Imagen que contiene firmar

Descripción generada automáticamente

Análisis y Diseño de Algoritmos Avanzados

8 de noviembre de 2021

Carlos Moises Chavez Jimenez

ITESM

A01637322

Actividad Integradora 2: Reflexión

**Reflexión**

Para empezar mi reflexión, me gustaría decir que quedó muy satisfecho del trabajo que hicimos para esta segunda actividad integradora. En esta ocasión, tuvimos que implementar más algoritmos y aplicarlos de diferente forma e incluso con diferentes lenguajes.

Antes de las funciones, tuvimos que leer el archivo de texto con los datos de entrada. No tuvimos ningún problema en ello, pero por una razón que veremos más adelante, dejamos las últimas coordenadas sin guardar.

Empezando con la función 1, pudimos reutilizar el código de actividades pasadas. Implementamos el algoritmo de Dijkstra para encontrar el camino más corto de un nodo a otro. No fue para nada difícil, porque tomábamos la matriz correspondiente a las distancias. Procesamos los datos de la matriz, y como salida desplegamos la distancia.

Para la segunda función se utilizó el algoritmo del agente viajero. El objetivo era recorrer cada nodo no más de una vez, y al final regresar al nodo inicial. En un principio pensamos que podíamos utilizar el algoritmo de Floyd, pero rectificamos por la forma en que se pedía la salida.

Para la función tres nos llevó mucho tiempo encontrar la forma de resolverlo. Pensamos en la idea de utilizar el algoritmo de la mochila, considerando que había un limite de flujo y dos matrices por utilizar. Pero, nos topamos con que no tenemos peso máximo (capacidad máxima para este caso). Al final, optamos por buscar otro algoritmo que abordara mejor la situación. Encontramos el algoritmo de Ford-Fulkerson que especial para manejar flujo en un grafo. Se toma el nodo inicial y el nodo final, y se hacen todos los recorridos posibles para llegar de un nodo a otro. Siempre tomando la capacidad de flujo residual mínima de cada recorrido. Al final, se suman las capacidades residuales mínimas que se encontraron y ese es flujo máximo.

Para la última función, decidimos implementarla en Python en vez de C++. Esto porque Python tiene una librería que maneja de manera sencilla el algoritmo a implementar. Este es el algoritmo de Voronoi. La librería scipy tiene funciones para este algoritmo. Te permite graficar (tomando coordenadas de entrada) los famosos diagramas de Voronoi basados en la distancia Euclidiana. Pensando en la actividad anterior, tuvimos muchos problemas para simular el algoritmo de Delaunay, por eso decidimos hacerlo en Python. Aquí sí leímos las coordenadas del archivo de texto, y graficamos el diagrama.

Esta actividad en general fue retadora. Pasamos mucho tiempo investigando diferentes algoritmos que resolvieran la situación que se nos presentaba. Tuvimos muchos problemas al principio al querer implementar la función 4 en C++, y eso nos quitó mucho tiempo. Pero gracias a que vimos que, en la entrega pasada, la implementación en Python fue más sencilla, nos quitamos de problemas y cambiamos de lenguaje. Creo que fue un buen aumento en la dificultad comparado con la actividad integradora pasada, y se me hizo una buena forma de demostrar los algoritmos que hemos aprendido a lo largo del curso.