ANÁLISIS DEL RETO

Student-1, **Re4**, Pablo Ramírez, p.ramirezt2@uniandes.edu.co, C.E. 202321722

Student-2, **Re3**, Manuel Ricardo Torres, mr.torres@uniandes.edu.co, C.E. 202321428

Requerimiento <<1>>

Descripción

```python
def req_1(data_structs, codigo, nivel):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1
    entry = mp.get(data_structs["paises"], codigo)
    jobs_pais = me.getValue(entry)

    lst_ofertas = lt.newList("ARRAY_LIST")

    for job in lt.iterator(jobs_pais):
        if codigo == job["country_code"] and nivel == job["experience_level"]:
            lt.addLast(lst_ofertas, job)

    ordenada = sort(data_structs, lst_ofertas, "date_crit")

    return ordenada
```

Este requerimiento se encarga de retornar una lista con las últimas N ofertas de trabajo más recientes ofrecidas en un país filtrando por un nivel de experticia dado.

Entrada	Estructura del modelo, código y nivel
Salidas	Una lista TAD ordenada cronológicamente y filtrada por país y nivel de experticia.
Implementado (Sí/No)	Implementado por Manuel Ricardo Torres

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista de Jobs	O(n)
Ordenar	O(nlogn)
TOTAL	***O(nlogn)***

Pruebas Realizadas

Se realizó la prueba de dicho requerimiento usando como datos de entrada 50 últimas ofertas, código de país "PL" y nivel de experticia “junior”.

Procesadores	**Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

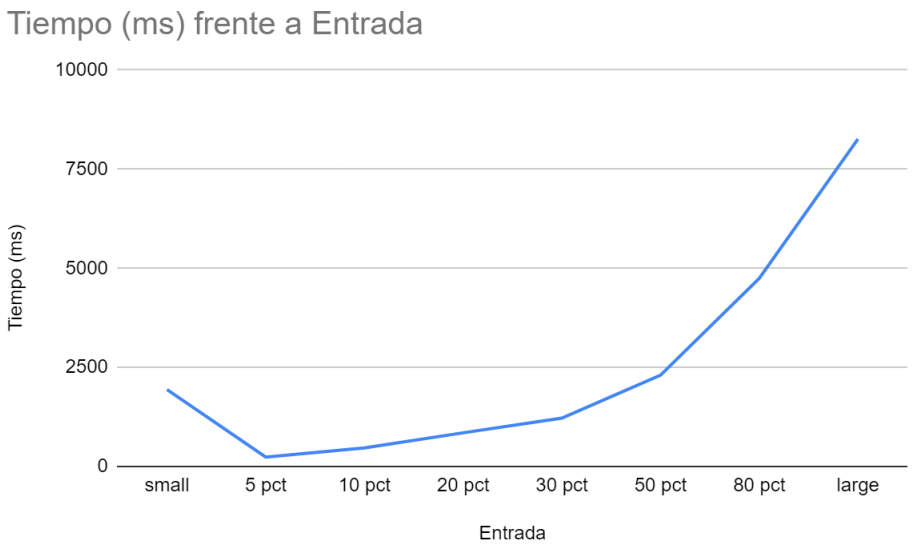
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	1941.154
5 pct	238.706
10 pct	466.343
20 pct	848.999
30 pct	1218.996
50 pct	2302.896
80 pct	4746.931
large	8260.926

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se puede observar, a medida que el tamaño de la entrada aumenta, el tiempo de ejecución también. Sin embargo, al ser complejidad de O(nlogn), no crece tan rápido como lo hace una función exponencial.

