**Documento de análisis**

* **Nombres:**

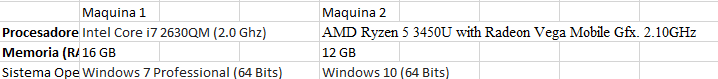
# Estudiante 1: Samuel Josué Freire Tarazona, 202111460, s.freire@uniandes.edu.co ----------->

# Requerimiento 3 (Individual) = Samuel Josué Freire Tarazona

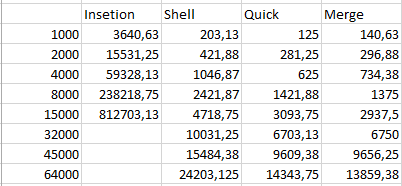
# Estudiante 2: José David Martínez Oliveros, 202116677, jd.martinezo1@uniandes.edu.co----------->

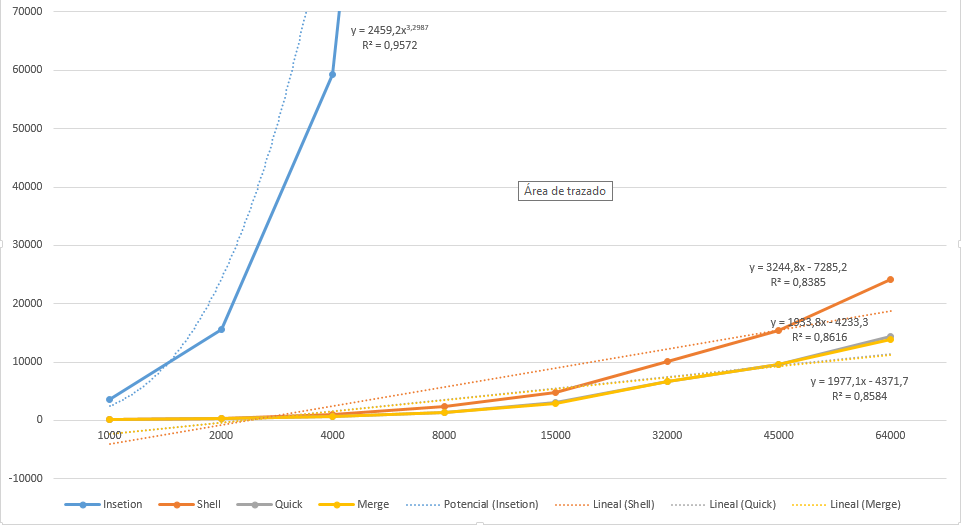
# Requerimiento 4 (Individual)= José David Martínez Oliveros

* **Propiedades de los sistemas operativos:**
  + Maquina 1: Samuel Josué Freire Tarazona
  + Maquina 2: José David Martínez Oliveros

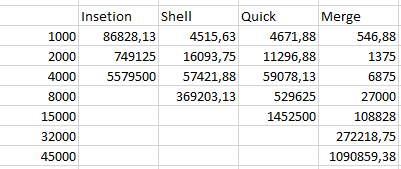


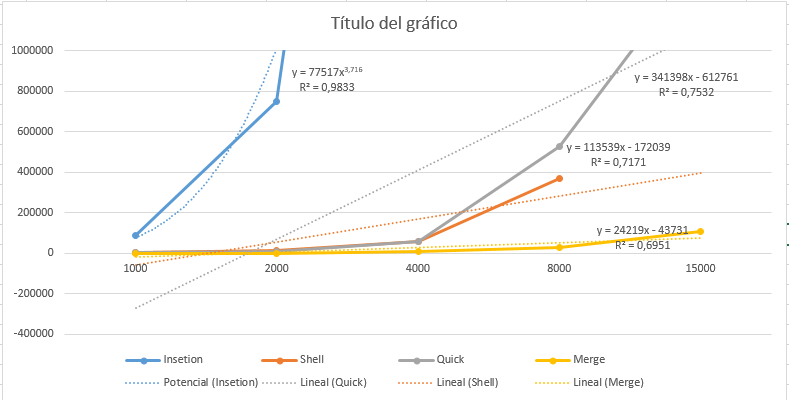
* **Pruebas de ordenamientos:**
* **Maquina 1:**

****

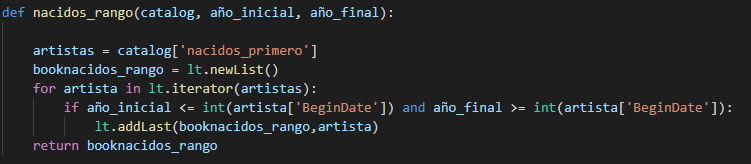
****

* **Maquina 2:**

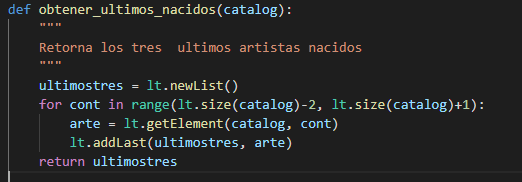




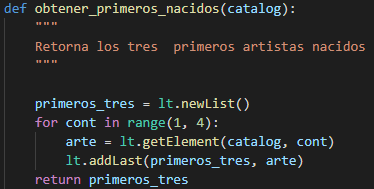
* **Conclusiones:**
  + Los algoritmos, como se vio en el laboratorio 4, funcionan como se enuncian teóricamente. Puesto que, cada uno tiene un orden crecimiento similar al que se conoce con anterioridad. Sin olvidar, que aquí se aplica un caso promedio, ni el mejor ni el peor. Por ejemplo, para el caso del insertion sort, podemos denotar que tiene un crecimiento potencial. Esto se puede deducir ya que a medida que aumentan los datos el tiempo aumenta de manera potencial. También, se puede ver por su ecuación y por su línea de tendencia. Ya que, esta tiende a ser de un orden potencial. Otro ejemplo, lo podemos ver con el caso del ordenamiento Shell. Este ordenamiento tiene un orden teórico de O(N^2), y como se ve en la gráfica cumple con su línea. Demás, si se ve la tendencia d esta grafica no damos cuenta que la gráfica tiene a crecer de manera exponencial, como una función de grado mayor a 1. Y por último, tenemos el ejemplo del ordenamiento Quick y Merge. Estos dos cumplen con ordenamientos logarítmicos que de cierta manera se ven en la gráfica. Ya que representan un crecimiento inferior a un exponencial pero superior a un constante.
  + Luego, las diferencias entre cada máquina se deben a su capacidad. Esto lo podemos denotar ya que para la maquina 1 se presentan mejores tiempo, en la mayoría de los ordenamientos. Además, de que su procesador no es tan diferente en términos de Hertz. Por ejemplo, Para el Insertion, se puede denotar que para la maquina los crecimientos son menores. A pesar de que no se logran las pruebas completas. Caso contrario, sucede con la maquina 2, donde se ve que dese el primer momento, el tiempo es superior.
  + Ahora, luego de haber hecho las pruebas y este pequeño análisis, se va decir cual ordenamiento es mejor y porque se utilizó en el reto. Para nuestro grupo, el ordenamiento que funciono de mejor manera fue el Merge; seguido de este, etsa el quick sort; luego, el Shell sort; y por utlimo está el Insertion. Cada posición está basado en las pruebas de tiempo hechas, ya que el Merge genero mejores números que los otros. Y el Insertion tiende a no cargar luego de cierta cantidad de datos. Por esto son los lugares so el ranking dado. Por lo que este fue el ordenamiento que usamos en las funciones de ordenamiento del reto. Ademas, de esto, más adelante se va a ver, pero este ordenamiento de Merge nos da mejores números en términos de los requerimientos y sus pruebas totales.
* **Análisis de complejidad:**
  + **Requerimiento 1:**
    - Primera función (nacidos\_rango):



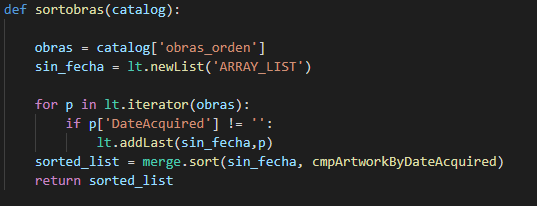
* O(n)
  + - Segunda función (obtener\_ultimos\_nacidos):



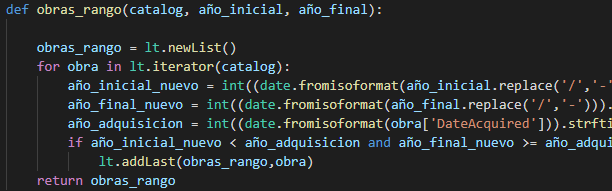
* O(n)
  + - Tercera función (obtener\_primeros\_nacidos):



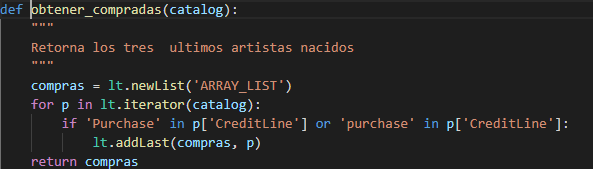
* O(n)
  + - COMPLEJIDAD GENERAL: O(n) + O(n) + O(n) = O(3n) = O(n)
  + **Requerimiento 2:**
    - Primera función (sortobras):



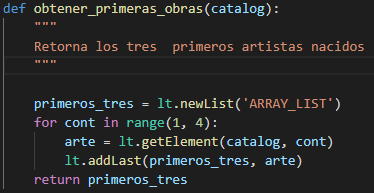
* O(n)
  + - Segunda función (obras\_rango):



