UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INTELIGENCIA ARTIFICIAL

PROYECTO 1

FECHA DE ENTREGA: 12 DE NOVIEMBRE DE 2020

SEMESTRE 2021-1

ALUMNOS:

Amezaga Campos Salvador
Avendaño Cabanillas Gustavo Eduardo
Carranza Escobar Luis Enrique
Guzmán Martínez Lizeth Yoseline

Índice general

Proyecto 1: Problema del Agente Viajero (TSP)

1.1. Descripción del problema

Se tiene un número de nodos (ciudades, localidades, tiendas, empresas, etc.) que deben ser visitados por una entidad (persona, agente viajero, automotor, avión, autobús, etc.), sin visitar 2 veces el mismo nodo. Si tenemos 3 nodos (a, b y c) por visitar, entonces tendríamos una función de combinaciones sin repetición c(3,2), es decir, tendríamos 6 posibles soluciones: abc, acb, bac, bca, cab, cba, para el caso de 4 nodos tendríamos 12 combinaciones, para 10 nodos tendríamos 90 combinaciones, para 100 ciudades tendríamos 9,900 combinaciones y así sucesivamente.
Como ejemplo en el problema del Ulises de Homero que intenta visitar las ciudades descritas en la Odisea exactamente una vez (16 ciudades) donde existen múltiples conexiones entre las diferentes ciudades, Grötschel y Padberg (1993) llegó a la conclusión de que existen 653,837'184,000 rutas distintas para la solución de este problema.

1.2. Características del TSP

TSP se encuentra clasificado como Problema de optimización Combinatoria, es decir, es un problema donde intervienen cierto número de variables donde cada variable puede tener N diferentes valores y cuyo número de combinaciones es de carácter exponencial, lo que da lugar a múltiples soluciones óptimas (soluciones que se calculan en un tiempo finito) para una instancia.
TSP es un problema considerado difícil de resolver, denominándose en lenguaje computacional NP-Completo, es decir, es un problema para el que no podemos garantizar que se encontrará la mejor solución en un tiempo de cómputo razonable. Para dar solución se emplean diferentes métodos, entre los cuales, los principales se denominan heurísticas cuyo objetivo es generar soluciones de buena calidad en tiempos de cómputo mucho más pequeños (soluciones óptimas tiempo – respuesta).

1.2.1. Usos del TSP

1. Tiempo de recorrido entre ciudades: horas, minutos, días, semanas, etc.
2. Distancia de recorrido entre ciudades: metros, kilómetros, millas, milímetros, etc.
3. Costo de traslado: dinero, desgaste de las piezas, gasto de energía, etc.
4. Las variables que se pueden adoptar dependen de cada problema, por ejemplo:
   - *a*) Circuitos electrónicos: cantidad de soldadura utilizada, menor espacio entre los puntos de soldadura de los circuitos, evitar el cruce entre las líneas de soldadura, tiempo de fabricación, distribución de los circuitos, entre otras.
   - *b*) Control de semáforos: Número de semáforos (nodos), tiempo de traslado entre semáforos, cantidad de autos que pasan por un punto, entre otras variables.
   - *c*) Previsión del tránsito terrestre: puntos en una ciudad, cantidad de vehículos, tiempo de traslado, tipos de vehículos, horas pico, correlación entre variables, regresión lineal, etc.
   - *d*) Entrega de productos: Peso de las entregas, número de entregas, nodos (domicilios) a visitar, recorridos, tiempos de traslado, tipo de vehículo, etc.
   - *e*) Estaciones de trabajo: secuencia de actividades, lugar de las herramientas (nodos), Tipo de herramientas, tiempo de uso, etc.
   - *f*) Edificación: puntos de edificación (construcciones), distancia entre las construcciones y los insumos, vehículos (grúas, camiones de volteo, etc.), cantidad de combustible que emplean, etc.
   - *g*) Entre otras variables

