**Universidad Tecnológica Centroamericana**

**UNITEC**

**Planificador de Rutas**

**Estudiante:**

|  |  |
| --- | --- |
| Dennis Cheong | 11241278 |

**Análisis de Algoritmos, Sección: 610**

**Catedrático: Jorge García**

**Tegucigalpa, M.D.C., 26/09/2016**

Índice

[Introducción 3](#_Toc462610057)

[Marco Teórico 4](#_Toc462610058)

[Implementación 6](#_Toc462610059)

[Resultados 7](#_Toc462610060)

[Conclusiones 8](#_Toc462610061)

[Anexos 8](#_Toc462610062)

[Referencias 10](#_Toc462610063)

# Introducción

Se debe desarrollar un módulo de planificación de rutas de distribución de sus productos en una ciudad. Lo que se desea es poder darle a cada conductor una lista de clientes, con sus ubicaciones y el orden en que deben ser visitados.

Para ello, se implementó un algoritmo que encuentra una solución al problema dada cualquier instancia del mismo.

NOTA: PUEDE UTILIZAR CÓDIGO OPEN SOURCE COMO BASE DE SU CÓDIGO —POR EJEMPLO LA INTERFAZ GRÁFICA- PERO NO DE SU ALGORITMO, EL ALGORITMO DEBE SER IMPLEMENTADO POR USTED.

# Marco Teórico

Los objetivos que se deben cumplir son los siguientes:

* La entrada del módulo sería la lista de los clientes con su ubicación y el número de camiones disponibles.
* La salida sería una lista de clientes ordenados por orden de visita para cada camión.
* El objetivo primordial es minimizar la distancia recorrida por cada camión
* El segundo objetivo es que todos los camiones recorran una ruta de aproximadamente la misma longitud que los demás, esto porque permitiría que el desgaste de cada camión sea similar.

Al analizar dicho problema y después de una serie de investigación. Se encontró que el problema planteado era similar al muy conocido problema de VRP (Vehicle Routing Problem).

Determinar la solución óptima resulta ser NP-Duro, como se probará ahorita mismo:

1. VRP es NP?

\* Existe algún algoritmo correcto que lo resuelva?

-Sí, se puede resolver el problema probando todas las combinaciones posibles hasta encontrar las rutas más óptimas.

\* El resultado es verificable en tiempo polinomial?

- No, no se puede verificar que la ruta dada es la más óptima en tiempo polinomial.

2. Es VRP reducible?

TSP-D

G = (V, E)

K = límite de distancia

Se puede encontrar una ruta que cubra todos los puntos una vez y que no se exceda a k?

VRP-D

G = (V, E)

N = número de camiones

K2 = límite de distancia

Existe N rutas que al sumarse sea K2 o menos?

TSP-D

Ruta

Si

G=G->

VRP-D

G->

N=1->

K->

Nada

No

K2=K->

Para este problema existen varias instancias:

\*Capacitated VRP (CVRP)

\*Capacitated VRP with time windows (CVRPTW)

\*Capacitated VRP with Pick-up and Deliveries and Time Windows (CVRPPDTW)

\*Multiple Depot VRP (MDVRP)

\*Multiple Depot VRP with Time Windows (MDVRPTW)

\*Periodic VRP (PVRP)

\*Periodic VRP with Time Windows (PVRPTW)

\*Split Delivery VRP (SDVRP)

\*Split Delivery VRP with Time Windows (SDVRPTW)

\*Vehicle Routing Problem with Pick-up and Deliveries (VRPPD)

Y al mismo tiempo, varios algoritmos que lo pueden resolver:

\*Exactas: Prueba todas las posibles combinaciones.

