j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

OBSERVACIONES DEL RETO 1

Tabla 1. Parámetros Máquinas

	Máquina 1	**Máquina 2**
Procesadores	Intel® Core™ i7-8550U CPU @ 1.8GHZ, 2.00GHz	Intel® Core™ i5-8250U CPU @ 1.6GHZ, 1800 Mhz
Memoria RAM (GB)	16GB	8.0 GB
Sistema Operativo	Windows 10 Home-64-bits	Windows 10 Home-64-bits

En la Tabla 1 se observan algunos parámetros de las máquinas utilizadas que pueden llegar a afectar el tiempo de ejecución y el tamaño de la muestra hasta la cual se alcanzan a cargar los datos. Para el análisis del reto 1 se realizará por los cuatro requerimientos:

1. Encontrar buenos videos por categoría y país (Grupal)

Parámetros Entrada:

Categoría: Se selecciona una categoría de acuerdo a su name del category-id.csv
País (country): Se selecciona un país (country) del videos-large.csv o videos-small.csv
Número de videos para listar (n): Se selecciona int(n) como la cantidad de videos que le gustaría al usuario visualizar.

Return: Debe retornar la cantidad n seleccionada de los videos con los parámetros: Nombre del video (title) − trending_date − Nombre del canal (channel_title) − publish_time − Reproducciones (views) − likes − dislikes.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

```python
def sortvideosbypais(catalog, size, pais, categoria):
    videos = catalog['videos']
    videospais = lt.newList()
    conet = 1
    start_time = time.process_time()
    for catg in lt.iterator(catalog['category']):
        if catg["name"] == categoria:
            break
        conet += 1
    categoriafinal = lt.getElement(catalog["category"], conet)
    for cont in range(1,  lt.size(catalog["videos"])):
        video = lt.getElement(videos, cont)
        if video["country"] == pais and video["category_id"] == categoriafinal["tag_id"]:

            lt.addLast(videospais, video)
    videospais = mt.sort(videospais, cmpVideosByViews)
    videospaisfinal = lt.subList(videospais, 0, size)
    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000

    return (videospaisfinal, elapsed_time_mseg)
```

Figura 1. Código implementado requerimiento 1

2. Encontrar video tendencia por país (Estudiante 2)

Parámetros Entrada:

País (country): Se selecciona un país (country) del videos-large.csv o videos-small.csv

Return: Esta función debe retornar, de acuerdo al parámetro país (country), el vídeo que más días lleva trending con la información: Nombre del video (title) − Nombre del canal (channel_title) − Pais (country) − Número de días como tendencia.

```python
def number_one_video(catalog, country):

    videos = catalog["videos"]
    sublist_country = lt.newList("ARRAY_LIST")
    start_time = time.process_time()

    for x in range(0, lt.size(catalog["videos"])):
        pays = lt.getElement(videos, x)["country"]
        if pays == country.lower():
            lt.addLast(sublist_country, lt.getElement(catalog['videos'], x))
    lt_country = sublist_country
    lt_country = mt.sort(lt_country, cmpfunctionByVideoid)

    sublist_title = lt.newList("ARRAY_LIST")
    j = lt_country["elements"]

    listu = [x["video_id"] for x in j]

    contador = 0
    initial_title = listu[0]
    for elemento in listu:
        freq = listu.count(elemento)
        if (freq > contador):
            contador = freq
            initial_title = elemento
    video_id = initial_title
    count = contador

    for x in range(1,  lt.size(videos)):
        ct = lt.getElement(videos, x)["country"]
        vid_id = lt.getElement(videos, x)["video_id"]
        if ct == country and vid_id == video_id:
            lt.addLast(sublist_title, lt.getElement(catalog['videos'], x))
    videoTrend = mt.sort(sublist_title, cmpVideosByViews)
    videoTrend = lt.subList(videoTrend, 0, 1)
    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000

    dic = {'title': videoTrend["elements"][0]["title"], "channel_title": videoTrend["elements"][0]["channel_title"],
           "country": country, "number_days": count, "elapsed_time_mseg": elapsed_time_mseg}
    return dic
```

Figura 2. Código implementado requerimiento 2

3. Encontrar video tendencia por categoría (Estudiante 1)

Parámetros Entrada:

Categoria(category_name): *se ingresa la categoría de video que se quiere filtrar la cual debe estar en el archivo category-id.csv para encontrar el video con más días en trending dentro de esa categoría.*

```python
def sortvideosbycattrending(catalog, categoria):
    videos = catalog['videos']
    videospais = lt.newList("ARRAY_LIST")
    start_time = time.process_time()
    conet = 1
    for catg in lt.iterator(catalog['category']):
        if catg["name"] == categoria:
            break
        conet += 1
    categoriafinal = lt.getElement(catalog["category"], conet)
    for cont in range(1,  lt.size(catalog["videos"])):
        video = lt.getElement(videos, cont)
        if video["category_id"] == categoriafinal["tag_id"]:

            lt.addLast(videospais, video)
    lista = []
    dic = {
    }
    max = 0
    titulo = ""
    for pos in range(1,  lt.size(videospais)):
        videoid = lt.getElement(videospais, pos)
        if videoid["video_id"] != "#NAME?":

            lista.append(videoid["video_id"])
            dic[videoid["video_id"]] = videoid

    for x in lista:
        if lista.count(x) > max:
            max = lista.count(x)
            titulo = x

    print("dias : " + str(max))
    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    print("Elapsed Time ms:", elapsed_time_mseg)
    return (dic[titulo])
```

Figura 3. Código implementado requerimiento 3

Return:

Debe de retornar el título de video, el nombre del canal del video, su categoría, y el número de días que estuvo

4. Buscar los videos con más likes (Grupal)

Parámetros Entrada:

País (country): Se selecciona un país (country) del videos-large.csv o videos-small.csv

Número de videos para listar (n): Se selecciona int(n) como la cantidad de videos que le gustaría al usuario visualizar.

Etiqueta del video (Tag): palabra relacionada con la que se va a buscar el los tags del vídeo obtenidos del parámetro tags de videos-large.csv o videos-small.csv

Return: Regresa los vídeos ordenados del mayor número de likes al menor de acuerdo a los parámetros insertados. Retorna Nombre del video (title) − Nombre del canal (cannel_title) − publish_time − Reproducciones (views) − likes − dislikes − tags.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

```python
def VideoByTagLikes(catalog, country, size, tag):

    videosct = lt.newList("ARRAY_LIST")
    start_time = time.process_time()

    for x in range(0, lt.size(catalog["videos"])):
        ct = lt.getElement(catalog['videos'], x)["country"]
        tag_c = lt.getElement(catalog['videos'], x)["tags"]
        if ct == country and (tag in str(tag_c)) == True:
            lt.addLast(videosct, lt.getElement(catalog['videos'], x))

    videosct = mt.sort(videosct, cmpVideosByLikes)
    final_lt = lt.subList(videosct, 1, size)

    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    print("Elapsed Time:", elapsed_time_mseg)
    return (final_lt)
```

Figura 4. Código implementado requerimiento 4

Consideraciones Preliminares: Dado el análisis visto en el laboratorio 4 y 5, en conjunto con la teoría presentada en clase, se optó por utilizar el algoritmo de ordenamiento recursivo mergesort (peor caso O(nlogn). Asimismo, aunque seleccionando la opción 1 se puede cambiar el tipo de EDA, se va a utilizar "ARRAY_LIST" ya que tiene un O(1) en todas las funciones que se implementaron con el TAD lista (newlist, size, getelement,firstelement).

