ACTIVIDAD VORAZ

Algoritmos de Prim y Kruskal

Alumnos

De la Mora Villaseñor Diego Gabriel || Código: 220576197
Lopez Esparza Angel Emanuel || Código: 223991772
Lopez Galván Melanie Montserrat || Código:220574046
Materia: Análisis de algoritmos

Sección: D06

Profesor: Lopez Arce Delgado Jorge Ernesto

Contenido

Introducción 2
Objetivos 3
Desarrollo 3
Resultados 4
Conclusiones 4
Bibliografía 5

Introducción

El uso de algoritmos basados en grafos es altamente utilizado para la verificación de rutas y costos para recorrer distintos caminos. Esta clase de algoritmos son un conjunto de instrucciones para explorar y analizar las relaciones entre nodos en una estructura de grafos. Algunos de los tipos más comunes son Kruskal, Prim y Dijkstra, en este caso nos concentraremos en los primeros dos algoritmos.

Kruskal: Este algoritmo fue diseñado en 1956 por Joseph B. Kruskal, investigador del Math Center. El objetivo del algoritmo de Kruskal o **Minimum Spanning Tree** (MST) es construir un árbol formado por arcos sucesivamente seleccionados de mínimo peso a partir de un grafo con pesos en los arcos.

Una de sus aplicaciones típicas es el diseño de redes telefónicas, la formulación de también ha sido aplicada para halar soluciones en diversas áreas como el diseño de redes de transporte, diseño de redes de telecomunicaciones, sistemas distribuidos, interpretación de datos climatológicos, visión artificial y búsqueda de superestructuras.

Dado un grafo *G* con nodos conectados por arcos con peso (coste o longitud): el peso o coste total de un árbol será la suma de pesos de sus arcos. Obviamente, árboles diferentes tendrán un coste diferente. El problema es entonces ¿cómo encontrar el árbol de coste total mínimo?

Una manera de encontrar la solución al problema del árbol de coste total mínimo, es la enumeración completa. Aunque esta forma de resolución es eficaz, no se puede considerar un algoritmo, y además no es nada eficiente.

Este problema fue resuelto independientemente por Dijkstra (1959), Kruskal (1956) y Prim (1957) y la existencia de un algoritmo polinomial (que todos ellos demostraron) es una grata sorpresa, debido a que un grafo con *N* vértices puede llegar a contener *N^N-2* subárboles. A lo largo de la historia se ha hecho un gran esfuerzo para encontrar un algoritmo rápido para este problema. El algoritmo de Kruskal es uno de los más fáciles de entender y probablemente el mejor para resolver problemas a mano.

Prim: En 1957, Robert Prim descubrió un algoritmo útil para la resolución del problema del algoritmo MST, el cual es un problema típico de optimización combinatoria. El algoritmo de Prim encuentra un árbol de peso total mínimo conectando nodos o vértices con arcos de peso mínimo del grafo sin formar ciclos.

Al igual que el algoritmo de Kruskal, el de Prim también ha sido aplicado para hallar soluciones en diversas áreas (diseño de redes de transporte, diseño de redes de telecomunicaciones - TV por cable, sistemas distribuidos, interpretación de datos climatológicos, visión artificial - análisis de imágenes - extracción de rasgos de parentesco, análisis de clusters y búsqueda de superestructuras de quasar, plegamiento de proteínas,

reconocimiento de células cancerosas, y otros). También se ha utilizado para encontrar soluciones aproximadas a problemas NP-Hard como el del “viajante de comercio”.

Consiste en dividir los nodos de un grafo en dos conjuntos: procesados y no procesados. Al principio, hay un nodo en el conjunto procesado que corresponde a el equipo central; en cada interacción se incrementa el grafo de procesados en un nodo (cuyo arco de conexión es mínimo) hasta llegar a establecer la conexión de todos los nodos del grafo a procesar.

Objetivos

Generales:

1. Implementar los algoritmos de búsqueda con grafos de Prim y Kruskal en Python.
2. Crear una GUI que muestre los resultados para ser comparados entre los dos algoritmos implementados.

Particulares:

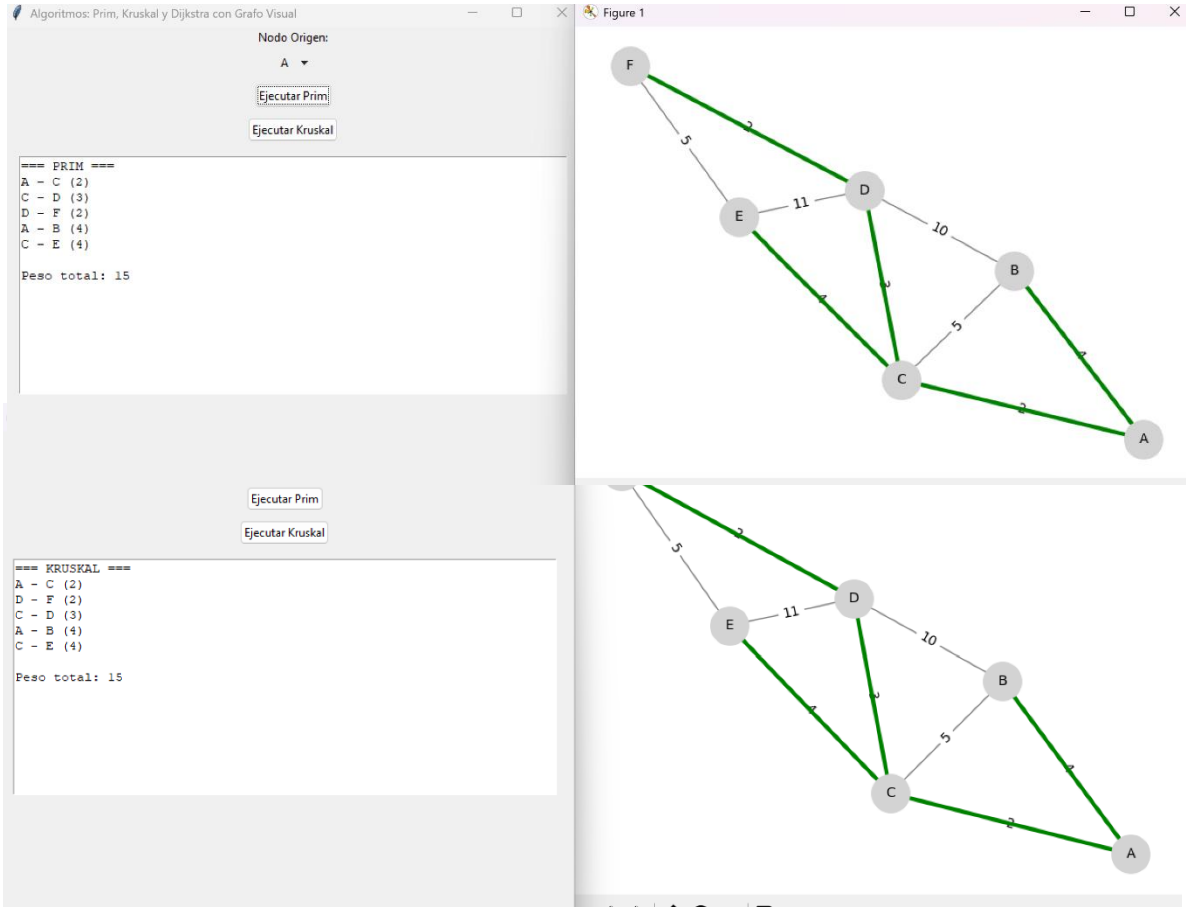
1. Implementar algoritmo de Prim con heap.
2. Implementar algoritmo de Kruskal usando Union-Find.
3. Mostrar las diferencias en el proceso de resolución del problema.

Desarrollo

Para la creación de los algoritmos creamos archivos individuales para cada uno, en este caso fue más fácil implementar el algoritmo de Prim debido al uso de la biblioteca heapq que cuenta con módulos para operaciones con heap, además de hacer uso de listas para la creación de los índices que mantiene el mínimo como el índice 0, con lo que se hace todo el trabajo relativamente.

En cambio, para la implementación de kruskal creamos una clase para Union-Find con compresión de caminos que permite hace dos operaciones principales, find que permite saber a que conjunto pertenece un elemento y unión que une dos conjuntos. Para esta clase hacemos uso de diccionarios que guarda el representante de cada elemento y después se encuentran los representantes rx y ry de los conjuntos donde están x y, y si son distintos los une, de forma que se une el árbol más pequeño debajo del más grande y si tienen el mismo rango uno se convierte en padre y a ese se le aumenta el rango.

Resultados



Conclusiones

Los algoritmos implementados, a pesar de presentar resultados similares en casi todos los casos son útiles en distintos ámbitos como se menciono previamente, como se puede percibir en los resultados a pesar de ser similar esto puede cambiar en grafos con mayor cantidad de nodos. Nos pareció bastante interesante la implementación debido a la cantidad de implementaciones de estos algoritmos.

Bibliografía

Grafos - software para la construcción, edición y análisis de grafos. (n.d.-a). Upv.es. Retrieved November 20, 2025, from https://arodrigu.webs.upv.es/grafos/doku.php?id=algoritmo_kruskal

Grafos - software para la construcción, edición y análisis de grafos. (n.d.-b). Upv.es. Retrieved November 20, 2025, from https://arodrigu.webs.upv.es/grafos/doku.php?id=algoritmo_prim