-Branch and bound (máximo 100 nodes) (Fisher 1994)

-Branch and cut

\*Heuristicas:

-Savings: Clark and Wright (1964)

-Matching Based

-Multi-route Improvement Heuristics

-Thompson and Psaraftis (1993)

-Van Breedam (1994)

-Kinderwater and Savelsbergh (1997)

\*2-Phase Algorithm:

-Cluster-First, Route-Second Algorithms

-Fisher and Jaikumar (1981)

-The Petal Algorithm

-The Sweep Algorithm

-Taillard (1993)

-Route-First, Cluster-Second Algorithms

\*Meta-Heuristics:

-Ant Algorithms

-Constraint Programming

-Deterministic Annealing

-Genetic Algorithms

-Simulated Annealing

-Tabu Search

-Granular Tabu

-The adaptative memory procedure

-Kelly and Xu (1999)

# Implementación

En este proyecto se implementó el algoritmo de Clarke and Wright. Este algoritmo se basa en savings, que es la distancia que se “salva” cuando se decide unir un nodo con otro. Este valor se saca mediante la siguiente fórmula:

Saving(n1,n2) = distancia(Origen,n1) + distancia(Origen,n2) – distancia(n1,n2)

Además de este particular requerimiento, es necesario que cada nodo posea un peso. También el algoritmo requiere de un límite de peso, para controlar prácticamente la cantidad de rutas que queremos y un valor distancia considerablemente equitativa entre cada ruta.

En este caso, el peso de cada nodo se tomó como la distancia entre dicho nodo con el origen. El límite que se utilizó fue la suma total de todos los nodos con el origen, todo esto divido entre la cantidad de camiones.

El algoritmo se corre varias veces, hasta que la cantidad de rutas resultante sea igual a la cantidad de camiones. Si la cantidad de rutas es mayor a la cantidad de camiones, se incrementará por 1% el límite actual. Caso contrario, se reducirá 1% del límite actual

Puede encontrar el código fuente del proyecto en el siguiente enlace: <https://github.com/elco45/Route_Planner>

# Resultados

Utilizando un archivo de prueba con nodos en:

(0,0)(0,2)(0,4)(1,5)(2,3)(4,1)(5,2) (100,1)

Número de camiones: 1

Ruta 0 distancia: 124.21784109681518

Número de camiones: 2

Ruta 0 distancia: 24.212841221808937

Ruta 1 distancia: 100.00499987500625

Número de camiones: 7

Ruta 0 distancia: 2.0

Ruta 1 distancia: 4.0

Ruta 2 distancia: 5.0990195135927845

Ruta 3 distancia: 3.605551275463989

Ruta 4 distancia: 4.123105625617661

Ruta 5 distancia: 5.385164807134504

Ruta 6 distancia: 100.00499987500625

PROBLEMAS

- El archivo entrada.txt contiene demasiado nodos, no se supo si el algoritmo implementado resulta ser óptimo.

- El algoritmo de Clarke and Wright requiere que todos los nodos tengan un peso. También se requiere tener un límite de peso, todo el algoritmo circula alrededor de este valor.