1.3. Descripción de la solución

Algoritmos propuestos para dar solución a TSP
El Problema del Agente Viajero puede resolverse de diferentes maneras:

- Enumeración de todas las soluciones factibles. Es decir, enlistar todas las posibles soluciones al problema, calcular sus costos asociados, e identificar, por comparación, cuál es la solución con el costo más conveniente.
- Métodos exactos. También llamados algoritmos óptimos, intentan descartar familias enteras de posibles soluciones, tratando así de acelerar la búsqueda y llegar a una óptima. Los que más se usan para resolver el TSP son Ramificación y Acotamiento, y Ramificación y Corte.
- Heurísticas. Son métodos obtienen buenas soluciones en tiempos de cómputo muy cortos, aunque sin garantizar la solución única.

1.3.1. Búsqueda Voraz (Greedy Search)

El Algoritmo escoge en cada paso al mejor elemento posible, conocido como el elemento más prometedor. Se elimina ese elemento del conjunto de candidatos ( ) y, acto seguido, comprueba si la inclusión de este elemento en el conjunto de elementos seleccionados produce una solución factible.
En caso de que así sea, se incluye ese elemento en S. Si la inclusión no fuera factible, se descarta el elemento. Iteramos el bucle, comprobando si el conjunto de seleccionados es una solución y, si no es así, pasando al siguiente elemento del conjunto de candidatos.
Una vez finalizado el bucle, el algoritmo comprueba si el conjunto S es una solución o no, devolviendo el resultado apropiado en cada caso.

1.3.2. Caractéristicas del algoritmo:

- El término Voraz (Greedy) ó Avaro es porque en cada paso trata de situarse tan cerca del objetivo como pueda, seleccionando el nodo con menor función de evaluación f(n)
- Esta función debe de informar sobre la calidad del estado:
    1. Coste de alcanzar la solución desde el estado actual
    2. Independiente del coste del camino hasta el estado actual
- Utiliza funciones heurísticas del estado
- Expande el nodo más cercano al objetivo, asumiendo que probablemente conduzca más rápidamente a la solución.

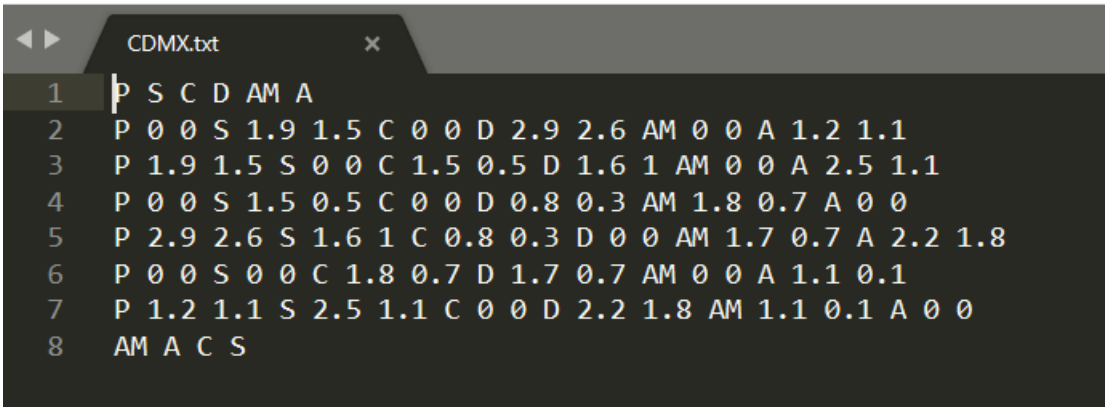
- Al igual que los otros métodos estudiados es necesario verificar los “callejones sin salidas” (no expandir estados repetidos)
- Las funciones heurísticas capturan el conocimiento adicional que se tiene del problema.
- No necesariamente brinda la solución óptima

