Universidad de Santiago de Chile
Departamento de ingeniería informática
Ingeniería civil informática
Algoritmos Avanzados
Laboratorio

2-2018

Resolución de un problema de obtención de recursos mediante enumeración explícita

Kevin Arévalo Fernández
kevin.arevalo@usach.cl
www.github.com/arefer

RESUMEN: *En este documento se aborda una modificación del problema del comerciante viajero, que trata, de encontrar la ruta más óptima entre un conjunto de puntos o nodos. El problema fue resuelto mediante enumeración explícita -fuerza bruta- y se concluye que no es la mejor forma de resolverlo.*

PALABRAS CLAVE: enumeración explícita, fuerza bruta, grafo.

ABSTRACT: *This document deals with a modification of the traveling merchant problem, which tries to find the most optimal route among a set of points or nodes. The problem was solved by explicit enumeration - brute force - and it is concluded that it is not the best way to solve it.*

KEYWORDS: *explicit enumeration, brute force, graph.*

1 INTRODUCCIÓN

Este paper se desarrolla en el marco del curso de Algoritmos Avanzados de la carrera de Ingeniería Civil en Informática. Corresponde al desarrollo del Laboratorio número uno de dicho curso y se espera aplicar y comprender las ventajas y desventajas de la utilización de la técnica de enumeración explícita o, también llamada comúnmente fuerza bruta, mediante la resolución de un problema planteado. El objetivo es ampliar la gama de estrategias para generar algoritmos mediante la generalización de las conclusiones con respecto a este método para cualquier problema general.

2 DESCRIPCIÓN DEL PROBLEMA

Dada una ciudad capital, y un conjunto de puntos de recolección de recursos, que están todos conectados entre sí (y también existe un camino entre la capital y cada uno de ellos). El problema consiste en encontrar la ruta más corta que parta desde la ciudad capital, pase por todos los puntos de recolección y vuelva a la capital de forma que se genere el menor costo posible, sabiendo que el camino entre un punto y otro tiene un costo conocido (y no son todos necesariamente iguales) y que se debe pasar por todos y cada uno de ellos exactamente una vez. Además se sabe que el costo de ir desde cualquier punto de recolección a la ciudad es constante e igual 1.

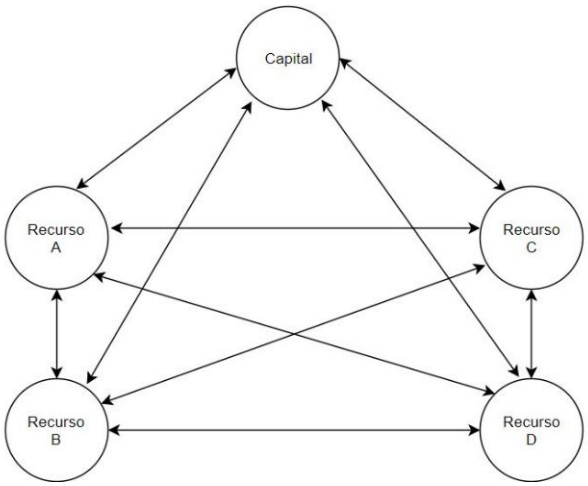

Figura 1: Representación de la ciudad y los puntos de recolección de recursos.

3 MARCO TEÓRICO

Enumeración explícita: Consiste en una técnica de resolución de problemas que se basa en generar todas las posibles soluciones, para luego filtrar aquellas que son válidas, de acuerdo a las características del problema, y finalmente escoger la mejor de ellas según un criterio en particular.

Grafo: Es una estructura algebraica que representa un conjunto de vértices o nodos y sus relaciones (aristas). Se denota por G(V,A). Adicionalmente estas aristas pueden tener algún tipo de característica de la relación.

Orden lexicográfico: Dados dos conjuntos a y b, decimos que a < b si ambas secuencias tienen un prefijo común de largo *i*, y el primer carácter diferente es $a_{i+1} < b_{i+1}$ (Ej con cadenas de caracteres: **merc**ed < **merc**urio, porque e < u). Además el largo *i* puede ser igual a cero.

4 DESCRIPCIÓN DE LA SOLUCIÓN

Para resolver el problema, se programa un algoritmo en el lenguaje C, abstrayendo la situación a un grafo. El algoritmo resuelve instancias generales del problema y es ejecutable en los sistemas operativos más usados (Mac, Windows, Linux).

4.1 IDEA

La idea a implementar es construir una lista con todos los puntos de recolección, y luego generar todas las permutaciones posibles de ella, en orden lexicográfico (que representan a un camino a seguir), pero además agregando un 0 al principio y al final, para indicar que se parte desde la capital y también se llega a ella. Mientras se vayan generando estas permutaciones, calcular el costo del camino representado e ir almacenando el mejor camino y su costo, para entregarlo como solución una vez revisadas todas las opciones.

4.2 REPRESENTACIÓN Y ESTRUCTURA

Para representar la situación, se utiliza un grafo de la forma:

$$G = \{(A_1,\ B_1,\ C_1\ ), ..., (A_n,\ B_n,\ C_n)\}$$

En donde A_i , B_i son puntos de recolección, y C_i es el costo de ir desde uno de ellos hacia el otro.

