Laboratorio 2: Backtracking

Javier Arredondo Contreras

Universidad de Santiago de Chile
javier.arredondo.c@usach.cl

16 de diciembre de 2018

Resumen

El presente artículo muestra la implementación de un algoritmo presentado por el cuerpo docente de la asignatura Algoritmos Avanzados de la Universidad de Santiago de Chile. Para dicho algoritmo se aplican métodos y técnicas basadas en Backtracking para la optimización de la problemática. El objetivo del estudio de este documento se basa principalmente en analizar los resultados obtenidos para luego profundizar en la eficacia y eficiencia de la solución propuesta.

I. Introducción

La asignatura de Algoritmos Avanzados, impartido por el Departamento de Ingeniería Informática tiene como objetivo que los estudiantes sean capaces de analizar en profundidad distintos tipos de algoritmos. Dentro de esta segunda experiencia se busca aplicar el diseño algorítmico de soluciones que brinda *Backtracking*; el cual dice que se deben buscar soluciones de acuerdo a ciertas restricciones; el término en español viene siendo traducido como vuelta atrás, dicha traducción es adecuada en la implementación del algoritmo, ya que a partir de ciertas condiciones el algoritmo retrocede en las posibles soluciones.

En base a este documento se ha presentado un problema por parte del cuerpo docente; el cual es brevemente explicado a lo largo del documento.

i. Objetivo principal

- Desarrollar una algoritmo que sea capaz de resolver el problema planteado, a través del uso de fuerza bruta.

ii. Objetivos secundarios

- Obtener la ruta óptima para el rey del reinado de *Clover*.
- Responder a las preguntas de análisis de un algoritmo: ¿Se detiene?, cuando se detiene ¿entrega una solución correcta?, ¿es eficiente?, ¿se puede mejorar? y ¿existe otro método?.
- Para obtener la eficiencia de la solución, se debe estimar el orden de complejidad del algoritmo desarrollado utilizando la notación $O()$.

iii. Estructura del documento

En primer lugar se presenta la descripción y contexto del problema, para luego exponer un breve marco teórico que ayuda a comprende los conceptos claves del documento. Posteriormente se realiza un análisis para el algoritmo, respondiendo las preguntas planteadas para así estudiar la eficiencia de dicho algoritmo.

II. Descripción del problema

El reino de Clover dispone de una gran cantidad de minerales y recursos dentro de sus fronteras, esto ha provocado que otros reinos sientan envidia de la situación actual del reino Clover. Es importante mencionar, que todos los recursos de dicho reino son llevados a la ciudad principal para el abastecimiento de sus súbditos.

A pesar de la buena situación del reino, éste posee solo un encargado para acarrear los recursos a la capital, es por esto que el rey de Clover, aprovechándose de esta situación quiere determinar la ruta más económica, cuyo camino pase por todas las ciudades del reino, con el fin de que la paga del encargado sea del menor costo posible.

III. Marco teórico

A continuación se describen conceptos utilizados a lo largo del documento:

- Grafo: Es una estructura de datos, la cual compone un conjunto de datos mediante esquemas gráficos. Dicha estructura, tiene la peculiaridad de relacionar los elementos del conjunto, con el fin de poder investigar y estudiar las interrelaciones del conjunto. Cada elemento del grafo se denomina vértice y la relación entre vértices se llama arista.
- Árbol: Es un tipo de grafo, donde cualquier vértice del árbol tiene solamente un camino que los comunica.
- Algoritmo: Cierta cantidad de instrucciones que permiten encontrar la solución a alguna problemática.
- Backtracking: Es un método de resolución de problemas que se basa en la filosofía de buscar con retroceso, este consiste en resolver problemas que deben satisfacer determinadas restricciones. A partir de estas restricciones se va desarrollando un cierto árbol en profundidad, en cada vértice se ramifica (se crea un hijo) de acuerdo a las condiciones de la solución, si la ramificación no cumple los requisitos de la solución, se vuelve al vértice padre, para así realizar otra ramificación.
- Eficiencia de un algoritmo: Serie de propiedades de un algoritmo, que están relacionados con el tiempo de ejecución y la utilización de recursos que este haga.

