ANÁLISIS DEL RETO

Daniel Corredor, 20212247, d.corredorl@uniandes.edu.co

Johan Garcia, 202012165, je.garciab1@uniandes.edu.co

Martin Rodriguez no ayudó en el análisis

Requerimiento <<1>>

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Un año
Salidas	Número total de programas y películas, los tres primero y últimos programas ordenados asociados a un año
Implementado (Sí/No)	Sí, Daniel Corredor

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
busca el valor de la llave asociada a año	O(1)
Ordena la lista de programas asociados al año con Shell sort	O(nlog2n)
Crea una lista con los 3 primero y últimos programas organizados	O(n)
TOTAL	***O(nlog2n)***

Pruebas Realizadas

Entrada	**Tiempo (ms)**	**Memoria**
1999	8.654	1.626
1989	8.231	1.389
1979	8.709	1.551

Tablas de datos

1999

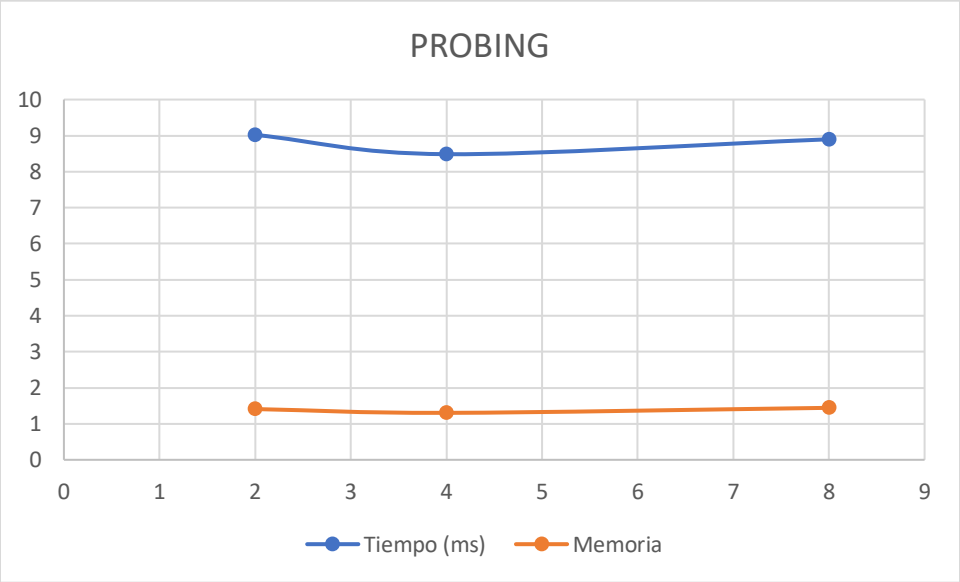
PROBING

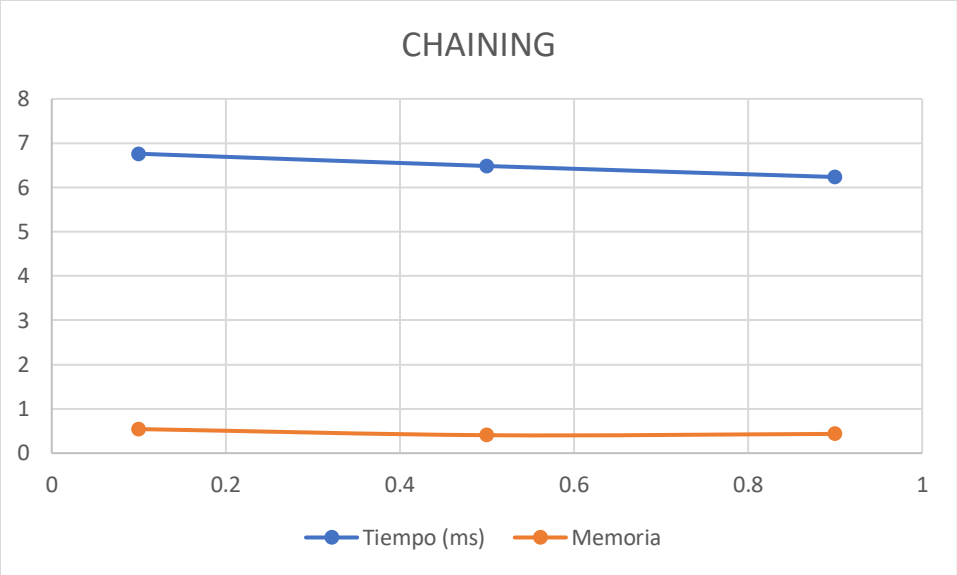
Factor de carga	Tiempo (ms)	Memoria
0.1	6.762	0.543
0.5	6.485	0.4052
0.9	6.236	0.4344