1.3.3. Pseudocódigo

1. Si n0 es meta, fin con éxito, devolviendo n0
2. Abiertos ← (n0); Cerrados ← ( )
3. Si Abiertos = ( ), fin devolviendo fallo
4. n ← primer elemento de Abiertos; eliminar n de Abiertos y llevarlo a Cerrados; Suc ← ( )
5. expandir n, colocando sus hijos en Suc, como hijos de n
6. Si alguno de los hijos de n es un nodo meta, fin con éxito, devolviendo el camino
7. eliminar de Suc cualquier nodo cuyo estado ya esté asociado a algún nodo de Abiertos o Cerrados
8. colocar los nodos de Suc en Abiertos
9. Reordenar Abiertos según valores crecientes de h(n)
10. Ir a 3

1.3.4. Codificación

Ocuparemos el lenguaje Java y el IDE Netbeans para el desarrollo del proyecto. Requeriremos un archivo en formato txt donde indicaremos las ciudades, pesos y heurísticas entre ellos, y al final de la linea la ruta que deseemos completar.
ejemplo:



Aprovechamos el paradigma de la programación orientada a objetos y tendremos 6 clases

1. El lector de texto: Esta clase tendrá los metodos para leer el archivo linea por linea y así obtener la información para solucionar el problema.
2. Matriz: A partir del archivo leidos introduciremos los valores especificados a sus respectivas matrices de pesos y heurísticas.
3. Arbol: Desde las matrices de datos generamos los nodos padres e hijos para reconocer las distintas rutas y nodos.
4. Nodos: Donde reconoceremos y manejaremos los distintos nodos que creemos.
5. Clase Main: Donde instanciaremos cada clase y finalmente mediante un main imprimiremos los distintos resultados.

