# ANÁLISIS DEL RETO

*Guillermo Antonio Villalba Escamilla, 202114000, g.villalba@uniandes.edu.co*

*Nicolás Ruiz Pérez, 202123608, n.ruizp2@uniandes.edu.co*

*Juan José Tovar, 202113204, jj.tovar@uniandes.edu.co*

## Requerimiento 1:

### Descripción

```
187 def TitlesByYear(catalog,first_year,last_year): #Función Principal Requerimiento 1
188     titlesList = lt.newList()
189     for stream in catalog:
190         for title in lt.iterator(catalog[stream]):
191             if (title["release_year"] != "unknown") and (title["type"] == "Movie"):
192                 if int(title["release_year"]) >= int(first_year) and int(title["release_year"]) <= int(last_year):
193                     title["streaming_service"] = stream
194                     lt.addLast(titlesList,title)
195     merg.sort(titlesList,cmpyear)
196     return titlesList
```

| | |
|---|---|
| **Entrada** | Catálogo, primer año, segundo año. |
| **Salidas** | Lista encadenada. |
| **Implementado (Sí/No)** | Sí |

### Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| Comparación | O(n) |
| addlast | O (1) |
| Merge sort | O(nlogn) |
| ***TOTAL*** | ***O(nlogn)*** |

### Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|
| Prueba 1 (0,5%) | 0,73 |
| Prueba 2 (5%) | 5,075 |

| | |
|---|---|
| Prueba 3 (20%) | 44,31 |
| Prueba 4 (30%) | 167,76 |
| Prueba 5 (50%) | 313,1 |
| Prueba 6 (100%) | 1217,33 |

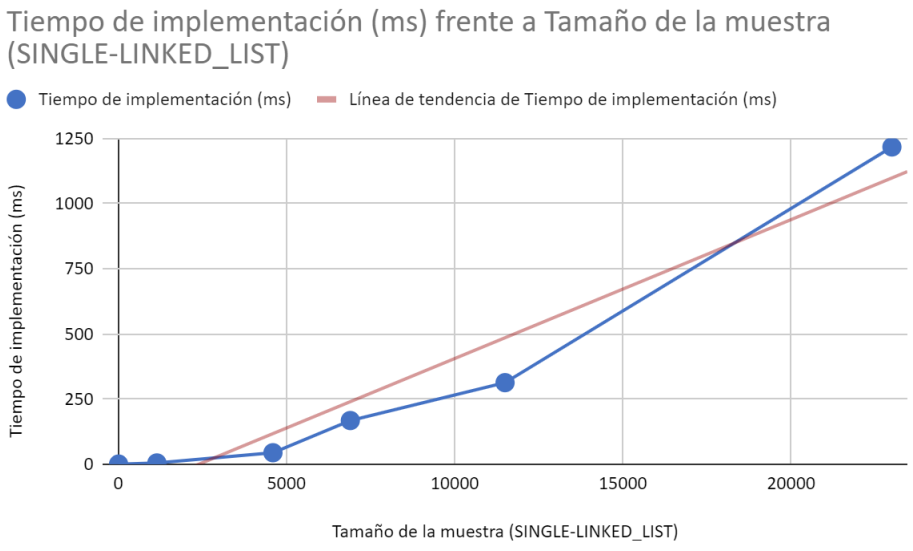
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Requerimiento 1 | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de implementación (ms)** | m | y-intercepto |
| 0,05% | 11,499 | 0,73 | 0,05331059975 | -127,6140373 |
| 5,00% | 1149,9 | 5,075 | | |
| 20,00% | 4599,6 | 44,31 | | |
| 30,00% | 6899,4 | 167,76 | | |
| 50,00% | 11499 | 313,1 | | |
| 100,00% | 22998 | 1217,33 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




## Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 2:

## Descripción

```python
def TitleByTime(catalog,firstDate,LastDate):#Función Principal Requerimiento 2
    returnlist = lt.newList()
    for streaming_platform in catalog:
        for title in lt.iterator(catalog[streaming_platform]):
            if (title["date_added"] != "unknown") and (title["type"] == "TV Show"):
                if DateCompare(title["date_added"],firstDate,LastDate) == True:
                    title["streaming_service"] = streaming_platform
                    lt.addLast(returnlist,title)
    merg.sort(returnlist, comparedate)
    return lt.size(returnlist),returnlist
```

| | |
|---|---|
| **Entrada** | Catálogo, primera fecha, segunda fecha. |
| **Salidas** | Lista encadenada. |
| **Implementado (Sí/No)** | Sí |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| Comparación | O(n) |
| Función DateCompare | O (1) |
| Merge sort | O(nlogn) |
| ***TOTAL*** | ***O(nlogn)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|
| Prueba 1 (0,5%) | 4,96 |
| Prueba 2 (5%) | 18,087 |
| Prueba 3 (20%) | 93,056 |
| Prueba 4 (30%) | 219,74 |
| Prueba 5 (50%) | 478,01 |
| Prueba 6 (100%) | 2011,77 |

