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---------->

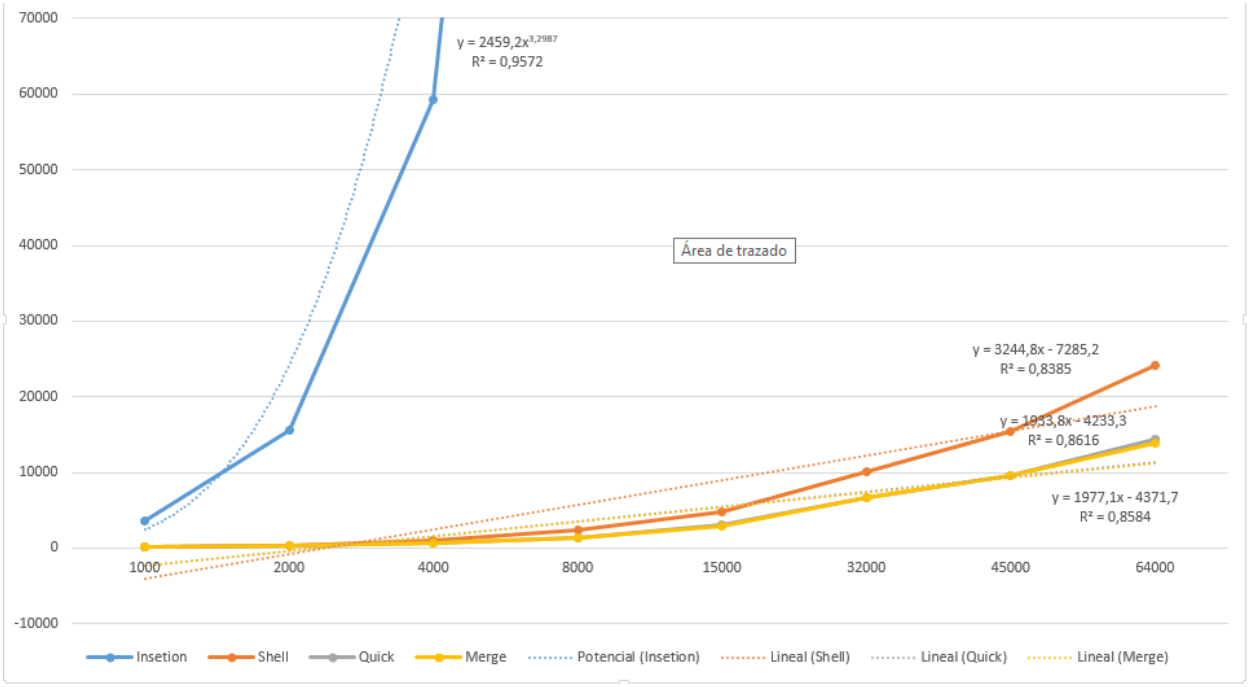
Requerimiento 4 (Individual)= José David Martínez Oliveros

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

	Maquina 1	Maquina 2
Procesadore	Intel Core i7 2630QM (2.0 Ghz)	AMD Ryzen 5 3450U with Radeon Vega Mobile Gfx. 2.10GHz
Memoria (RA	16 GB	12 GB
Sistema Ope	Windows 7 Professional (64 Bits)	Windows 10 (64 Bits)