CHAINING

Factor de carga	Tiempo (ms)	Memoria
2	9.02	1.409
4	8.484	1.304
8	8.898	1.443

Graficas





Análisis

Aunque la complejidad no es muy alta, la eficiencia es muy baja tiene una cantidad de tiempo muy alta a lo esperado

Requerimiento <<2>>

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Una fecha
Salidas	Número total de programas y películas, los tres primero y últimos programas ordenados asociados a esa fecha
Implementado (Sí/No)	Sí, Daniel Corredor

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
busca el valor de la llave asociada a la fecha	O(1)
Ordena la lista de programas asociados a la fecha con Shell sort	O(nlog2n)
Crea una lista con los 3 primero y últimos programas organizados	O(n)
TOTAL	***O(nlog2n)***

Tablas de datos

November 12, 2019

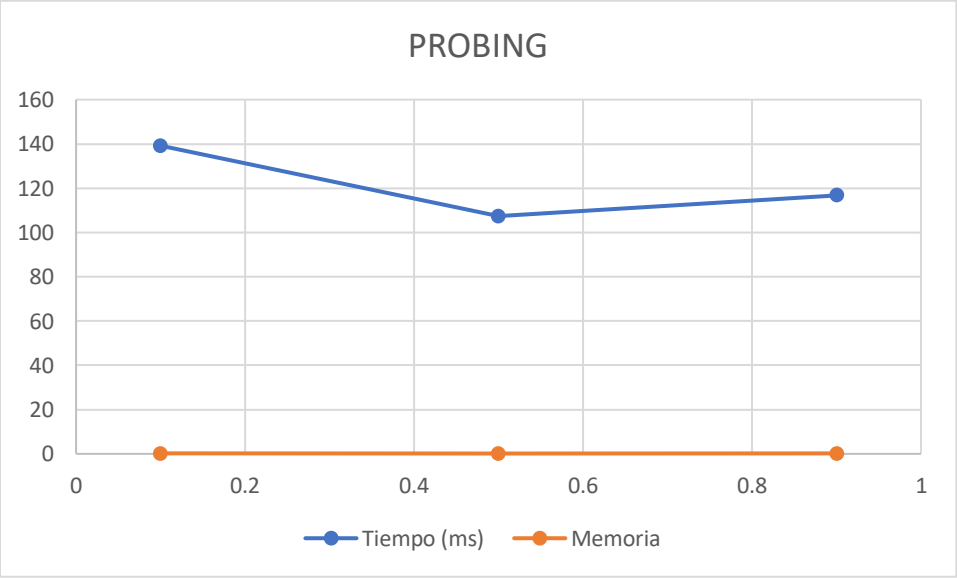
PROBING

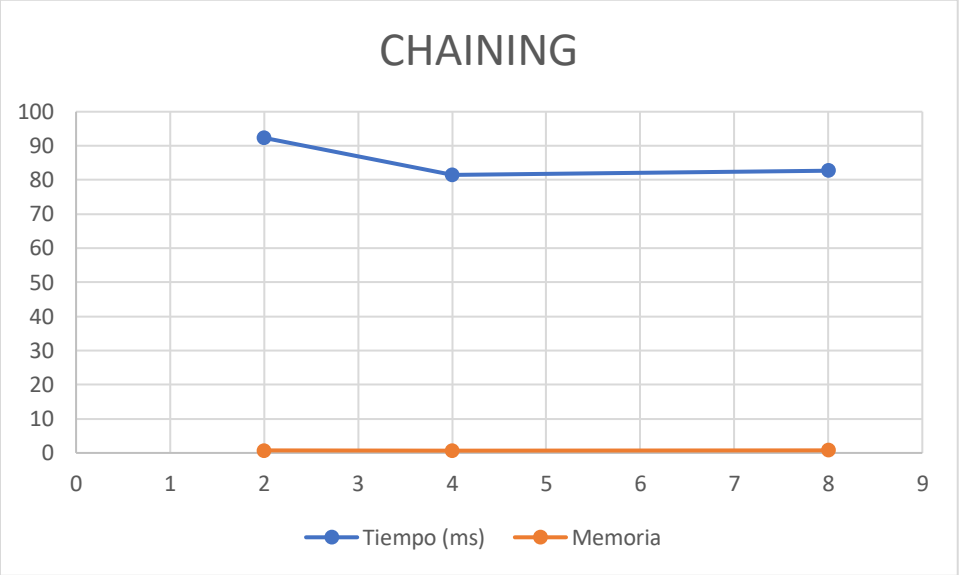
Factor de carga	Tiempo (ms)	Memoria
0.1	139.2	0.1911
0.5	107.4	0.1617
0.9	116.8	0.2072

