ANÁLISIS DEL RETO

Andres Romero, af.romerop1@uniandes.edu.co, 202312399.

Juan José Penha, j.penha@uniandes.edu.co, 202312307.

Jerónimo Vásquez, j.vasquezp2@uniandes.edu.co, 202223824.

Requerimiento <<1>>

```
def req_1(control, fecha_inicial, fecha_final):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1


    lista = lt.newList('ARRAY_LIST')

    datos = om.valueSet(control['fechaindex'])

    for i in lt.iterator(datos):
        for j in lt.iterator(i):
            lt.addLast(lista,{'time':j['time'],'mag':j['mag'], 'lat':j['lat'], 'long':j['long'], 'depth':j['depth'],
                              'sig': j['sig'], 'gap': j['gap'], 'nst': j['nst'], 'title': j['title'],
                               'cdi': j['cdi'],'mmi': j['mmi'], 'magType': j['magType'], 'code': j['code'], 'type':j['type']})
    r = lt.newList('ARRAY_LIST')
    for e in lista['elements']:
        if e['time'] >= fecha_inicial and e['time'] <= fecha_final:
            lt.addLast(r,e)




    quk.sort(r, compare)



    return r['elements']
```

Descripción

Entrada	El datastructs, la fecha inicial (Digitada por el usuario) y la fecha final (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Jerónimo Vásquez

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (acceder al árbol)	O (n)

Paso 3 (acceder al for)	O (n)
Paso 4 (acceder al segundo for anidado)	O (n^2)
Paso 5 (acceder al for no anidado)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N^2)***

Pruebas Realizadas

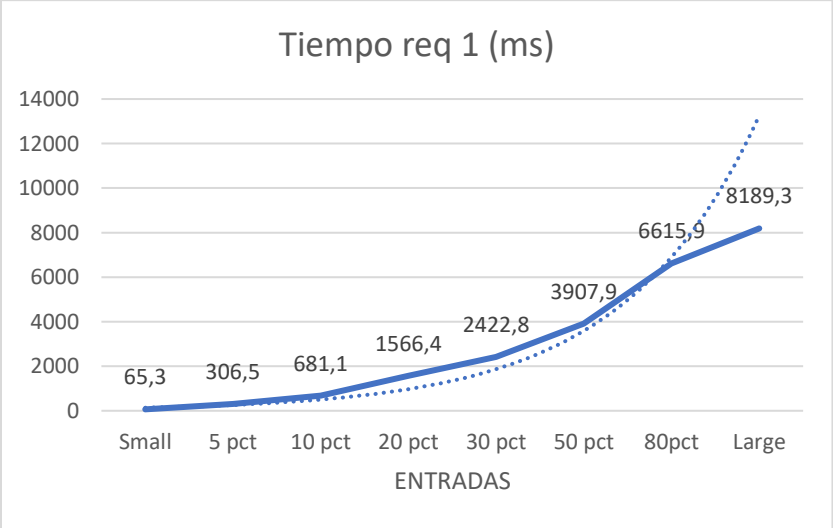
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	**Tiempo (ms)**
Small	65.3
5 pct	306.5
10 pct	681.1
20 pct	1566.4
30 pct	2422.8
50 pct	3907.9
80pct	6615.9
Large	8189.3

Graficas



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad lineal, la implementación de este requerimiento tiene un orden lineal O(n^2). Esto debido a que, el código tiene un for anidado que se encarga de acceder al array para agregar ciertos parámetros n cantidad de veces, dándole así una complejidad de O(n^2) en el peor de los casos.

Requerimiento <<2>>

```
def req_1(control, fecha_inicial, fecha_final):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1


    lista = lt.newList('ARRAY_LIST')

    datos = om.valueSet(control['fechaindex'])

    for i in lt.iterator(datos):
        for j in lt.iterator(i):
            lt.addLast(lista,{'time':j['time'],'mag':j['mag'], 'lat':j['lat'], 'long':j['long'], 'depth':j['depth'],
                              'sig': j['sig'], 'gap': j['gap'], 'nst': j['nst'], 'title': j['title'],
                               'cdi': j['cdi'],'mmi': j['mmi'], 'magType': j['magType'], 'code': j['code'], 'type':j['type']})
    r = lt.newList('ARRAY_LIST')
    for e in lista['elements']:
        if e['time'] >= fecha_inicial and e['time'] <= fecha_final:
            lt.addLast(r,e)




    quk.sort(r, compare)



    return r['elements']
```