Resultados Obtenidos por Requerimiento

Para obtener el tiempo de ejecución total por requerimiento se escogieron parámetros bases para comparar entre ambas máquinas. Esto se hizo con el fin de ver el distinto funcionamiento entre ambas máquinas y su complejidad.

Requerimiento 1
Categoría: Music
País: Canadá
N: 3

```
Indique tamaño de la muestra: 3
Indique el pais: canada
Indique la categoria: Music
18.28.05 ZAYN - Entertainer (Official Video) 2018-05-23T16:00:03.000Z ZaynVEVO 9501171 498326 12238
Elapsed Time ms: 0.0
18.28.05 ZAYN - Entertainer (Official Video) 2018-05-23T16:00:03.000Z ZaynVEVO 9501171 498326 12238
Elapsed Time ms: 0.0
17.26.12 Diljit Dosanjh | Raat Di Gedi (Official Video) Neeru Bajwa | Jatinder Shah | Arvindr Khaira 2017-12-23T11:32:18.000Z Speed Records 6973988 219682 6331
```

Figura 5. Resultado Obtenido req 1

Requerimiento 2
País: Canadá

```
Este es el req 2
Ingrese el país al que le gustaría ver el vídeo con más días como tendencia:
canada
{'title': 'Post Malone - rockstar ft. 21 Savage', 'channel_title': 'PostMaloneVEVO', 'country': 'canada', 'number_days': 8, 'elapsed_time_mseg': 133765.625}
```

Figura 6. Resultado Obtenido req 2

Requerimiento 3
Categoría: Music

```
Indique la categoria: Music
dias : 92
Elapsed Time ms: 135671.875
Childish Gambino - This Is America (Official Video) ChildishGambinoVEVO 10
```

Figura 7. Resultado requerimiento 3

Requerimiento 4
N: 3
País: Canadá
Tag: 2018

```
{'Title': 'VENOM - Official Trailer (HD)', 'channel_title': 'Sony Pictures Entertainment', 'Publish_time': '2018-04-24T03:45:03.000Z', 'views': '53071887', 'Likes': '1243479', 'Dislikes': '44414', 'Tags': '"Venom"|"Venom Movie"|"Venom (2018)"|"Marvel"|"Marvel Comics"|"Planet of the Symbiotes"|"Eddie Brock"|"Tom Hardy"|"Ruben Fleischer"|"Spider-man"|"Spider-man: Homecoming"|"Michelle Williams"|"Jenny Slate"|"Riz Ahmed"|"Spider-man Spinoff"|"We Are Venom"|"Peter Parker"|"Sony Pictures Entertainment"|"film"|"movie"|"official"|"official venom movie trailer"|"official trailer"|"sony pictures venom"'}
{'Title': 'VENOM - Official Trailer (HD)', 'channel_title': 'Sony Pictures Entertainment', 'Publish_time': '2018-04-24T03:45:03.000Z', 'views': '51436997', 'Likes': '1229060', 'Dislikes': '43653', 'Tags': '"Venom"|"Venom Movie"|"Venom (2018)"|"Marvel"|"Marvel Comics"|"Planet of the Symbiotes"|"Eddie Brock"|"Tom Hardy"|"Ruben Fleischer"|"Spider-man"|"Spider-man: Homecoming"|"Michelle Williams"|"Jenny Slate"|"Riz Ahmed"|"Spider-man Spinoff"|"We Are Venom"|"Peter Parker"|"Sony Pictures Entertainment"|"film"|"movie"|"official"|"official venom movie trailer"|"official trailer"|"sony pictures venom"'}
{'Title': 'Dua Lipa - IDGAF (Official Music Video)', 'channel_title': 'Dua Lipa', 'Publish_time': '2018-01-12T12:00:07.000Z', 'views': '24609441', 'Likes': '1210593', 'Dislikes': '31214', 'Tags': '"dua lipa"|"idgaf"|"dl1"|"I don\'t give a fuck"|"dua idgaf"|"dua lipa official"|"dua new video"|"dua lipa video"|"dua leepa"|"warner bros records"|"warner Bros"|"so i cut you off"|"dua lipa i dont give a fuck"|"Pop"|"Dance"|"2018"|"Debut Album"|"I Don\'t Give A F"|"Dueling Duas"|"Dueling Lipas"|"Double Dua"|"Double Lipa"|"Dueling Dua Lipas"|"Double Dua Lipas"'}
```

Figura 8. Resultado requerimiento 4

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

Resultado Requerimientos

Tabla 2. Parámetros Máquinas

	Maquina 1	**Máquina 2**	
Tamaño de la muestra (n) (ARRAY_LIST)	Tiempo (ms)	Tiempo (ms)	Complejidad
Req 1	5718.75	4390,25	O(n)
Req 2	34718.75	129437,5	O(n)
Req 3	4914.625	138187,5	O(n)
Req 4	890.625	765,25	O(n)

Tanto para el código utilizado para el primer y cuarto requerimiento se utiliza un único ciclo para obtener los parámetros requeridos en cada función, además se genera un merge sort de la lista resultante. Debido a esto, en el peor de los casos se deberá recorrer la longitud total de la lista; por lo que, el orden de crecimiento dependerá de la cantidad de elementos n -> O(n). En el caso del requerimiento 2 y 3 sucede algo similar, pues se realiza un filtro ya sea por categoría o país el cual depende de la cantidad de elementos n que contega la lista inicial. Esto hace que se recorra la lista de manera lineal para generar un primer filtro, también es necesario generar una nueva lista la cual debe ser nuevamente recorrida para poder hacer el filtrado final.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

Tabla 1

Tamaño de la muestra (LINKED_LIST)	Insertion Sort (ms)	Selection Sort (ms)	Shell Sort (ms)	Merge Sort (ms)	Quick Sort (ms)
1000	99593.75	69140.62	2828.15	31.25	31.25
2000	636390.625	567015.625	16718.75	78.12	95.32
4000			82656	156.25	218.75
8000			413859.3	328.12	426.82
16000			2008281.2	756.56	718.75
32000				1531.25	1489.62
64000				3218.75	3500.26
128000				7031.25	8141.35
256000				14812.54	
512000					

Tabla 2. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion sort	x	
Selection sort	x	
Shell sort	x	
Merge Sort	x	
Quick Sort	x	