CHAINING

Factor de carga	Tiempo (ms)	Memoria
2	92.34	0.698
4	81.49	0.664
8	82.77	0.753

Graficas





Análisis

Aunque la complejidad no es muy alta, la eficiencia es muy baja tiene una cantidad de tiempo muy alta a lo esperado

Requerimiento <<4>>

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Un genero
Salidas	Número total de programas y películas, los tres primero y últimos programas ordenados asociados a un genero
Implementado (Sí/No)	Sí, Daniel Corredor

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
busca el valor de la llave asociada a el director buscado	O(1)
Ordena la lista de programas asociados al genero con Shell sort	O(nlog2n)

Crea una lista con los 3 primero y últimos programas organizados	O(n)
TOTAL	***O(nlog2n)***

Pruebas Realizadas

Entrada	**Tiempo (ms)**	**Memoria**
Fantasy	0.368	1.883
Action	0.332	1.242
Adventure	0.467	0.172

Tablas de datos

Entrada John Hughes

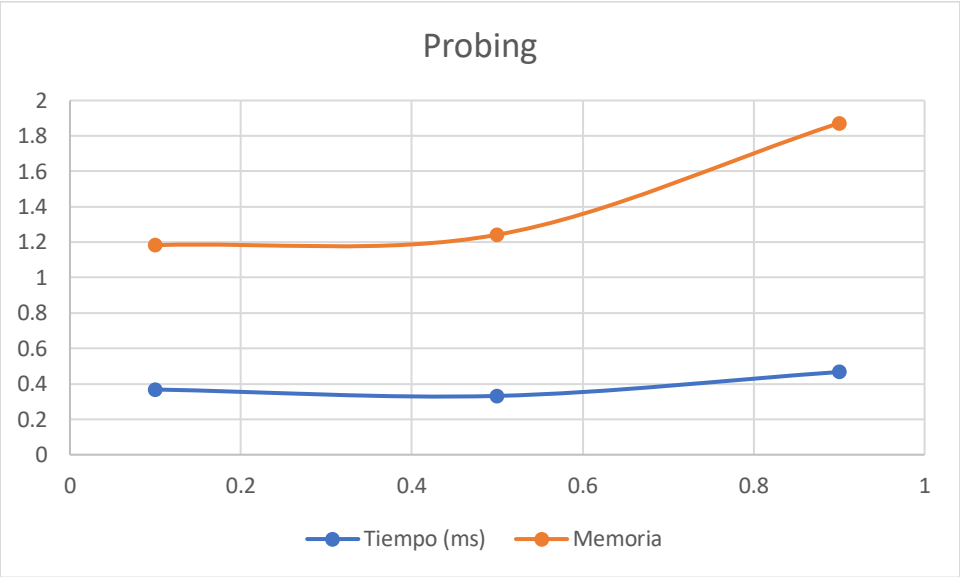
PROBING

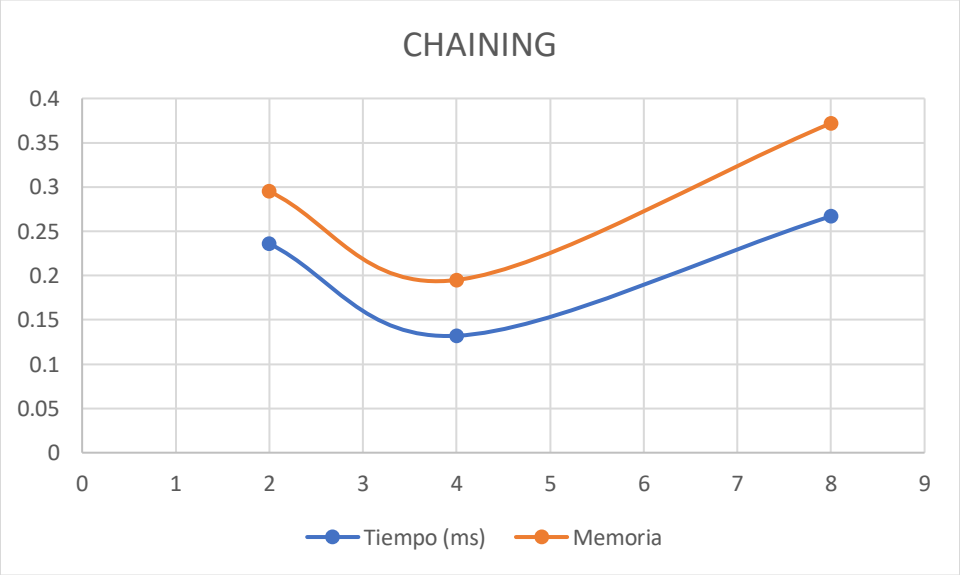
Factor de carga	**Tiempo (ms)**	**Memoria**
0.1	0.368	1.183
0.5	0.332	1.242
0.9	0.467	1.872

CHAINING

Factor de carga	**Tiempo (ms)**	**Memoria**
2	0.236	0.295
4	0.132	0.195
8	0.267	0.372

Graficas





Análisis

Se esperaba que los valores de carga fueran más eficientes cuando hay un factor de carga de 0.5 para Probing y 4 para Chaining, esto se ve muy claro al disminuir la memoria y tiempo para chaining y un poco claro para probing.

Requerimiento <<6>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Un director
Salidas	Numero total de programas y películas, lista con la cantidad de programas y películas por género, los tres primero y últimos programas ordenados asociados a un director
Implementado (Sí/No)	Sí, Daniel Corredor

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
busca el valor de la llave asociada a el director buscado	O(1)
Ordena la lista de programas asociados al director con Shell sort	O(nlog2n)
Crea una lista con los 3 primero y últimos programas organizados	O(n)
TOTAL	***O(nlog2n)***

Pruebas Realizadas

Entrada	**Tiempo (ms)**	**Memoria**
John Hughes	0.368	1.883
Natty Kumar	0.332	1.242
Matteo Garrone	0.467	0.172