1.4. Experimentos

Experimento 1

```
P S C D AM A
P 0 0 S 1.9 1.5 C 0 0 D 2.9 2.6 AM 0 0 A 1.2 1.1
P 1.9 1.5 S 0 0 C 1.5 0.5 D 1.6 1 AM 0 0 A 2.5 1.1
P 0 0 S 1.5 0.5 C 0 0 D 0.8 0.3 AM 1.8 0.7 A 0 0
P 2.9 2.6 S 1.6 1 C 0.8 0.3 D 0 0 AM 1.7 0.7 A 2.2 1.8
P 0 0 S 0 0 C 1.8 0.7 D 1.7 0.7 AM 0 0 A 1.1 0.1
P 1.2 1.1 S 2.5 1.1 C 0 0 D 2.2 1.8 AM 1.1 0.1 A 0 0
S P D
```

```
run:
El archivo tiene 8 líneas.
El número de ciudades es de: 6
La ciudad de inicio es la número: 1
La ciudad de inicio es: S

Peso del hijo: 1.9
Heu del hijo: 1.5
Ciudad hija: P


Peso del hijo: 1.5
Heu del hijo: 0.5
Ciudad hija: C


Peso del hijo: 1.6
Heu del hijo: 1.0


Peso del hijo: 2.5
Heu del hijo: 1.1

--------------------------------------------------
Ciudad: 2
Peso del hijo: 0.8
Heu del hijo: 0.3
Ciudad hija: D


Peso del hijo: 1.8
Heu del hijo: 0.7


Ciudad Nodo: D
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 4
--------------------------------------------------
------------------------------------------------
Ciudad: 3
Peso del hijo: 2.9
Heu del hijo: 2.6
Ciudad hija: P


Peso del hijo: 1.6
Heu del hijo: 1.0
Ciudad hija: S


Peso del hijo: 1.7
Heu del hijo: 0.7
Ciudad hija: AM


Peso del hijo: 2.2
Heu del hijo: 1.8


Ciudad Nodo: AM
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
------------------------------------------------
```

```
-----------------------------------------------
Ciudad: 4
Peso del hijo: 1.8
Heu del hijo: 0.7
Ciudad hija: C


Peso del hijo: 1.1
Heu del hijo: 0.1
Ciudad hija: A


Ciudad Nodo: A
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 3
-----------------------------------------------
-----------------------------------------------------
Ciudad: 5
Peso del hijo: 1.2
Heu del hijo: 1.1
Ciudad hija: P


Peso del hijo: 2.5
Heu del hijo: 1.1


Peso del hijo: 2.2
Heu del hijo: 1.8


Ciudad Nodo: P
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
-----------------------------------------------------
Ciudades visitadas:
S C D AM A P
El peso total es de: 9.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
El archivo tiene 8 líneas.
El número de ciudades es de: 6
La ciudad de inicio es la número: 1
La ciudad de inicio es: S

Peso del hijo: 1.9
Heu del hijo: 1.5
Ciudad hija: P


Peso del hijo: 1.5
Heu del hijo: 0.5
Ciudad hija: C


Peso del hijo: 1.6
Heu del hijo: 1.0


Peso del hijo: 2.5
Heu del hijo: 1.1


-------------------------------------------------------------
Ciudad: 2
Peso del hijo: 0.8
Heu del hijo: 0.3
Ciudad hija: D
```

```
------------------------------------------------------------
Ciudad: 2
Peso del hijo: 0.8
Heu del hijo: 0.3
Ciudad hija: D


Peso del hijo: 1.8
Heu del hijo: 0.7


Ciudad Nodo: D
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 4
------------------------------------------------------------
------------------------------------------------------------
Ciudad: 3
Peso del hijo: 2.9
Heu del hijo: 2.6
Ciudad hija: P


Peso del hijo: 1.6
Heu del hijo: 1.0
Ciudad hija: S


Peso del hijo: 1.7
Heu del hijo: 0.7
Ciudad hija: AM
```


```
Peso del hijo: 1.7
Heu del hijo: 0.7
Ciudad hija: AM
```


```


Peso del hijo: 2.2
Heu del hijo: 1.8


Ciudad Nodo: AM
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
------------------------------------------------------------
------------------------------------------------------------
Ciudad: 4
Peso del hijo: 1.8
Heu del hijo: 0.7
Ciudad hija: C


Peso del hijo: 1.1
Heu del hijo: 0.1
Ciudad hija: A


Ciudad Nodo: A
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 3
------------------------------------------------------------
```

```
Ciudad Nodo: A
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 3
------------------------------------------------------------------
------------------------------------------------------------------
Ciudad: 5
Peso del hijo: 1.2
Heu del hijo: 1.1
Ciudad hija: P


Peso del hijo: 2.5
Heu del hijo: 1.1


Peso del hijo: 2.2
Heu del hijo: 1.8


Ciudad Nodo: P
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
------------------------------------------------------------------
Ciudades visitadas:
S C D AM A P
El peso total es de: 9.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experimento 2

```
run:
El archivo tiene 8 líneas.
El número de ciudades es de: 6
La ciudad de inicio es la número: 4
La ciudad de inicio es: AM

Peso del hijo: 1.8
Heu del hijo: 0.7
Ciudad hija: C


Peso del hijo: 1.7
Heu del hijo: 0.7
Ciudad hija: D


Peso del hijo: 1.1
Heu del hijo: 0.1
Ciudad hija: A


------------------------------------------------------------------
Ciudad: 5
Peso del hijo: 1.2
Heu del hijo: 1.1
Ciudad hija: P


Peso del hijo: 2.5
Heu del hijo: 1.1
```

```
------------------------------------------------------------
Ciudad: 5
Peso del hijo: 1.2
Heu del hijo: 1.1
Ciudad hija: P


Peso del hijo: 2.5
Heu del hijo: 1.1


Peso del hijo: 2.2
Heu del hijo: 1.8


Ciudad Nodo: P
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
------------------------------------------------------------
------------------------------------------------------------
Ciudad: 0
Peso del hijo: 1.9
Heu del hijo: 1.5
Ciudad hija: S