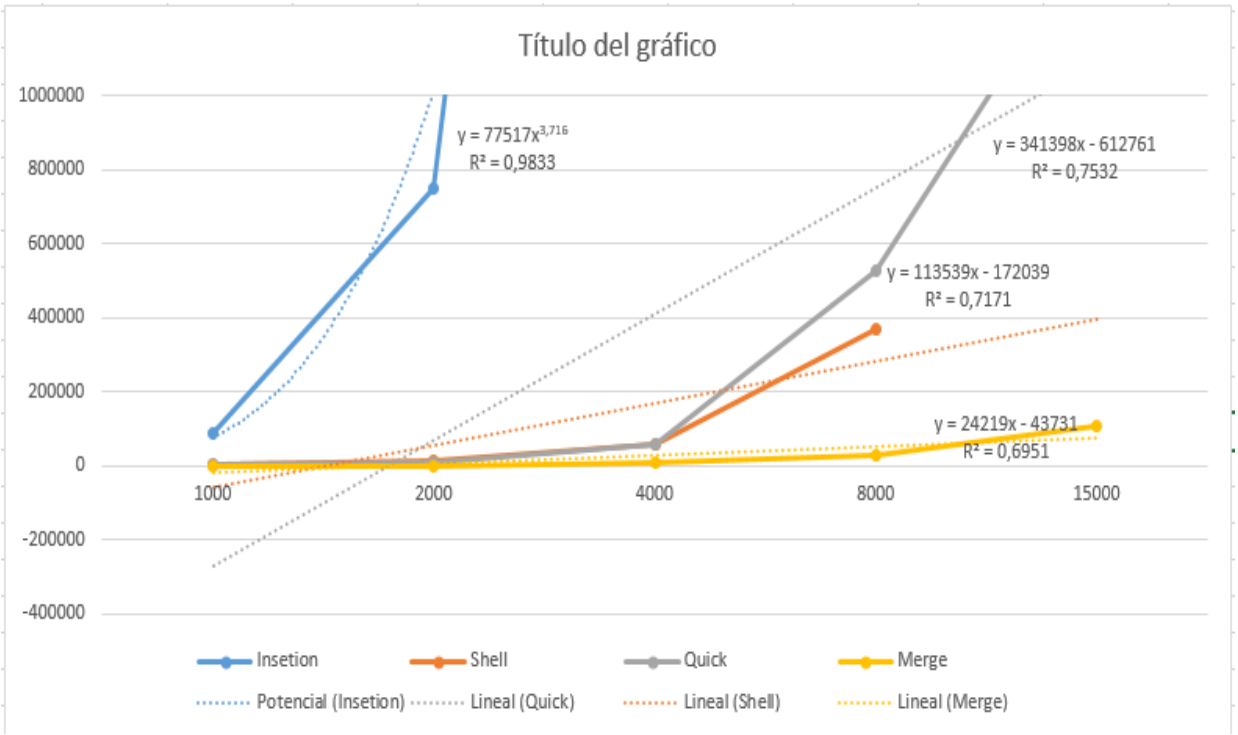
- **Pruebas de ordenamientos:**
  - **Maquina 1:**

	Insetion	Shell	Quick	Merge
1000	3640,63	203,13	125	140,63
2000	15531,25	421,88	281,25	296,88
4000	59328,13	1046,87	625	734,38
8000	238218,75	2421,87	1421,88	1375
15000	812703,13	4718,75	3093,75	2937,5
32000		10031,25	6703,13	6750
45000		15484,38	9609,38	9656,25
64000		24203,125	14343,75	13859,38



- **Maquina 2:**

	Insetion	Shell	Quick	Merge
1000	86828,13	4515,63	4671,88	546,88
2000	749125	16093,75	11296,88	1375
4000	5579500	57421,88	59078,13	6875
8000		369203,13	529625	27000
15000			1452500	108828
32000				272218,75
45000				1090859,38



- **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):

```
def nacidos_rango(catalog, año_inicial, año_final):

    artistas = catalog['nacidos_primero']
    booknacidos_rango = lt.newList()
    for artista in lt.iterator(artistas):
        if año_inicial <= int(artista['BeginDate']) and año_final >= int(artista['BeginDate']):
            lt.addLast(booknacidos_rango,artista)
    return booknacidos_rango
```

      - O(n)
    - Segunda función (obtener_ultimos_nacidos):

```
def obtener_ultimos_nacidos(catalog):
    """
    Retorna los tres  ultimos artistas nacidos
    """
    ultimostres = lt.newList()
    for cont in range(lt.size(catalog)-2, lt.size(catalog)+1):
        arte = lt.getElement(catalog, cont)
        lt.addLast(ultimostres, arte)
    return ultimostres
```

      - O(n)
    - Tercera función (obtener_primeros_nacidos):

```
def obtener_primeros_nacidos(catalog):
    """
    Retorna los tres  primeros artistas nacidos
    """

    primeros_tres = lt.newList()
    for cont in range(1, 4):
        arte = lt.getElement(catalog, cont)
        lt.addLast(primeros_tres, arte)
    return primeros_tres
```

      - O(n)
    - COMPLEJIDAD GENERAL: O(n) + O(n) + O(n) = O(3n) = O(n)

- **Requerimiento 2:**
  - Primera función (sortobras):

```
def sortobras(catalog):

    obras = catalog['obras_orden']
    sin_fecha = lt.newList('ARRAY_LIST')

    for p in lt.iterator(obras):
        if p['DateAcquired'] != '':
            lt.addLast(sin_fecha,p)
    sorted_list = merge.sort(sin_fecha, cmpArtworkByDateAcquired)
    return sorted_list
```

