Optimal route: An algorithm that simplifies the distribution of merchandise

Juan Diego Gutierrez

Carla Daniela Rendón

Medellín, 1/11/2018

Data Structures

For the design of the data structure, first we apply a matrix of arrays; to divide the map into different quadrants. Clients and recharging stations that belong to a quadrant will be stored in an array

To each vehicle we assign a quadrant to travel, where the vehicle will visit all customers in that quadrant including charging stations, we apply a greedy algorithm in each quadrant to travel the nodes

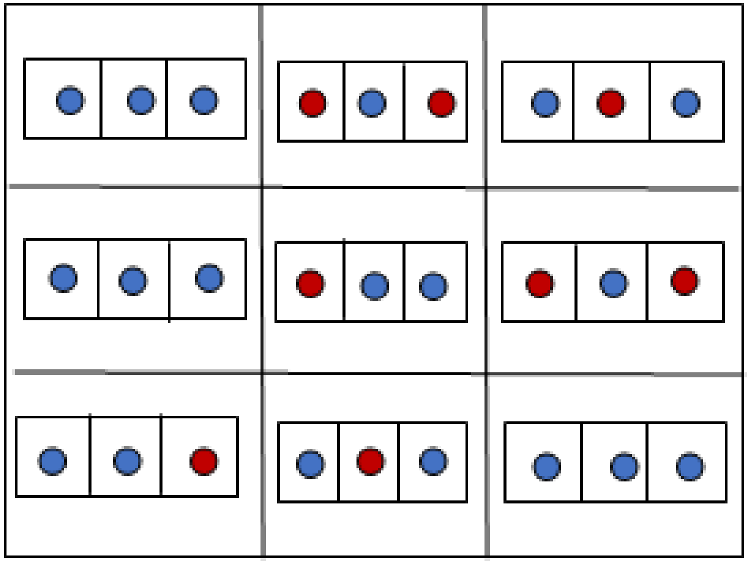


Figure 1

Figure 1: A matrix of arrays, where the blue nodes are clients and the red nodes are load stations

Algorithm and Complexity

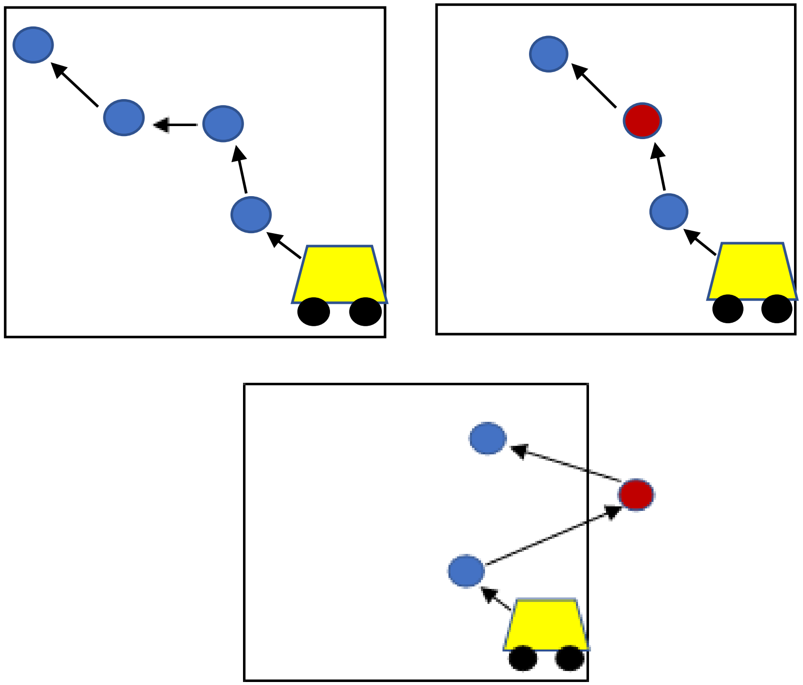


Figure 2

Figure 2: The possible cases that can occur when a car travels its quadrant

OPERATION	COMPLEXITY
Read the file	O(N), N being the number of lines
Divide into Quadrants	O(N), N being the number of vehicles
Create Quadrants	O(N x M), N being the rows and M the columns
Create Node	O(1)
Add Node	O(N), N being the index in the array where the node is stored
Travel	O(N x M x P), N being the rows, M the columns and P the number of nodes in the array
Calculate Distance	O(1)
Calculate Time	O(1)
Finish Travel	O(N), N being the number of clients in the quadrant
Add Routes	O(N), N being the index of the array where the node is stored
Print Routes	O(N), N being the number of quadrants
Print Time	O(1)
Total Complexity	O(N x M x P), N being the rows, M the columns and P the number of nodes in the quadrant. N x M it tends to be N ^ 2 since N and M are very close or equal

Table 1: Time complexity of the algorithm.