Requerimiento <<2>>

Descripción

```python
def req_2(data_structs, empresa, ciudad):
    """
    Función que soluciona el requerimiento 1
    """
    # TODO: Realizar el requerimiento 1
    entry = mp.get(data_structs["ciudades"], ciudad)
    jobs_ciudad = me.getValue(entry)

    lst_ofertas = lt.newList("ARRAY_LIST")

    for job in lt.iterator(jobs_ciudad):
        if empresa == job["company_name"]:
            lt.addLast(lst_ofertas, job)

    ordenada = sort(data_structs, lst_ofertas, "date_crit")

    return ordenada
```

Este requerimiento se encarga de retornar una lista con las últimas N ofertas de trabajo más recientes de una empresa ingresando su nombre y una ciudad.

Entrada	Estructura del modelo, empresa y ciudad
Salidas	Una lista TAD ordenada cronológicamente y filtrada por el nombre de una empresa y ciudad.
Implementado (Sí/No)	Implementado por Manuel Ricardo Torres

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista de Jobs	O(n)
Ordenar	O(nlogn)
TOTAL	***O(nlogn)***

Pruebas Realizadas

Se realizó la prueba de dicho requerimiento usando como datos de entrada 50 últimas ofertas, nombre de la empresa “mBank” y ciudad “Warszawa”.

Procesadores	**Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

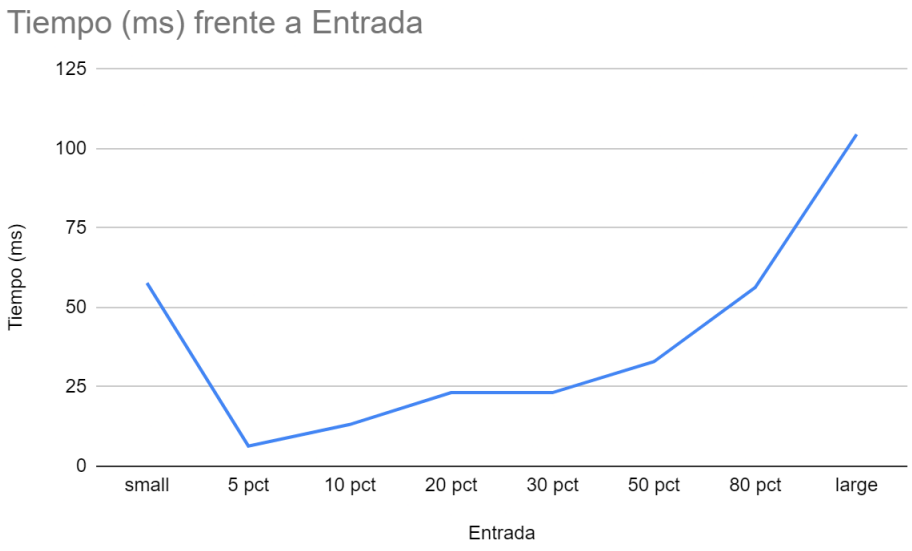
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	57.705
5 pct	6.376
10 pct	13.152
20 pct	23.182
30 pct	23.193
50 pct	32.924
80 pct	56.313
large	104.503

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se puede observar, a medida que el tamaño de la entrada aumenta, el tiempo de ejecución también. Sin embargo, al ser complejidad de O(nlogn), no crece tan rápido como lo hace una función exponencial.

Requerimiento <<3>>

Descripción

```python
def req_3(data_structs, empresa, fecha_inicial, fecha_final):
    """
    Función que soluciona el requerimiento 3
    """
    # TODO: Realizar el requerimiento 3
    entry = mp.get(data_structs["empresas"], empresa)
    jobs_empresa = me.getValue(entry)

    ofertas = lt.newList("ARRAY_LIST")

    #inicio contadores de tipos de experticia
    junior = 0
    mid = 0
    senior = 0

    for oferta in lt.iterator(jobs_empresa):
        large_date = datetime.strptime(oferta["published_at"], "%Y-%m-%dT%H:%M:%S.%fZ")
        date = str(large_date.date())
        if date>=fecha_inicial and date<=fecha_final:
            lt.addLast(ofertas,oferta)
            if oferta["experience_level"] == "junior":
                junior += 1
            elif oferta["experience_level"] == "mid":
                mid += 1
            elif oferta["experience_level"] == "senior":
                senior += 1

    ordenada = sort(data_structs,ofertas,"country_range_date_crit")
    elements = [ordenada,lt.size(ordenada), junior, mid, senior]

    return elements
```

Este requerimiento se encarga de retornar una lista con las ofertas de trabajo publicadas por una empresa en un rango de fechas dado.