Peso del hijo: 2.9
Heu del hijo: 2.6

Ciudad Nodo: S
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 3
------------------------------------------------------------
------------------------------------------------------------
Ciudad: 1
Peso del hijo: 1.5
Heu del hijo: 0.5
Ciudad hija: C


Peso del hijo: 1.6
Heu del hijo: 1.0


Peso del hijo: 2.5
Heu del hijo: 1.1


Ciudad Nodo: C
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
------------------------------------------------------------
Ciudades visitadas:
AM A P S C
El peso total es de: 8.9
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
El archivo tiene 9 líneas.
El número de ciudades es de: 7
La ciudad de inicio es la número: 3
La ciudad de inicio es: EstadodeMexico

Peso del hijo: 130.0
Heu del hijo: 500.0
Ciudad hija: Chihuahua


Peso del hijo: 80.0
Heu del hijo: 300.0
Ciudad hija: Zacatecas


------------------------------------------------------------
Ciudad: 2
Peso del hijo: 60.0
Heu del hijo: 350.0
Ciudad hija: Tamaulipas


Peso del hijo: 300.0
Heu del hijo: 2000.0
```

Experimento 3

```
Peso del hijo: 80.0
Heu del hijo: 300.0
Ciudad hija: Zacatecas


------------------------------------------------------------
Ciudad: 2
Peso del hijo: 60.0
Heu del hijo: 350.0
Ciudad hija: Tamaulipas


Peso del hijo: 300.0
Heu del hijo: 2000.0


Ciudad Nodo: Tamaulipas
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 1
------------------------------------------------------------
Ciudades visitadas:
EstadodeMexico Zacatecas Tamaulipas
El peso total es de: 790.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
El archivo tiene 9 líneas.
El número de ciudades es de: 7
La ciudad de inicio es la número: 3
La ciudad de inicio es: EstadodeMexico

Peso del hijo: 130.0
Heu del hijo: 500.0
Ciudad hija: Chihuahua


Peso del hijo: 80.0
Heu del hijo: 300.0
Ciudad hija: Zacatecas


------------------------------------------------------------
Ciudad: 2
Peso del hijo: 60.0
Heu del hijo: 350.0
Ciudad hija: Tamaulipas


Peso del hijo: 300.0
Heu del hijo: 2000.0


Ciudad Nodo: Tamaulipas
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 1
```

Experimento 4

```
Ciudad: 1
Peso del hijo: 120.0
Heu del hijo: 650.0
Ciudad hija: Oaxaca


Ciudad Nodo: Oaxaca
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 1
------------------------------------------------------------
------------------------------------------------------------
Ciudad: 4
Peso del hijo: 90.0
Heu del hijo: 2000.0
Ciudad hija: Chiapas


Ciudad Nodo: Chiapas
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 2
------------------------------------------------------------
------------------------------------------------------------
Ciudad: 5
Peso del hijo: 300.0
Heu del hijo: 2000.0
Ciudad hija: Zacatecas


Peso del hijo: 100.0
```

```
------------------------------------------------------------
Ciudad: 5
Peso del hijo: 300.0
Heu del hijo: 2000.0
Ciudad hija: Zacatecas


Peso del hijo: 100.0
Heu del hijo: 720.0
Ciudad hija: Yucatan


Ciudad Nodo: Yucatan
Peso Nodo: 0.0
Heuristica: 0.0
Número de hijos: 1
------------------------------------------------------------
Ciudades visitadas:
EstadodeMexico Zacatecas Tamaulipas Oaxaca Chiapas Yucatan
El peso total es de: 4470.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experimento 5

