# **Laboratory practice No. 2**

|  |  |
| --- | --- |
| **Sara María Castrillón Ríos**  Universidad Eafit  Medellín, Colombia  smcastril1@eafit.edu.co | **Valeria Guerra Zapata**  Universidad Eafit  Medellín, Colombia  vguerraz@eafit.edu.co |

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

**3.1** Initially we did a permutation without repetition in order to generate every possible path through brute force and then verify which of them are correct.

**3.2** O(n\*\*m) where n represents the number of vertices, and m the number of edges, it means , every possibility per vertex.

**3.3** Time was estimated with:

“import time

start=time.time()

.

.

.

end = time.time()

print(end-start)”



This algorithm is not going to be so efficient with a large number of vertices because it was a brute force approach and the fact to generate every possibility and then verify it is correct is such a hard work, plus the complexity is exponential.

**3.4** Initially we did a permutation without repetition in order to generate every possible table of queens and it was stored in a list, at the same time, the input was read to identify de wrong box and go to compare them together.

**3.5** O(n\*\*2 \* m)

**3.6** We have a complexity of O(n\*\*2 \* m), since in the verificar\_posiciones method we have two nested fors that are evaluated according to the number of queens we have (n), and then we have another for that is evaluated according to the number of bad squares (m)

***4) Practice for midterms***

***4.1.1.*** maximo < actual

**4.1.2.** O(n\*\*2)

**4.2.1.** arr, k+1

**4.2.2.** O(n)

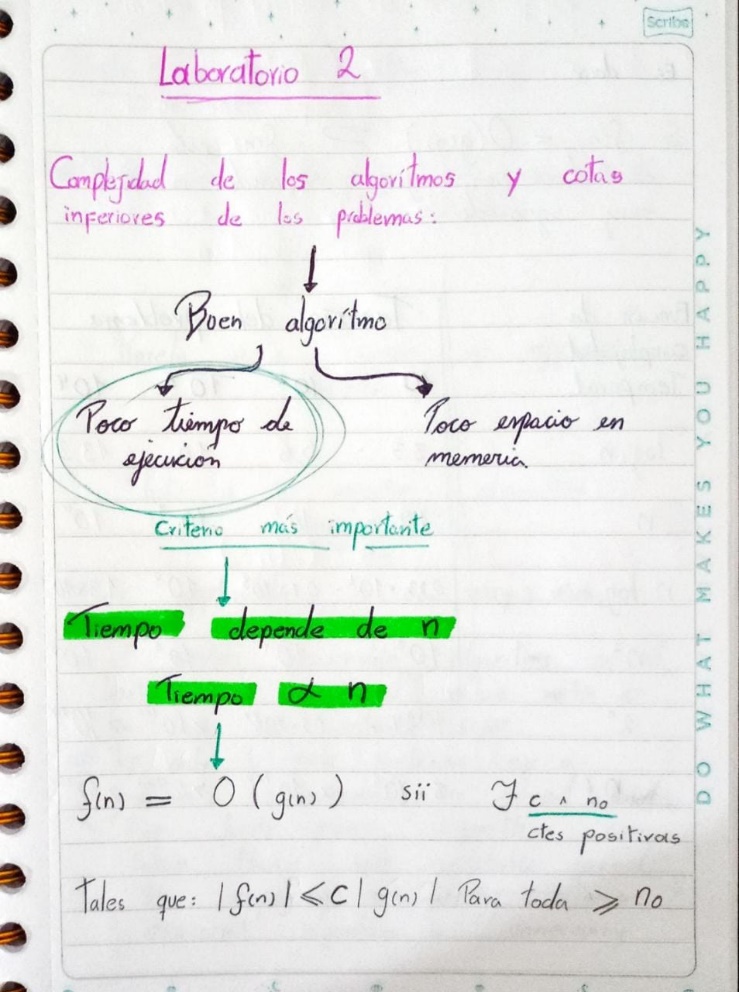
**4.3.1** i - m

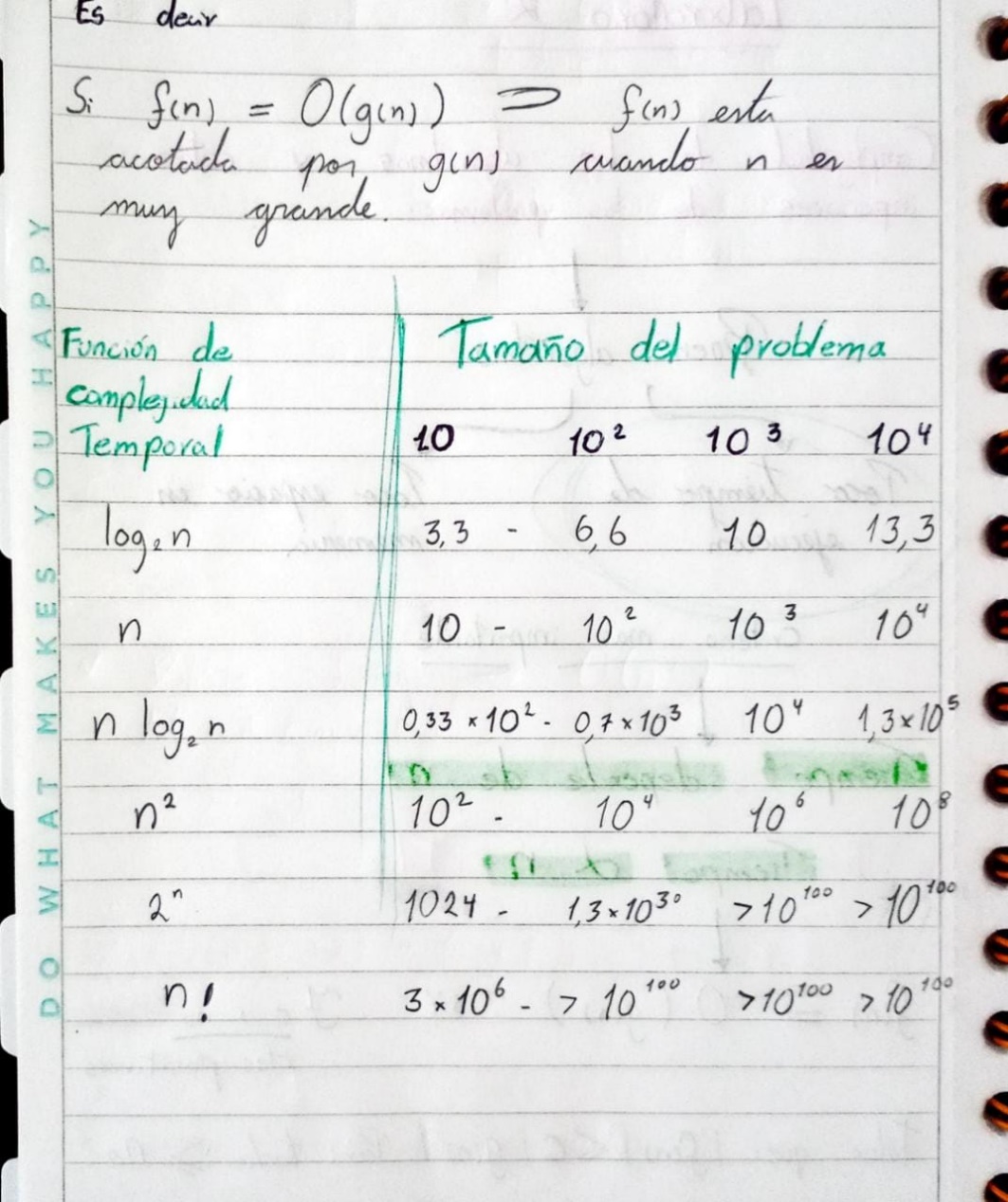
**4.3.2** n

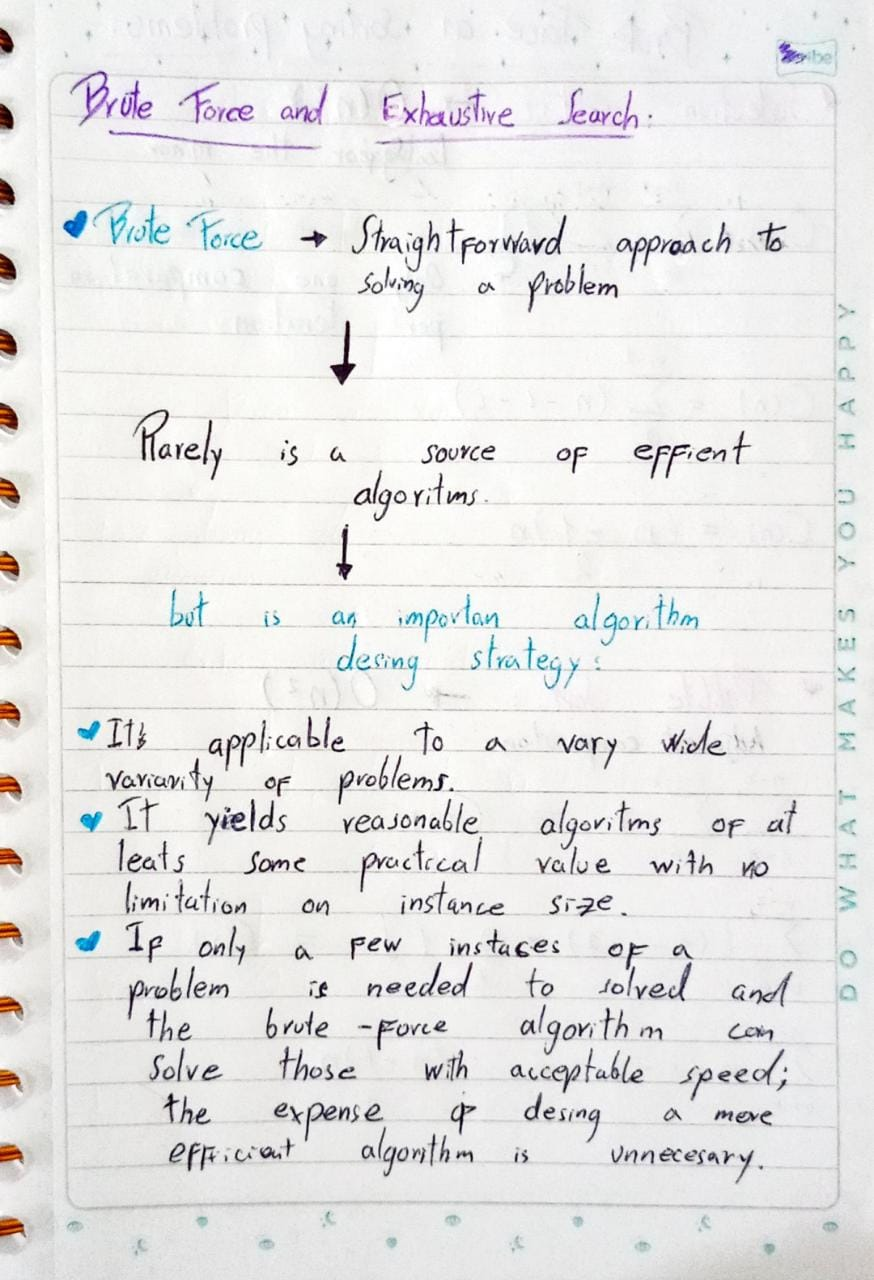
**4.3.3.** O(n\*m)

