ANÁLISIS DEL RETO

Rodrigo García Aguirre, 202212375,r.garciaa2@uniandes.edu.co

Daniel Gomez, 202215189, de.gomezc12@uniandes.edu.co

Juan Sebastian Castro, 202011784, j.castro14@uniandes.ed

Procesador	Ryzen 9 4900Hs
Memoria Ram	16
Sistema Operativo	Windows 11

Requerimiento 1:

Descripción

```
def req_1(info):
    """
    Función que soluciona el requerimiento 1 O(N)
    """
    lista=lt.newList("ARRAY_LIST") #Por convencion, la posicion 1 sera 2012, 2 sera 2013 y asi sucesivamente
    anio_base=2011
    lt.addFirst(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    # TODO: Realizar el requerimiento 1
    for i in lt.iterator(info):
        anio=lt.getElement(i,1)-anio_base  #Anio es la posicion para la lista
        saldo=lt.getElement(i,10)
        if lt.getElement(lista,anio)==0 or saldo > lt.getElement(lt.getElement(lista,anio),10):
            lt.changeInfo(lista,anio,i)
    return lista
```

Entrada	Catálogo
Salidas	ARRAY_LIST, Tabúlate con datos solicitados.

Implementado (Sí/No)	Sí (GRUPAL)

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Asignación	O(1)
Comparación	O(1)
Recorrido (for)	O (n)
changeInfo	O(1)
TOTAL	***O(n)***

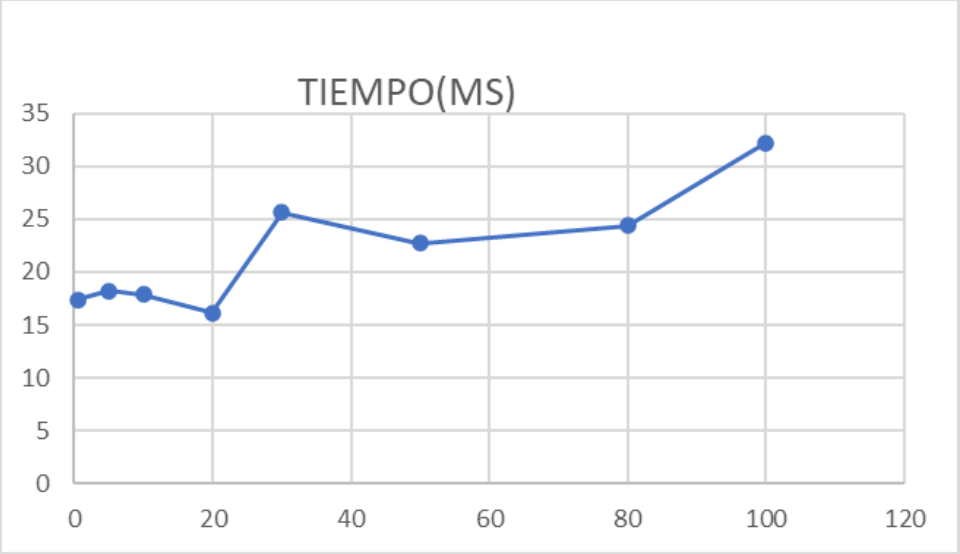
Pruebas Realizadas

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%)	17,41
Prueba 2 (5%)	18,02
Prueba 3 (10%)	17,9
Prueba 4 (20%)	16,13
Prueba 5 (30%)	26,07
Prueba 5 (50%)	23,4
Prueba 5 (80%)	24,7
Prueba 6 (100%)	32,2

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 que nos muestra la gráfica tiene tendencia N, lo cual nos verifica que el algoritmo es O(N)

Requerimiento 2:

Descripción

```python
def req_2(info):
    """
    Función que soluciona el requerimiento 2
    """
    # TODO: Realizar el requerimiento 2
    lista=lt.newList("ARRAY_LIST") #Por convencion, la posicion 1 sera 2012, 2 sera 2013 y asi sucesivamente
    anio_base=2011
    lt.addFirst(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    # TODO: Realizar el requerimiento 1
    for i in lt.iterator(info):
        anio=lt.getElement(i,1)-anio_base  #Anio es la posicion para la lista
        saldo=lt.getElement(i,11)
        if lt.getElement(lista,anio)==0 or saldo > lt.getElement(lt.getElement(lista,anio),11):
            lt.changeInfo(lista,anio,i)

    return lista
```

Entrada	Catálogo
Salidas	ARRAY_LIST, Tabúlate con datos solicitados.
Implementado (Sí/No)	Sí (GRUPAL)

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Asignación	O(1)
Comparación	O(1)
Recorrido(for)	O (n)
ChangeInfo	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

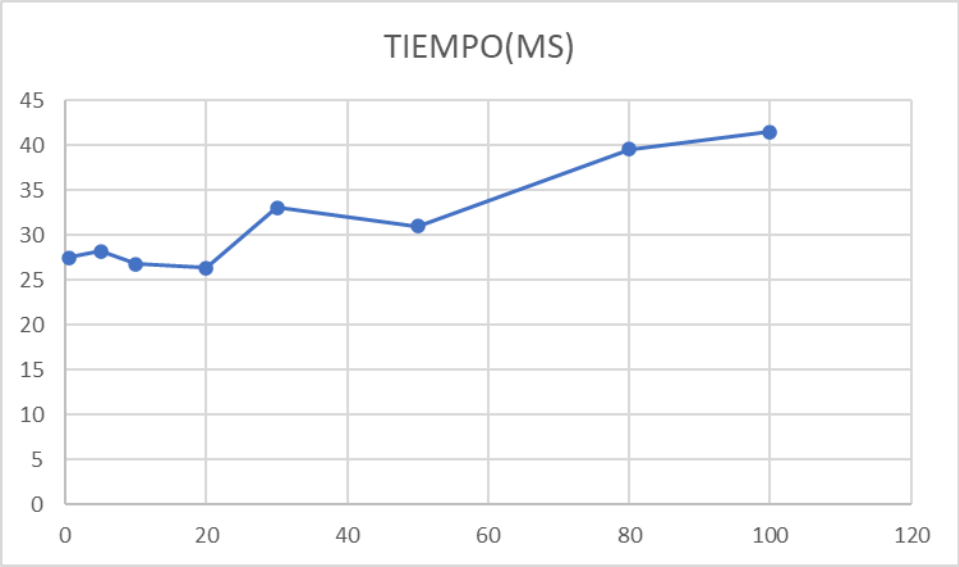
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (ms)**
(0,5%)	27,46
(5%)	28,17

(10%)	26,79
(20%)	26,65
(30%)	33,07
(50%)	31,01
(80%)	39,55
(100%)	41,2

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 que nos muestra la gráfica tiene tendencia N, lo cual nos verifica que el algoritmo es O(N)

Requerimiento 3:

Descripción

```python
def req_3(info):
    """
    Función que soluciona el requerimiento 3 INDIVIDUAL O(N)
    """
    # TODO: Realizar el requerimiento 2
    lista=lt.newList("ARRAY_LIST") #Por convencion, la posicion 1 sera 2012, 2 sera 2013 y asi sucesivamente
    anio_base=2011
    lt.addFirst(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    # TODO: Realizar el requerimiento 1
    for i in lt.iterator(info):
        anio=lt.getElement(i,1)-anio_base  #Anio es la posicion para la lista
        saldo=lt.getElement(i,12)
        if lt.getElement(lista,anio)==0 or saldo < lt.getElement(lt.getElement(lista,anio),12):
            lt.changeInfo(lista,anio,i)

    return lista
```

Entrada	catálogo
Salidas	ARRAY_LIST, Tabúlate con datos solicitados.
Implementado (Sí/No)	Sí (Individual:) Rodrigo García Aguirre