Tablas de datos

Entrada John Hughes

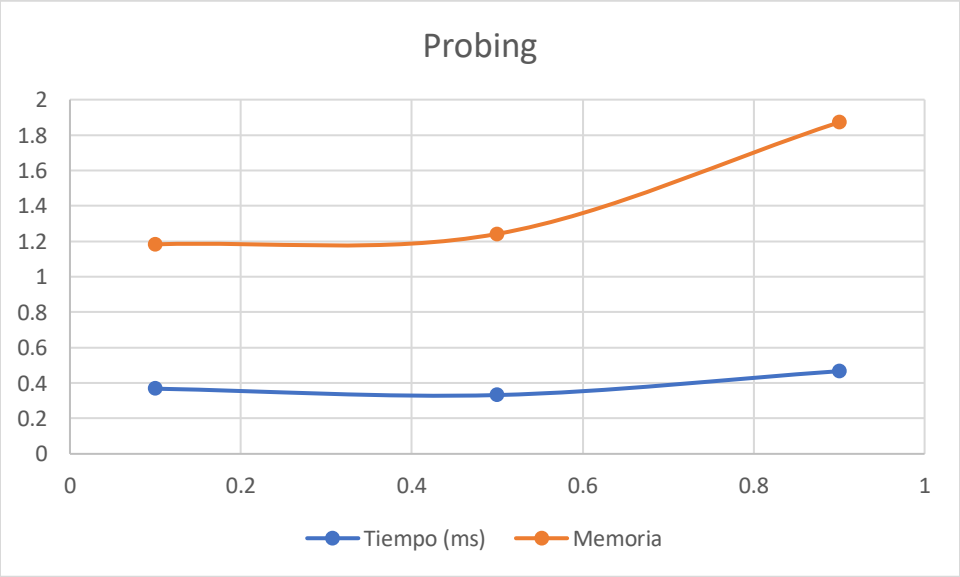
PROBING

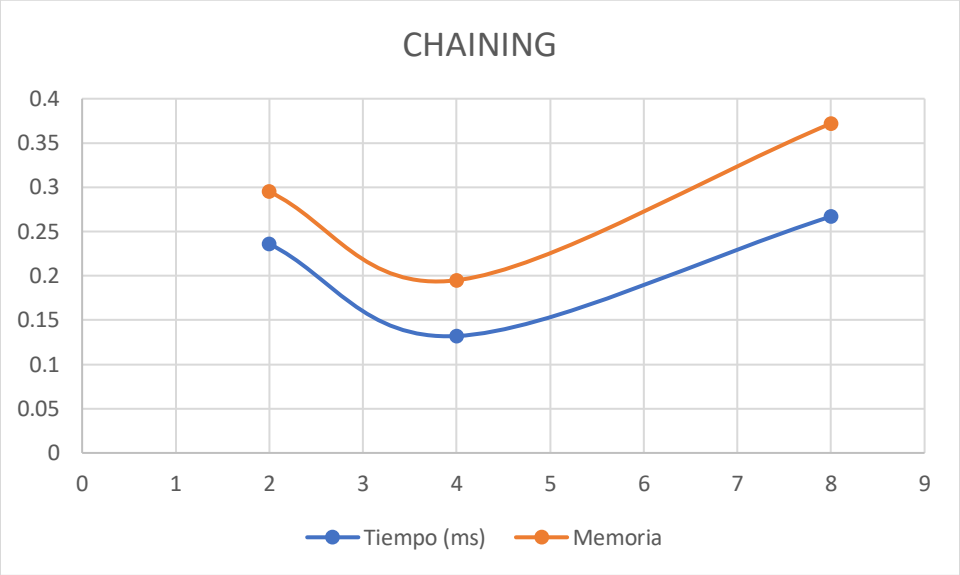
Factor de carga	Tiempo (ms)	Memoria
0.1	0.368	1.183
0.5	0.332	1.242
0.9	0.467	1.872

CHAINING

Factor de carga	Tiempo (ms)	Memoria
2	0.236	0.295
4	0.132	0.195
8	0.267	0.372

Graficas





Análisis

Se esperaba que los valores de carga fueran más eficientes cuando hay un factor de carga de 0.5 para Probing y 4 para Chaining, esto se ve muy claro al disminuir la memoria y tiempo para chaining y un poco claro para probing.

Requerimiento <<7>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Un rank
Salidas	El grupo total de programas y películas, por cada genero se presento el nombre, cantidad por streaming, cantidad por película y programa
Implementado (Sí/No)	Sí, Daniel Corredor y Johan Garcia

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
busca el valor de la llave asociada a cada genero	O(n)
Poner cada una de las listas de programas asociadas a cada genero en una lista adt	O(n)
Organizar la lista con Shell sort por los que mas tiene contenido	O(nlogn)

Recorre la lista con la cantidad Rank (M), y agrega información por cada genero recorriendo cada programa (L)	N*L*M
TOTAL	***O(***N*L*M***)***

Pruebas Realizadas

Entrada	**Tiempo (ms)**	**Memoria**
1	22.67	0.1128
5	30.98	0.1332
10	39.44	0.1728