Algorithm design criteria

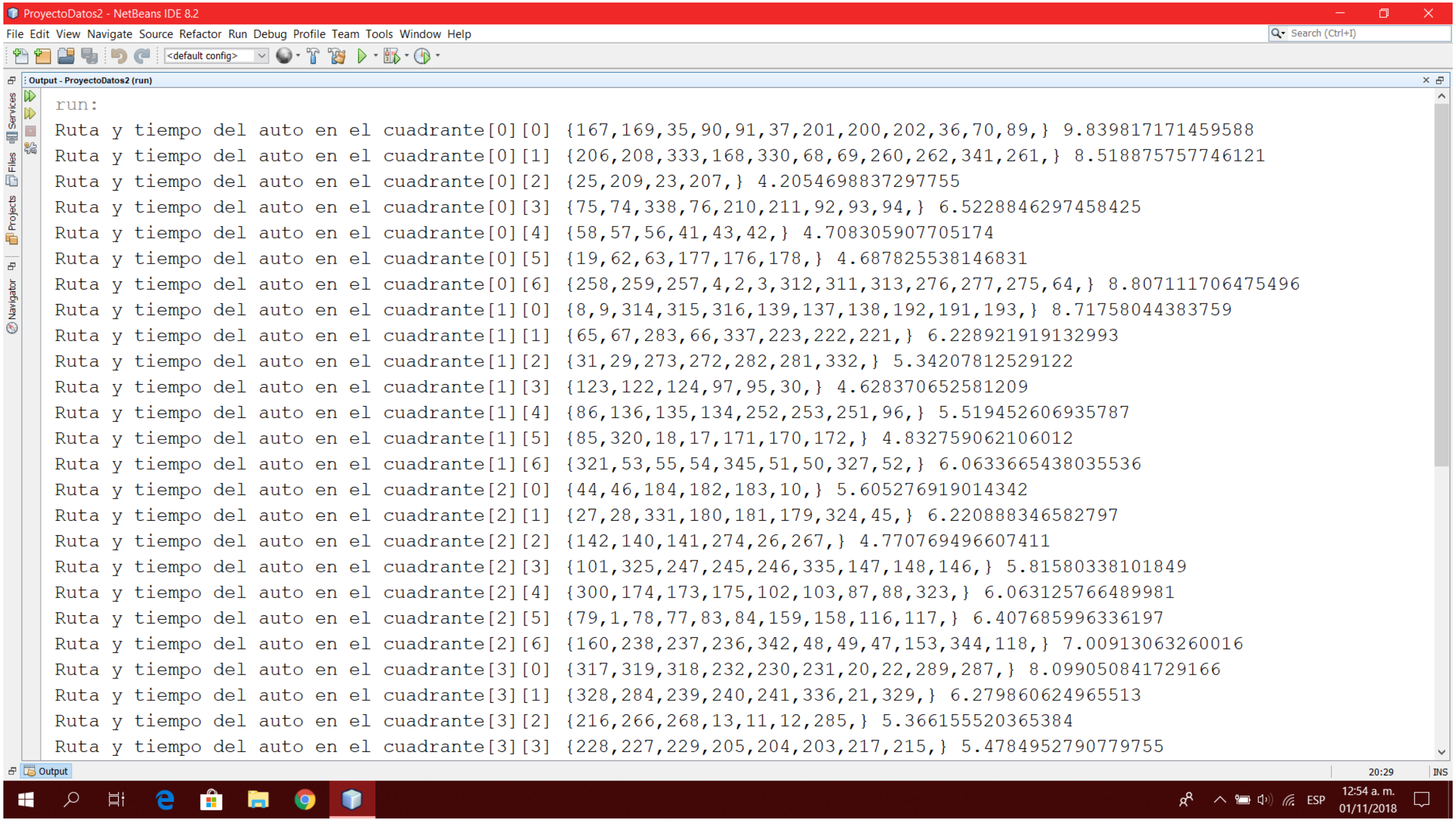
We decided to apply this data structure because we consider that it is optimal to solve the problem. In the given problem we have a map with many nodes, applying directly a greedy algorithm, it would not be the most efficient solution, but, by dividing the map into quadrants, we create smaller maps where we reduce the number of nodes, optimizing the execution of greedy algorithm, applying it to quadrants with few nodes.

Also in our data structure we implement an algorithm that optimizes the division of the map into quadrants, calculating the two closest factors that multiplied give us the number of quadrants.