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Asignación	O(1)
Comparación	O(1)
Recorrido(for)	O (n)
ChangeInfo	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

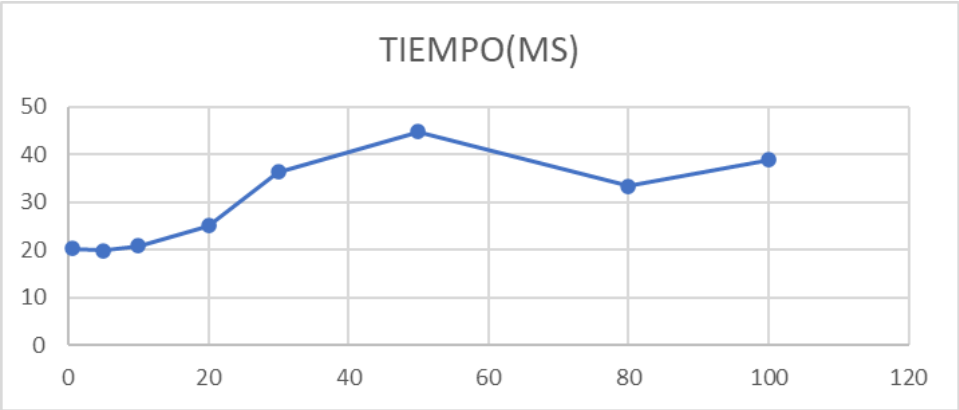
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (ms)**
(0,5%)	20,283
(5%)	19,91
(10%)	20,83
(20%)	25,02
(30%)	36,38
(50%)	44,74

(80%)	33,35
(100%)	39,5

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 que nos muestra la gráfica tiene tendencia N, lo cual nos verifica que el algoritmo es O(N)

Requerimiento 4:

```
201
202 def req_4(info):
203     """
204     Requerimiento No. 4 (Individual): Encontrar el subsector económico con los mayores
205     costos y gastos de nómina para todos los años disponibles    """
206
207     lista=lt.newList("ARRAY_LIST") #Por convencion, la posicion 1 sera 2012, 2 sera 2013 y asi sucesivamente
208     anio_base=2011
209     lt.addFirst(lista,0)
210     lt.addLast(lista,0)
211     lt.addLast(lista,0)
212     lt.addLast(lista,0)
213     lt.addLast(lista,0)
214     lt.addLast(lista,0)
215     lt.addLast(lista,0)
216     lt.addLast(lista,0)
217     lt.addLast(lista,0)
218     lt.addLast(lista,0)
219     # TODO: Realizar el requerimiento 1
220     for i in lt.iterator(info):
221         anio=lt.getElement(i,1)-anio_base  #Anio es la posicion para la lista
222         saldo=lt.getElement(i,9)
223         if lt.getElement(lista,anio)==0 or saldo < lt.getElement(lt.getElement(lista,anio),9):
224             lt.changeInfo(lista,anio,i)
225
226     return lista
227
```

Descripción

Entrada	Catalogo
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(1)
Comparación	O(1)
Recorrido(for)	O (n)
ChangeInfo	O(1)
TOTAL	***O(n)***