### Tablas de datos

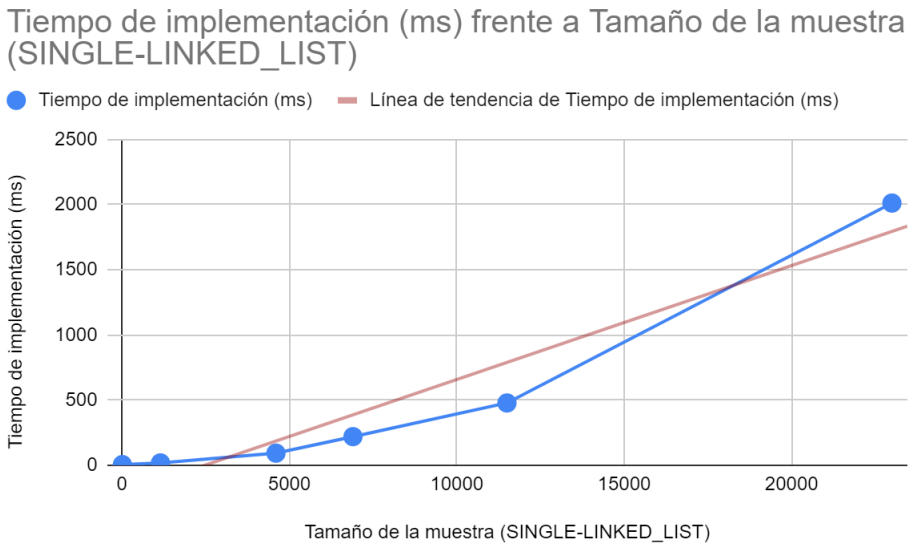
Las tablas con la recopilación de datos de las pruebas.

| **Requerimiento 2** | | |
|---|---|---|

| Porcentaje de la muestra [pct] | Tamaño de la muestra (SINGLE-LINKED_LIST) | Tiempo de implementación (ms) | m | y-intercepto |
|---|---|---|---|---|
| 0,05% | 11,499 | 4,96 | 0,08741530874 | -216,1092655 |
| 5,00% | 1149,9 | 18,087 | | |
| 20,00% | 4599,6 | 93,056 | | |
| 30,00% | 6899,4 | 219,74 | | |
| 50,00% | 11499 | 478,01 | | |
| 100,00% | 22998 | 2011,77 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




## Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 3:

## Descripción

```python
def TitlesByActor(actor,catalog): #Función Principal Requerimiento 3
    TV_count = 0
    Movie_count = 0
    titles = lt.newList()
    for streaming_service in catalog:
        for title in lt.iterator(catalog[streaming_service]):
            if (title["cast"] != "") and (actor in title["cast"]):
                if title["type"] == "Movie":
                    Movie_count += 1
                else:
                    TV_count += 1
                title["streaming_platform"] = streaming_service
                title["cast"] = title["cast"].strip()
                lt.addLast(titles,title)
    merg.sort(titles,ActorCompare)
    return titles,TV_count,Movie_count
```

| | |
|---|---|
| **Entrada** | Nombre del actor, catálogo |
| **Salidas** | Tupla (lista encadenada, numero de shows, numero de películas) |
| **Implementado (Sí/No)** | Sí |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| Comparación | O(n) |
| addlast | O (1) |
| Merge sort | O(nlogn) |
| Operador In | O (n) |
| ***TOTAL*** | ***O(nlogn)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|
| Prueba 1 (0,5%) | 0,15 |
| Prueba 2 (5%) | 0,43 |
| Prueba 3 (20%) | 1,45 |
| Prueba 4 (30%) | 2,15 |
| Prueba 5 (50%) | 4,37 |
| Prueba 6 (100%) | 12,817 |