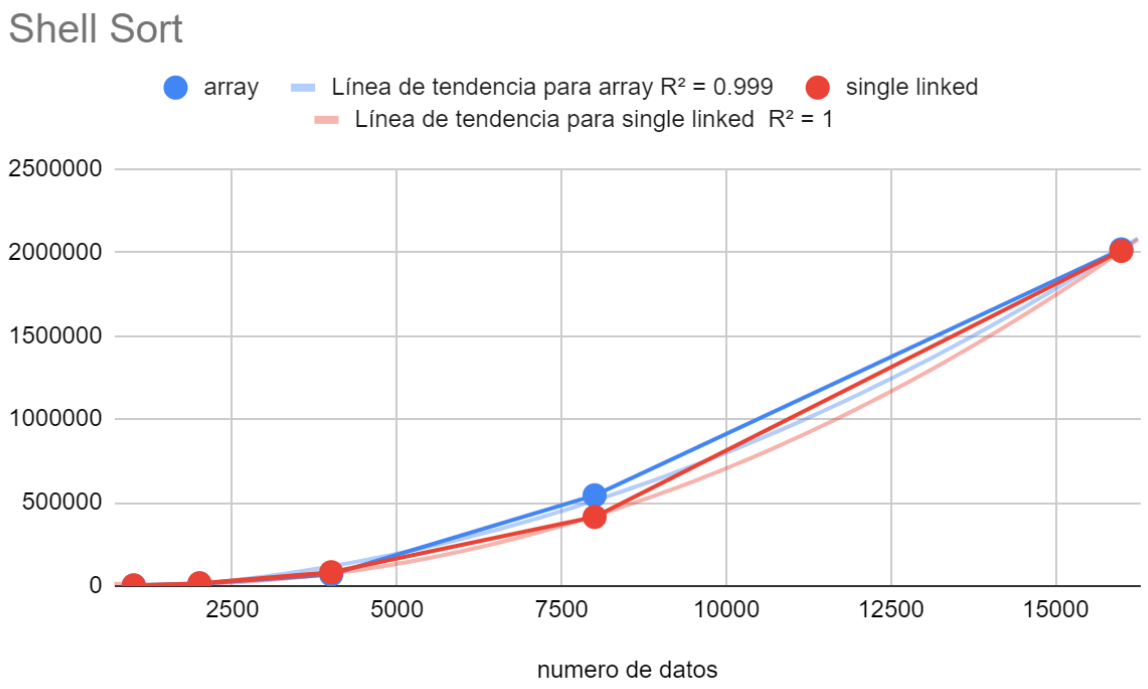
Tabla 3. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

- Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 1.**

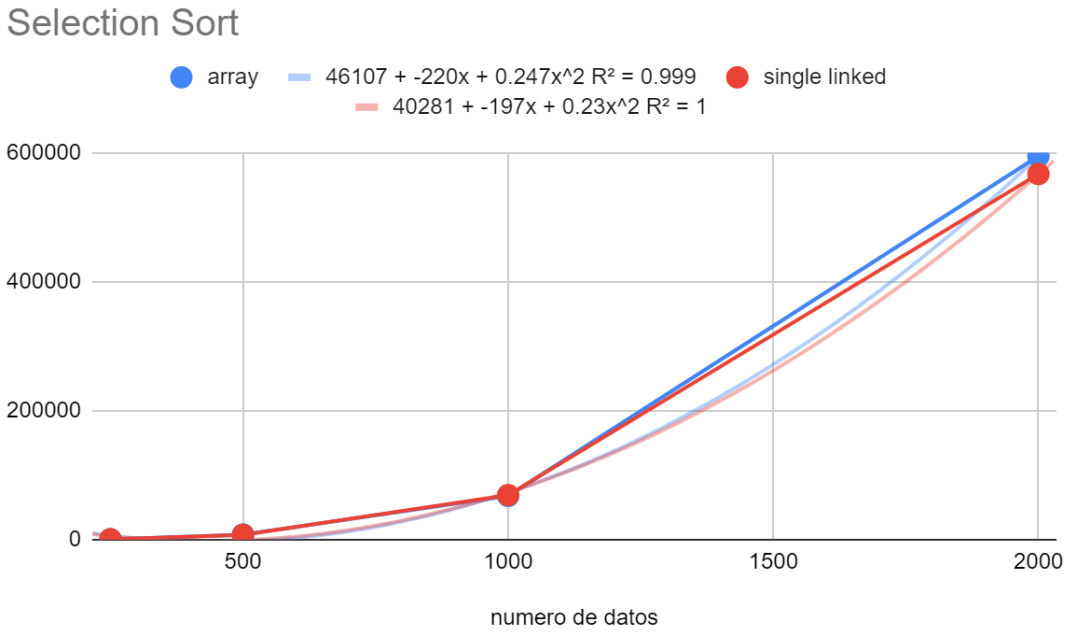
- Comparación de rendimiento para Shell Sort.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



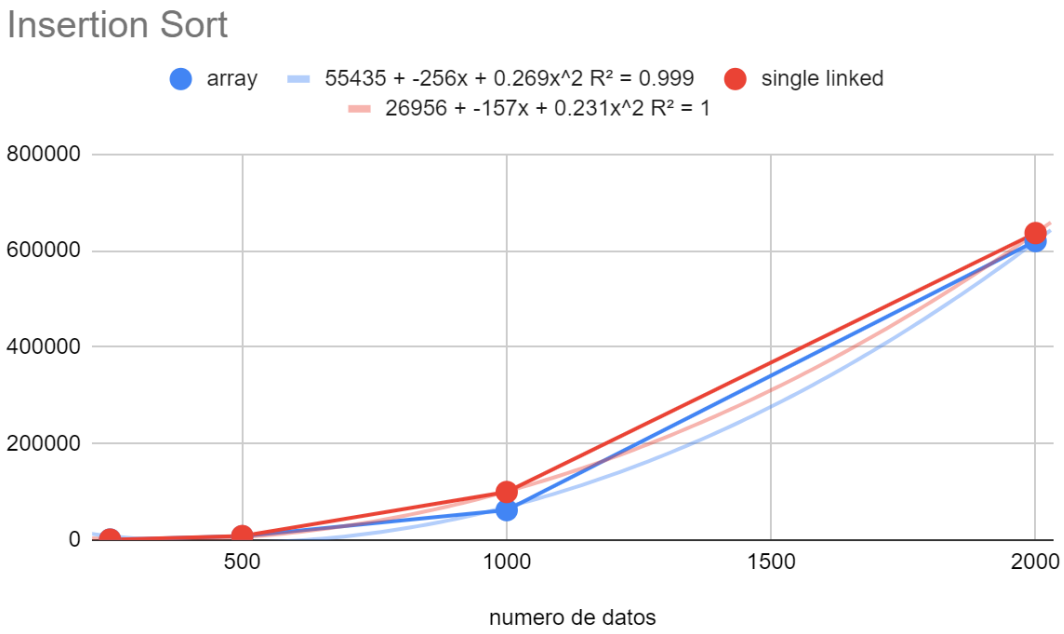
- Comparación de rendimiento para Selection Sort.