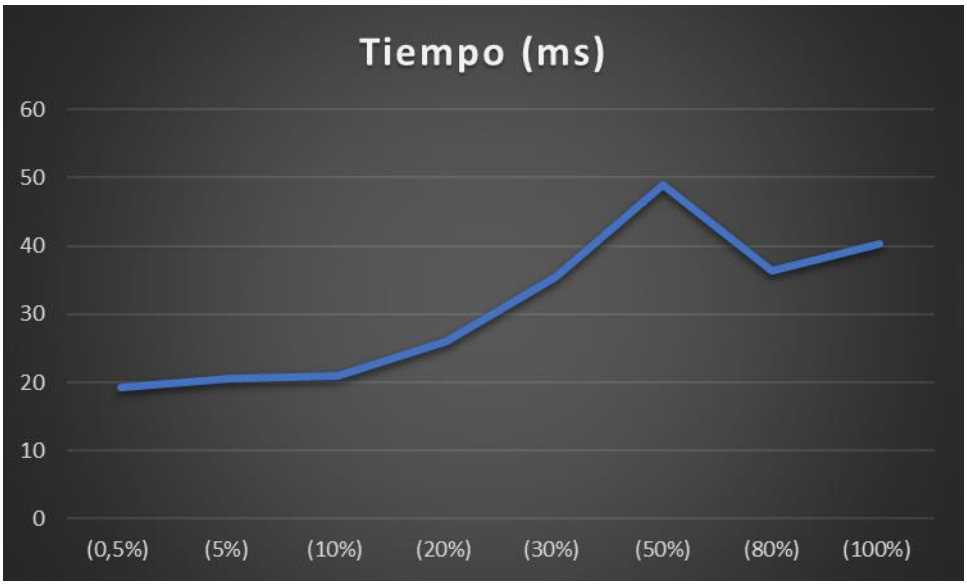
Pruebas Realizadas

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)**
(0,5%)	19,265
(5%)	20,654
(10%)	20,893
(20%)	26,024
(30%)	35,324
(50%)	48,968
(80%)	36,365
(100%)	40,245

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.

Al ver la gráfica podemos darnos cuenta de cómo aumenta el tiempo de ejecución del programa al aumentar la cantidad de datos ingresados esto nos permite concluir que la complejidad algorítmica esta correcta y es de O(N)

Requerimiento 5:

```
def req_5(data_structs):
    """
    Función que soluciona el requerimiento 5
    """
    # TODO: Realizar el requerimiento 5
    pass
    lista=lt.newList("ARRAY_LIST") #Por convencion, la posicion 1 sera 2012, 2 sera 2013 y asi sucesivamente
    anio_base=2011
    lt.addFirst(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)
    lt.addLast(lista,0)

    for i in lt.iterator(data_structs):
        anio=lt.getElement(i,1)-anio_base  #Anio es la posicion para la lista
        saldo=lt.getElement(i,13)
        if lt.getElement(lista,anio)==0 or saldo > lt.getElement(lt.getElement(lista,anio),13):
            lt.changeInfo(lista,anio,i)

    return lista
```

Descripción

Entrada	Catálogo,
Salidas	Arraylist
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Asignación	O(1)
Comparación	O(1)
Recorrido(for)	O (n)
ChangeInfo	O(1)
TOTAL	***O(n)***