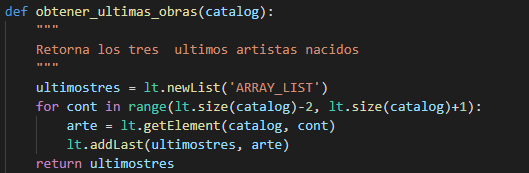
* O(n)
  + - Tercera función (obras\_compradas):



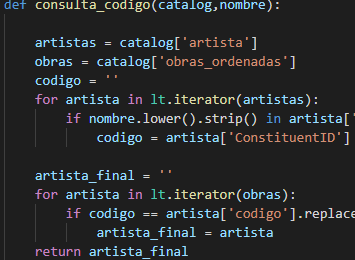
* O(n)
  + - Cuarta función (obtener\_primeras\_obras):



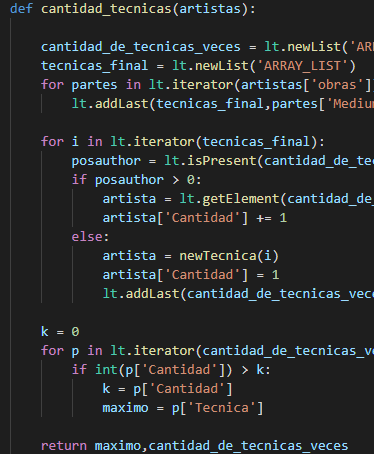
* O(n)
  + - Quinta función (obtener\_ultimas\_obras):



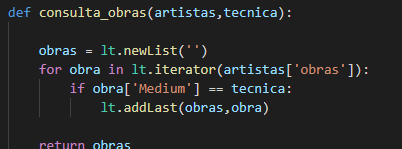
* O(n)
  + - COMPLEJIDAD GENERAL: O(n) + O(n) + O(n)+O() + O(n) = O(5n+) = O(n)
  + **Requerimiento 3:**
    - Primera función (consulta\_codigo):



* O(n)
  + - Segunda función (cantidad\_tecnicas):



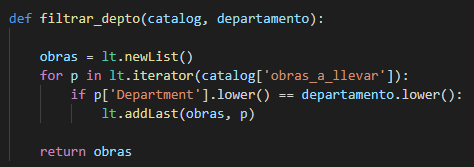
* O(n)
  + - Tercera función (consulta\_obras):



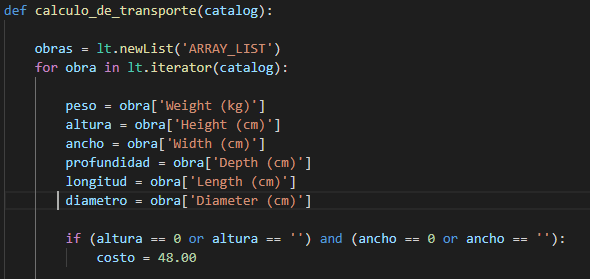
* O(n)
  + - COMPLEJIDAD GENERAL: O(n) + O(n) + O(n) = O(3n) = O(n)
  + **Requerimiento 4:**
    - función (consulta\_codigo):



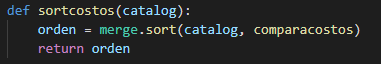
* O(o(n^2 + log n))
  + **Requerimiento 5:**
    - Primera función (filtrar\_depto):



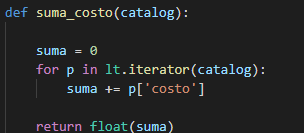
* O(n)
  + - Segunda función (Calcular\_transporte):



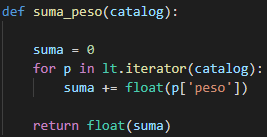
* O(n)
  + - Tercera función (sortcostos):



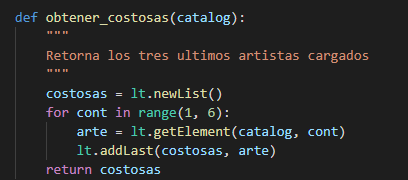
* O(n)
  + - Cuarta función (suma\_costo):



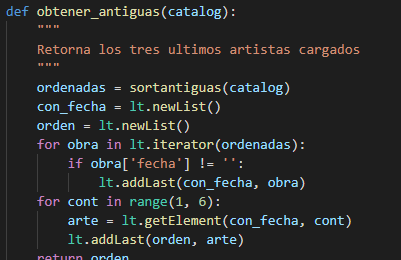
* O(n)
  + - Quinta función (suma\_peso):



* O(n)
  + - Sexta función (obtener costosas):



* O(n)
  + - Séptima función (obtener antiguas):



* O(n)
  + - COMPLEJIDAD GENERAL: O(n) + O(n) + O(n)+O() + O(n)+ + O(n) + O(n) = O(7n+) = O(n)
* **Pruebas de requerimiento:**