4		
1	2	5
1	3	7
1	4	8
2	3	10
2	4	2
3	4	9

Cuadro 1: *Ejemplo de archivo de entrada.*

IV. Descripción de la solución

Idea: Dado el mapa del reinado, el cual se define como se muestra en el Cuadro 1, donde la primera línea representa las ciudades a analizar, el resto de líneas son los valores que el encargado cobra por viajar (ciudad origen, ciudad destino y costo monetario). Esta entrada se representa en un grafo, donde cada nodo es una ciudad y las aristas son las conexiones y distancias entre ellas. Luego, a partir de la capital, se comienzan a generar posibles caminos a cada ciudad adyacente, a partir del camino generado se verifica si cumple con las condiciones del conjunto solución, acá se tienen tres casos posibles:

1. Si se cumplen con las condiciones del conjunto solución se sigue analizando el camino generado, donde a partir del último nodo visitado, se prosigue en analizar los nodos adyacentes no visitados.

2. Si no se cumplen con las condiciones del conjunto solución, se descarta el camino generado y se devuelve al nodo padre. A partir del nodo padre se realiza el análisis a los nodos adyacentes no visitados.

3. El conjunto solución contiene todas las ciudades visitadas, por lo cual la próxima ciudad visitada será la capital. A partir del camino generado, se calcula el costo de la ruta y se guarda la solución parcial encontrada. Posteriormente se realiza un retroceso en el camino generado, y se analiza otras posibles rutas a partir de la última ciudad visitada, es decir, se realiza 1) o 2) nuevamente.

Es importante mencionar, que el conjunto solución son las ciudades visitadas a lo largo del algoritmo, donde se restringe el paso por una ciudad dos veces y la condición de análisis corresponde a la comparación del costo del conjunto solución en ese momento con el costo de la solución parcial. Si el costo del conjunto solución es menor que la solución parcial, se dice que se cumple la condición de análisis.

Para encontrar la solución al problema general, se ha realizado una división en subproblemas, con el fin de solucionar cada uno de estos de forma separada.

I. Lectura del archivo

Para la lectura del archivo se implemento una estructura de datos; un grafo. Para así poder almacenar la información de la entrada en dicha estructura y posteriormente procesarlos.

Operaciones implementadas para el grafo:

- *readGraph()*: Crea una matriz de adyacencia que representa un grafo.
- *showGraph()*: Muestra el grafo en pantalla.

II. Encontrar ruta óptima

Dado que esto aún sigue siendo un problema no abordable, se divide en nuevos subproblemas:

1. **Implementación de estructura *route*:** Se implementa un tipo de dato abstracto, que simula el comportamiento que debe tener una ruta. Con esto se pueden realizar:
   - *add()*: Agrega una ciudad al conjunto de ciudades visitadas.
   - *inRoute()*: Verifica si una ciudad ya se encuentra en la ruta prevista.
   - *getCurrent()*: Obtiene el último nodo agregado a la ruta.
   - *makeCopy()*: Realiza una copia de la ruta.
2. **Análisis del conjunto solución:** Para verificar que el conjunto solución en cuestión, cumple con las condiciones para ser una solución posible, se verifica a través de las siguientes funciones:
   - *isSolution()*: Verifica que el conjunto solución actual haya pasado por todas las ciudades del reino.
   - *isPosible()*: Verifica que a partir de una ciudad de origen, se pueda realizar el viaje a una ciudad de destino. Para esto se verifica que el origen esté en el conjunto solución y que el destino no esté en dicho conjunto.
3. **Ruta óptima con *Backtracking*:** En la búsqueda de la ruta óptima se utiliza un caso base, el cual es verificado a partir de la función *isSolution()*, si la ruta es solución y tiene menor costo que la ruta óptima parcial se guarda la ruta encontrada. Si la ruta analizada no es solución; se verifica para cada nodo adyacente si es posible ir, con la función *isPosible()*, de ser posible se agrega al conjunto solución y se realiza *backtracking* con el nueva ruta. Éste proceso recursivo se realiza hasta encontrar la solución óptima.