Entrada	Estructura de datos del modelo, la empresa digitada por el usuario, fecha inicial y fecha final.
Salidas	Una lista de los elementos que se piden: la lista ordenada, la cantidad de datos, y la cantidad de datos de cada nivel de experticia.
Implementado (Sí/No)	Fue implementado por Manuel Ricardo Torres

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista de Jobs	O(n)
Ordenar	O(nlogn)
TOTAL	***O(nlogn)***

Pruebas Realizadas

Se realizó la prueba con los datos de entrada “ClickUp” como empresa, fecha de inicio “2022-01-01" y fecha final “2022–12-31"

Procesadores	**Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

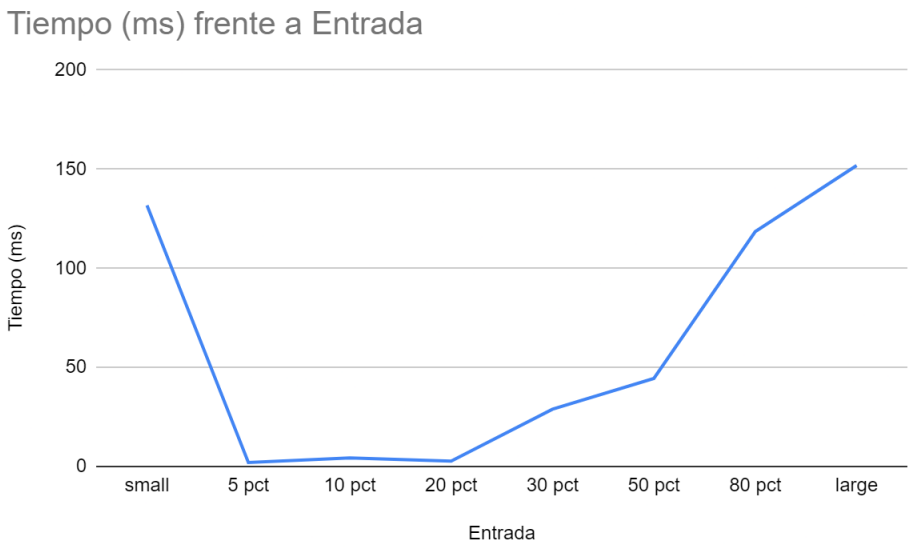
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	131.758
5 pct	1.987
10 pct	4.259
20 pct	2.699
30 pct	28.880
50 pct	44.395
80 pct	118.498
large	151.852

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se puede observar, a medida que el tamaño de la entrada aumenta, el tiempo de ejecución también. Sin embargo, al ser complejidad de O(nlogn), no crece tan rápido como lo hace una función exponencial.

Requerimiento <<4>>

Descripción

```
def req_4(data_structs, codigo, fecha_inicio, fecha_fin):
    """
    Función que soluciona el requerimiento 4
    """
    # TODO: Realizar el requerimiento 4
    pais_filtrado = mp.get(data_structs["paises"], codigo)
    pais = me.getValue(pais_filtrado)
    ofertas = lt.newList("ARRAY_LIST")

    for job in lt.iterator(pais):
        large_date = datetime.strptime(job["published_at"], "%Y-%m-%dT%H:%M:%S.%fZ")
        date = str(large_date.date())
        if date>=fecha_inicio and date<=fecha_fin:
            lt.addLast(ofertas, job)

    ordenada = sort(data_structs, ofertas, "date_company_crit")
    return ordenada
```

Este requerimiento se encarga de retornar una lista con las ofertas de trabajo publicadas en un país dado su código y un rango de fechas.