Table of the Time Execution

Number of Vehicles	Execution Time
6	1068 ms
20	2001 ms
42	2069 ms
100	2015 ms
200	2070 ms
300	2082 ms

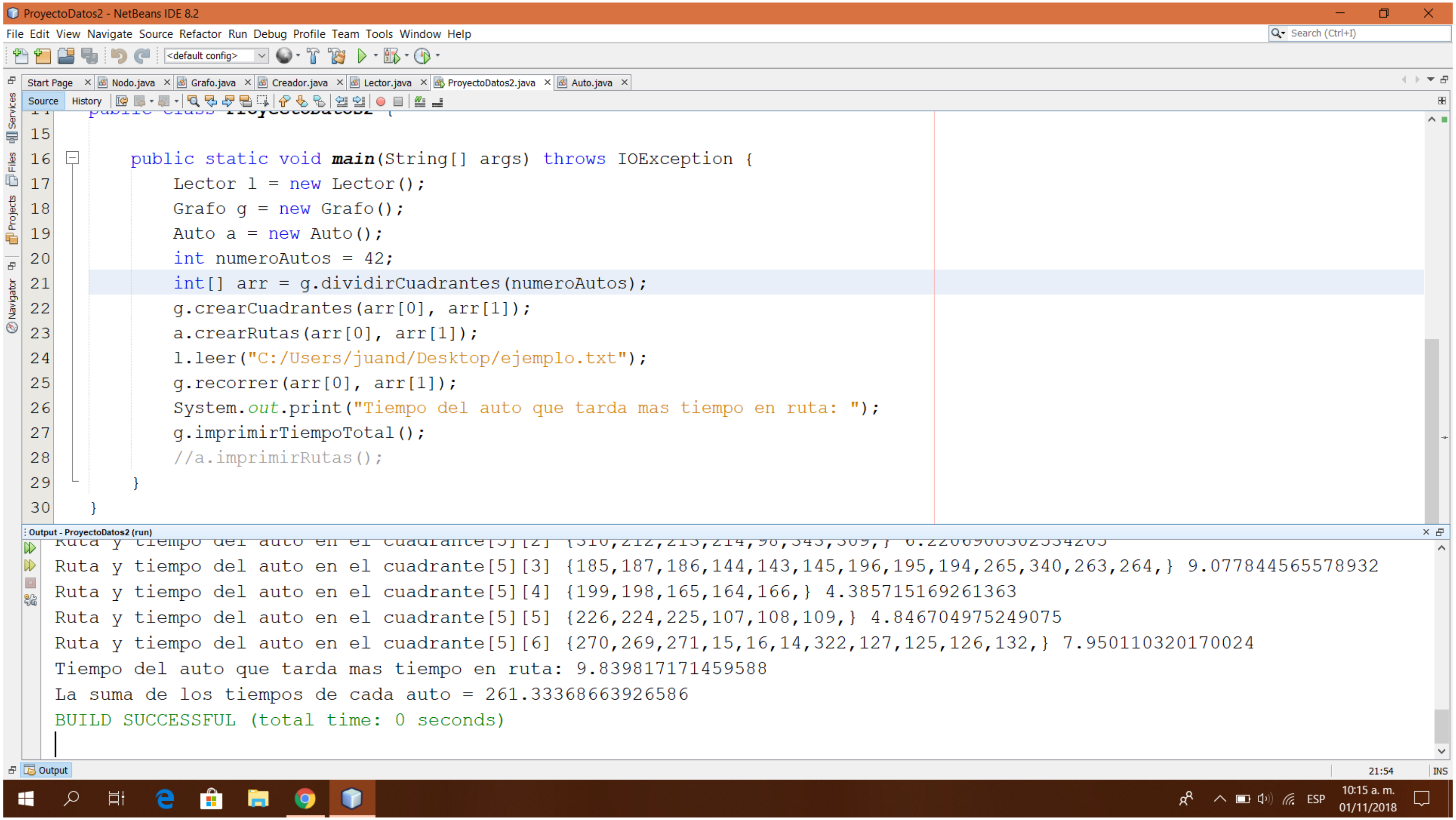
Software prototype



ProyectoDatos2 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Output - ProyectoDatos2 (run)
Services
Files
Projects
Navigator
Output
20:29
INS
ESP
12:54 a. m.
01/11/2018

```
run:
Ruta y tiempo del auto en el cuadrante[0][0] {167,169,35,90,91,37,201,200,202,36,70,89,} 9.839817171459588
Ruta y tiempo del auto en el cuadrante[0][1] {206,208,333,168,330,68,69,260,262,341,261,} 8.518875757746121
Ruta y tiempo del auto en el cuadrante[0][2] {25,209,23,207,} 4.2054698837297755
Ruta y tiempo del auto en el cuadrante[0][3] {75,74,338,76,210,211,92,93,94,} 6.5228846297458425
Ruta y tiempo del auto en el cuadrante[0][4] {58,57,56,41,43,42,} 4.708305907705174
Ruta y tiempo del auto en el cuadrante[0][5] {19,62,63,177,176,178,} 4.687825538146831
Ruta y tiempo del auto en el cuadrante[0][6] {258,259,257,4,2,3,312,311,313,276,277,275,64,} 8.807111706475496
Ruta y tiempo del auto en el cuadrante[1][0] {8,9,314,315,316,139,137,138,192,191,193,} 8.71758044383759
Ruta y tiempo del auto en el cuadrante[1][1] {65,67,283,66,337,223,222,221,} 6.228921919132993
Ruta y tiempo del auto en el cuadrante[1][2] {31,29,273,272,282,281,332,} 5.34207812529122
Ruta y tiempo del auto en el cuadrante[1][3] {123,122,124,97,95,30,} 4.628370652581209
Ruta y tiempo del auto en el cuadrante[1][4] {86,136,135,134,252,253,251,96,} 5.519452606935787
Ruta y tiempo del auto en el cuadrante[1][5] {85,320,18,17,171,170,172,} 4.832759062106012
Ruta y tiempo del auto en el cuadrante[1][6] {321,53,55,54,345,51,50,327,52,} 6.0633665438035536
Ruta y tiempo del auto en el cuadrante[2][0] {44,46,184,182,183,10,} 5.605276919014342
Ruta y tiempo del auto en el cuadrante[2][1] {27,28,331,180,181,179,324,45,} 6.220888346582797
Ruta y tiempo del auto en el cuadrante[2][2] {142,140,141,274,26,267,} 4.770769496607411
