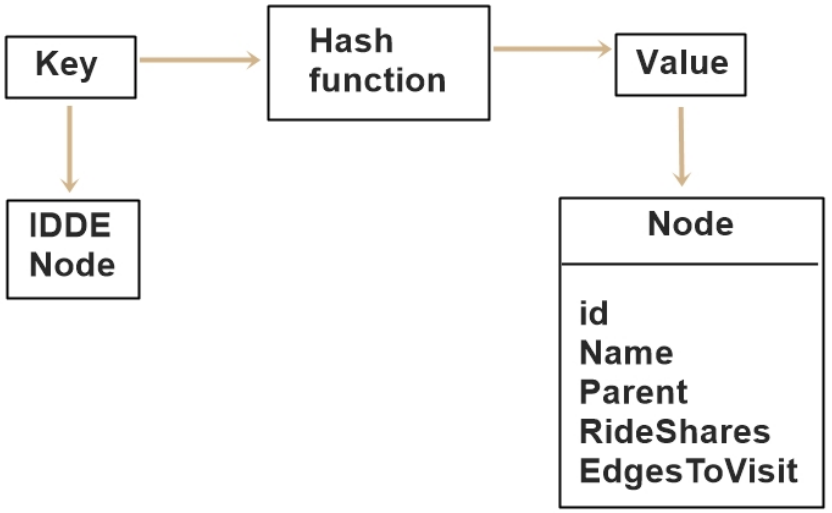
Route optimization algorithm for ridesharing

María Sofía Uribe
Isabel Cristina Graciano
Medellín, 2019/05/16

Data Structures



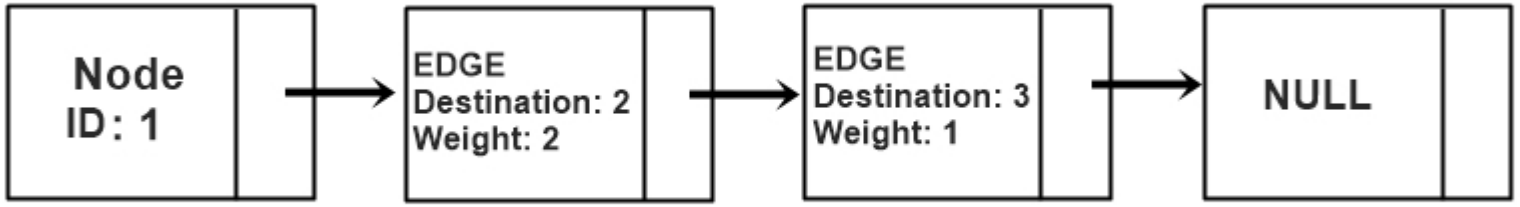


Figure 1: *A hash function which has a key (Id node) and a value (Node).*
Figure 2: *Simply linked list of the edges of a node. Edge is a class that contains Destination and Weight*

Algorithm and Complexity

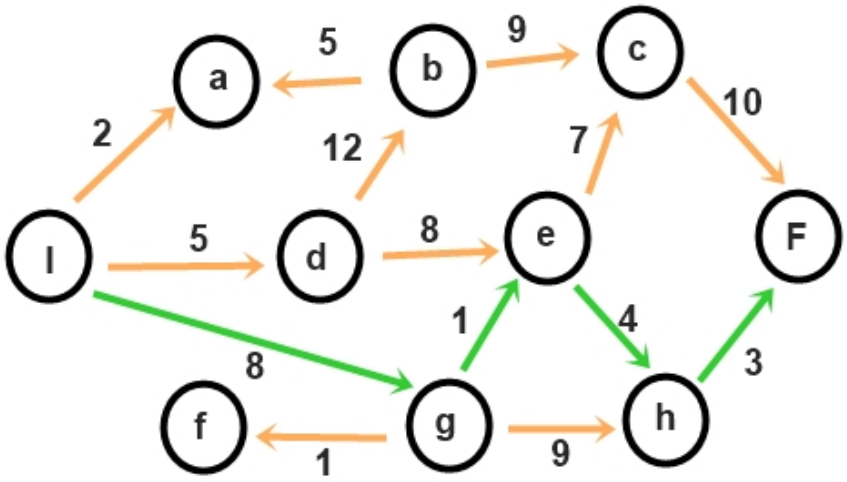


Figure 2: Directed search

Subproblem	Worst Case Complexity
Creating the graph(Reading the file and creating the graph)	O(V+E)
Search(A*) (Reconstruct one short path to 1)	O(V+E)
Assign Vehicles	O(V^2+EV)
Total complexity	O(V^2+EV)

Table 1: Time complexity of every method in the program and the total complexity of the algorithm.

