**Pontificia Universidad Católica Madre y Maestra**

**Facultad de Ciencias de la ingeniería,**

**Departamento de Ingeniería en Sistemas y Computación**

**Diseño y análisis de Algoritmos**



Trabajo:

T2-Viajante de comercio

Materia:

Diseño y análisis de Algoritmos

ST-ISC-405-T-001

Estudiantes:

Francis Cáceres

2012-0559

Profesor:

Juan R. Núñez

Introducción

El Problema del Agente Viajero (TSP por sus siglas en inglés) o problema del viajante, responde a la siguiente pregunta: Dada una lista de ciudades y las distancias entre cada par de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y regresa a la ciudad origen? Este es un problema NP-duro dentro en la optimización combinatoria, muy importante en la investigación de operaciones y en la ciencia de la computación.

El problema fue formulado por primera vez en 1930 y es uno de los problemas de optimización más estudiados. Es usado como prueba para muchos métodos de optimización. Aunque el problema es computacionalmente complejo, una gran cantidad de heurísticas y métodos exactos son conocidos, de manera que, algunas instancias desde cien hasta miles de ciudades pueden ser resueltas.

Desarrollo

En esta tarea aplicarás uno o más algoritmos para el problema del viajante de comercio, tales como el algoritmo de programación dinámica. Aquí tenemos archivos de datos (escoger uno) de Canada, Argentina, Chile, Birmania en los que se describe una instancia de TSP. La primera línea indica el número de ciudades. Cada ciudad es un punto en el plano, y cada línea subsiguiente indica número de la ciudad i y luego las coordenadas o direcciones x e y de la ciudad i. La distancia entre dos ciudades se define como la distancia euclidiana --- es decir, dos ciudades en ubicaciones (x, y) y (z, w) que la distancia \ /- (x-z)^ 2 + (y-w)^2 - entre ellos. En la respuesta, escriba el costo mínimo de una visita del viajante de comercio para esta instancia, redondeado al número entero más cercano.

No pude colocar las ciudades, pero si pude simular la coloración de vértices que en este caso seria las ciudades. En el método siguiente se obtiene el costo mínimo para visitar las ciudades, ayudado por otros más:

public int COST(int currentNode,int inputSet[],int setSize)  
{  
 if(setSize==0)  
 return weight[currentNode][0];  
 int min=INF,minindex=0;  
 int setToBePassedOnToNextCallOfCOST[]=new int[n-1];  
 for(int i=0;i<setSize;i++)  
 {  
 int k=0;  
 for(int j=0;j<setSize;j++)  
 {  
 if(inputSet[i]!=inputSet[j])  
 setToBePassedOnToNextCallOfCOST[k++]=inputSet[j];  
 }  
 int temp=COST(inputSet[i],setToBePassedOnToNextCallOfCOST,setSize-1);  
 if((weight[currentNode][inputSet[i]]+temp) < min)  
 {  
 min=weight[currentNode][inputSet[i]]+temp;  
 minindex=inputSet[i];  
 }  
 }  
 return min;  
}

Mientras que este construye el tour a seguir:

public void constructTour()  
{  
 int previousSet[]=new int[n-1];  
 int nextSet[]=new int[n-2]; for(int i=1;i<n;i++)  
 previousSet[i-1]=i;  
 int setSize=n-1;  
 tour[0]=MIN(0,previousSet,setSize);  
 for(int i=1;i<n-1;i++)  
 {  
 int k=0;  
 for(int j=0;j<setSize;j++)  
 {  
 if(tour[i-1]!=previousSet[j])  
 nextSet[k++]=previousSet[j];  
 }  
 --setSize;  
 tour[i]=MIN(tour[i-1],nextSet,setSize);  
 for(int j=0;j<setSize;j++)  
 previousSet[j]=nextSet[j];  
 }  
 display();  
}

Para un mejor vistazo al código completo puede visitar este link (El código fuente está basado en Java):

<https://github.com/SuitMonkey/T2/blob/master/src/Main.java>

Aquí está el resultado:

