**ANÁLISIS DEL RETO**

Juan Lago, 202021359, j.lagoa@uniandes.edu.co

Juan David Ortiz, 202222483, jd.ortizp1@uniandes.edu.co

Mauricio Martínez, 2023314461, m.martinezu@uniandes.edu.co

### **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 <<1>>**

## **Descripción**

A computer screen shot of text

Description automatically generated

Este requerimiento se encarga de retornar los últimos N partidos jugados por cierto equipo de algún país según su condición en el encuentro. Lo primero que hace es crear una lista vacia, luego define cual es la condición del encuentro dada como entrada y guarda cada partido en el que aquel pais halla jugado como la condición dada dentro de la lista previamente creada.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, el pais equipo el cual se va a buscar, la condición del encuentro en la que esta el pais |
| **Salidas** | Una lista con todos los partidos en los que el pais jugo como la condición entregada(hometeam o awayteam) |
| **Implementado (Sí/No)** | Si. Implementado por Juan Lago |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator** | **O(n)** |
| Addlast() | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.288 |
| 5 pct | 1.385 |
| 10 pct | 2.254 |
| 20 pct | 4.455 |
| 30 pct | 8.564 |
| 50 pct | 10.506 |
| 80 pct | 18.66 |
| large | 31.596 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.288 |
| 5 pct | Dato2 | 1.385 |
| 10 pct | Dato3 | 2.254 |
| 20 pct | Dato4 | 4.455 |
| 30 pct | Dato5 | 8.564 |
| 50 pct | Dato6 | 10.506 |
| 80 pct | Dato7 | 18.66 |
| large | Dato8 | 31.596 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

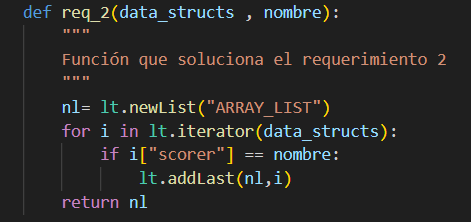
## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<2>>**

## **Descripción**



Este requerimiento se encarga de retornar una lista con la información de los primeros N goles anotados por cierto jugador.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, nombre del jugador |
| **Salidas** | Lista con todos los goles anotados por el jugador especificado |
| **Implementado (Sí/No)** | Si. Implementado por Juan Lago |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator** | **O(n)** |
| Addlast | O(1) |
| ***TOTAL*** | ***O(1)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.19 |
| 5 pct | 0.336 |
| 10 pct | 0.606 |
| 20 pct | 1.188 |
| 30 pct | 1.717 |
| 50 pct | 2.855 |
| 80 pct | 4.759 |
| large | 5.451 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.19 |
| 5 pct | Dato2 | 0.336 |
| 10 pct | Dato3 | 0.606 |
| 20 pct | Dato4 | 1.188 |
| 30 pct | Dato5 | 1.717 |
| 50 pct | Dato6 | 2.855 |
| 80 pct | Dato7 | 4.759 |
| large | Dato8 | 5.451 |
|  |  |  |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<3>>**

## **Descripción**

A screen shot of a computer program

Description automatically generated

Este requerimiento se encarga de retornar una lista con la información de los partidos que disputo un equipo especificado entre dos fechas de tiempo determinadas y también cuenta cuantos fueron jugando como locales y cuantos jugando como visitantes

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, nombre del equipo, fecha inicial y fecha final del periodo de consulta |
| **Salidas** | Lista con todos los partidos del equipo en cierto tiempo, partidos locales, partidos visitantes |
| **Implementado (Sí/No)** | Si. Implementado por Juan Lago |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator de i** | **O(i)** |
| Addlast | O(1) |
| Lt.iterator de i con lt.iterator de j | O(i\*j) |
| ***TOTAL*** | ***O(i\*j)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 4.5 |
| 5 pct | 29.37 |
| 10 pct | 63.98 |
| 20 pct | 145.95 |
| 30 pct | 258.07 |
| 50 pct | 591.16 |
| 80 pct | 1222.98 |
| large | 1821.02 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 4.5 |
| 5 pct | Dato2 | 29.37 |
| 10 pct | Dato3 | 63.98 |
| 20 pct | Dato4 | 145.95 |
| 30 pct | Dato5 | 258.07 |
| 50 pct | Dato6 | 591.16 |
| 80 pct | Dato7 | 1222.98 |
| large | Dato8 | 1821.02 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<4>>**

