Laboratorio de Fuerza Bruta

Cristian Eduardo Espinoza Silva

Universidad de Santiago de Chile
cristian.espinoza.s@usach.cl

19 de noviembre de 2018

Resumen

En el presente informe se muestra el problema y la solución correspondiente al primer laboratorio del curso de Algoritmos Avanzados. La solución se elabora usando la técnica de Enumeración implicita tambien llamada Fuerza Bruta.Se realiza un análisis de los resultados obtenidos, viendo la eficacia y eficiencia de la solución.

I. Introducción

En el presente Paper de la asignatura de algoritmos avanzados recae en la aplicación de uno de los tópicos que se estudia, en esta oportunidad tendremos la aplicación de la resolución de problema basada en "Fuerza Bruta". La resolución de problema mencionada anteriormente la aplicaremos al problema planteado en esta ocasión, el cual consiste en encontrar el camino mínimo para llegar a una a una ciudad, se debe tener en cuenta que se representará gráficamente como un grafo completo, de esto hablaremos en las siguientes secciones del informe. Finalmente, tendremos un análisis que contendrá la eficiencia y eficacia del algoritmo planteado, además de dar a conocer técnicas que ayudarían a mejorar los aspectos mencionados anteriormente, con el fin de concluir finalmente si el algoritmo propuesto es efectivamente eficiente.

I. Objetivo Principal

Desarrollar un algoritmo que, mediante fuerza bruta resuelva el problema planteado, el cual consiste en resolver el problema del vendedor viajero, obteniendo la cantidad mínimo de movimiento entre sus nodos. Los costos de cada uno de los viajes serán entregados mediante un archivo de texto, al igual que la cantidad de ciudades que tendremos. Por otro lado, tendremos que la solución sera entregada mediante otro archivo de texto que contendrá la cantidad de costo recorrido y el camino obtenido para ese costo en especifico.

II. Objetivo Secundario

1. Obtener el camino mínimo para resolver el problema del vendedor viajero.

2. Responder las preguntas principales de un algoritmo, tales como: ¿Se detiene?, ¿Cuando se detiene?, ¿Es eficiente?, ¿Se puede mejorar?, ¿Existen otros métodos?.

3. En el caso de la eficiencia, estimar el orden de complejidad que adopta el algoritmo implementado.

III. Estructura del informe

En el informe se presentan la descripción del problema , luego un marco teórico que define conceptos clave que se utilizaran a lo largo del informe, se procede a describir y analizar la solución obtenida para el problema para finalmente, hacer un análisis de esta respondiendo las preguntas anteriormente planteadas, haciendo énfasis en la eficiencia del algoritmo.

II. Descripción del problema

El Reino Clover es conocido y envidiado por los otros reinos por poseer una gran cantidad de minerales y demás recursos dentro de sus fronteras. Todos estos recursos son llevados a su ciudad principal para el abastecimiento de sus súbditos. Además, desea que lleguen todo sus minerales en el menor tiempo posible a la capital. Por ende, debemos crear un algoritmo utilizando la técnica de Fuerza Bruta que entrega el camino con le costo mínimo para llegar a la ciudad que desea llegar el Reino Clover. Esta solución se entregara en un archivo de texto, donde se indicara el costo que se requiere y el camino a seguir para obtener ese costo mínimo calculado anteriormente.

III. Marco teórico

1. Algoritmo: Secuencia de instrucciones finita que permite encontrar la solución a un determinado problema.
2. Nodo: Es uno de los elementos de un grafo. Cada nodo será una estructura o registro que dispondrá de varios campos, y al menos uno de esos campos será un puntero o referencia a otro nodo, de forma que, conocido un nodo, a partir de esa referencia, será posible en teoría tener acceso a otros nodos de la estructura.
3. Matriz de adyacencia: Es una matriz cuadrada que se utiliza como una forma de representar relaciones binarias. Donde se crea una matriz cero, cuyas columnas y filas representan los nodos del grafo.
4. Eficiencia de un algoritmo: Corresponde a las propiedades del algoritmo que nos permite identificar cuanto tiempo tarda un algoritmo en completarse y cuantos recursos usara. Por lo tanto una mayor eficiencia significara que el algoritmo usara menos recursos y tardara menos tiempo en resolver un problema.
5. Tiempo de ejecución: Expresión algebraica que indica el tiempo en que demora ejecutar un algoritmo, en función de la entrada de este. Las instrucciones mas básicas poseen un tiempo de ejecución constante 1.
6. Orden de complejidad: Corresponde a la estimación del tiempo de ejecución usando la cota superior asintótica de esta.
7. Enumeración implícita: Consiste en una forma de resolución de problemas en la cual se deben encontrar todos los candidatos posibles a solución, y luego dentro de ese conjunto encontrar el subconjunto que cumpla las condiciones dadas.
8. Lenguaje de programación C: Lenguaje de programación imperativo-procedural lanzado en 1972. Es un lenguaje de nivel medio que destaca por el manejo manual de memoria, poder crear tipos de datos compuestos y estructuras, además de las características propias de su paradigma.