Entrada	Estructuras de datos del modelo, código del país, fecha inicial y fecha final

Salidas	Una lista filtrada dado un país por un rango de fechas y ordenada alfabética y cronológicamente por el nombre del compañía y fecha de publicación.
Implementado (Sí/No)	Implementado por Pablo Ramírez

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista de Jobs	O(n)
Ordenar	O(nlogn)
TOTAL	***O(nlogn)***

Pruebas Realizadas

Se realizó la prueba de dicho requerimiento usando como datos de entrada "PL" como país, fecha de inicio "2023-06-01" y fecha final "2023-12-01".

Procesadores	**Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

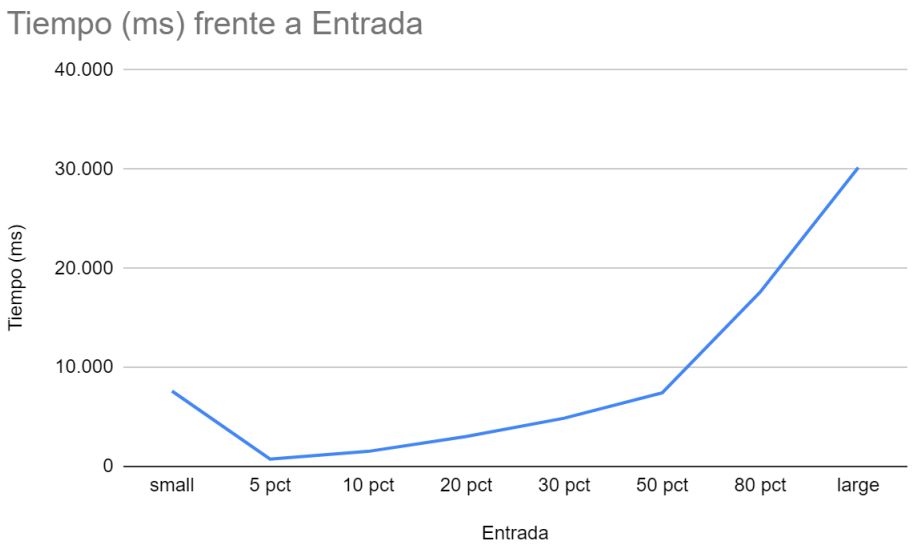
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	7606.919
5 pct	745.832
10 pct	1518.468
20 pct	3015.648
30 pct	4870.415
50 pct	7438.770
80 pct	17604.897
large	30164.516

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se puede observar, a medida que el tamaño de la entrada aumenta, el tiempo de ejecución también. Sin embargo, al ser complejidad de O(nlogn), no crece tan rápido como lo hace una función exponencial.

Requerimiento <<6>>

Descripción

Este requerimiento se encarga de retornar una tabla de hash con las N ciudades con mayor cantidad de ofertas de trabajo dado un año de publicación y un nivel de experticia.

Entrada	Estructura de datos del modelo, N ciudades, año y nivel de experticia
Salidas	Una tabla de hash cuyas llaves son las N ciudades con mayor número de ofertas y cuyos valores son las ofertas de trabajo de cada ciudad respectivamente.
Implementado (Sí/No)	Implementado por Pablo Ramírez

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista de jobs	O(n)
Recorrer la lista de employments	O(n)
TOTAL	***~O(2n)***

Pruebas Realizadas

Se realizo las pruebas con el número de ciudades 25, año “2022” y nivel de experticia “senior”.