# Conclusiones

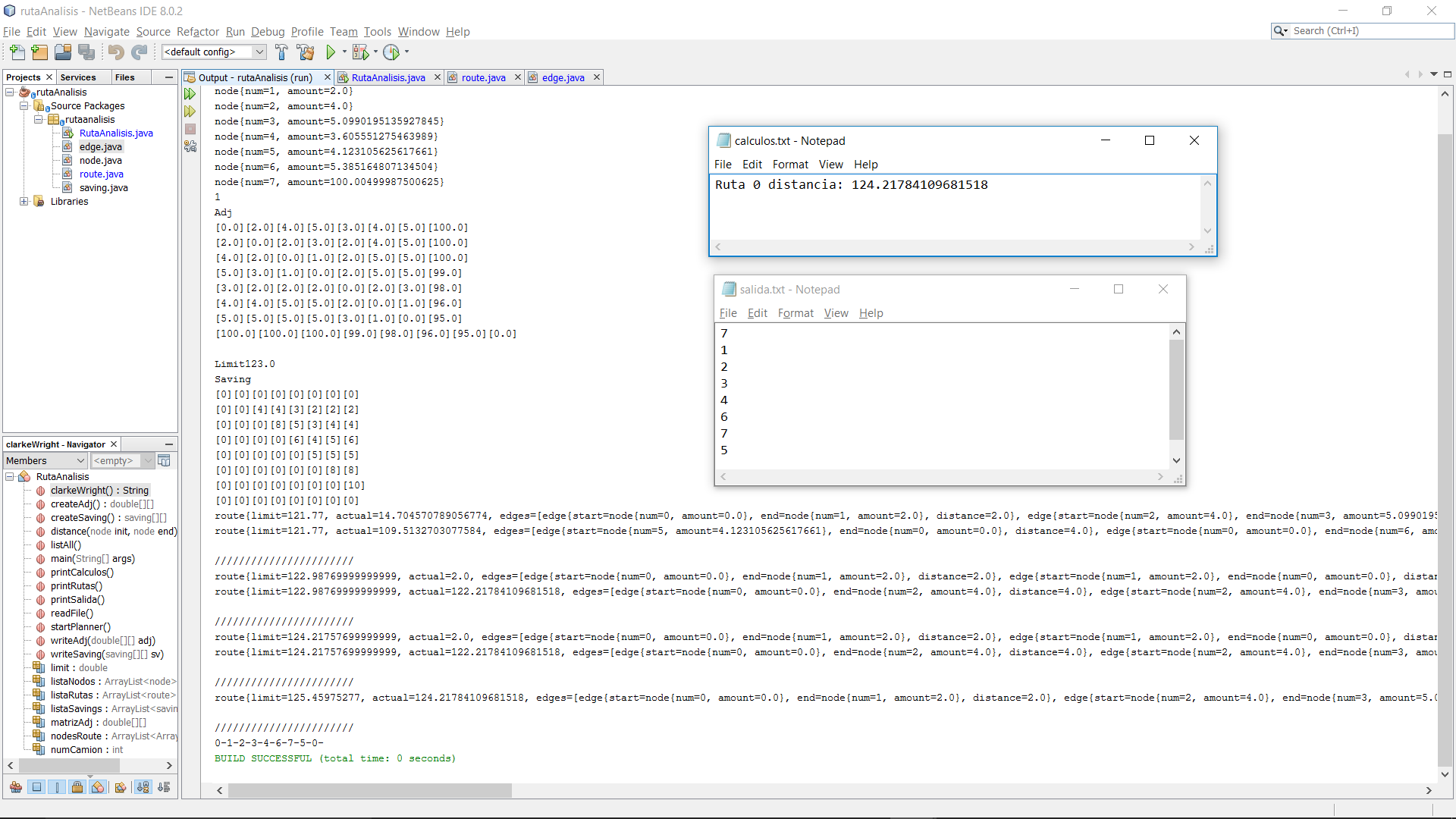
Los valores resultantes parecen cumplir con los requisitos del proyecto. Aunque no se sabe con certeza si el valor es el óptimo, se puede apreciar que está considerablemente cercano a una respuesta óptima. Posiblemente, la respuesta correcta solo se puede solucionar al probar todas las posibles combinaciones.