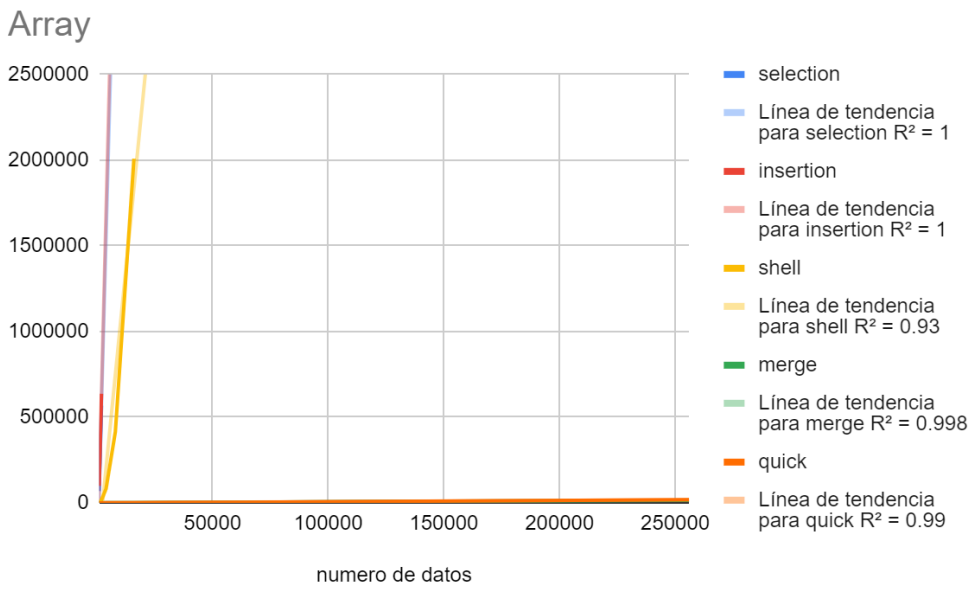
- Comparación de rendimiento para Insertion Sort.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



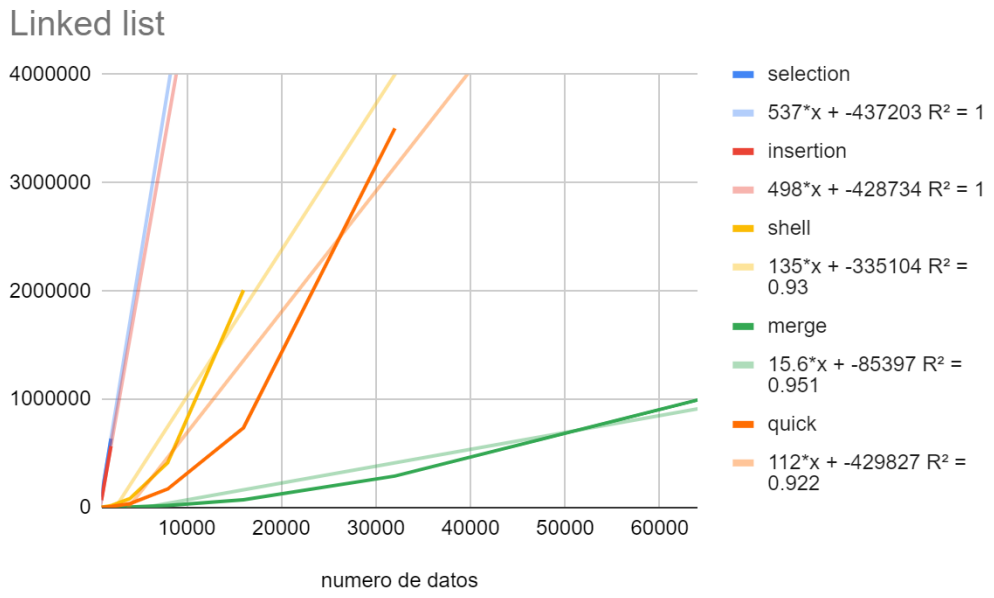
- Comparación de rendimiento ARRAYLIST.



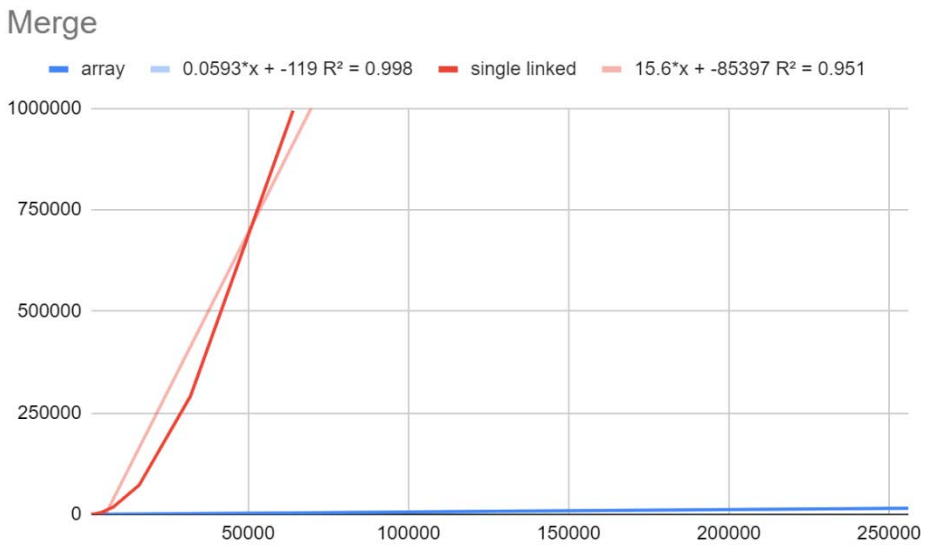
- Comparación de rendimiento LINKED_LIST.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



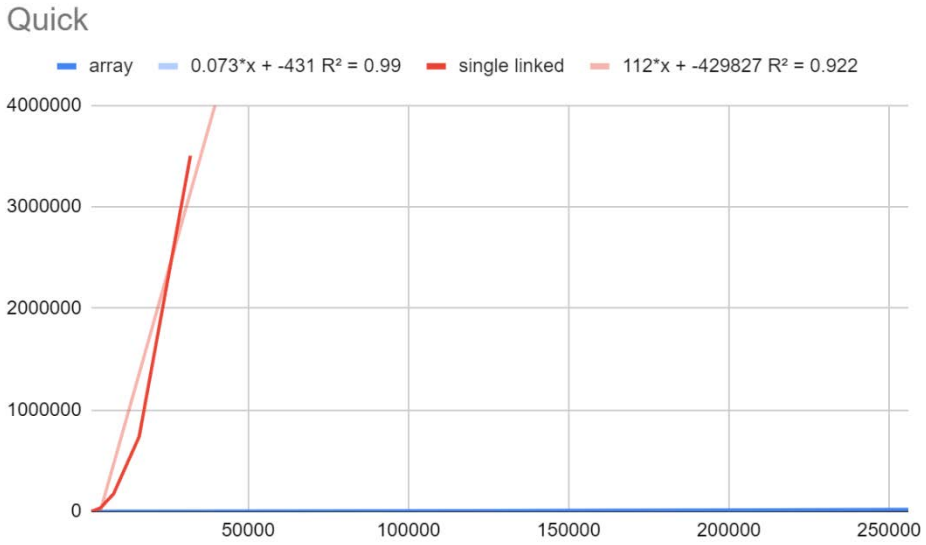
- Comparación de rendimiento para Merge Sort.