Procesadores	**Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

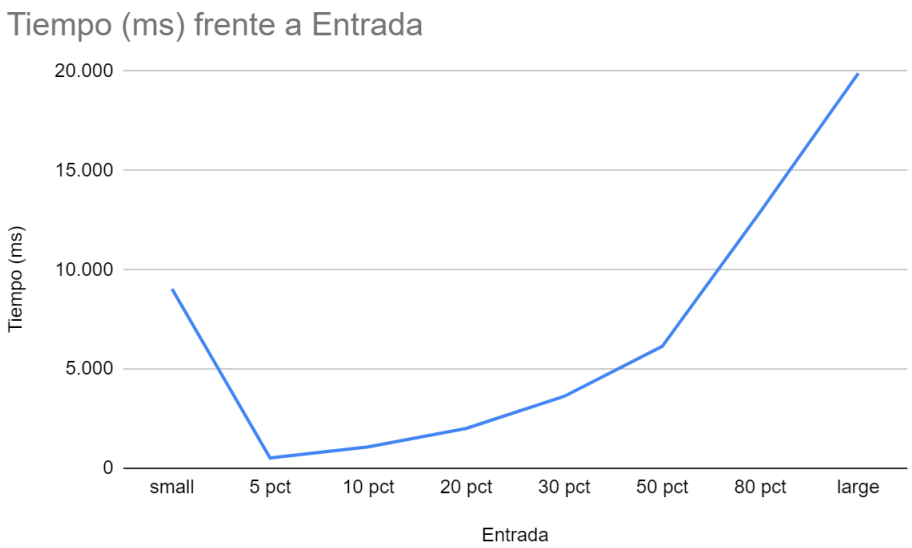
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	9031.801
5 pct	519.401
10 pct	1073.479
20 pct	2007.370
30 pct	3621.860
50 pct	6143.499
80 pct	12918.278
large	19896.225

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se observó, aunque se cargan los datos con especificaciones para que la muestra sea muy grande, se observa que la complejidad temporal aumenta linealmente.

Requerimiento <<7>>

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, numero del top de países de la consulta, y el año y mes de la consulta.
Salidas	Retorna una lista con dos listas. La primera, contiene la información general de la consulta: número total de ofertas y de ciudades y el país y la ciudad con mayores ofertas. Por otro lado, la segunda lista contiene 3 listas, las cuales contienen la siguiente información para cada una de las diversas experticias: número de habilidades, la habilidad más y menos solicitada, el promedio del nivel mínimo requerido, el número de empresas y aquella que tiene mayor y menor ofertas, y, por último, el número de empresas que tienen múltiples sedes.
Implementado (Sí/No)	Implementado por Manuel Ricardo Torres

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer las ofertas para crear una lista a ordenar	O(n)
Recorrer nuevamente para organizar las estructuras de datos	O(n)
Ordenar la lista	***O(nlogn) (Con pocos elementos)***
TOTAL	***O(nlogn) (Muy poco probable)*** ***O(2n) (Común)***

Pruebas Realizadas

Se realizó la prueba de dicho requerimiento usando como datos de entrada 10 últimos países, el año 2023 y el mes 07.

Procesadores	**Apple Silicon M1 Pro**

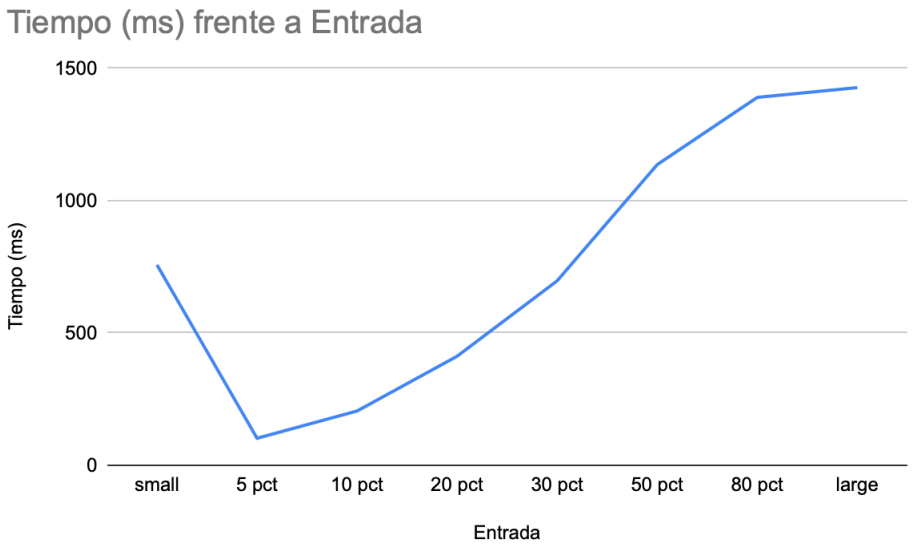
Memoria RAM	16 GB
Sistema Operativo	MacOS Sonoma