# Anexos

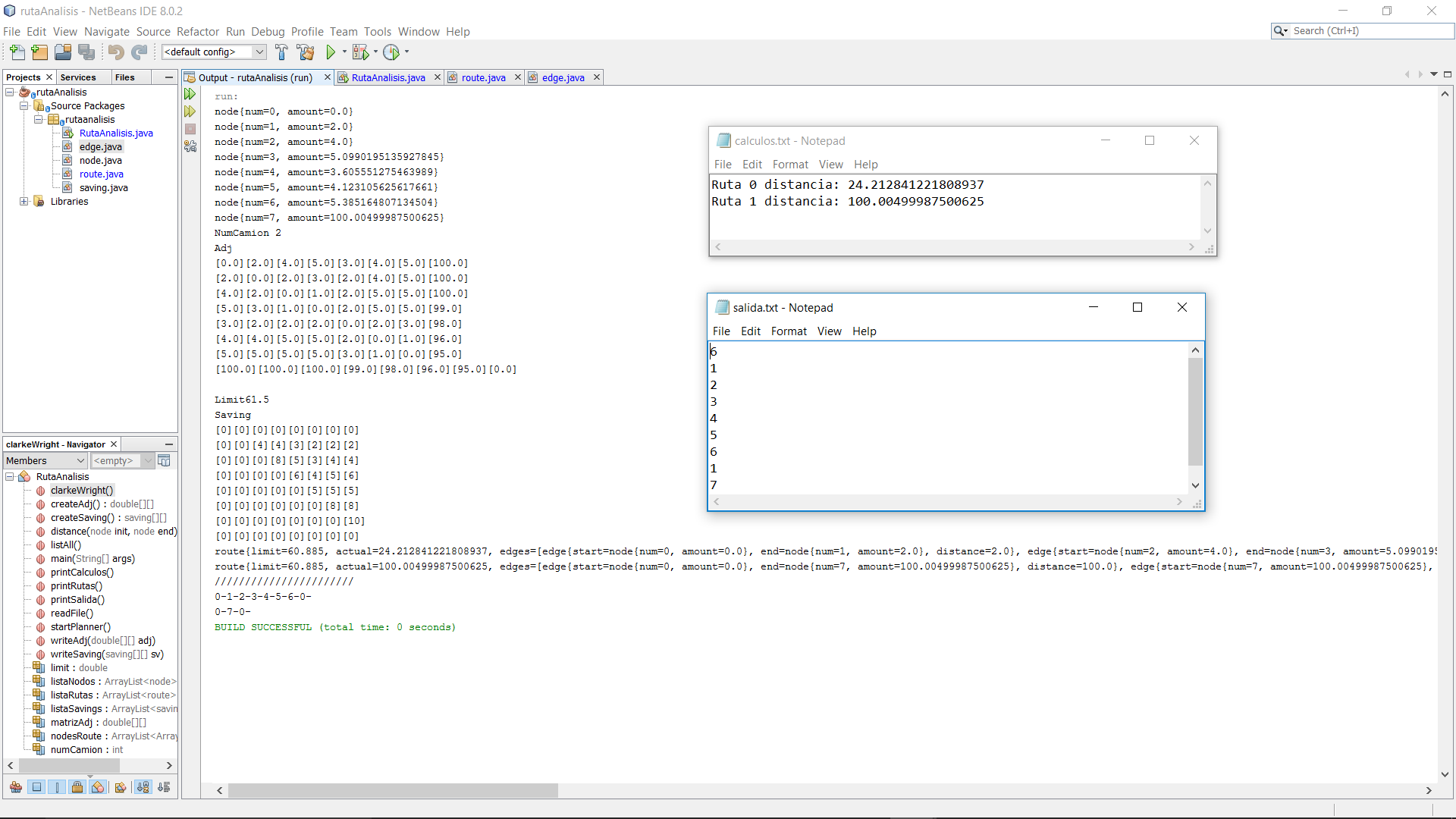
Utilizando un archivo de prueba con nodos en:

(0,0)(0,2)(0,4)(1,5)(2,3)(4,1)(5,2) (100,1)