## **Descripción**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Este requerimiento se encarga de retornar una lista con la información de los partidos de un torneo especificado entre dos fechas ingresadas. También retorna el total de partidos, la cantidad de países participantes, de ciudades donde se disputaron y el total de partidos definidos por penal.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, fecha inicial, fecha final, nombre del torneo |
| **Salidas** | Lista con todos los partidos jugados en el torneo, total de partidos, países involucrados, ciudades donde se disputo, definiciones por penal |
| **Implementado (Sí/No)** | Si. Implementado por Juan David Ortiz |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator de i** | **O(i)** |
| Addlast | O(1) |
| Lt.iterator de i con lt.iterator de j | O(i\*j) |
| Addlast | O(1) |
| ***TOTAL*** | ***O(i\*j)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 4.35 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<5>>**

## **Descripción**

A computer screen shot of a program code

Description automatically generated

Este requerimiento se encarga de retornar una lista con los goles anotados por un jugador dado y entre dos fechas también dadas, además se retorna las que han sido desde el punto penal y las que han sido por auto gol.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, fecha inicial, fecha final, nombre del jugador |
| **Salidas** | Lista con las anotaciones del jugador en determinado tiempo, anotación por punto penal y anotaciones de autogol |
| **Implementado (Sí/No)** | Si. Implementado por Juan Lago |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator de i** | **O(n)** |
| Addlast | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 2.44 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<6>>**

## **Descripción**

Este requerimiento se encarga de retornar una lista con la información de los primeros N goles anotados por cierto jugador.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, nombre del jugador |
| **Salidas** | Lista con todos los goles anotados por el jugador especificado |
| **Implementado (Sí/No)** | Si. Implementado por Juan Lago |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator** | **O(n)** |
| Addlast | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.05 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 7.24 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<7>>**

## **Descripción**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

Este requerimiento se encarga de retornar una lista con la información de N cantidad de mejores jugadores de competencias oficiales dentro de un periodo de tiempo especificado. Se retorna también la cantidad de anotadores encontrada en total, la cantidad de partidos jugados por los anotadores, así como el total de goles, penales y autogoles.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, numero del top de jugadores, fecha inicial, fecha final |
| **Salidas** | Lista con los jugadores y sus estadísticas de goles, jugadores revisados en total, partidos jugados, goles anotados totales, penales y autogoles totales. |
| **Implementado (Sí/No)** | Si. Implementado por Mauricio Martinez |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Iterador de i e iterador de j** | **O(i\*j)** |
| Addlast | O(1) |
| Addlast | O(1) |
| ***TOTAL*** | ***O(i\*j)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.05 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento <<8>>**

## **Descripción**

A computer screen shot of a program code

Description automatically generated

Este requerimiento se encarga de retornar una lista con la información de los primeros N goles anotados por cierto jugador.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, nombre del jugador |
| **Salidas** | Lista con todos los goles anotados por el jugador especificado |
| **Implementado (Sí/No)** | Si. Implementado por Juan Lago |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Lt.iterator** | **O(n)** |
| Addlast | O(1) |
| ***TOTAL*** | ***O(1)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 |
| Memoria RAM | 16 RAM |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.05 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento Ejemplo**

## **Descripción**



Este requerimiento se encarga de retornar un dato de una lista dado su ID. Lo primero que hace es verificar si el elemento existe. Dado el caso que exista, retorna su posición, lo busca en la lista y lo retorna. De lo contrario, retorna None.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, ID. |
| **Salidas** | El elemento con el ID dado, si no existe se retorna None |
| **Implementado (Sí/No)** | Si. Implementado por Juan Andrés Ariza |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Buscar si el elemento existe (isPresent) | O(n) |
| Obtener el elemento (getElement) | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 7 4800HS with Radeon Graphics |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.05 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.