INSTITUTO POLITECNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERIA CAMPUS ZACATECAS

Alumno:

Mariel López Beltrán

Docente:

Roberto Oswaldo Cruz Leija

Grupo:

3CM1

Asignatura:

Análisis de algoritmos

Tarea:

TSP

Fecha de entrega:

07/11/2019

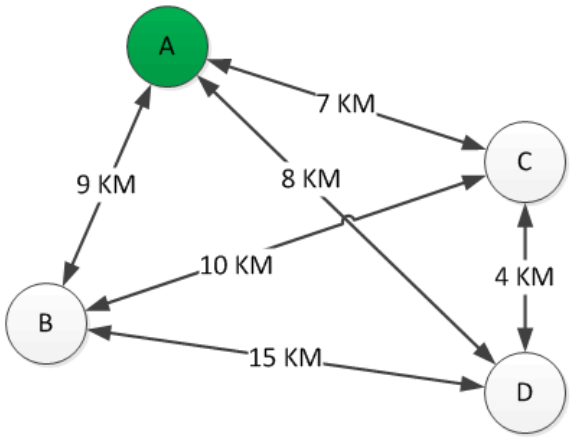
Introducción:

En el presente proyecto se llevara a cabo la programación del TSO o denominado como el problema del viajante o agente viajero cabe destacar que la programación de dicho algoritmo conlleva la aplicación del algoritmo de Dijkstra el cual nos permite calcular las distancias de las ciudades en las cuales se determinara cuál de ellas es la ruta más corta para que en este caso nuestro viajero logre recorrer todas las ciudades ingresadas con el camino más corto

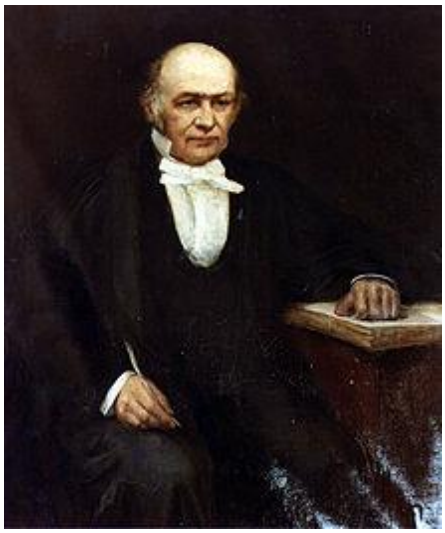
Cabe destacar que es posible que no se pueda llevar acabo al 100% la programación de dicho programa debido a la complejidad que tiene para poder programarlo

Marco teórico:

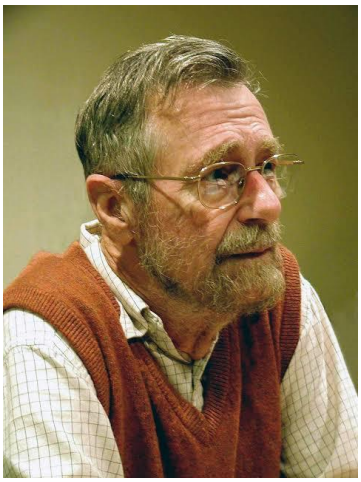
El problema del vendedor viajero, problema del vendedor ambulante, problema del agente viajero o problema del viajante (*TSP* por sus siglas en inglés (Travelling Salesman Problem)), responde a la siguiente pregunta: dada una lista de ciudades y las distancias entre cada par de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen? Este es un problema NP-Hard 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.



El TSP tiene diversas aplicaciones aún en su formulación más simple, tales como: la planificación, la logística y en la fabricación de circuitos electrónicos. Un poco modificado, aparece como: un sub-problema en muchas áreas, como en la secuencia de ADN. En esta aplicación, el concepto de “ciudad” representa, por ejemplo: clientes, puntos de soldadura o fragmentos de ADN y el concepto de “distancia” representa el tiempo de viaje o costo, o una medida de similitud entre los fragmentos de ADN. En muchas aplicaciones, restricciones adicionales como el límite de recurso o las ventanas de tiempo hacen el problema considerablemente difícil. El TSP es un caso especial de los Problemas del Comprador Viajante (travelling purchaser problem).