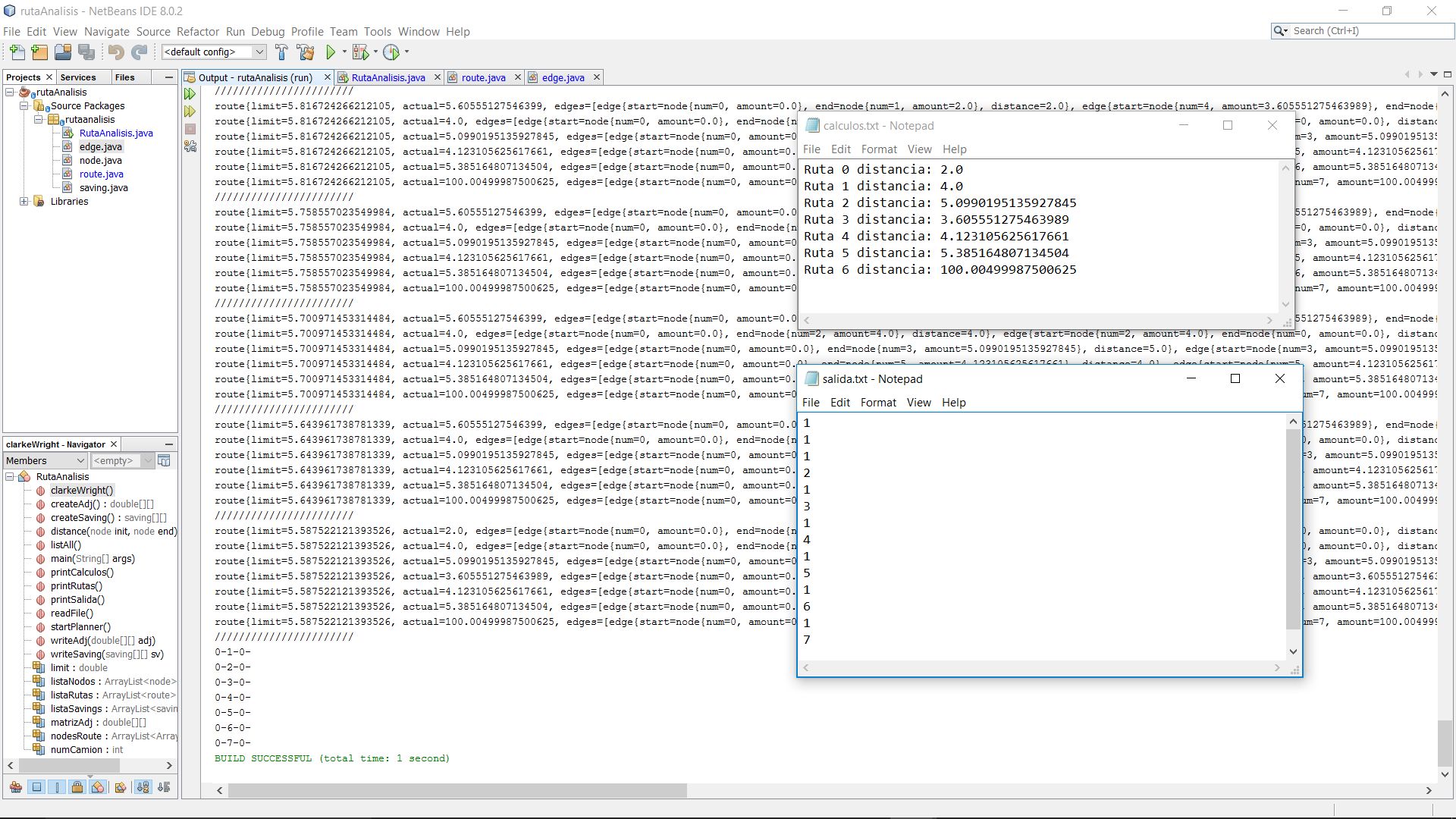
Número camiones = 1



Número de camiones = 2



Número de camiones = 7



# Referencias

(25 de 9 de 2016). Obtenido de http://web.mit.edu/urban\_or\_book/www/book/chapter6/6.4.12.html

(25 de 9 de 2016). Obtenido de http://pure.au.dk/portal-asb-student/files/36025757/Bilag\_E\_SAVINGSNOTE.pdf

(25 de 9 de 2016). Obtenido de http://neo.lcc.uma.es/vrp/solution-methods/heuristics/savings-algorithms/

*mayfe*. (25 de 9 de 2016). Obtenido de http://www.mafy.lut.fi/study/DiscreteOpt/CH6.pdf

*The vrp web*. (25 de 9 de 2016). Obtenido de http://www.bernabe.dorronsoro.es/vrp/

*Youtube*. (25 de 9 de 2016). Obtenido de https://www.youtube.com/watch?v=-EkM8CmHb5Q