IV. Descripción de la solución

La cantidad de nodos ingresados se representaran mediante un árbol, el cual contenderá cada una de las ciudades que se ingresan en el archivo de texto con su respectivo peso de cada una de las arista. Además, se debe tener en cuenta que se debe agregar un nodo extra que no estará indicado en el archivo, el cual va a representar la ciudad inicial y a la vez también representara la ciudad final.

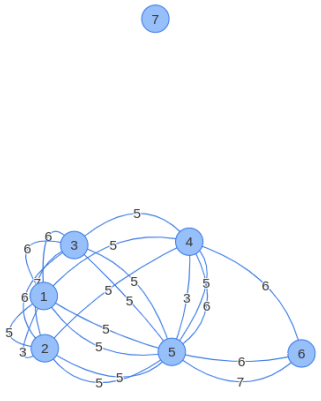

Figura 1: *Representación gráfica en formato de nodos*

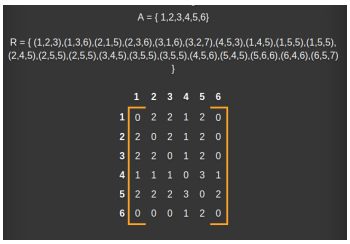

Figura 2: *Representación gráfica en formato de matriz de adyacencia*

1. Idea: Se desea tratar el problema del vendedor viajero mediante enumeración implícita que nos ayudara a generar todas las posibles combinaciones que existan. Para poder darle una representación un poco más gráfica al problema ocuparemos un grafo el cual es fuertemente conexo (Debido a que tenemos todas las ciudades conectadas con todas), esto nos permitirá ver con mayor facilidad cada una de las posibilidades que se tiene para llegar al destino final. Con respecto a los pesos que tendrán cada uno de los viajes a las distintas ciudades estos se almacenaran en una matriz de adyacencia, para así tener cada valor que se demora en llegar de una ciudad a otra, esto para poder resolver lo que nos pide el enunciado que es encontrar el camino menor, así pudiendo al final del problema cuando tenemos la gama entera de caminos generados poder calcular cada uno de sus pesos y dejar solamente el o los caminos que fueron los mínimos.

2. Funciones y procedimientos: A continuación dejaremos en claro cada una de las funciones y algoritmos utilizados para llegar a la solución del problema, tenemos que tener en cuenta que las funciones utilizadas se separan en 5 sub-secciones, las cuales son:

   a) Funciones auxiliares: Funciones que se encargan de realizar subproblemas que se obtienen al ir desarrollando cada uno sucesos que se enfrentan para llegar a la solución final del problema. Dentro de están funciones tenemos como:

      1) PrintCurrent: La cual se encarga de ir dando a conocer en que parte del códigos vamos actualmente, imprimiendo cada uno de los caminos generados y su respectivo costo de viaje.
      2) Factorial: Tenemos que como su nombre lo dice se encarga de calcular el factorial de un numero.
      3) RevisarListaFinal: Se encarga de verificar si el camino generado ya se encuentra en la lista final de caminos generados.
      4) verNumero: Se encarga de verificar si un numero "x", se encuentra en la lista.
      5) reiniciarLista: Se encarga de reiniciar una lista, y asignar a cada uno de sus valores en valor -1.
      6) LiberarMemoria: Función que se encarga de liberar la memoria solicita al comienzo del programa para la matriz con la que se trabaja a lo largo del programa.

   b) Funciones de manejo de archivo: Funciones que se encargar de cargar y escribir los archivos que se ocupan para el comienzo y finalización del programa.

      1) LoadMatriz: Se encarga de crear la matriz que se ocupara a lo largo del programa ,además de asignar a cada una de sus variables los valores obtenidos al momento de leer el archivo de entrada.
      2) saveTablero: Se encarga de crear el archivo de salida, donde se coloca el costo total del camino mínimo que se obtuvo y además de colocar el recorrido de ese camino mínimo mencionado anteriormente.

   c) Función principal: Función que se encarga de crear la enumeración implícita o también conocida como Fuerza Bruta.

d) Función de manejos de matrices: Funciones que se encargar de manejar las matrices que se ocupan dentro del programa, además de una función auxiliar que ayuda a ir viendo la creaciones de cada uno de los valores que va obteniendo dicha matriz.

1) CreateBoard: Función que se encarga de crear la matriz que se utilizara dentro del programa, pidiendo memoria para cada una de los variables que contenderá y además de pedir memoria para la misma estructura que representa a la matriz en cuestión.
2) PrintMatriz: Función que se encarga de mostrar cada uno de los valores que se tiene actualmente, esto nos sirve, para poder ir verificando el funcionamiento de cada una de las funciones que se crean y realizan cambios a la matriz.

e) Función de permutación: Funciones que se encarga de crear cada una de las permutaciones que se tiene con los nodos ingresados en el archivo de texto, esto nos sirve para verificar cada una lo de los caminos posibles que se pueden tener, dentro de esta secciones tenemos las siguientes funciones:

1) Process: Función que se encarga de ingresar cada numero generado en la combinación actual a la lista que contiene la matriz.
2) Sawp: Función que se encarga de realizar un cambio en los valores de las dos variables ingresadas como parámetros.
3) Reverse: Función que se encarga de realizar una inversión en el numero generado en la combinación.
4) B: Función que se encarga de verificar si el numero ingresado como parámetro es par.
5) lexperms: Función que se encarga de generar la combinación completo de números, teniendo en cuenta que genera la combinación de números de atrás hacia adelante.
6) permutación: Función que se encarga de unir a cada una de las funciones que se mencionan anteriormente, donde el termino de esta función entrega la combinación completa de cada uno de los caminos que se puede generar con la cantidad de nodos ingresados.

V. Análisis de la solución y resultados

1. Análisis de la solución

a) ¿El algoritmo se detiene?: Si se detiene, al momento después de generar cada una de los caminos con sus respectivos pesos, y entregar el que contiene le peso mínimo.

b) ¿Resuelve el problema?:Si resuelve el problema, ya que luego de obtener todos los posibles caminos, elige el que tenga el menor peso.

c) ¿Es eficiente?: No, ya que al utilizar fuerza bruta se revisan estados que podrían ser descartados de raíz inmediatamente, haciendo que el orden de complejidad crezca.

d) ¿Se puede mejorar?: Si, cualquier otro método que no sea fuerza bruta sera mas eficiente en encontrar la respuesta.

e) ¿Existe otro método?: Si, por ejemplo usando la misma búsqueda en anchura, pero detener el algoritmo apenas se encuentre la primera solución, pues ya que es búsqueda en anchura, el primero en ser encontrado sera efectivamente el camino mas corto.

2. Complejidad de la solución

   A continuación tendremos una tabla donde se vera reflejado cada uno de los orden que adopta cada una de las funciones implementadas en el programa realizado.

Función	***Orden – O( )***
printCurrent	N²
factorial	n
revisarListaFinal	N²
viewNumber	n
reiniciarList	n
freeMemory	n
loadMatriz	N²
saveTablero	N²
bruteForce	(n!)²
printBoard	N²
createBoard	N²
process	n
swap	1
reverse	n
B	1
lexperms	n
permutación	n

Figura 3: *Tabla que contiene orden de cada una de las funciones*

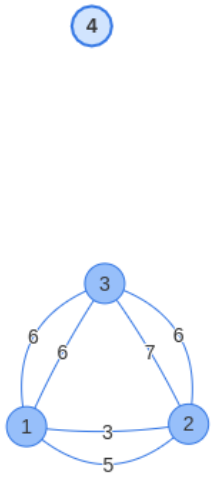

Figura 4: *Representación gráfica en formato de nodos*

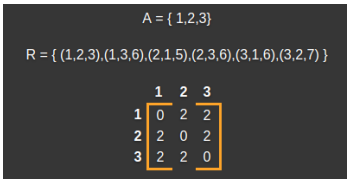

Figura 5: *Representación gráfica en formato de matriz de adyacencia*

VI. Traza de la solución

A continuación realizaremos el paso a paso que va a realizando nuestro algoritmo implementado para poder abordar el problema descrito anteriormente:

1. Paso 1: Primero realizamos la creación de la matriz de adyacencia, la cual nos permite realizar una representación gráfica en formato de nodos, donde lo que se genera se puede ver de la siguiente forma, tener en cuenta que no tenemos el nodos inicial que vendría siendo la ciudad de donde se inicia y de igual forma de donde se tiene que llegar al finalizar el recorrido por las ciudades:

2. Paso 2: Realizamos cada una de las combinaciones posibles que se pueden hacer con los nodos ingresados, a continuación tendremos cada una de los caminos posibles que se pueden generar con los nodos que se esta trabajando actualmente:

   a) 1-2-3

   b) 1-3-2

   c) 2-1-3

   d) 2-3-1

   e) 3-1-2

   f) 3-2-1

   Debemos tener en cuenta que en esta sección consideramos todo los caminos, es decir, los que son iguales por la transición que se tiene en cada uno de los caminos, de igual forma se consideran.

3. Paso 3: Procedemos a calcular cada uno de los pesos que contiene los caminos generados intermitente, además tener en consideración que debemos agregar 2 unidades de peso adicional, donde se considera la ida de la ciudad inicial y el regreso a esa misma ciudad que en ambas direcciones tiene un peso de 1, desde cualquier ciudad:

   a) 1-2-3 = 10 + 2 = 12.

   b) 1-3-2 = 12 + 2 = 14.

   c) 2-1-3 = 11 + 2 = 13.

   d) 2-3-1 = 13 + 2 = 15.

   e) 3-1-2 = 9 + 2 = 11.

   f) 3-2-1 = 11 + 2 = 13.

4. Paso 4: Por ultimo buscamos el camino que da el camino mínimo verificando cada uno de los pesos que se obtuvieron, donde se obtiene el siguiente camino:

   a) 3-1-2 = 9 + 2 = 11.

   Donde por ultimo, se realiza la creación del archivo, donde se ingresa en su primera linea el peso total del camino y a continuación de esa se genera el camino que obtiene ese peso mínimo mencionado anteriormente.

```
1  17
2  5 - 6 - 3 - 2 - 4 - 7 - 1
```

Figura 6: *Formato de salida de la solución*

VII. Conclusión

La solución entregada cumple efectivamente con lo pedido en el enunciado, es decir que este entrega el camino mínimo y su respectivo peso, además de cumplir con el requisito de usar fuerza bruta para encontrar la solución. El algoritmo propuesto responde correctamente a las preguntas de análisis del algoritmo.

En cuanto a la eficiencia de este, el algoritmo tiene un orden de complejidad bastante alto, el cual es el siguiente :

1. Analizamos cada una de las funciones mostradas en la figura 3, llegamos a un análisis que acotando superiormente, el orden de complejidad de la solución propuesta es de:

$$O(n) = (n!)^2 \tag{1}$$

La ecuación planteada anteriormente nos indica lo poco eficiente del algoritmo, ya que tarda bastante en entregar el resultado esperado. Esto se debe a que en primera instancia el algoritmo busca todos los caminos a la solución, es decir todos los caminos posibles que nos lleven a resolver el problema del vendedor viajero con su respectiva restricción que es el camino mínimo, y luego de entre estos se elige el que tome el menor tiempo posible, es decir, el que contiene el peso mínimo.

Una forma de mejorar este algoritmo seria, por ejemplo, detener el algoritmo apenas encuentre la primera solución, ya que con la búsqueda en anchura, la primera solución encontrada sera efectivamente la mas corta.

Por ultimo. podemos observar que fuerza bruta es eficaz en encontrar la solución, pero no es nada eficiente en hacerlo. Sin embargo, muchas veces resulta mas fácil de implementar la solución usando fuerza bruta, además que puede servir en casos en que específicamente se necesite obtener todos los candidatos a solución posibles.

Referencias

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

[Generador de grafos] Implementación de grafos.