ASSIGNMEN 2: COMPUTATIONAL COMPLEXITY.

JOSÉ LUIS CUMBRERA SÁNCHEZ

1.INTRODUCTION.

We are going to use Prim´s algorithm in our work. The context is créate a graph with a certain number of mandatories nodes and certain amount of non-mandatories nodes.

We are going to analyze the computability of the program and we are going to understand why are we obtaining that results.

2.METHODOLOGY.

I´m using a software called CodeBlock++ to compile my c++ code. The draw has been made in Gnuplot.

I´m using a graph with 10 mandatories nodes and 0 to 12 non-mandatories nodes, i think that with that we can obtain the waited results.

I used an randomized graph because we don´t need to attend on the inicialization, ins´t important in our chase.

For each k, i have calculated the number of combinations possibles with our amount of non-mandatory nodes taking only k, lets call it N. After that, i have done Prim N times in that specific iteration. I simulated the combination always using the same graph, because in terms of complexity is irrelevant.

3.RESULT AND DISCUSSIONS.

We are going to use big Oh (O) to classify the algorithm because we always have to do it for our worst chase. In this chase, the worst chase is only obtain an output.

3.1.Big Oh(O) complexity.

We have an know order O(n^2). We can divine it only watching what the algorithm does:

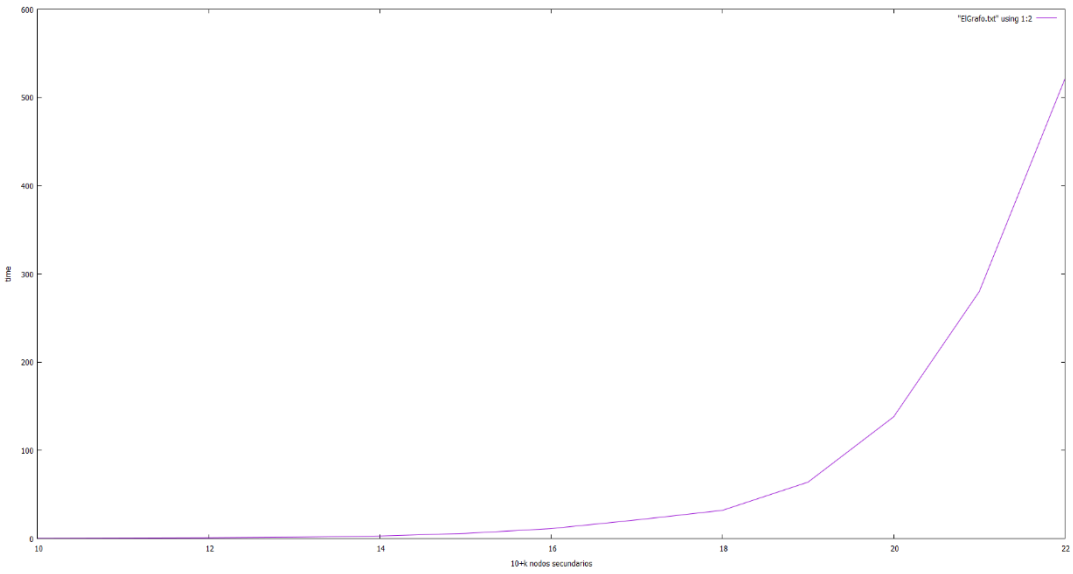
We start in A (example) and we have 3 option, we select the minumin cost option (B) and now we have {A,B}.

We repeat the algorithm using now all the options in A and B, selecting the best one.

As we can see, we are revisiting nodes, so, in the worst chase We visit the graph twice, so the order is O(n^2).

3.2.Output.

As we can see, the obtained result is a clasical output for an O(n^2) problem.



Here we have the obtained result in bars:

10+0	0.37
10+1	0.60
10+2	0.88
10+3	1.58
10+4	2.92
10+5	5.87
10+6	11.29
10+7	21.19
10+8	32
10+9	64
10+10	138
10+11	280
10+12	521

4. CONCLUSIONS.

This algorith is a typical logaritmic time algorithm, as we can see, if our chase is a low chase one, that algorithm resolves the problema is a correct amount of time, but, is the chase so big… , well you can see it in the draw, we have a chase whose output will be obtained in a few minutes, hours… So watching this example, we can see the importance of try to find a logarithmic or a linearithmic time solution, but not always is easy!

With this example we can answer the typical question : Why this is important?, well study computational complexity is important because we can discover how to implement more efficient algorithms and we can improve our live with this (making our mobiles faster for example).