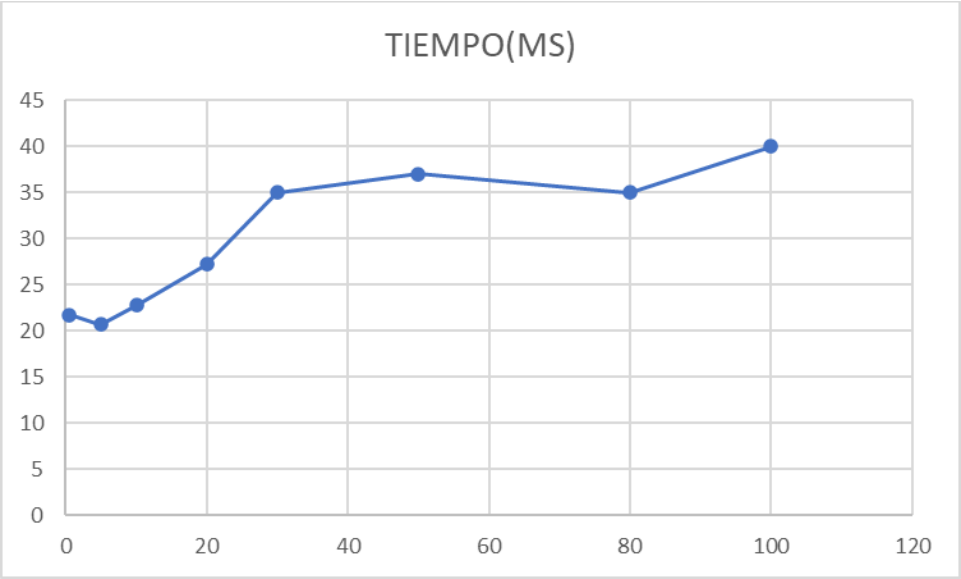
Pruebas Realizadas

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)**
(0,5%)	21,23
(5%)	20,71
(10%)	22,76
(20%)	27,20
(30%)	35,90
(50%)	37,42
(80%)	35,35
(100%)	40,8

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.

Al ver la gráfica podemos apreciar el crecimiento que tiene y podemos decir que la complejidad algorítmica es O(n)

Requerimiento 6:

```
def req_6(data_structs,anio):
    """
    Función que soluciona el requerimiento 6
    """
    lista_anio = lt.newList("ARRAY_LIST")

    for dato_impuesto in lt.iterator(lista_anio):
        if dato_impuesto["Año"] == anio:
            lt.addlast(lista_anio,dato_impuesto)


    lista_sectores = lt.newList("ARRAY_LIST")

    lista_control_sectores = lt.newList("ARRAY_LIST")

    for dato_filtrado in lt.iteratpr(lista_anio):
        if lt.ispresent(lista_control_sectores, lt.getElement(dato_filtrado,5)) == 1:
            lt.addlast(lista_control_sectores,lt.getElement(dato_filtrado,5))
            dato_filtrado_sector = {"Nombre sector economico":lt.getElement(dato_filtrado,5)}
            lt.addlast(lista_sectores,dato_filtrado_sector)
        else:
            for dato_filtrado_lista_sectores in lt.iterator(lista_sectores):
                if lt.getElement(dato_filtrado_lista_sectores,5) == lt.getElement(dato_filtrado,5):
                    dato_filtrado_lista_sectores["Total ingresos netos sector economico"] += dato_filtrado["Total ingresos netos sector economico"]


    lista_subsectores_por_sector = lt.newList("ARRAY_LIST")
    for sector in lt.iterator(lista_sectores):
        lista_interna_subsectores = lt.newList("ARRAY_LIST")
        lista_control = lt.newList("ARRAY_LIST")
        if lt.ispresent(lista_control,sector["Nombre subsector economico"]) == 0:
            lt.addlast(lista_control_sectores,lt.getElement(dato_filtrado,7))
            dato_filtrado_sector = {"Nombre sector economico":lt.getElement(dato_filtrado,7)}
            lt.addlast(lista_sectores,dato_filtrado_sector)
        else:
            for dato_filtrado_lista_sectores in lt.iterator(lista_sectores):
                if lt.getElement(dato_filtrado_lista_sectores,5) == lt.getElement(dato_filtrado,7):
                    dato_filtrado_lista_sectores["Total ingresos netos sector economico"] += dato_filtrado["Total ingresos netos sector economico"]
    return None
```

Descripción

Entrada	Catálogo, Año
Salidas	ARRAYLIST, tabulate con la información solicitada
Implementado (Sí/No)	Sí (INCOMPLETO)

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Primer ciclo	O(n)
Buscar elemento	O(n)
Segundo ciclo	***O(n ∧ 2)***
TOTAL	***O(n ∧ 2)***

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)**
(0,5%)	
(5%)	
(20%)	

(30%)	
(50%)	
(100%)	

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.

Requerimiento 7:

Descripción

Entrada	Catálogo, Top n, Año Inicial, Año Final
Salidas	Tabla con las actividades económicas
Implementado (Sí/No)	No

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pruebas Realizadas

Tablas de datos

Graficas

Análisis