Tablas de datos

Rank = 5

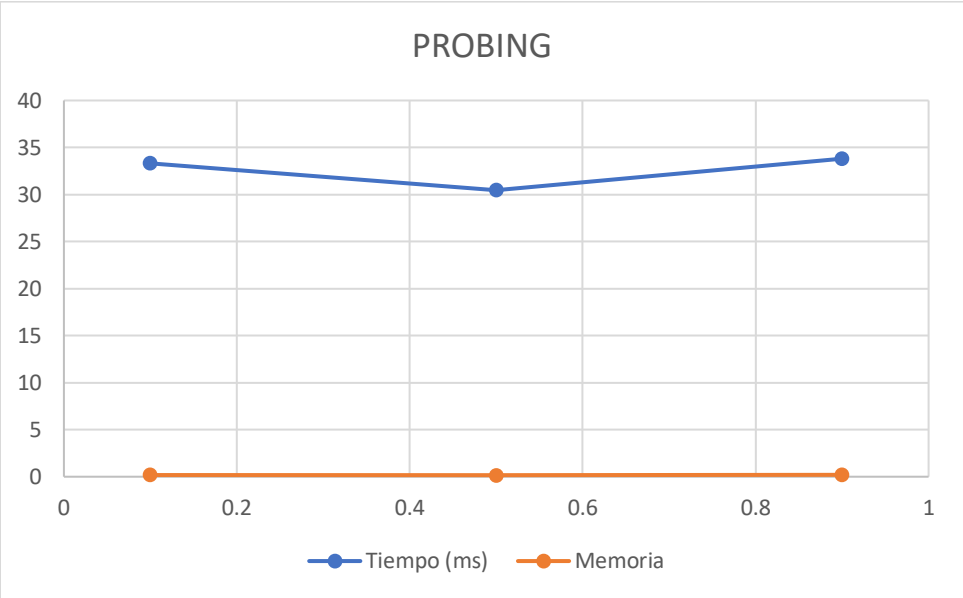
PROBING

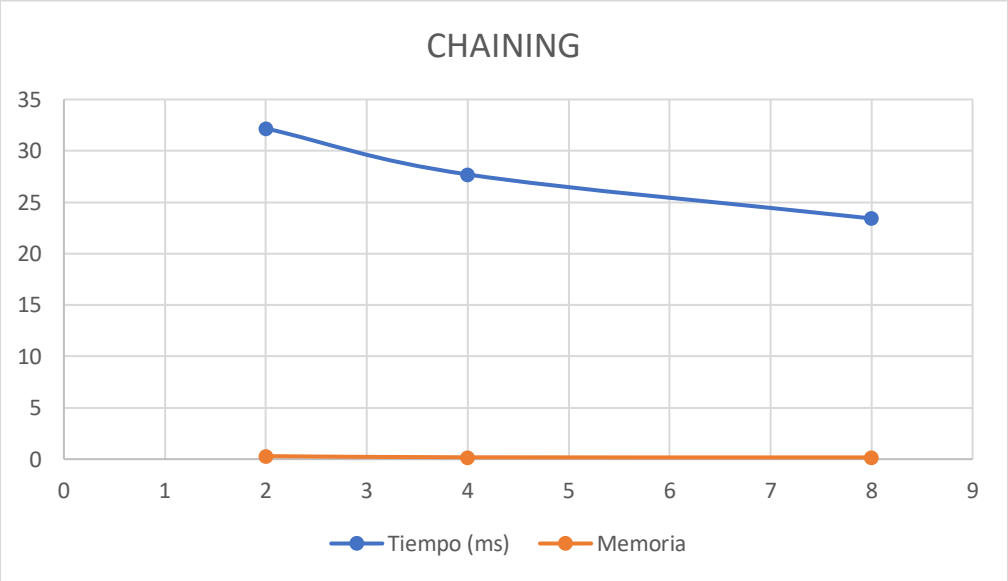
Factor de carga	Tiempo (ms)	Memoria
0.1	30.33	0.1817
0.5	30.46	0.1623
0.9	33.81	0.2014

CHAINING

Factor de carga	Tiempo (ms)	Memoria
2	32.18	0.295
4	27.7	0.1812
8	23.45	0.1711

Graficas





Análisis

En este caso la cantidad del Rank influía bastante en la eficiencia de la función sin embargo, inesperadamente la eficiencia en chaining semi proporcionalmente factor de carga , probablemente necesitaba mas elementos por poner en la función mp.newMap .

Requerimiento <<8>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Un genero y un rank
Salidas	El grupo total de programas y películas por actor, por cada actor se presentó el nombre, cantidad por streaming, cantidad por película y programa, lista de programas y películas, lista de actores y directores con los que el actor participó
Implementado (Sí/No)	Sí, Daniel Corredor

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Invoca el requerimiento 4 para listar los programas de un genero	O(nlogn)
Recorre la lista (M), por cada programa busca cada uno de los actores (N) y mete todos los programas asociados a cada actor	O(MN)
Organizar la lista con Shell sort por los actores con mas programas	O(nlogn)
Recorre la lista con la cantidad Rank (M), y agrega información por cada actor recorriendo cada programa (L)	N*L*M
TOTAL	***O(***N*L*M***)***

Pruebas Realizadas

Fantasy

Entrada	**Tiempo (ms)**	**Memoria**
1	5148.22	16.52
5	5988.09	20.47
10	7391.17	26.03

