An approach to Carpooling, an implemented solution for traffic and contamination problems.

Alfredo José Ospino Ariza
Jonatan David Acevedo López
Medellín, 21/05/2019

Data Structures

Figure 1: *A HashMap of Linked Lists that holds all the nodes and also a Linked List that takes in all nodes that have been already visited.*

Algorithm and Complexity

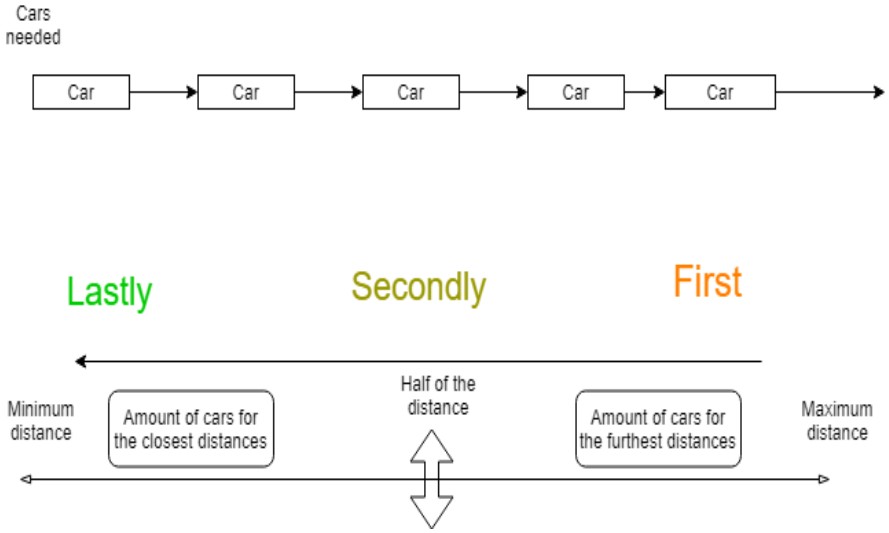

Figure 2: The addition of cars into the list of used an filled cars

	Subproblems	**Complexity**	**Total Complexity**
N *= the number of nodes the map has.* ***A*** *= the amount of line that are read of the data set.* ***C*** *= the number of cars that are being used to "car-pool".* ***L*** *= the length of the list of the first value of the map.*	Adding nodes(vertexes) to the map.	O(A)	**O(L*Log N)**
	Obtaining the furthest, "middlest", closest away node and adding the filled cars to the list.	O(L)	
	Finding the cars needed to do the carpooling.	O(L)	

Table 1: The Complexity of the algorithm

Algorithm design criteria

We based ourselves on something similar to a Binary Search, because this solution is very convenient for our problem, which is finding the least number of needed cars in a list, so at the end it is not very different to the previously mentioned algorithm.

	Dataset 205	***Dataset 11***	***Dataset 4***
Best case	0.0003 s	0.00025s	0.000241 s
Average case	0.00045 s	0.000425s	0.000395 s
Worst case	0.00062 s	0.0006 s	0.00055 s

Time and Memory Consumption

	Dataset 205	Dataset 11	Dataset 4
Memory consumption	8.4 MB	16 MB	16 MB

Figure 3: Memory consumption of the carpooling algorithm.