Descripción

Entrada	El datastructs, la fecha inicial (Digitada por el usuario) y la fecha final (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Juan José

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (acceder al árbol)	O (n)
Paso 3 (acceder al for)	O (n)
Paso 4 (acceder al segundo for anidado)	O (n^2)
Paso 5 (acceder al for no anidado)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N^2)***

Pruebas Realizadas

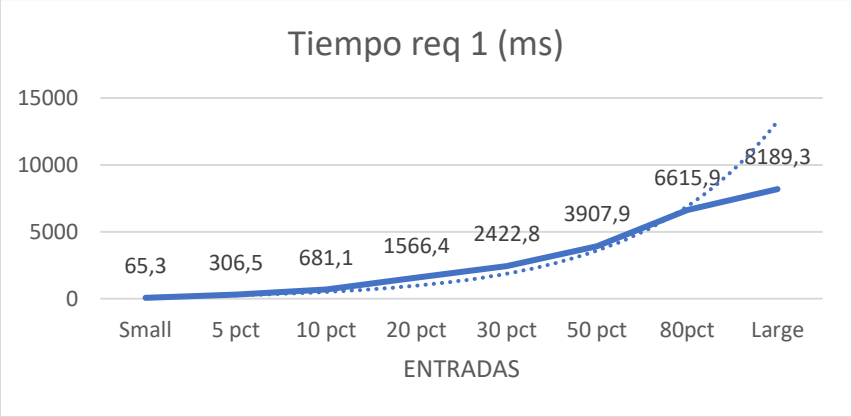
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	**Tiempo (ms)**
Small	626.5
5 pct	
10 pct	
20 pct	
30 pct	
50 pct	
80pct	
Large	

Graficas



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad lineal, la implementación de este requerimiento tiene un orden lineal O(n^2). Esto debido a que, el código tiene un for anidado que se encarga de acceder al array para agregar ciertos parámetros n cantidad de veces, dándole así una complejidad de O(n^2) en el peor de los casos

Requerimiento <<3>>

```python
def req_3(control,magnitud,profundidad):
    """
    Función que soluciona el requerimiento 3
    """
    # TODO: Realizar el requerimiento 3
    lista = lt.newList('ARRAY_LIST')
    r = lt.newList('ARRAY_LIST')

    datos = om.values(control['magnitud'], magnitud, om.maxKey(control['magnitud']))

    for i in lt.iterator(datos):
        for j in lt.iterator(i):
            lt.addLast(lista,{'time':j['time'],'mag':j['mag'], 'lat':j['lat'], 'long':j['long'], 'depth':j['depth'],
                              'sig': j['sig'], 'gap': j['gap'], 'nst': j['nst'], 'title': j['title'],
                                'cdi': j['cdi'],'mmi': j['mmi'], 'magType': j['magType'], 'code': j['code'], 'type':j['type']})

    for e in lista['elements']:
        if e['depth'] <= profundidad:
            lt.addLast(r, e)


    quk.sort(r, compare)

    return r['elements']
```

Descripción

Entrada	El datastructs, la magnitud (Digitada por el usuario) y la profundidad (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Jerónimo Vásquez

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (acceder al árbol)	O (n)
Paso 3 (acceder al for)	O (n)
Paso 4 (acceder al segundo for anidado)	O (n^2)
Paso 5 (acceder al for no anidado)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N^2)***

Pruebas Realizadas

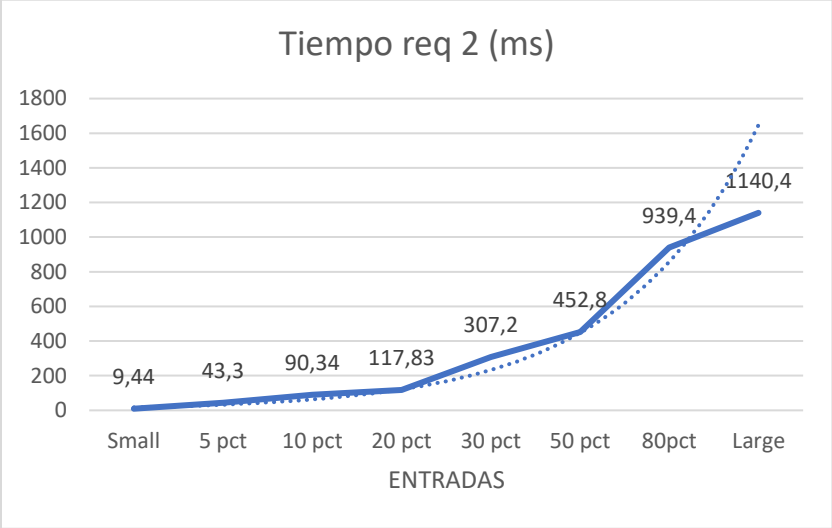
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	Tiempo (ms)
Small	7.22
5 pct	46.3
10 pct	125.8
20 pct	202.2
30 pct	286.9
50 pct	521.9
80pct	1125.9
Large	2861.5