Tablas de datos

Fantasy, 5

PROBING

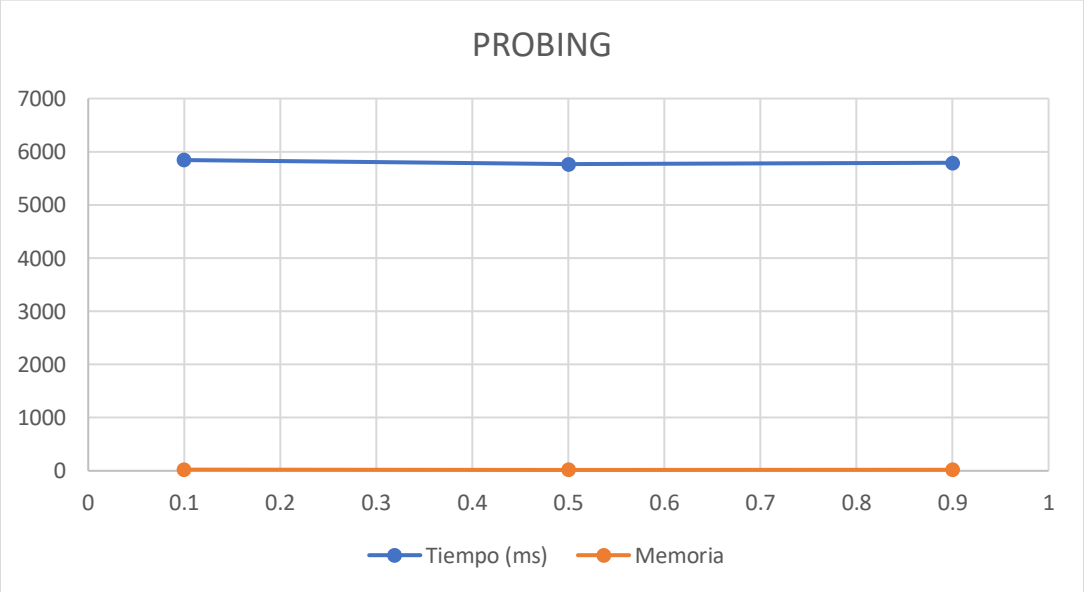
Factor de carga	Tiempo (ms)	Memoria
0.1	5843.75	22.83

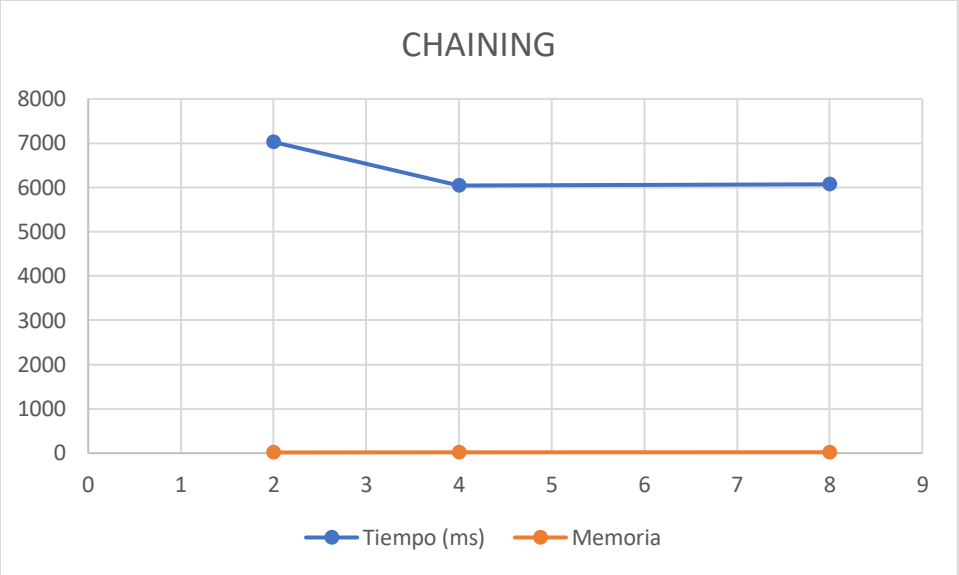
0.5	5768.2	20.02
0.9	5791.07	19.76

CHAINING

Factor de carga	Tiempo (ms)	Memoria
2	7025.78	19.09
4	6044.14	18.56
8	6072.45	21.16

Graficas





Análisis

En este caso la cantidad del Rank influía bastante en la eficiencia de la función ya que el recorrido para obtener las listas y la información de cada actor es un factor muy importante en la complejidad del algoritmo

Análisis reto 1-2

Los arboles tienen una mayor eficiencia que las listas utilizadas en el reto 1, además de tener una complejidad mejor o igual ya que en la mayoría de funciones la complejidad depende mayormente de el ordenamiento de la lista utilizada.