- Comparación de rendimiento para Quick Sort.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



Maquina 2

Resultados

Tamaño de la muestra (ARRAY_LIST)	Insertion Sort (ms)	Selection Sort (ms)	Shell Sort (ms)	Merge Sort (ms)	Quick Sort (ms)
1000	708,33	854,16	31,25	26,04	31,25
2000	2942.708	2963,54	93,75	67,7	62,5
4000	11239,58	13557,29	203,125	145,83	135,41
8000	49604,16	54479,16	416,66	281,25	276,04
16000	206218.75	239218,75	1088,54	687,5	630,2
32000	781343,75	941031,25	2859,375	1354,16	1515.625
64000	Supera los 30 minutos	Supera los 40 min	6104,16	1946,16	2905,33
128000	-	-	16088,54	5614,583	6354,16
256000	-	-	35458,33	12713,54	13296,875
512000	-	-	-	-	-

Tabla 4. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación arreglo.

Tamaño de la muestra (LINKED_LIST)	Insertion Sort (ms)	Selection Sort (ms)	Shell Sort (ms)	Merge Sort (ms)	Quick Sort (ms)
1000	52125	40260,41	2864,375	250	1494,79
2000	386609,25	368812,5	13702,95	927,08	7833,3
4000	3271484.38	2907437,5	56718,75	3875	32307,29

8000	Tiempo mayor a 45 min	Tiempo mayor a 45 min	289171,875	16255,20	159906,25
16000	-	-	1294656,25	62911,45	683203,125
32000	-	-	Excede el tiempo de 30 min	260796,875	3090109,375
64000	-	-	-	963140,625	Excede tiempo de 45 min
128000	-	-	-	Excede 45 minutos	-
256000	-	-	-	-	-
512000	-	-	-	-	-

Tabla 5. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion sort	x	
Selection sort	x	
Shell sort	x	
Merge Sort	x	
Quick Sort	x	

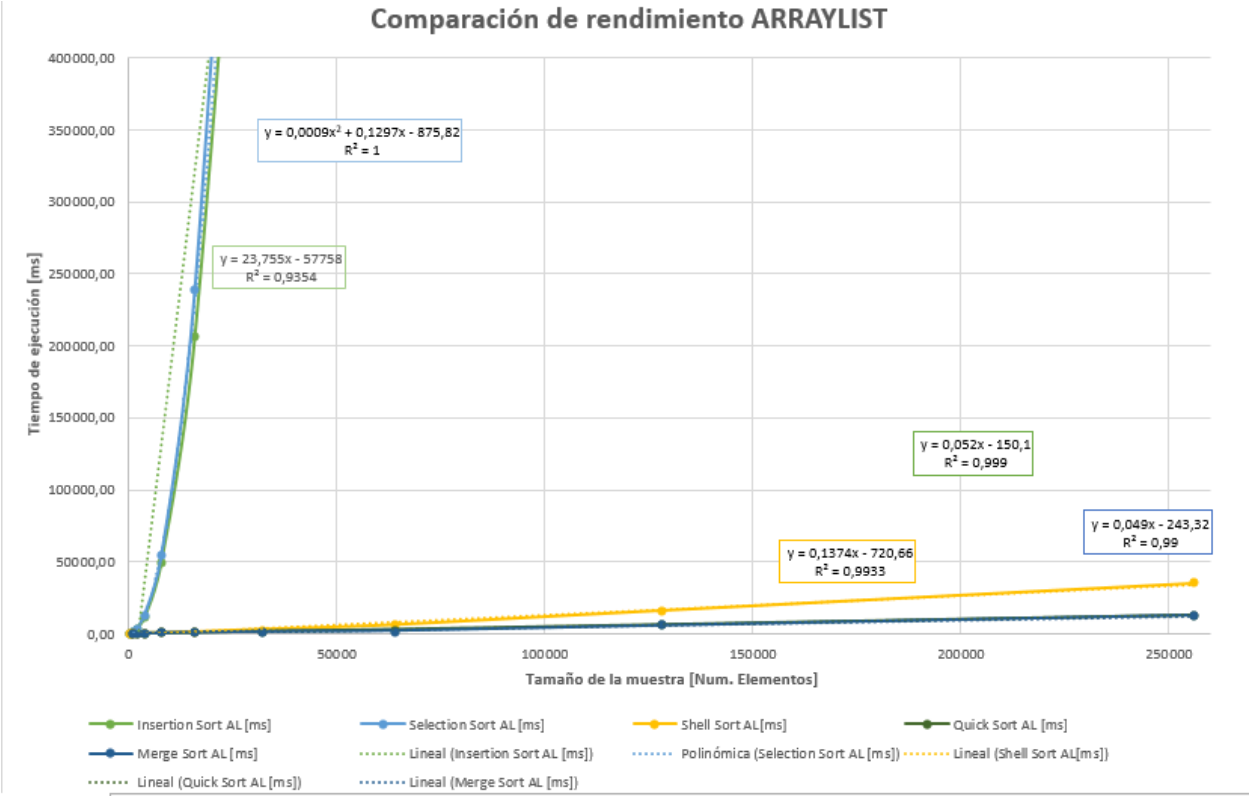
Tabla 6. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