Graficas



Análisis

A pesar de que la complejidad analizada sea de O(n^2) la gráfica demuestra que el incremento de esta es de orden exponencial, lo que nos llega a concluir que esta toma de datos fue optima y no se empleó o no hizo parte del peor de los casos

Requerimiento <<4>>

```python
def req_1(control, fecha_inicial, fecha_final):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1


    lista = lt.newList('ARRAY_LIST')

    datos = om.valueSet(control['fechaindex'])

    for i in lt.iterator(datos):
        for j in lt.iterator(i):
            lt.addLast(lista,{'time':j['time'],'mag':j['mag'], 'lat':j['lat'], 'long':j['long'], 'depth':j['depth'],
                              'sig': j['sig'], 'gap': j['gap'], 'nst': j['nst'], 'title': j['title'],
                               'cdi': j['cdi'],'mmi': j['mmi'], 'magType': j['magType'], 'code': j['code'], 'type':j['type']})
    r = lt.newList('ARRAY_LIST')
    for e in lista['elements']:
        if e['time'] >= fecha_inicial and e['time'] <= fecha_final:
            lt.addLast(r,e)




    quk.sort(r, compare)



    return r['elements']
```

Descripción

Entrada	El datastructs, la fecha inicial (Digitada por el usuario) y la fecha final (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Juan José

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (acceder al árbol)	O (n)
Paso 3 (acceder al for)	O (n)
Paso 4 (acceder al segundo for anidado)	O (n^2)
Paso 5 (acceder al for no anidado)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N^2)***

Pruebas Realizadas

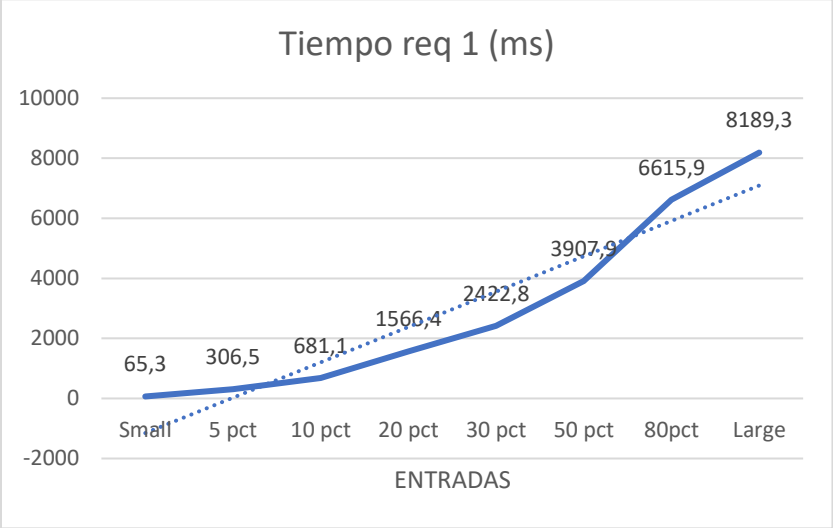
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	Tiempo (ms)
Small	12.3
5 pct	84.9
10 pct	196.3
20 pct	411.6
30 pct	680.2
50 pct	1215.6
80pct	4604.3
Large	6140.2

Graficas



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad lineal, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, el código tiene un for anidado que se encarga de acceder al array para agregar ciertos parámetros n cantidad de veces, dándole así una complejidad de O(n) en el peor de los casos

Requerimiento <<5>>

```python
def req_1(control, fecha_inicial, fecha_final):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1


    lista = lt.newList('ARRAY_LIST')

    datos = om.valueSet(control['fechaindex'])

    for i in lt.iterator(datos):
        for j in lt.iterator(i):
            lt.addLast(lista,{'time':j['time'],'mag':j['mag'], 'lat':j['lat'], 'long':j['long'], 'depth':j['depth'],
                              'sig': j['sig'], 'gap': j['gap'], 'nst': j['nst'], 'title': j['title'],
                               'cdi': j['cdi'],'mmi': j['mmi'], 'magType': j['magType'], 'code': j['code'], 'type':j['type']})
    r = lt.newList('ARRAY_LIST')
    for e in lista['elements']:
        if e['time'] >= fecha_inicial and e['time'] <= fecha_final:
            lt.addLast(r,e)




    quk.sort(r, compare)



    return r['elements']
```