Algorithm design criteria

After the analysis of some possible solutions, we concluded that is is easier to use HashTables because it allows us to reduce the required time for searching and insertion and LinkedLists so we can have in each value of the HashTable a node.

To find the shortest path among the nodes in the Graph we used A* algorithm because it has a low complexity in the worst case performance so it will not increase the performance of our algorithm and it can find a path between multiple points.

Time and Memory Consumption

Dataset	1	2	3	4	5
Memory Consumption	0 MB	0 MB	8 MB	10 MB	11 MB

Figure 3: Memory consumption of the ridesharing algorithm based on five given dataset.

Software prototype

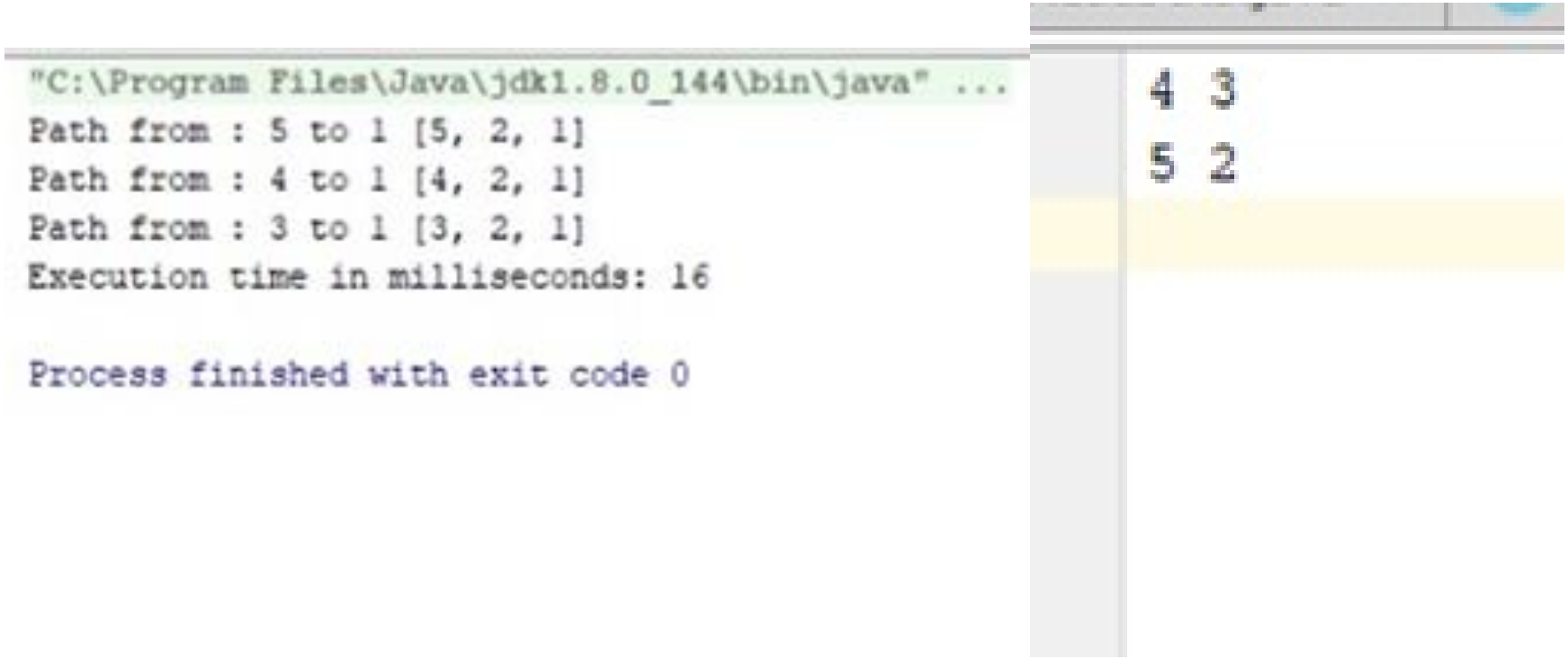


Figure 4: Execution of the program.
Figure 5: Result of the program