Graficas

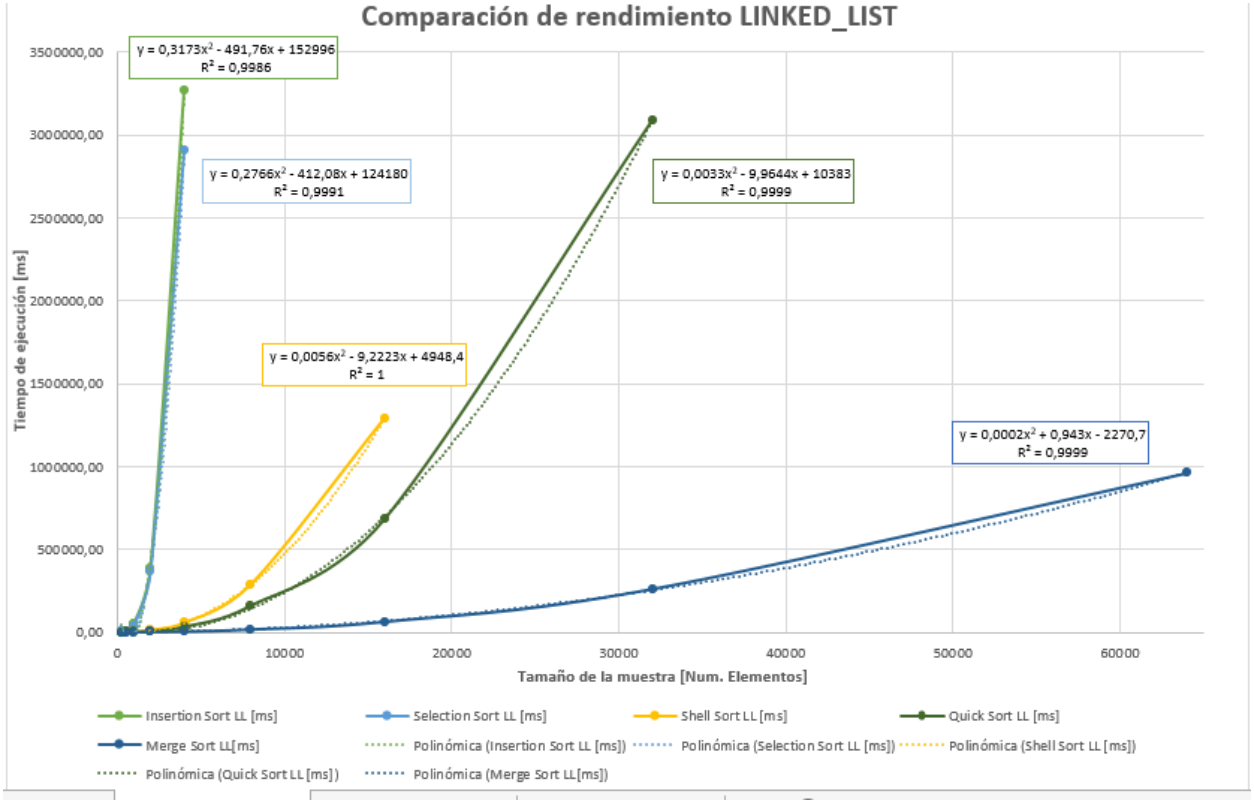
- Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 2.**
    - Comparación de rendimiento ARRAYLIST.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



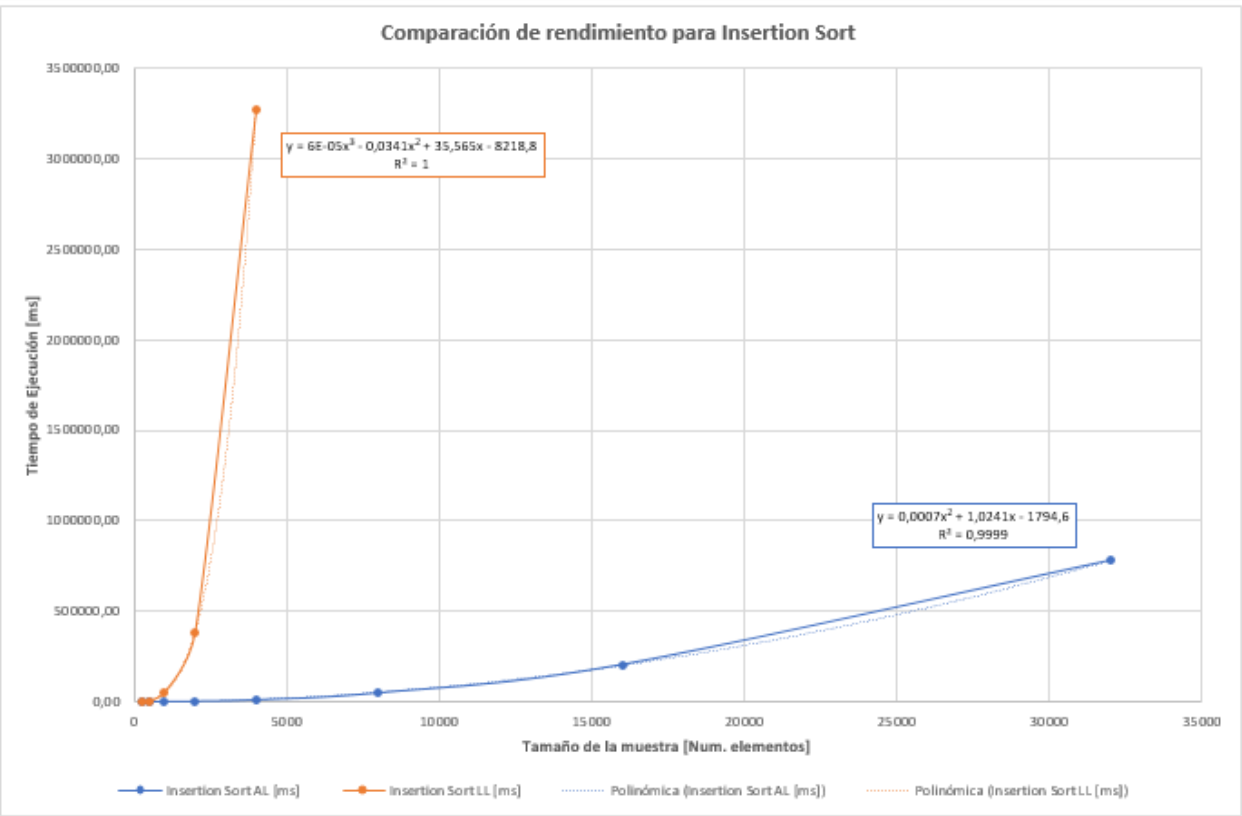
- Comparación de rendimiento LINKED_LIST.