Descripción

Entrada	El datastructs, la fecha inicial (Digitada por el usuario) y la fecha final (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Jerónimo Vásquez

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (acceder al árbol)	O (n)
Paso 3 (acceder al for)	O (n)
Paso 4 (acceder al segundo for anidado)	O (n^2)
Paso 5 (acceder al for no anidado)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N^2)***

Pruebas Realizadas

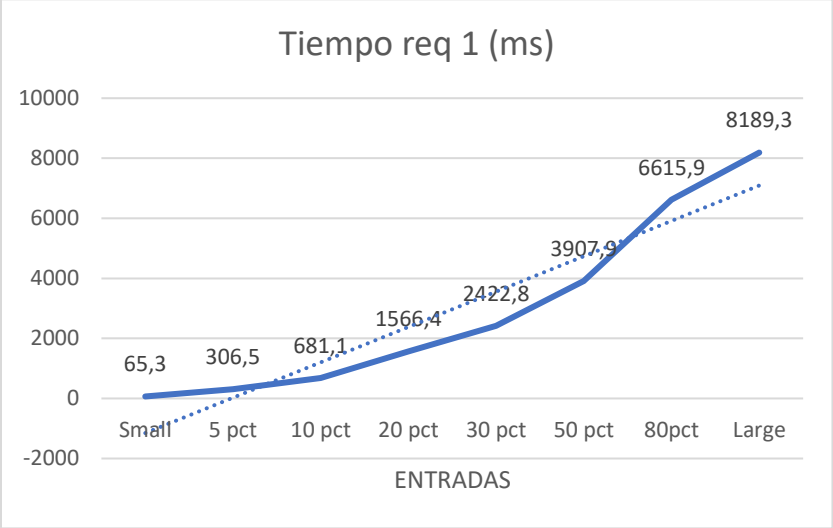
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	Tiempo (ms)
Small	26.8
5 pct	219.6
10 pct	508.2
20 pct	1183.0
30 pct	2055.5
50 pct	3549.5
80pct	13323
Large	17318

Graficas



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad lineal, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, el código tiene un for anidado que se encarga de acceder al array para agregar ciertos parámetros n cantidad de veces, dándole así una complejidad de O(n) en el peor de los casos

Requerimiento <<6>>

```
def req_1(control, fecha_inicial, fecha_final):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1


    lista = lt.newList('ARRAY_LIST')

    datos = om.valueSet(control['fechaindex'])

    for i in lt.iterator(datos):
        for j in lt.iterator(i):
            lt.addLast(lista,{'time':j['time'],'mag':j['mag'], 'lat':j['lat'], 'long':j['long'], 'depth':j['depth'],
                              'sig': j['sig'], 'gap': j['gap'], 'nst': j['nst'], 'title': j['title'],
                               'cdi': j['cdi'],'mmi': j['mmi'], 'magType': j['magType'], 'code': j['code'], 'type':j['type']})
    r = lt.newList('ARRAY_LIST')
    for e in lista['elements']:
        if e['time'] >= fecha_inicial and e['time'] <= fecha_final:
            lt.addLast(r,e)




    quk.sort(r, compare)



    return r['elements']
```

Descripción

Entrada	El datastructs, la fecha inicial (Digitada por el usuario) y la fecha final (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Jerónimo Vásquez

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (acceder al árbol)	O (n)
Paso 3 (acceder al for)	O (n)
Paso 4 (acceder al segundo for anidado)	O (n^2)
Paso 5 (acceder al for no anidado)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N^2)***

