# **Laboratory practice No. 2: Brute Force / Exhaustive Search**

|  |  |
| --- | --- |
| **Juan David Echeverri Villada**  Universidad Eafit  Medellín, Colombia  jdecheverv@eafit.edu.co | **Juan Sebastián Guerra Hernández**  Universidad Eafit  Medellín, Colombia  jsguerrah@eafit.edu.co |

**3) Practice for final project defense presentation**

**3.1** Firstly, we read the txt to create the graph according to what is specified in it. Then, we start doing the permutations to get all the possible ways to go to each vertex only once, and during the iteration, we discard all the paths that are not possible (because there is not an edge that connect those two vertex). Finally, we sum all the weights of the edges in each path, comparing them and finally returning the pair of the lowest distance and its respective path.

**3.2** The complexity of the algorithm es de O(!V + E^2) being V the number of vertex and E the number of edges of the graph.

**3.3** No, since it is a brute force algorithm it has a very high complexity. In a case where we must test it with 50 clients, the number of process done is equal to 50! = 3.04x10^64. Now, if we use a 3Ghz processor, it will take an estimate of 3.21x10^44 millenniums to find the best route to attend 50 clients.

**3.4** To solve the exercise 2.1, we use 2 arrays. In the first one, we save the game board with the positions of the queens. The index represents the column and the number saved in that position of the array is the row. In the second one, we save, using the same mechanics as before, and array list that in each position contains the cells of the board that cannot be used (the broken ones). Having those two arrays, our algorithm starts to check the possible solutions making sure there is no collisions with other queens and avoiding the broken cells, giving recursively the total of possible solutions.

**3.5** The complexity of the algorithm is O(n^3)

**3.6** The meaning of the variable “n” is the number of queens.

***4) Practice for midterms***

* + 1. actual > maximo
    2. O(n^2), Being n the number of elements in the array a

***4.2.1*** arr, k+1

***4.2.2*** O(n!), Being n the number of elements in the array arr.

***4.3.1*** return i-j

***4.3.2*** return n

***4.3.3*** O(n), Being n the length of the text.

***4.4.1*** temp % 10

***4.4.2*** Opc. B

***4.5.1*** i+1

***4.5.2*** left == right