Un camino es representado por un arreglo de la forma:

$$C = \{X_1,\ X_2,\ \dots\ ,\ X_n\}$$

En donde los X_i son puntos de recolección. Además un camino tiene la particularidad de que su dimensión es $n+2$, siendo n el número de puntos de recolección, y $X_1 = X_n = 0$ porque todo camino comienza desde la capital y llega a la capital (representada con 0).

4.3 PSEUDOCÓDIGO

```
BruteForce(nodos, numNodos, grafo,
numAristas): void
   mejorCamino = nodos
   mejorCosto = costoTotal(nodos)
   permutacion = 1

   mientras permutacion == 1:
```

```
    si permutar(nodos):
       costo = costoTotal(nodos)
       si costo < mejorCosto:
          mejorCamino = nodos
          mejorCosto = costo
    sino:
       permutacion = 0

mostrar(mejorCamino)
```

la función permutar, retorna la próxima permutación del conjunto pasado como parámetro, en orden lexicográfico, siempre que sea posible.
Los parámetros *numNodos , grafo, numAristas* son utilizados por la función *costoTotal*, pero no se muestra expllícitamente en el pseudocódigo por simplicidad.

4.4 TRAZA

Entrada = g = {(1,2,5),
(1,3,7),(1,4,8),(2,3,10),(2,4,2),(3,4,9)}

numAristas = 6
nodos = {1,2,3,4}
numNodos = 4

bruteForce(nodos, numNodos, g, numAristas)
mejorCamino = nodos
mejorCosto = costo({1,2,3,4}) = 26
permutacion = 1

** Entra al ciclo **
primera permutación = nodos=
{1,2,4,3}
costo = costo({1,2,4,3}) = 18
costo < mejorCosto →
mejorCosto = costo
mejorCamino = nodos
.
.
.

Séptima permutación = nodos =
{2,1,4,3}
costo = costo({2,1,4,3}) = 24
costo > mejorCosto

.
.
.
Así hasta la n-ésima permutación
** Sale del ciclo **
mostrar(mejorCamino)

4.5 ORDEN DE COMPLEJIDAD

Como el proceso de permutar tiene una complejidad ~ *O(n)*, y para un conjunto de n nodos, existen $n!$ permutaciones, entonces la complejidad del algoritmo es

$$\sim O(n * n!)$$

5 ANÁLISIS DE LA SOLUCIÓN

5.1 ANÁLISIS DE IMPLEMENTACIÓN

¿Se detiene?
Sí, se detiene cuando termina de realizar todas las permutaciones posibles.
Cuando se detiene ¿Obtiene la solución?
Sí obtiene la solución, porque compara el costo de todos los caminos posibles.
¿Se puede mejorar?
Sí se puede mejorar, quizás cambiando la estructura que representa al grafo, para acceder a los costos de los nodos de forma más eficiente.
Otra idea: Utilizar un método goloso, como el algoritmo de Dijkstra.

5.2 EJECUCIÓN

Para usar el programa, se necesita el archivo de entrada ubicado en la misma carpeta que el código fuente, y con nombre “entrada.in”

Dado un conjunto de **n** nodos o puntos de recolección, en la primera línea del archivo debe indicarse dicho número **n**, seguido de un salto de línea y a continuación $\binom{n}{2}$ relaciones de la forma (A, B, costo), separadas por saltos de línea, de forma que se enlisten todos los costos. Se asume además que las relaciones son conmutativas, y en el archivo deben encontrarse sólo una vez, es decir, si se posee la relación (A, B, C1), no debe encontrarse la relación (B, A, C1). Otro supuesto del problema es que los puntos de recolección deben representarse con números enteros distintos de cero, y que el costo de ir desde la Capital (representada como nodo cero) hasta cualquier punto de recolección es de 1.

La salida se muestra por pantalla y además s e escribe en un archivo con nombre "salida.out" en la misma carpeta del ejecutable.

Compilación y ejecución :

- Para compilar el programa:

```
$ gcc main.c functions.c -o out
```

- para ejecutar el programa:

```
$ ./out
```

o

```
> out.exe
```

Para compilar el programa en modo debugger:

```
$ gcc main.c functions.c -D DEBUG -o out
```

6 CONCLUSIONES

En general se resuelve el problema planteado, obteniendo la solución más óptima, pero a costa de un algoritmo demasiado ineficiente, considerando su orden de complejidad ~ $O(n * n!)$.
Es posible afirmar que la técnica de fuerza bruta o enumeración explícita es ineficiente para cualquier problema de entradas muy grandes.

7 REFERENCIAS

- "Algoritmos: Teorías y aplicaciones" , Mónica Villanueva Ilufi ,2002 , disponible en :http://www.udesantiagovirtual.cl/moodle2/pluginfile.php?file=%2F117134%2Fmod_resource%2Fcontent%2F1%2FAlgoritmos-apuntes.pdf

- "plantilla Paper Lab Avanzados" , Nicolás Gutiérrez , 2018 , disponible en : www.ejemplo.com/ngutierrez/Avanzados
- Orden lexicográfico. (2008, marzo 17). *Enciclopedia,* De la Enciclopedia Libre Universal en Español.. Disponible en http://enciclopedia.us.es/index.php?title=Orden_lexicogr%C3%A1fico&oldid=408436.