V. Análisis de la solución y resultados

1. **¿El algoritmo se detiene?**

   El algoritmo efectivamente se detiene, ya que al generar las soluciones que se pueden realizar a partir de las restricciones propuestas, el algoritmo se detiene cuando no tiene más posibles rutas que verificar.

2. **¿Resuelve el problema?**

   Se resuelve el problema con éxito, ya que se logra encontrar la ruta óptima que el encargado debe seguir.

3. **¿Es eficiente?**

   Viendo el Cuadro 2, se puede observar que el algoritmo no es sumamente eficiente, ya que su complejidad es exponencial $O(n^n)$. Esto significa que a un mayor número de ciudades, más tiempo demorará en ejecutarse el programa. A pesar de este orden de complejidad, *backtracking* tiene como ventaja que puede desechar ramificaciones

Función	Orden de complejidad
readGraph()	$O(n)$
showGraph()	$O(n^2)$
add()	$O(1)$
inRoute()	$O(n)$
getCurrent()	$O(1)$
makeCopy()	$O(n)$
isSolution()	$O(1)$
isPosible()	$O(1)$
backtracking()	$O(n^n)$

Cuadro 2: *Complejidad por funcionalidad*

enteras de posibles soluciones que no cumplen con ciertas restricciones, pero a pesar de esto no es el algoritmo más eficiente para enfrentar esta problemática.

4. **¿Se puede mejorar?**

   El algoritmo contó con todas las posibles mejoras, con la finalidad de generar la menor cantidad de soluciones ineficientes. Por lo cual el algoritmo no cuenta con posibles mejoras.

5. **¿Existen otros métodos?**

   Claramente existen otros métodos, como por ejemplo resolverlo como en la experiencia pasada con Fuerza Bruta u otros como metodo *Greedy*.

VI. Traza

Dado el archivo de entrada de Cuadro 3, se procede a aplicar el algoritmo desarrollado. Para una mayor comprensión se recomienda observar la Figura 1. Desde la capital se traza quienes son las ciudades a las que se pueden ir; entonces para cada ciudad visitada se aplica la misma lógica. La secuencia Capital-1-2-3-Capital, es la ruta óptima. Al comparar la ruta óptima con las otras posibles secuencias, se van descartando, por lo cual no se crean más ramificaciones posibles.

3		
1	2	1
1	3	5
2	3	1

Cuadro 3: *Entrada para la traza.*

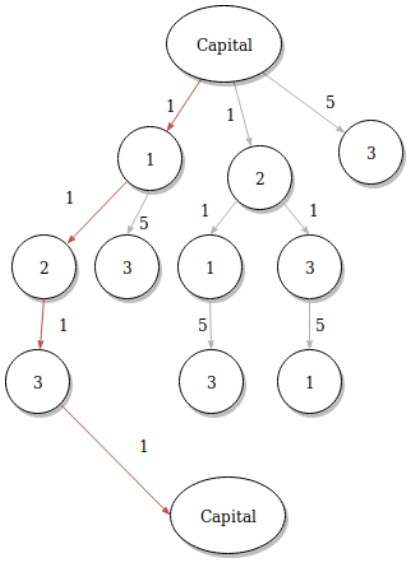

Figura 1: *Traza de ejemplo.*

VII. Conclusiones

Recordando la complejidad del algoritmo generado, $O(n^n)$ se puede decir que el algoritmo no es eficiente, dado que no es polinómico. Este orden tiene como desventaja que a mayor número de ciudades, el orden crecerá exponencialmente. Si comparamos el algoritmo con el de la experiencia pasada, se puede concluir que de cierto modo *backtracking* tiene mejor rendimiento, ya que Fuerza Bruta genera todas las posibles soluciones y finalemente escoge la adecuada. A partir de ésta premisa se puede decir que *backtracking* es una mejor considerable de Fuerza Bruta, ya que a partir de las ramificaciones creadas, es capaz de desechar posibles combinaciones que Fuerza Bruta genera.

Referencias

[M. Villanueva, 2002] Algoritmos: Teoría y Aplicaciones