j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

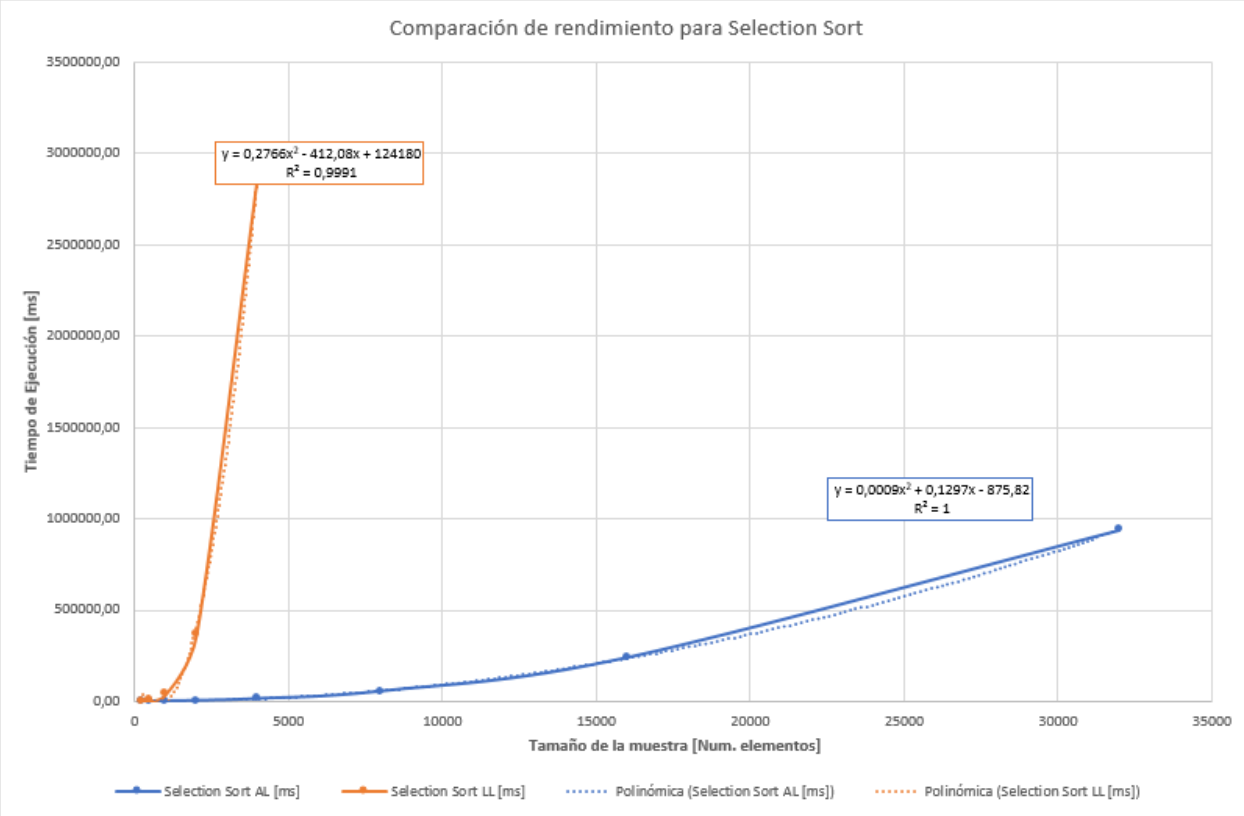
- Comparación de rendimiento para Insertion Sort.



- Comparación de rendimiento para Selection Sort.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

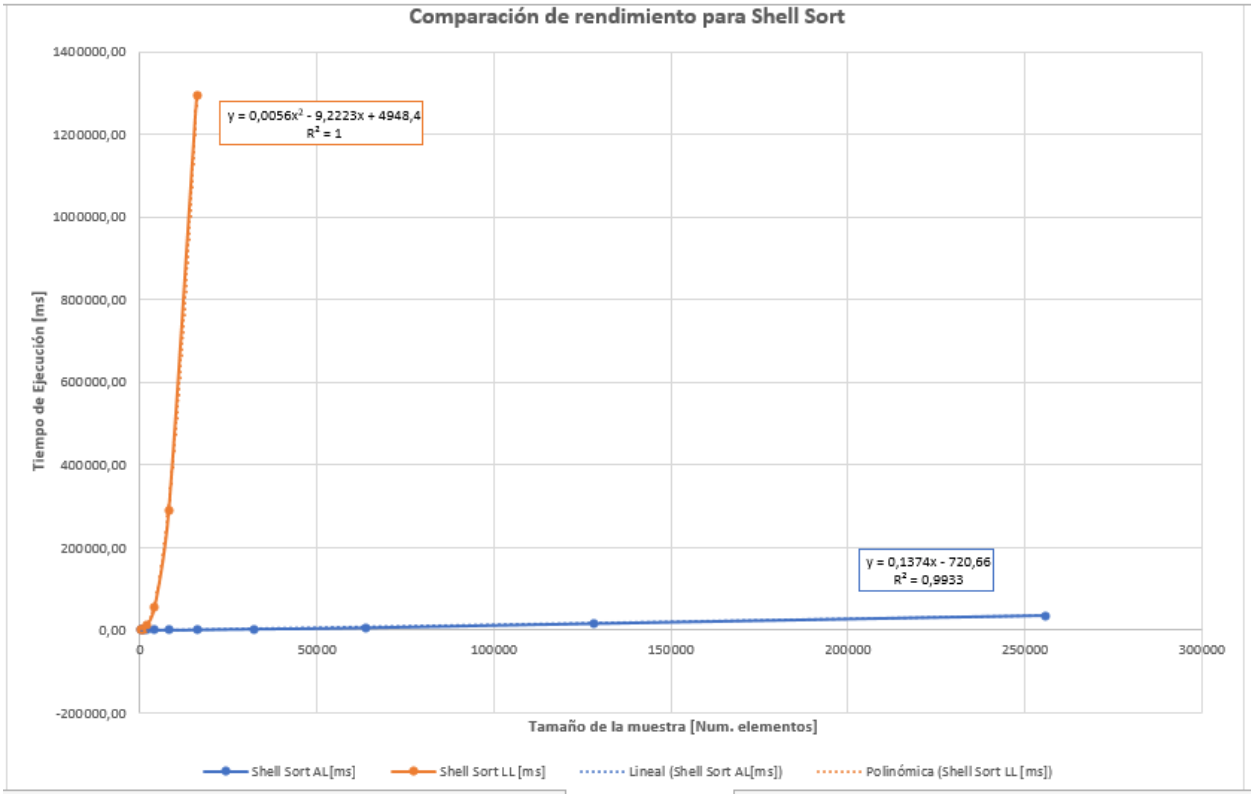
Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