    - O(nlog$n$)
  - Segunda función (obras_rango):

```
def obras_rango(catalog, año_inicial, año_final):

    obras_rango = lt.newList()
    for obra in lt.iterator(catalog):
        año_inicial_nuevo = int((date.fromisoformat(año_inicial.replace('/','-'
        año_final_nuevo = int((date.fromisoformat(año_final.replace('/','-'))).
        año_adquisicion = int((date.fromisoformat(obra['DateAcquired'])).strfti
        if año_inicial_nuevo < año_adquisicion and año_final_nuevo >= año_adqui
            lt.addLast(obras_rango,obra)
    return obras_rango
```

    - O(n)
  - Tercera función (obras_compradas):

```
def obtener_compradas(catalog):
    """
    Retorna los tres  ultimos artistas nacidos
    """
    compras = lt.newList('ARRAY_LIST')
    for p in lt.iterator(catalog):
        if 'Purchase' in p['CreditLine'] or 'purchase' in p['CreditLine']:
            lt.addLast(compras, p)
    return compras
```

    - O(n)

- Cuarta función (obtener_primeras_obras):

```
def obtener_primeras_obras(catalog):
    """
    Retorna los tres  primeros artistas nacidos
    """

    primeros_tres = lt.newList('ARRAY_LIST')
    for cont in range(1, 4):
        arte = lt.getElement(catalog, cont)
        lt.addLast(primeros_tres, arte)
    return primeros_tres
```

  - O(n)

- Quinta función (obtener_ultimas_obras):

```
def obtener_ultimas_obras(catalog):
    """
    Retorna los tres  ultimos artistas nacidos
    """
    ultimostres = lt.newList('ARRAY_LIST')
    for cont in range(lt.size(catalog)-2, lt.size(catalog)+1):
        arte = lt.getElement(catalog, cont)
        lt.addLast(ultimostres, arte)
    return ultimostres
```

  - O(n)

- COMPLEJIDAD GENERAL: O(n) + O(n) + O(n)+O($n\ \log n$) + O(n) = O(5n+log$n$) = O(nlog$n$)

- **Requerimiento 3:**
  - Primera función (consulta_codigo):

```
def consulta_codigo(catalog,nombre):

    artistas = catalog['artista']
    obras = catalog['obras_ordenadas']
    codigo = ''
    for artista in lt.iterator(artistas):
        if nombre.lower().strip() in artista['
            codigo = artista['ConstituentID']

    artista_final = ''
    for artista in lt.iterator(obras):
        if codigo == artista['codigo'].replace
            artista_final = artista
    return artista_final
```

    - O(n)

- Segunda función (cantidad_tecnicas):

```
def cantidad_tecnicas(artistas):

    cantidad_de_tecnicas_veces = lt.newList('AR
    tecnicas_final = lt.newList('ARRAY_LIST')
    for partes in lt.iterator(artistas['obras']
        lt.addLast(tecnicas_final,partes['Mediu

    for i in lt.iterator(tecnicas_final):
        posauthor = lt.isPresent(cantidad_de_te
        if posauthor > 0:
            artista = lt.getElement(cantidad_de_
            artista['Cantidad'] += 1
        else:
            artista = newTecnica(i)
            artista['Cantidad'] = 1
            lt.addLast(cantidad_de_tecnicas_vec

    k = 0
    for p in lt.iterator(cantidad_de_tecnicas_v
        if int(p['Cantidad']) > k:
            k = p['Cantidad']
            maximo = p['Tecnica']

    return maximo,cantidad_de_tecnicas_veces
```

  - O(n)
- Tercera función (consulta_obras):

```
def consulta_obras(artistas,tecnica):

    obras = lt.newList('')
    for obra in lt.iterator(artistas['obras']):
        if obra['Medium'] == tecnica:
            lt.addLast(obras,obra)

    return obras
```

  - O(n)
- COMPLEJIDAD GENERAL: O(n) + O(n) + O(n) = O(3n) = O(n)