Matriz de pesos

0.0 24.0 71.0 10.0 16.0 13.0 24.0 21.0 25.0 87.0 14.0 21.0 19.0

24.0 0.0 17.0 15.0 83.0 12.0 95.0 25.0 40.0 15.0 13.0 35.0 57.0

71.0 17.0 0.0 35.0 92.0 80.0 17.0 85.0 18.0 26.0 94.0 14.0 12.0

10.0 15.0 35.0 0.0 70.0 86.0 13.0 11.0 15.0 46.0 10.0 12.0 98.0

16.0 83.0 92.0 70.0 0.0 66.0 10.0 17.0 94.0 79.0 87.0 58.0 37.0

13.0 12.0 80.0 86.0 66.0 0.0 16.0 15.0 17.0 54.0 22.0 88.0 99.0

24.0 95.0 17.0 13.0 10.0 16.0 0.0 24.0 67.0 17.0 18.0 11.0 70.0

21.0 25.0 85.0 11.0 17.0 15.0 24.0 0.0 26.0 10.0 16.0 23.0 20.0

25.0 40.0 18.0 15.0 94.0 17.0 67.0 26.0 0.0 17.0 16.0 65.0 60.0

87.0 15.0 26.0 46.0 79.0 54.0 17.0 10.0 17.0 0.0 67.0 12.0 11.0

14.0 13.0 94.0 10.0 87.0 22.0 18.0 16.0 16.0 67.0 0.0 10.0 12.0

21.0 35.0 14.0 12.0 58.0 88.0 11.0 23.0 65.0 12.0 10.0 0.0 50.0

19.0 57.0 12.0 98.0 37.0 99.0 70.0 20.0 60.0 11.0 12.0 50.0 0.0

Matriz heuristica.

0.0 5.0 3.0 1.0 3.0 7.0 0.0 3.0 7.0 5.0 2.0 4.0 7.0

5.0 0.0 4.0 2.0 1.0 4.0 9.0 9.0 3.0 8.0 7.0 7.0 9.0

3.0 4.0 0.0 5.0 0.0 3.0 3.0 1.0 5.0 2.0 0.0 5.0 6.0

1.0 2.0 5.0 0.0 0.0 2.0 9.0 2.0 8.0 6.0 5.0 8.0 7.0

3.0 1.0 0.0 0.0 0.0 3.0 2.0 6.0 9.0 6.0 9.0 6.0 1.0

7.0 4.0 3.0 2.0 3.0 0.0 8.0 5.0 6.0 7.0 5.0 7.0 9.0

0.0 9.0 3.0 9.0 2.0 8.0 0.0 9.0 8.0 2.0 9.0 1.0 1.0

3.0 9.0 1.0 2.0 6.0 5.0 9.0 0.0 9.0 3.0 0.0 0.0 9.0

7.0 3.0 5.0 8.0 9.0 6.0 8.0 9.0 0.0 4.0 4.0 3.0 0.0

5.0 8.0 2.0 6.0 6.0 7.0 2.0 3.0 4.0 0.0 9.0 7.0 6.0

2.0 7.0 0.0 5.0 9.0 5.0 9.0 0.0 4.0 9.0 0.0 1.0 0.0

4.0 7.0 5.0 8.0 6.0 7.0 1.0 0.0 3.0 7.0 1.0 0.0 4.0

7.0 9.0 6.0 7.0 1.0 9.0 1.0 9.0 0.0 6.0 0.0 4.0 0.0

La ciudad de inicio es el número: 0
La ciudad de inicio es: C1

Ciudades visitadas:
C1 C4 C8 C10 C13 C11 C12 C7 C5 C1
El peso total es de: 120.0
BUILD SUCCESSFUL (total time: 0 seconds)

Bibliografía

[1] Lea, Doug: *Programación concurrente en java*. Addisson-Wesley, 2001.

[2] Martín, Antonio: *Programador Certificado Java 2*. Alfaomega Grupo Editor, 2008.

[3] Sierra, Antonio Martín: *Java*. Alfaomega Gurpo Editor, 2018.

[4] Stephen Stelting, Olav Maassen: *Patrones de diseño aplicados a Java*. Pearson Educacion, 2003.