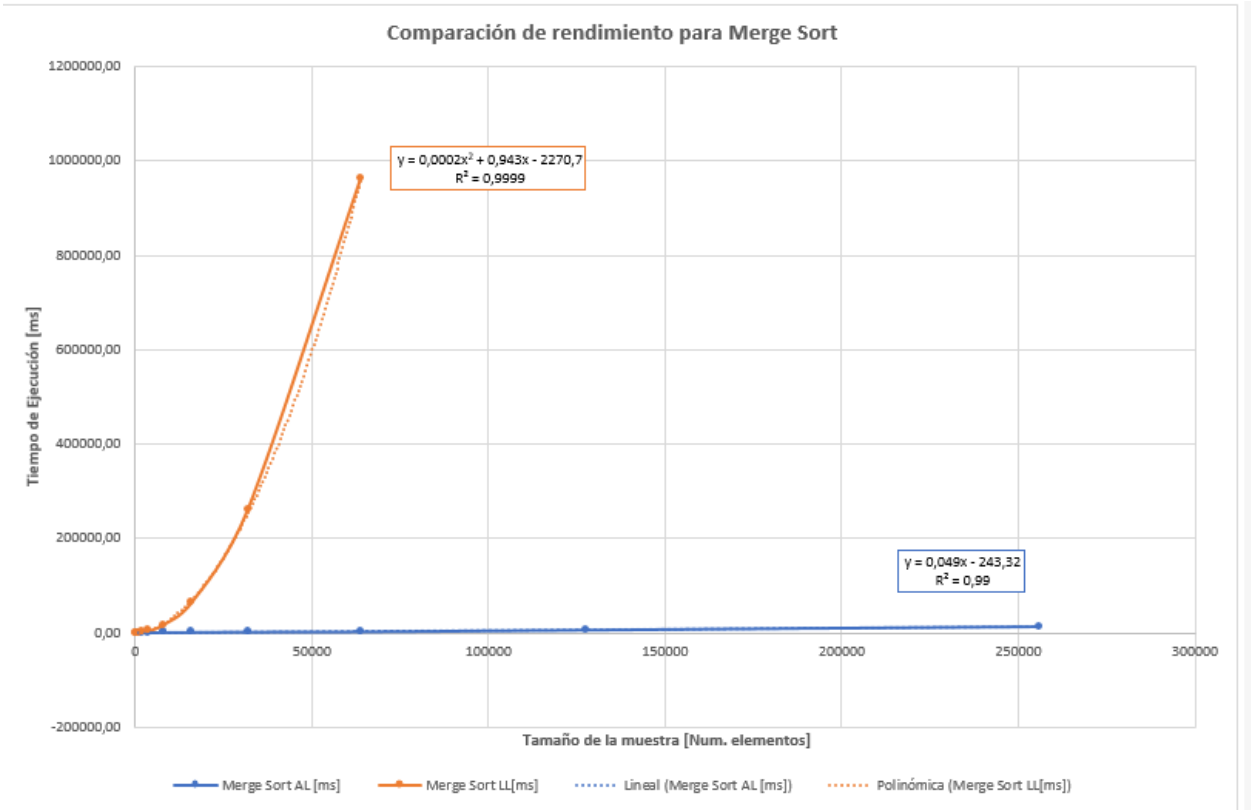
- Comparación de rendimiento para Shell Sort.

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531



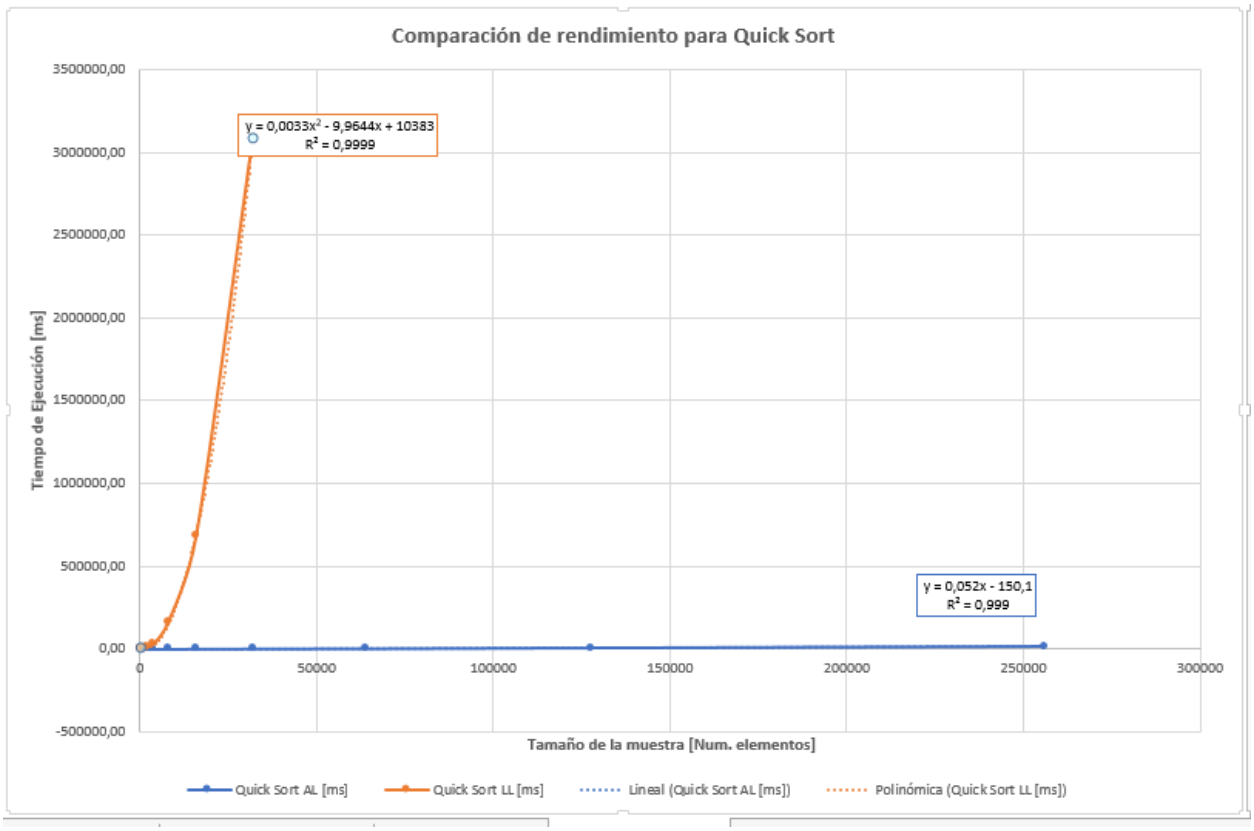
- Comparación de rendimiento para Merge Sort.



j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

- Comparación de rendimiento para Quick Sort.



Análisis

1. ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?

Independientemente de la estructura de datos, los tipos de ordenamiento iterativo y recursivo actuaron en su mayoría de acuerdo a lo esperado y visto en la teoría. En el caso particular de selection sort actuó con O(n^2) para ambos casos, insertion O(n^2), shell, merge y quick actuaron con O(n) para array_list y O(n^2) para linked list. Por un lado, los algoritmos iterativos y quicksort actuaron en su peor caso, mientras que mergesort no actuó ni en su mejor o peor caso O (n (log n)).

2. ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?

Si existen diferencias entre máquinas especialmente en el tiempo de ejecución y el tamaño de la muestra hasta la cual se alcanzan a cargar los datos. En su mayoría, las gráficas actuaron con O(n^2).

3. De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

Las diferencias en el tiempo de ejecución se pueden deber a los parámetros mencionados en la parte superior del laboratorio como puede ser el procesador y memoria RAM. Además, el tener otras aplicaciones abiertas de escritorio pudo ser un factor que aumentó el tiempo de ejecución.

4. ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?

De acuerdo a las muestras obtenidas tanto para los ordenamientos iterativos y recursivos la mejor EDA es el array_list.

5. Para el caso analizado de ordenamiento de los videos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento estudiados (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor eficiencia en tiempo) para ordenar la mayor cantidad de videos.

Mayor eficiencia: A. Merge_Sort

B. Quick Sort

C. Shell Sort

D. Selection Sort

E. Insertion Sort