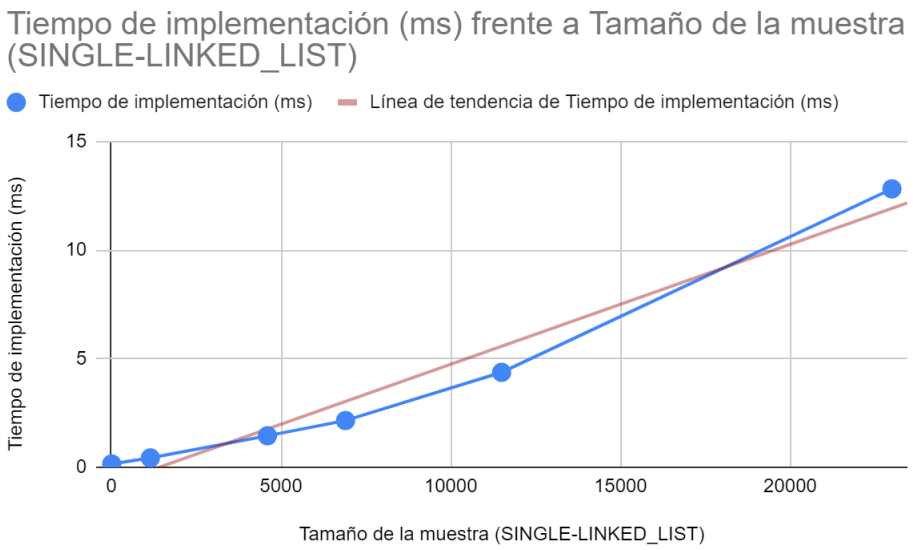
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Requerimiento 3 | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de Implementacion (ms)** | m | y-intercepto |
| 0,05% | 11,499 | 0,15 | 0,000552 1122892 | -0,7781965855 |
| 5,00% | 1149,9 | 0,43 | | |
| 20,00% | 4599,6 | 1,45 | | |
| 30,00% | 6899,4 | 2,15 | | |
| 50,00% | 11499 | 4,37 | | |
| 100,00% | 22998 | 12,817 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




# Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 4:

## Descripción

```
263 def genero(generos, catalog): #principal req 4
264     contadortv = 0
265     contadorpelicula = 0
266     generoscu = lt.newList()
267     for streaming_service in catalog:
268         for title in lt.iterator(catalog[streaming_service]):
269             if (title["listed_in"] != "" ) and (generos in title["listed_in"]):
270                 if title["type"] == "Movie":
271                     contadorpelicula += 1
272                 else:
273                     contadortv += 1
274                 title["streaming_platform"] = streaming_service
275                 title["listed_in"] = title["listed_in"].strip()
276                 lt.addLast(generoscu,title)
277     merg.sort(generoscu,cmpgen)
278     return generoscu, contadortv, contadorpelicula
```

| | |
|---|---|
| **Entrada** | Catalogo y Género a buscar |
| **Salidas** | Lista encadenada, número entero de películas y número entero de shows |
| **Implementado (Sí/No)** | Sí |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| comparación | O(n) |
| Operador in | O(n) |
| Addlast | O(n) |
| Merge Sort | O(nlogn) |
| ***TOTAL*** | ***O(nlog)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|
| Prueba 1 (0,5%) | 0,32 |
| Prueba 2 (5%) | 0,97 |
| Prueba 3 (20%) | 2,59 |
| Prueba 4 (30%) | 9,3 |
| Prueba 5 (50%) | 24,1 |
| Prueba 6 (100%) | 69,58 |