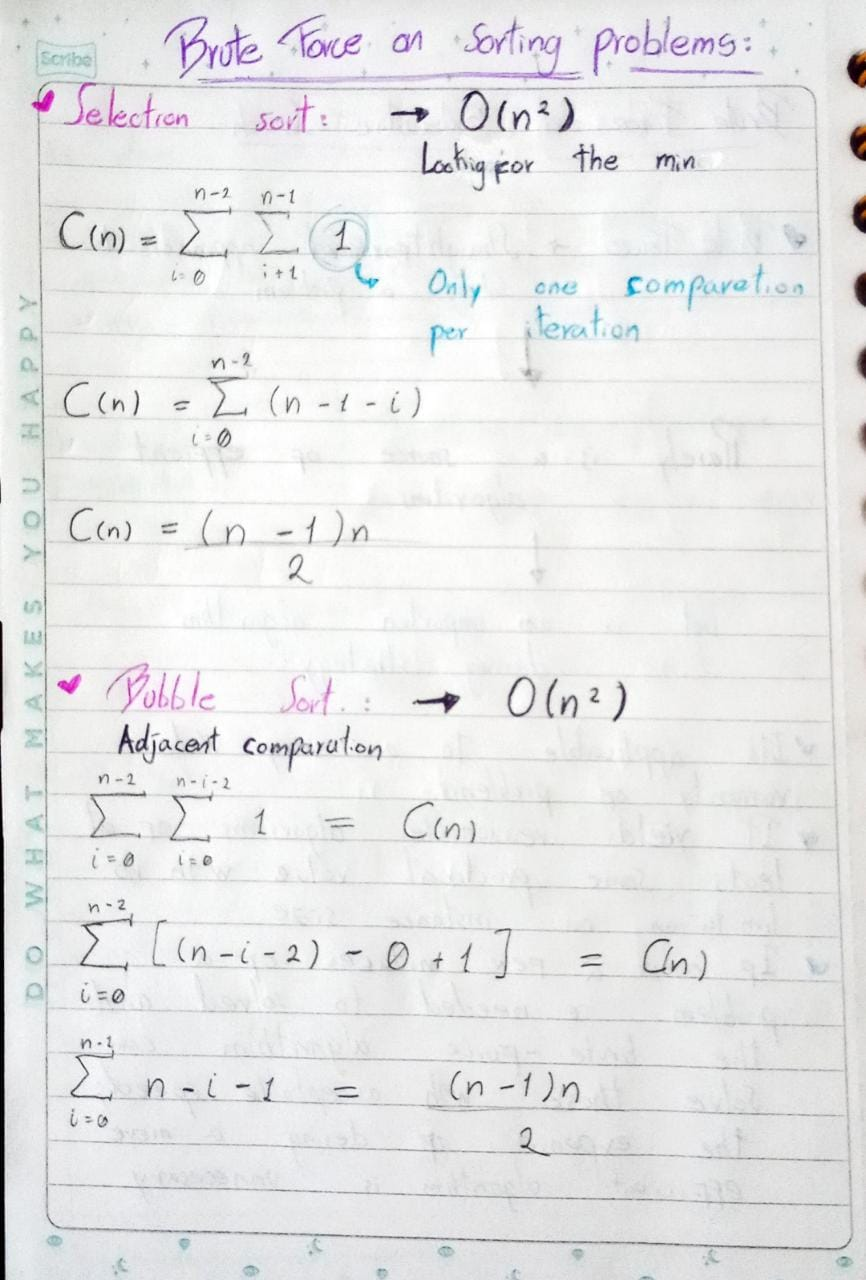
**4**

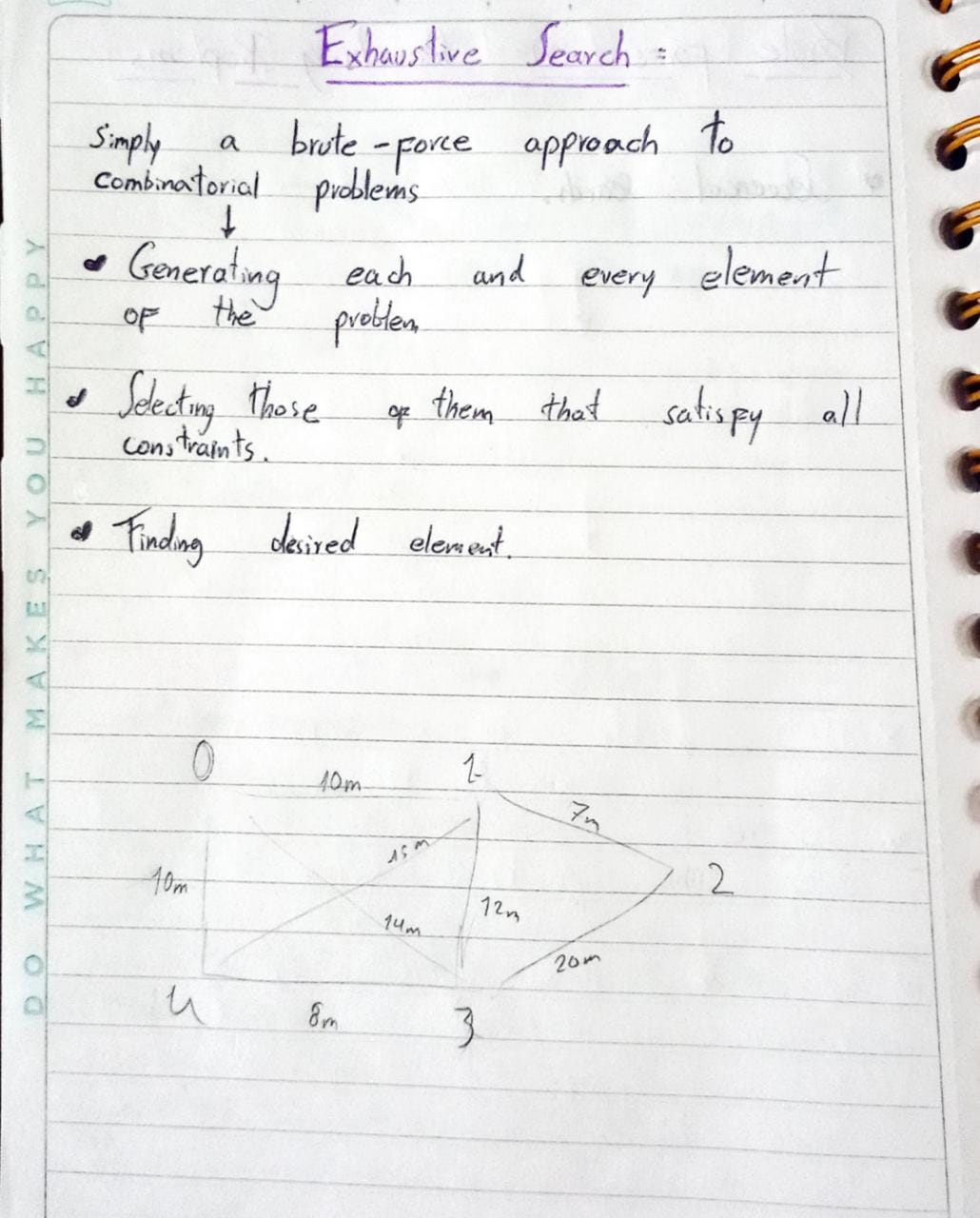
***5) Recommended reading***

******

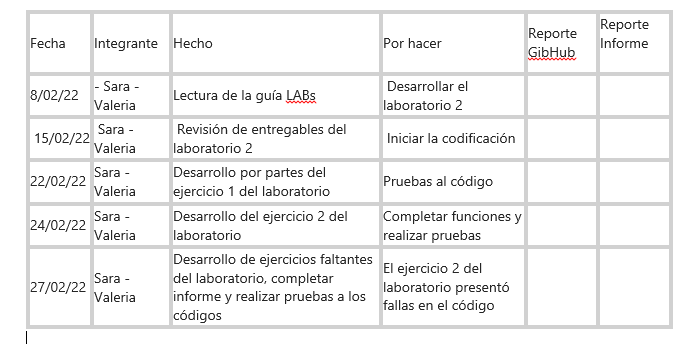








**6)** **Team work and gradual progress**

******