Ruta y tiempo del auto en el cuadrante[2][3] {101,325,247,245,246,335,147,148,146,} 5.81580338101849
Ruta y tiempo del auto en el cuadrante[2][4] {300,174,173,175,102,103,87,88,323,} 6.063125766489981
Ruta y tiempo del auto en el cuadrante[2][5] {79,1,78,77,83,84,159,158,116,117,} 6.407685996336197
Ruta y tiempo del auto en el cuadrante[2][6] {160,238,237,236,342,48,49,47,153,344,118,} 7.00913063260016
Ruta y tiempo del auto en el cuadrante[3][0] {317,319,318,232,230,231,20,22,289,287,} 8.099050841729166
Ruta y tiempo del auto en el cuadrante[3][1] {328,284,239,240,241,336,21,329,} 6.279860624965513
Ruta y tiempo del auto en el cuadrante[3][2] {216,266,268,13,11,12,285,} 5.366155520365384
Ruta y tiempo del auto en el cuadrante[3][3] {228,227,229,205,204,203,217,215,} 5.4784952790779755
```

Software prototype



```java
public class ProyectoDatos2 {

    public static void main(String[] args) throws IOException {
        Lector l = new Lector();
        Grafo g = new Grafo();
        Auto a = new Auto();
        int numeroAutos = 42;
        int[] arr = g.dividirCuadrantes(numeroAutos);
        g.crearCuadrantes(arr[0], arr[1]);
        a.crearRutas(arr[0], arr[1]);
        l.leer("C:/Users/juand/Desktop/ejemplo.txt");
        g.recorrer(arr[0], arr[1]);
        System.out.print("Tiempo del auto que tarda mas tiempo en ruta: ");
        g.imprimirTiempoTotal();
        //a.imprimirRutas();
    }
}
```

```
Ruta y tiempo del auto en el cuadrante[5][2] {310,212,213,214,98,343,309,} 6.2206900302534205
Ruta y tiempo del auto en el cuadrante[5][3] {185,187,186,144,143,145,196,195,194,265,340,263,264,} 9.077844565578932
Ruta y tiempo del auto en el cuadrante[5][4] {199,198,165,164,166,} 4.385715169261363
Ruta y tiempo del auto en el cuadrante[5][5] {226,224,225,107,108,109,} 4.846704975249075
Ruta y tiempo del auto en el cuadrante[5][6] {270,269,271,15,16,14,322,127,125,126,132,} 7.950110320170024
Tiempo del auto que tarda mas tiempo en ruta: 9.839817171459588
La suma de los tiempos de cada auto = 261.33368663926586
BUILD SUCCESSFUL (total time: 0 seconds)
```