# Laboratory practice No. 4: Greedy Algorithm

**Jonathan Sebastián Acevedo Castrillón**
Universidad Eafit
Medellín, Colombia
jsacevedoc@eafit.edu.co

**Manuel Alejandro Gutierrez Mejía**
Universidad Eafit
Medellín, Colombia
magutierrm@eafit.edu.co

## 3) Practice for final project defense presentation

**3.1** For this case we use an auxiliary array in which we will mark the visited nodes and an array of nodes in which we will add the chosen elements to represent the route, meaning that, the solution of the problem whose previous process begins by taking an initial node and starting from this, it calculate the distances with all their subsequent nodes that are not marked as visited in the auxiliary arrangement, and the nearest neighbor to the initial node will be chosen to update the accumulated distance plus the distance of the nearest neighbor. The initial node is saved as visited and the chosen successor is taken as the initial node for the next iterations until the destination is reached.

**3.2** No, the algorithm will not always deliver the optimal solution because it takes a node and visits its neighbors to calculate the distances, chooses the lowest cost and repeats the process, in this way you can skip paths that at a first glance have a distance that it does not convince the algorithm but then its successors can derive in a path whose accumulated distance is smaller. The condition for the algorithm to deliver at least one solution is that the graph to be traversed is totally connected since, by its nature, it will try to go through all the nodes and then return to the initial one.

**3.3** The algorithm can be adapted for the case of delivering orders to the city of Medellin's homes in this way, first of all, only considering as nodes to travel, the places to which an order must be delivered, and to have a greater efficiency, calculate the cost based on the time that it takes to get from one place to another, maybe so the algorithm could be used.

**3.4** We use a Linked List for save the routes of the morning and the evening because in the Linked List the access is O(n) and the insertion is O(1). This algorithm works like a sorting algorithm because first we sort all the Linked List so the first element is the longest route on the morning or in the evening, with this we make sure that the last position in the Linked List is the shortest route in the morning or in the evening so when the algorithm takes all the routes, if the sumatory is higher to d so the algorithm multiplies the shortest routes with r then we make sure that result is the minimal.

**3.5** O(n+m)

**3.6** The number of the routes in the evening is “m”.
The number of the routes in the morning is “n”.

## *4) Practice for midterms*

***4.1*** i = j;
***4.2*** *min > adjacencyMatrix[element][i]*
***4.3*** Optional
***4.4***
***4.4.1*** temp/2
***4.4.2*** temp + minimo
***4.4.3*** O(1)
***4.5*** Optional
***4.6***
***4.6.1*** i+1
***4.6.2*** res+1
***4.6.3*** i
***4.6.4*** 2