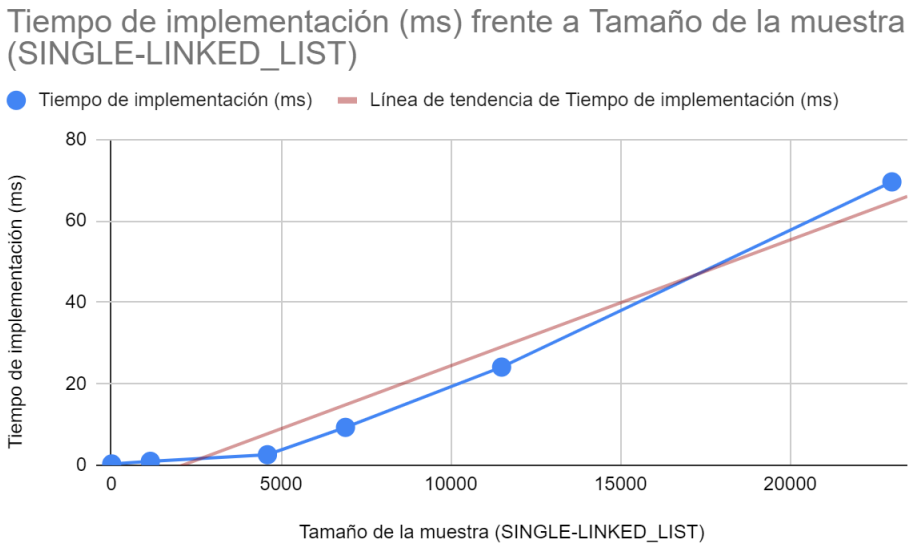
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Requerimiento 4 | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de implementación (ms)** | m | y-intersepto |
| 0,05% | 11,499 | 0,32 | 0,003091280279 | -6,486122922 |
| 5,00% | 1149,9 | 0,97 | | |
| 20,00% | 4599,6 | 2,59 | | |
| 30,00% | 6899,4 | 9,3 | | |
| 50,00% | 11499 | 24,1 | | |
| 100,00% | 22998 | 69,58 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




# Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 5:

## Descripción

```python
def producedAt(catalog,country): #Principal req 5
    movies = 0
    TV_Shows = 0

    country_catalog = lt.newList()
    for streaming in catalog:
        for title in lt.iterator(catalog[streaming]):
            title["streaming_platform"] = streaming
            for on_country in title["country"].split(","):
                on_country = on_country.strip()
                if country in on_country:
                    lt.addLast(country_catalog, title)
                    if title['type'] == 'Movie':
                        movies += 1
                    else:
                        TV_Shows += 1
    merg.sort(country_catalog,cmpCountry)

    return movies,TV_Shows,country_catalog

```

| | |
|---|---|
| **Entrada** | Catálogo, nombre del país. |
| **Salidas** | Tupla (Número de películas, numero de shows, lista encadenada) |
| **Implementado (Sí/No)** | Si |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| Comparación pais | O(n) |
| addlast | O(n) |
| Merge sort | O(nlogn) |
| Operador In | 0(n) |
| Operador [] | O(n) |
| ***TOTAL*** | ***O(nlogn)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|

| | |
|---|---|
| Prueba 1 (0,5%) | 0,07 |
| Prueba 2 (5%) | 0,41 |
| Prueba 3 (20%) | 1,81 |
| Prueba 4 (30%) | 3,42 |
| Prueba 5 (50%) | 6,64 |
| Prueba 6 (100%) | 15,48 |

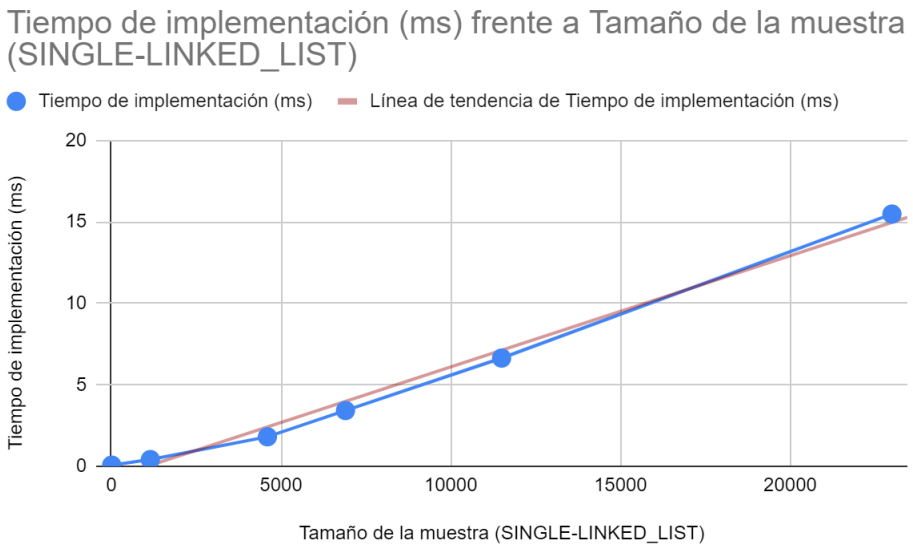
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| **Requerimiento 5** | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de implementación (ms)** | m | y-intercepto |
| 0,05% | 11,499 | 0,07 | 0,0006823035548 | -0,7242768289 |
| 5,00% | 1149,9 | 0,41 | | |
| 20,00% | 4599,6 | 1,81 | | |
| 30,00% | 6899,4 | 3,42 | | |
| 50,00% | 11499 | 6,64 | | |
| 100,00% | 22998 | 15,48 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.



## Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 6:

## Descripción

```python
def TitlesByDirector(catalog,director): #Función Principal Requerimiento 6
    type_count = {}
    streaming_count = {}
    listed_in_count = {}
    directorTitles = lt.newList()
    for streaming in catalog:
        for title in lt.iterator(catalog[streaming]):
            for i_dic in title["director"].split(","):
                i_dic = i_dic.strip()
                if i_dic == director:
                    if title["type"] not in type_count:
                        type_count[title["type"]] = 1
                    else:
                        type_count[title["type"]] += 1
                    title["streaming_platform"] = streaming
                    if title["streaming_platform"] not in streaming_count:
                        streaming_count[title["streaming_platform"]] = {"Movie":0,"TV Show":0}
                        streaming_count[title["streaming_platform"]][title["type"]] += 1
                    else:
                        streaming_count[title["streaming_platform"]][title["type"]] += 1
                    for genre in title["listed_in"].split(","):
                        genre = genre.strip()
                        if genre not in listed_in_count:
                            listed_in_count[genre] = 1
                        else:
                            listed_in_count[genre] += 1
                    lt.addLast(directorTitles,title)
    merg.sort(directorTitles,cmpTitlesByDirector)
    return directorTitles,type_count,streaming_count,listed_in_count
```

| | |
|---|---|
| **Entrada** | Catálogo, nombre del directo |
| **Salidas** | Tupla (Lista encadenada, diccionario con información del tipo de programa, diccionario con la información de las plataformas de streaming, diccionario con la información de los géneros) |
| **Implementado (Sí/No)** | Sí |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| Operador in | O(n) |
| Addlast | O(n) |
| Merge Sort | O(nlogn) |
| Operador [] | O(n) |
| ***TOTAL*** | ***O(nlogn)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| Entrada | Tiempo (ms) |
|---|---|
| Prueba 1 (0,5%) | 0,066 |
| Prueba 2 (5%) | 0,31 |
| Prueba 3 (20%) | 1,403 |
| Prueba 4 (30%) | 2,78 |
| Prueba 5 (50%) | 5,67 |
| Prueba 6 (100%) | 11,26 |

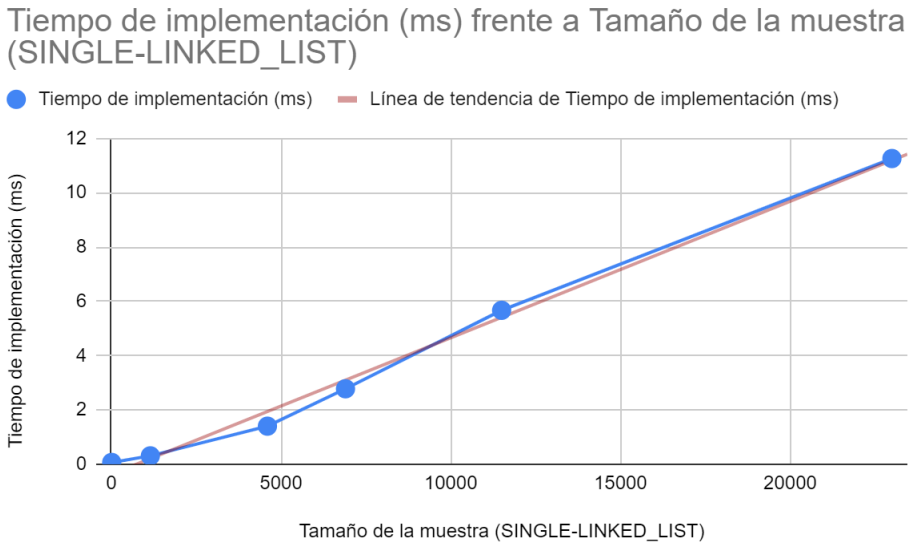
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Requerimiento 6 | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de implementación (ms)** | m | y-intersepto |
| 0,05% | 11,499 | 0,066 | 0,0005026735272 | -0,3692960145 |
| 5,00% | 1149,9 | 0,31 | | |
| 20,00% | 4599,6 | 1,403 | | |
| 30,00% | 6899,4 | 2,78 | | |
| 50,00% | 11499 | 5,67 | | |
| 100,00% | 22998 | 11,26 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.



# Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 7:

## Descripción

```python
def topGenres(catalog,TopN): # Función Princial Requerimiento 7
    genres_dict = {}
    top_genres = lt.newList()
    top_Num = lt.newList()
    for streaming in catalog:
        for title in lt.iterator(catalog[streaming]):
            title["streaming_platform"] = streaming
            for genre in title["listed_in"].split(","):
                genre = genre.strip()
                if genre not in genres_dict:
                    genres_dict[genre] = lt.newList()
                    lt.addLast(genres_dict[genre],title)
                else:
                    lt.addLast(genres_dict[genre],title)
    i = 0
    while i < int(TopN):
        max_ = None
        genreName = None
        for key in genres_dict:
            if max_ == None:
                genreName = key
                max_ = genres_dict[key]
            elif lt.size(genres_dict[key]) > lt.size(max_):
                genreName = key
                max_ = genres_dict[key]
        genres_dict.pop(genreName)
        lt.addLast(top_genres,{"listed_in": genreName,"titles": max_})
        i += 1
    for elems in lt.iterator(top_genres):
        lt.addLast(top_Num,topGenresInPlatform(elems))
    return top_Num, len(genres_dict)
```

| | |
|---|---|
| **Entrada** | Catálogo, número de Top |
| **Salidas** | Tupla (Lista encadenada, tamaño de la lista) |
| **Implementado (Sí/No)** | Sí |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Asignación | O(n) |
| Operador In | O(n) |
| Comparación | O(n) |
| Addlast | O(k) |
| Función topGenresInPlatform | O(n) |
| ***TOTAL*** | ***O(n)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|
| Prueba 1 (0,5%) | 1,70 |
| Prueba 2 (5%) | 5,63 |
| Prueba 3 (20%) | 12,52 |
| Prueba 4 (30%) | 14,71 |
| Prueba 5 (50%) | 20,08 |
| Prueba 6 (100%) | 71,33 |

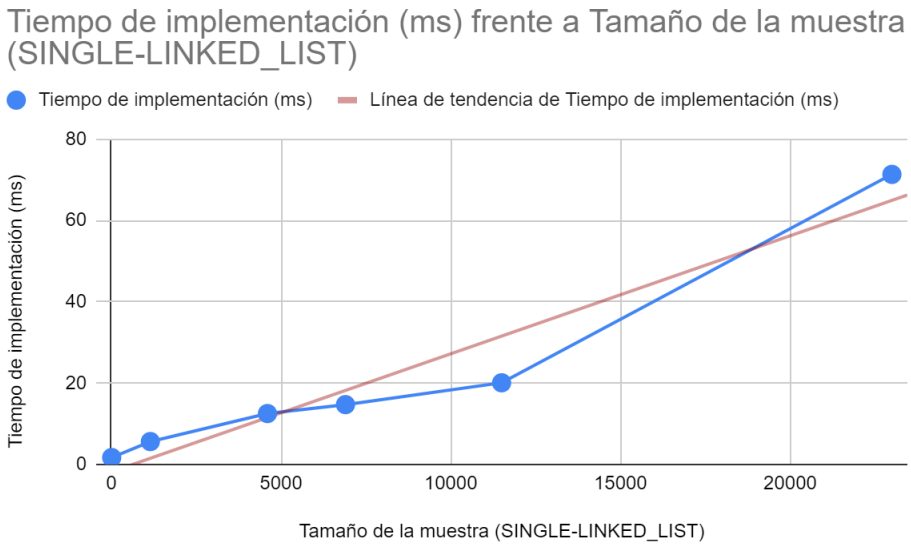
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| **Requerimiento 7** | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de implementación (ms)** | m | y-intersepto |
| 0,05% | 11,499 | 1,7 | 0,002903244755 | -1,823245218 |
| 5,00% | 1149,9 | 5,63 | | |
| 20,00% | 4599,6 | 12,52 | | |
| 30,00% | 6899,4 | 14,71 | | |
| 50,00% | 11499 | 20,08 | | |
| 100,00% | 22998 | 71,33 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




## Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

La línea de tendencia muestra un crecimiento constante, con pendiente menor a 1 en relación con la complejidad nlogn.

