Tecnológico de Monterrey

Análisis y diseño de algoritmos avanzados

Grupo 570

Reflexión

Fernanda Ríos Juárez- A01656047

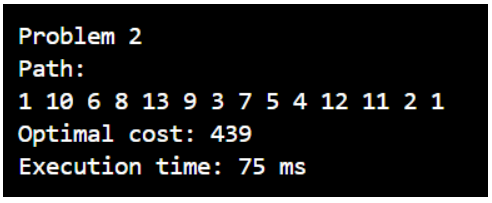
22 de mayo del 2024

En la materia se vieron diversos algoritmos,de los cuales, varios podían emplearse para la resolución de la actividad, sin embargo, al igual que en la actividad integral pasada, las complejidades de estos fueron los diferenciadores para saber cuáles emplear para la solución.

Para la solución de la primera parte, donde el problema 1 era un problema de árbol recubridor mínimo (MST), se empleó el algoritmo de Prim. Esto se debe a que, si bien su complejidad es básicamente la misma que del algoritmo Kruskal, O(E log V) al emplear una lista de adyacencia, su implementación es mucho más sencilla que la de Kruskal.

Para la segunda parte del, donde el problema 2 era un problema del agente viajero. Este problema, en general, es un problema bastante costoso, lo cual hace compleja la elección del mejor algoritmo para resolverlo. Entre las mejores opciones se encontraban el algoritmo de programación dinámica y el de branch and bound. Por lo visto en clase, el algoritmo de programación dinámica tiene una complejidad de $O(2^n(n + m))$, mientras que el de Branch and Bound tiene una de complejidad de $O(2^n)$. Por esta razón, se empleó el algoritmo de Branch and bound para la resolución del problema 2. Sin embargo, hubieron varias complicaciones para implementar el algoritmo, haciendo así que se tuviera una complejidad de $O(2^n * n^2)$ en el peor de los casos. La parte cuadrada de esta complejidad se debe a que se empleó una matriz de adyacencia en lugar de una lista de adyacencia como para los otros dos problemas. La razón de esto es porque el algoritmo de Branch and Bound necesita verificar rápidamente el costo entre cualquier par de nodos varias veces, haciendo así que la matriz de adyacencia sea la opción ideal porque ofrece el acceso a esta verificación en un tiempo lineal, además de que es lo más común al usar este algoritmo.

Además, se hicieron además pruebas usando la librería Chrono, el tiempo de ejecución del problema 2 es de 75 ms para el segundo archivo de entrada, el cual es el más largo con la finalidad de verificar que el tiempo de ejecución fuera bueno al menos para el archivo más pesado.



Para el problema 3, el cual era un problema de flujo máximo en redes, también había un par de algoritmos que se podían usar. Algunos de ellos siendo relativamente similares en cuestión de la complejidad. Si bien el de mejor complejidad es el algoritmo Orlin debido a que es de O(nm), nadie del equipo tiene buen entendimiento de este algoritmo aún habiendo investigado un poco de este y fue preferible no complicarnos con su implementación. Por ende, se optó por la mejor opción vista en clase, la cual fue el algoritmo de Dinic, cuya complejidad es de O(n^2m).

Referencias

OpenAI. "Análisis de complejidad algorítmica para el TSP." ChatGPT, versión GPT-4. https://chatgpt.com/share/681573c5-81f4-8000-bdea-b6a946284fab.

RODRÍGUEZ TELLO E.A. (2025, 21 abril). Problema del agente viajero (TSP)

RODRÍGUEZ TELLO E.A. (2025, 7 abril). Algoritmos en grafos, Flujo máximo en redes

RODRÍGUEZ TELLO E.A. (2025, 31 marzo). Algoritmos en grafos, Árbol recubridor mínimo (MST)