Diseño y Análisis de Algoritmos

Práctica 6

Problema del Viajante de Comercio (TSP)

Realizado por ……. Stephan Brommer Gutiérrez
Contacto……... alu0101493497@ull.edu.es

Universidad de La Laguna

ÍNDICE

Introducción

En esta práctica abordaremos el conocido Problema del Viajante de Comercio (TSP), un desafío clásico en la teoría de la computación y la optimización combinatoria. El TSP implica encontrar la ruta más corta que un vendedor debe seguir para visitar todos los nodos de un grafo ponderado (ciudades en este caso) exactamente una vez y luego regresar al nodo de origen.

Para resolver este problema, implementaremos tres enfoques diferentes:

- **Algoritmo de fuerza bruta**: Probaremos todas las posibles combinaciones de rutas y seleccionaremos la más corta.

- **Algoritmo voraz**: En cada paso, seleccionaremos el nodo más cercano no visitado como siguiente paso.

- **Algoritmo de programación dinámica**: Diseñaremos un algoritmo eficiente que use la técnica de programación dinámica para encontrar la solución óptima.

Además, desarrollaremos un generador de instancias para crear casos de prueba con diferentes números de nodos, lo que nos permitirá experimentar con el rendimiento de nuestros algoritmos en instancias de diferentes tamaños.

Nuestro objetivo será implementar estos algoritmos de manera eficiente, experimentar con instancias de prueba de diferentes tamaños y analizar el rendimiento y la precisión de cada enfoque. Al finalizar, presentaremos los resultados en una tabla que incluya los valores de las soluciones encontradas por cada algoritmo y los tiempos de ejecución asociados, esto se observará en el siguiente apartado.

Cabe destacar que, como parámetros hemos permitido la introducción de un tiempo límite dado en minutos, y si se excede, aparecerá en la tabla en el apartado del tiempo de ejecución del algoritmo la palabra “EXCESIVO”, y como coste y camino mínimo aparecerá el mejor resultado encontrado hasta el momento. En el caso de la programación dinámica si el tiempo límite se excediera durante el algoritmo de HeldKarp, que es el que he escogido para su resolución, el coste sería el mínimo encontrado hasta el momento y el camino estaría vacío, y si se excede el tiempo límite durante la reconstrucción del camino mínimo, tanto el camino como el coste serán los mejores hasta el momento.

Tabla con los resultados obtenidos para los distintos tamaños de problemas

Instancia	Valor Fuerza Bruta	Tiempo Fuerza Bruta (µs)	Valor Prog. Dinámica	Tiempo Prog. Dinámica (µs)	Valor Voraz	Tiempo Voraz (µs)
4 nodos	55	12	55	21	85	7
5 nodos	169.117	22	169.117	38	189.893	7
7 nodos	217.227	781	217.227	196	242.015	15
10 nodos	260.567	644125	260.567	2881	280.15	37
15 nodos	214.302	EXCESIVO	153.635	233514	255.167	118
18 nodos	492.413	EXCESIVO	233.608	2.78378e+06	342.505	189
20 nodos	543.609	EXCESIVO	186.4	1.56399e+07	337.619	223
40 nodos	1497.18	EXCESIVO	No válido	No válido	343.973	1727
100 nodos	4638.12	EXCESIVO	No válido	No válido	647.438	28230

Nota: Se debe tener en cuenta que los tiempos de ejecución están expresados en microsegundos para evitar que el tiempo de ejecución del algoritmo voraz siempre muestre 0 ms.

Es importante destacar que el algoritmo de programación dinámica puede enfrentar limitaciones de memoria en máquinas virtuales con conjuntos de datos de tamaño significativo. Por ejemplo, a partir de los 20 nodos, es posible que el tamaño de la tabla de memoria supere la capacidad de mi máquina virtual, lo que puede resultar en un error de "Segmentation fault (core dumped)". Destacar también que, se ha establecido un tiempo límite de 5 minutos para la ejecución de los algoritmos.

Al observar la tabla de resultados, se nota que el tiempo de ejecución del algoritmo voraz siempre es más rápido que el de los otros dos algoritmos. Además, el tiempo de ejecución del algoritmo de fuerza bruta tiende a aumentar significativamente a partir de los 10 nodos y a ser superior a los 5 minutos a partir de los 15 nodos. Se aproxima al tiempo de ejecución del algoritmo de programación dinámica hasta los 5 nodos.

Conclusiones extraídas

Tras todos los experimentos vamos a mostrar una serie de conclusiones extraídas de los tres algoritmos y de la estructura del programa:

- **Eficiencia y complejidad computacional**
  - El algoritmo de fuerza bruta es efectivo para encontrar la solución óptima, pero su complejidad factorial lo hace poco práctico para instancias grandes.
  - El algoritmo voraz proporciona soluciones rápidas, pero no garantiza la óptima debido a su naturaleza de elección.
  - El algoritmo de programación dinámica ofrece una solución óptima con una complejidad temporal más manejable que el enfoque de fuerza bruta, pero ocupa mucha más memoria.
- **Adaptabilidad y flexibilidad**
  - El tiempo límite de ejecución configurable permite controlar el rendimiento de los algoritmos y evitar tiempos de ejecución excesivos, sobre todo está pensado para el algoritmo de fuerza bruta.
  - La capacidad de generar instancias aleatorias facilita la experimentación y el análisis del comportamiento de los algoritmos en diferentes escenarios.
- **Patrón estrategia y principios SOLID**
  - Se utiliza el patrón estrategia para modularizar y desacoplar las diferentes implementaciones de los algoritmos de resolución del TSP. Cada algoritmo se encapsula en una clase que implementa una interfaz común, lo que permite que sean intercambiables y que se agreguen nuevos algoritmos fácilmente sin modificar el código existente. Esto proporciona flexibilidad y extensibilidad al programa. Además se hace uso de una buena estructura y diseño siguiendo los principios SOLID.