# Requerimiento 8:

## Descripción

```python
def ActorTop(catalog,N): #Función Principal Requerimiento 8
    actor_dict = {}
    top_actors = lt.newList()
    top_N = lt.newList()
    for stream in catalog:
        for title in lt.iterator(catalog[stream]):
            title["streaming_platform"] = stream
            for cast in title["cast"].split(","):
                cast = cast.strip()
                if cast not in actor_dict:
                    actor_dict[cast] = lt.newList()
                    lt.addLast(actor_dict[cast],title)
                else:
                    lt.addLast(actor_dict[cast],title)
    i = 0
    while i < int(N):
        max_ = None
        name = None
        for key in actor_dict:
            if max_ == None:
                name = key
                max_ = actor_dict[key]
            elif lt.size(actor_dict[key]) > lt.size(max_):
                name = key
                max_ = actor_dict[key]
        actor_dict.pop(name)
        lt.addLast(top_actors,{"name":name,"titles":max_})
        i += 1
    for i in lt.iterator(top_actors):
        lt.addLast(top_N,TopActorPropierties(i))
    return top_N,len(actor_dict)
```

| | |
|---|---|
| **Entrada** | Catálogo, número de Top |
| **Salidas** | Tupla (Lista encadenada, tamaño de la lista) |
| **Implementado (Sí/No)** | Sí |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| | |
|---|---|
| Asignación | O(n) |
| Operador In | O(n) |
| Comparación | O(n) |
| Addlast | O(k) |
| Función TopActorPropierties | O(nlogn) |
| ***TOTAL*** | ***O(nlogn)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| **Entrada** | **Tiempo (ms)** |
|---|---|
| Prueba 1 (0,5%) | 20,68 |
| Prueba 2 (5%) | 78,16 |
| Prueba 3 (20%) | 287,29 |
| Prueba 4 (30%) | 608,62 |
| Prueba 5 (50%) | 923,86 |
| Prueba 6 (100%) | 1609,38 |

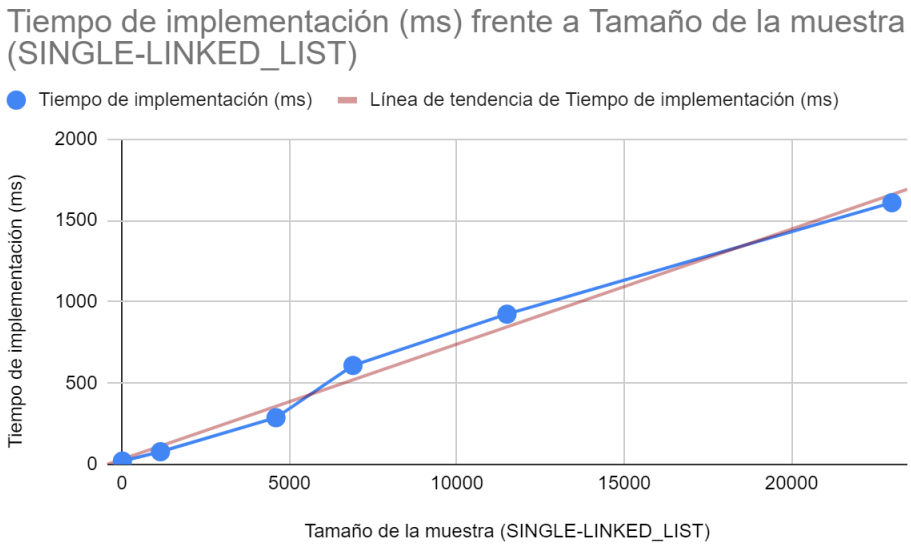
## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| **Requerimiento 8** | | | | |
|---|---|---|---|---|
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (SINGLE-LINKED_LIST)** | **Tiempo de implementación (ms)** | m | y-intersepto |
| 0,05% | 11,499 | 20,68 | 0,07077001173 | 31,77671995 |
| 5,00% | 1149,9 | 78,16 | | |
| 20,00% | 4599,6 | 287,29 | | |
| 30,00% | 6899,4 | 608,62 | | |
| 50,00% | 11499 | 923,86 | | |
| 100,00% | 22998 | 1609,38 | | |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




## Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Este es el requerimiento que más tiempo consume debido al número de componentes que abarca. Considerando la magnitud de datos que maneja, es de los más constantes con el incremento de tiempo en nlogn.