Pruebas Realizadas

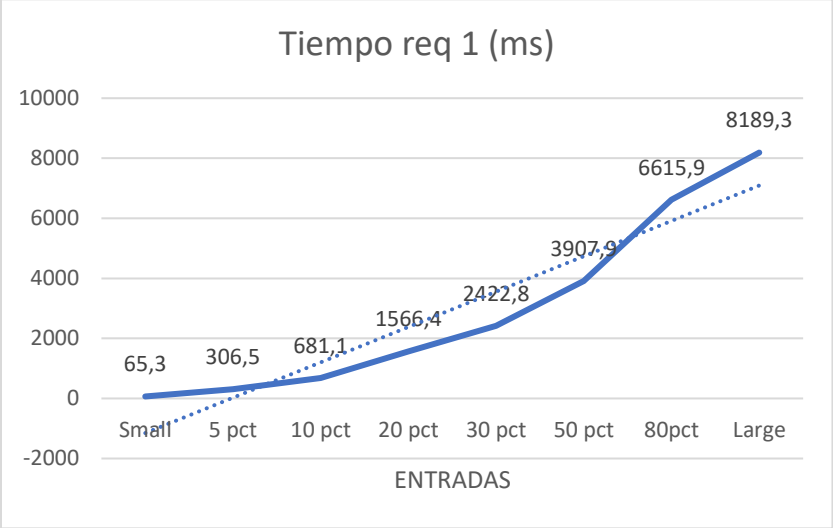
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	Tiempo (ms)
Small	1.38
5 pct	7.52
10 pct	13.0
20 pct	32.7
30 pct	49.9
50 pct	92.6
80pct	435.2
Large	440.6

Graficas



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad lineal, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, el código tiene un for anidado que se encarga de acceder al array para agregar ciertos parámetros n cantidad de veces, dándole así una complejidad de O(n) en el peor de los casos

Requerimiento <<7>>

```python
def req_7(control, anio, region, p):
    """
    Función que soluciona el requerimiento 7
    """
    # TODO: Realizar el requerimiento 7

    lista = lt.newList('ARRAY_LIST')
    c = control['fecha']

    map = mp.get(c, anio)

    control_r = me.getValue(map)

    for i in control_r['elements']:
        if region in i['title']:
            lt.addLast(lista, i)
    quk.sort(lista, compare)


    return lista
```

Descripción

Entrada	El datastructs, año (Digitada por el usuario) y la región (Digitada por el usuario)
Salidas	Un array list con toda la información necesaria
Implementado (Sí/No)	Si, implementado por Jerónimo Vásquez

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 (crear un array)	O (1)
Paso 2 (variable c como parámetro del árbol)	O (1)
Paso 3 (acceder al árbol)	O (n)
Paso 4 (acceder al segundo for)	O (n)
Paso 5 (acceder al condicional)	O (1)
Paso 6 (retornar r)	O (n)
Total	***O (N)***

Pruebas Realizadas

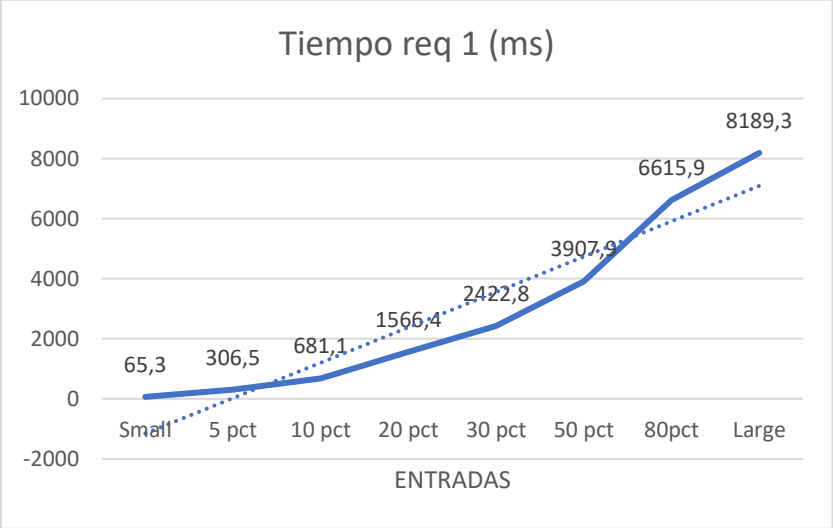
Procesadores	**AMD Ryzen 7 5700U with Radeon Graphics**
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11

Se operará todo desde el mismo computador en donde solo estará abierto visual studio para no afectar los hilos para la realización de la toma de tiempos de ejecución

Tablas de datos

Entrada	Tiempo (ms)
Small	0.69
5 pct	3.12
10 pct	4.48
20 pct	9.48
30 pct	15.9
50 pct	28.6
80pct	100.1
Large	132.5

Graficas



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad lineal, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, el código tiene un for anidado que se encarga de acceder al array para agregar ciertos parámetros n cantidad de veces, dándole así una complejidad de O(n) en el peor de los casos