Programación Dinámica: TSP

Guillermo I. Bautista G. *Estudiante, UPIIZ*

Resumen—La solución de problemas se puede clasificar según el problema, y se pueden encontrar problemas de soluciones rápidas, llamados en programación problemas logarítmicos que a pesar de la cantidad de entradas no tardan mas que $log(C)$ en resolverse, y por otro lado tenemos problemas que son tan difíciles de clasificar que los tenemos apartados en una sección denominan NP (no polinomiales) esto quiere decir que son problemas de demora exponencial o algo más tardado. se atacará un problema famoso de este tipo según distintas formas de programación llegando a una solución propuesta.

Index Terms—TSP, algoritmo, dinámico, grafo, ciudad, nodo, vértice.

1. Introduction

La programación dinámica es la forma de optimizar los recursos (memoria y tiempo), a la hora de resolver una problemática, trata, de dada una solución antigua encontrar una ayuda para una nueva solución, y de preferencia no repetir procesos en el camino, entendido esto podemos distinguir varias formas de resolver un problemas, soluciones iterativas, recursivas, dinámicas, etc, dependiendo del problema podremos atacar de mejor manera un problema o de peor manera.

2. Travelling Salesman Problem (TSP)

Dado un conjunto de ciudades y distancias entre ellas, cual es el camino más corto para recorrer todas las ciudades volviendo a la ciudad donde empecemos.

3. algoritmo de Held-Karp, y solución personal

Consiste en tener tu matriz de distancias y recorrerla de tal manera que según tu punto inicial veas todos los caminos posibles a una siguiente ciudad, después que veas los caminos posibles de una segunda ciudad dados los primeros caminos:

$$\phi = \text{nodo inicial}$$

$$[\phi, \{n\}] = \text{peso de nodo inicial a nodo n}$$

$$[\phi, \{n, m\}] = \text{peso mínimo de nodo inicial a}$$

nodo n pasando por m o viseversa

como se puede observar se trata de conseguir valores mínimos para los casos 1,2,3,...,n. que aunque no parece lo más óptimo simplifica el problema a un algoritmo

$$O(2^{10} * 10^2)$$

a diferencia de buscar todos los casos que difiere a un problema

$$O(n!)$$

un servidor no pudo implementar de manera eficiente el algoritmo de Help-Karp, pero tampoco quiso simplemente implementar un algoritmo a fuerza bruta (en este caso recursivo), así que mezclo un poco las ideas para tener una solución que pudiera competir contra el algoritmo puramente dinámico.

- *Unidad Interdisciplinaria de Ingenierías campus Zacatecas*
- *Instituto Politécnico Nacional*

```
run:
Camino es: 4 -> 8 -> 5 -> 0 -> 2 -> 1 -> 6 -> 3 -> 7 -> 9 -> 4 Peso de : 225
BUILD SUCCESSFUL (total time: 1 second)
```

Figura 1. Camino iniciado en 4, donde vemos que tarda 1 segundo en ejecución

```
run:
Camino es: 0 -> 2 -> 1 -> 6 -> 3 -> 7 -> 9 -> 4 -> 8 -> 5 -> 0 Peso de : 225
Camino es: 1 -> 6 -> 3 -> 7 -> 9 -> 4 -> 8 -> 5 -> 0 -> 2 -> 1 Peso de : 225
Camino es: 2 -> 1 -> 6 -> 3 -> 7 -> 9 -> 4 -> 8 -> 5 -> 0 -> 2 Peso de : 225
Camino es: 3 -> 7 -> 9 -> 4 -> 8 -> 5 -> 0 -> 2 -> 1 -> 6 -> 3 Peso de : 225
Camino es: 4 -> 8 -> 5 -> 0 -> 2 -> 1 -> 6 -> 3 -> 7 -> 9 -> 4 Peso de : 225
Camino es: 5 -> 0 -> 2 -> 1 -> 6 -> 3 -> 7 -> 9 -> 4 -> 8 -> 5 Peso de : 225
Camino es: 6 -> 3 -> 7 -> 9 -> 4 -> 8 -> 5 -> 0 -> 2 -> 1 -> 6 Peso de : 225
Camino es: 7 -> 9 -> 4 -> 8 -> 5 -> 0 -> 2 -> 1 -> 6 -> 3 -> 7 Peso de : 225
Camino es: 8 -> 5 -> 0 -> 2 -> 1 -> 6 -> 3 -> 7 -> 9 -> 4 -> 8 Peso de : 225
Camino es: 9 -> 4 -> 8 -> 5 -> 0 -> 2 -> 1 -> 6 -> 3 -> 7 -> 9 Peso de : 225
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 2. Iteración entre caminos, donde vemos que no tarda ni 1 segundo en ejecución

4. Resultados

buscando cuanto recorrí en la ruta más corta de manera recursiva y asignando el camino de forma dinámica encontré una particularidad un poco extraña en el algoritmo a pseudo-fuerza bruta: de complejidad: $O(n!)$ bajo a $O((n-1)! * e)$ La e se genera al asignar la ruta nueva (ya encontrado el camino), y $(n-1)!$ se genera al asignar un punto de inicio fijo, independiente de que camino quiero tomar, como ya se qué camino es el más corto y este tiene conectada a todas las ciudades, puedo recorrer esa solución para empezar desde la ciudad deseada.

github: https://github.com/Memotets/AdA/tree/master/src/Tsp.

5. Conclusiones

El uso de distintas soluciones para un mismo problema puede afectar de manera drástica el tiempo de resolución del mismo, en mi caso al mezclar una idea óptima con una idea poco óptima resulto en una solucion mejor que la peor, pero que esta muy cerca de la misma, dado que lo que hace que mis tiempos sean razonables es la propia matriz de ciudades y que como guardo mis resultados solo mando a llamar el método que genera caminos cortos 1 vez y no n. lo que se asemeja más a las prácticas de programación dinámica.