- **Requerimiento 4:**
  - función (consulta_codigo):

```
def dicc_orden(catalog_mayor):
    obras = catalog_mayor["obra_de_arte"]
    dicc_todo = {}

    for obra in lt.iterator(obras):
        id = obra['ObjectID']

        if id != '':
            codigos = obra['ConstituentID']
            nuevos_codigos = codigos.replace("[","")
            nuevos_codigos = nuevos_codigos.replace("]","")
            nuevos_codigos = nuevos_codigos.split(",")
            artistas = catalog_mayor['artista']

            for codigo in nuevos_codigos:
                nuevo = codigo.strip()



                for p in lt.iterator(artistas):

                    if nuevo == p['ConstituentID']:
                        if p['Nationality'] == '' or p['Nationality'] == "Nati
                            nacionalidad = 'Unknown'
```

- O(o(n^2 + log n))

- **Requerimiento 5:**
  - Primera función (filtrar_depto):

```
def filtrar_depto(catalog, departamento):

    obras = lt.newList()
    for p in lt.iterator(catalog['obras_a_llevar']):
        if p['Department'].lower() == departamento.lower():
            lt.addLast(obras, p)

    return obras
```

- O(n)

  - Segunda función (Calcular_transporte):

```
def calculo_de_transporte(catalog):

    obras = lt.newList('ARRAY_LIST')
    for obra in lt.iterator(catalog):

        peso = obra['Weight (kg)']
        altura = obra['Height (cm)']
        ancho = obra['Width (cm)']
        profundidad = obra['Depth (cm)']
        longitud = obra['Length (cm)']
        diametro = obra['Diameter (cm)']

        if (altura == 0 or altura == '') and (ancho == 0 or ancho == ''):
            costo = 48.00
```

- O(n)

- Tercera función (sortcostos):

```
def sortcostos(catalog):
    orden = merge.sort(catalog, comparacostos)
    return orden
```

- O(n$\log n$)

- Cuarta función (suma_costo):

```
def suma_costo(catalog):

    suma = 0
    for p in lt.iterator(catalog):
        suma += p['costo']

    return float(suma)
```

- O(n)

- Quinta función (suma_peso):

```
def suma_peso(catalog):

    suma = 0
    for p in lt.iterator(catalog):
        suma += float(p['peso'])

    return float(suma)
```

- O(n)

- Sexta función (obtener costosas):

```
def obtener_costosas(catalog):
    """
    Retorna los tres ultimos artistas cargados
    """
    costosas = lt.newList()
    for cont in range(1, 6):
        arte = lt.getElement(catalog, cont)
        lt.addLast(costosas, arte)
    return costosas
```

- O(n)

- Séptima función (obtener antiguas):

```
def obtener_antiguas(catalog):
    """
    Retorna los tres ultimos artistas cargados
    """
    ordenadas = sortantiguas(catalog)
    con_fecha = lt.newList()
    orden = lt.newList()
    for obra in lt.iterator(ordenadas):
        if obra['fecha'] != '':
            lt.addLast(con_fecha, obra)
    for cont in range(1, 6):
        arte = lt.getElement(con_fecha, cont)
        lt.addLast(orden, arte)
    return orden
```

- O(n)

- COMPLEJIDAD GENERAL: O(n) + O(n) + O(n)+O($n \log n$) + O(n)+ + O(n) + O(n) = O(7n+log n) = O(nlog n)

- **Pruebas de requerimiento:**

Pruebas requerimiento 1				
	1800-1900	1900-1920	1945-1980	Promedio
Maquina 1	31,25	15,63	15,63	20,8366667
Maquina 2	31,25	46,88	31,25	36,46
Pruebas requerimiento 2				
	1900/01/01-2	1929/11/19-1	1950/12/01-1	Promedio
Maquina 1	2843,75	2437,5	2687,5	2656,25
Maquina 2	5406,25	5078,13	6218,75	5567,71
Pruebas requerimiento 3				
	Libero badii	Chip Lord	Vladimir Bur	Promedio
Maquina 1	31,25	15,63	15,63	20,8366667
Maquina 2	46,88	31,25	31,25	36,46
Pruebas requerimiento 4				
	obras total	obras total	obras total	Promedio
Maquina 1	1686043,75	1465449,36	1654467,39	1601986,83
Maquina 2	1710758,22	1908467,55	1658604,86	1759276,88
Pruebas requerimiento 5				
	Drawings & P	Photography	Painting & Sc	Promedio
Maquina 1	8187,5	3328,13	390,63	3968,75333
Maquina 2	17828,13	7468,75	859,375	8718,75167