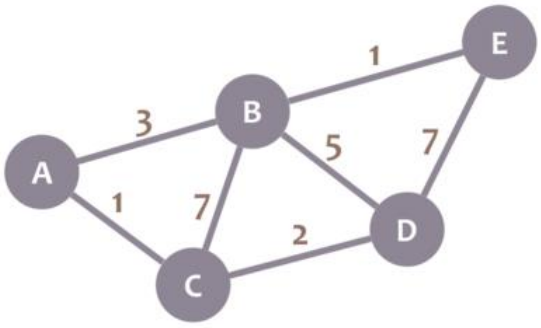


Algoritmo de Dijkstra. También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

Características del algoritmo:

- Es un algoritmo greddy.
- Trabaja por etapas, y toma en cada etapa la mejor solución sin considerar consecuencias futuras.
- El óptimo encontrado en una etapa puede modificarse posteriormente si surge una solución mejor

El algoritmo de Dijkstra te permite calcular la ruta más corta entre un nodo (tú eliges) y **todos los demás nodos en el grafo**. Encontrarás una descripción del algoritmo al final de esta página, pero ¡vamos a estudiarlo con un ejemplo explicado! Calculamos la distancia más corta entre el nodo C y los demás nodos del grafo:



Pruebas de ejecución:

```
        TSP f = new TSP("abcdef");
         f.agregarRutas('a','b', 3);
         f.agregarRutas('a','e', 6);
         f.agregarRutas('a','f',10);

         f.agregarRutas('b','c', 5);
         f.agregarRutas('b','e', 2);
         f.agregarRutas('c','d', 8);

         f.agregarRutas('c','e', 9);
         f.agregarRutas('c','f', 7);
         f.agregarRutas('d','f', 4);



         String resp = f.encontrarRuta('a', 'd');
         System.out.println(resp);
    }
}
```

```
utput - TSP (run)
run:
14: a f d
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
    public static void main(String[] args) {
        TSP f = new TSP("abcdefg");
         f.agregarRutas('a','b', 3);
         f.agregarRutas('a','c', 9);
         f.agregarRutas('b','c',2);

         f.agregarRutas('b','d', 7);
         f.agregarRutas('b','e', 1);
         f.agregarRutas('c','d', 7);

         f.agregarRutas('c','e', 1);
         f.agregarRutas('e','d', 5);
         f.agregarRutas('e','f', 9);

         f.agregarRutas('d', 'f', 2);
         f.agregarRutas('d', 'g', 8);
         f.agregarRutas('f', 'g', 4);


         String resp = f.encontrarRuta('a', 'g');
         System.out.println(resp);
    }
}
```

Output - TSP (run)

```
run:
15: a b e d f g
BUILD SUCCESSFUL (total time: 0 seconds)
```

Conclusiones:

Las pruebas de ejecución que se presentan en la parte superior nos dan un breve indicio acerca de cómo trabaja dicho algoritmo, en este caso el algoritmo pide el nodo en que se desea iniciar y el nodo en el que se desea terminar, cabe destacar que se necesita ingresar las ciudades en el presente caso que se requieren es decir las ciudades que se conectan además de ingresar el valor que estas tienen, en palabras simples esto lo podríamos definir que se requiere ingresar un grafo tal como el algoritmo de Dijistra requiere para llevar a cabo la solución de las ciudades

De acuerdo al problema que se decidió llevar a cabo podemos aclarar que en su totalidad no se logró llevar a cabo la resolución del problema del agente viajero debido a la complejidad que este conllevaba es decir como venía especificado, sin embargo, lo que se logró hacer es la programación del método de Dijistra que como fue visto en el marco teórico es necesario para llevar a cabo la resolución de dicho problema.