos densos	Grafos dispersos o desconectados por componentes
Complejidad	O(E log V)	O(E log E) ≈ O(E log V)

COMENTARIOS SOBRE ALGORITMOS EN LO PARTICULAR:

En los casos de los algoritmos de PRIM y KRUSKAL, ambos llevan realizan el MST, aunque con variaciones en cuanto a los tiempos de ejecución, como se aprecia a continuación

PRIM	KRUSKAL
Peso total: 18	**Peso total: 18**
Tiempo: 3.378494 s.	**Tiempo: 2.908298 s**

Nota: resultados tomados de ejecuciones de programa.

Entonces, a la luz de los resultados obtenidos, podemos afirmar que, si bien es verdad, la ejecución u obtención del MST por parte de uno y otro, tiene una variación en cuanto a sus tiempos, la misma, al menos por lo que respecta al grafo implementado para esta práctica, no es abismal, sin embargo, en virtud de las complejidades propias de uno y otro, a medida que el grafo (nodos y aristas) fuese mayor, el tiempo de ejecución, indudablemente, aumentaría a la par.

CONCLUSIONES:

La realización de esta práctica nos permitió comprender de manera clara y visual el funcionamiento de los algoritmos voraces, es decir, pudimos llevar a la práctica, los conceptos que hemos venido tratando a lo largo de las semanas anteriores en clase, asimismo, comprobamos las diferencias que existen por lo que toca a la ejecución de los algoritmos PRIM y KRUSKAL, por lo que toca a sus complejidades temporales.

Si bien es verdad, de modo opcional hemos implementado DIJKSTRA, igualmente hemos llevado a la práctica lo aprendido en torno a la búsqueda de camino de menor peso dentro de grafos, todo esto, de la mano con el concepto de “algoritmo voraz”, buscando la solución optima a un problema planteado.

A modo de comentario de cierre, queremos añadir que, la realización de esta práctica, nos ha permitido reforzar nuestro entendimiento acerca del funcionamiento y lógica detrás de los algoritmos aquí referidos, esto es, mediante un acercamiento empírico.

REFERENCIAS

Para esta práctica, hemos tomado referencias del sitio W3School, en concreto…

W3Schools. Dijkstra's Algorithm

https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php

W3Schools. Prim's Algorithm

https://www.w3schools.com/dsa/dsa_algo_mst_prim.php

W3Schools. Kruskal's Algorithm

https://www.w3schools.com/dsa/dsa_algo_mst_kruskal.php