Laboratory practice No. 4
Voracious Algorithms

Juan Camilo Guerrero Alarcón
Universidad Eafit
Medellín, Colombia
jcguerrera@eafit.edu.co

Juan José Escudero Valencia
Universidad Eafit
Medellín, Colombia
jjescuderv@eafit.edu.co

Santiago Pulgarín Vásquez
Universidad Eafit
Medellín, Colombia
spulgarinv@eafit.edu.co

3) Practice for final project defense presentation

3.1. The data structure used for this algorithm was an arraylist, in which we store the children nodes of the graph, later with this structure is traversed in an ordinary way and already placed in it and executing step by step, we started to draw and compare the minimum cost of each route between the axes and it is stored in a temporary variable.

3.2. The implementation of a voracious algorithm would not be adequate for this type of problem since it is affected by complexity, and it would not be an optimal solution. In order for the algorithm to work basically, a complete directed graph is available, to guarantee a unique solution otherwise the initial node would return.

3.3. One way to adapt it would be by visiting only the nodes where a certain order is associated with the address. One way to calculate these distances could be to establish different nodes where this said address.

3.4. The algorithm adds morning routes and afternoon routes to different arrays, once it's done, it sorts them in an ascending way using a tuned form of Quicksort. Then, it takes the first element in the first array and the last element in the second array. If its sum is greater than the maximum d given, it means that the driver is working extra hours, so the algorithm takes those extra hours and sums it to a variable that starts as 0 (it gets the amount of extra hours by calculating the sum of the elements minus d). The algorithm continues taking the next element of the first array and the previous element of the second array, doing the same process until it finishes checking the arrays.
At the end, the algorithm returns the product between the total extra hours and the amount of money that is paid for one extra hour.

3.5. O(nlogn). The longest process is sorting the arrays, java provides a method which is a tuned Quicksort with a nlogn complexity.

3.6. Being n the given number of drivers.

4) *Practice for midterms*

4.1 *i=j*
4.2 *min>adjacencyMatrix[element][i]*
1.4
  1.4.1 temp/2
  1.4.2 temp + minimo
  1.4.3 O(1)
1.6
  1.6.1 i + 1
  1.6.2 res +1
  1.6.3 i
  1.6.4 2

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473