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	757
5 pct	102
10 pct	205
20 pct	412
30 pct	697
50 pct	1136
80 pct	1390
large	1427

Graficas



Análisis

Como se puede observar, a medida que la entrada crece, la complejidad temporal aumenta. Esto, se debe a que, al tener una complejidad linearítmica (nlogN) o lineal (n) (dependiendo del caso), tiene que recorrer y ordenar una mayor cantidad de elementos.

Requerimiento <<8>>

Descripción

Este requerimiento se encarga de retornar una lista de elementos ordenada con los países con mayor y menor ofertas salariales para una experticia en un periodo de tiempo dada la divisa de publicación.

Entrada	Estructuras de datos del modelo, nivel de experticia, divisa, fecha inicial y fecha final.
Salidas	Una lista filtrada con los países ordenados de mayor a menor promedio de oferta salarial.
Implementado (Sí/No)	Implementado por Manuel Ricardo Torres y Pablo Ramírez

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista de Jobs	O(n)
Ordenar	O(nlogn)
TOTAL	***O(nlogn)***

Pruebas Realizadas

Se realizó la prueba de dicho requerimiento usando como nivel de experticia “indiferente”, divisa “pln”, fecha inicial “2022-01-01" y fecha final “2023-01-01"

Procesadores	**Apple Silicon M1 Pro**
Memoria RAM	16 GB
Sistema Operativo	MacOS Sonoma

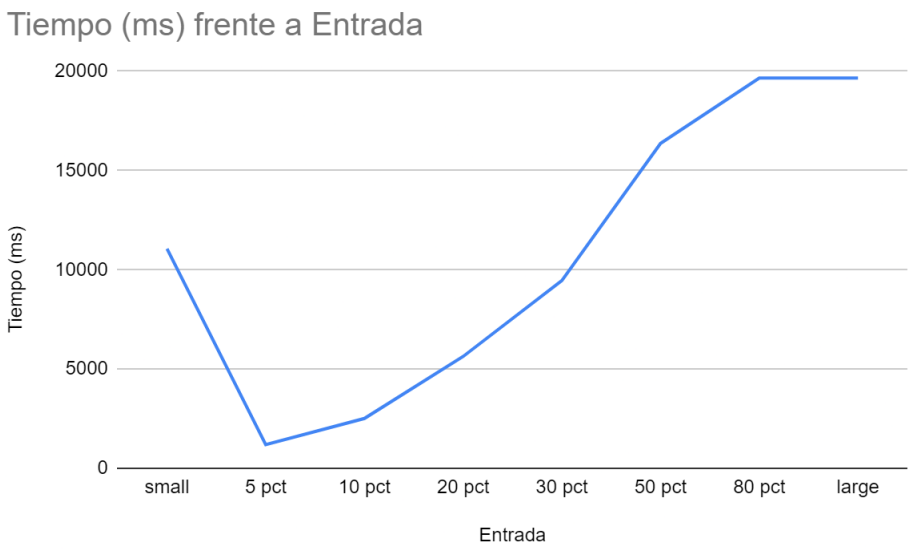
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	11061
5 pct	1188
10 pct	2509
20 pct	5633
30 pct	9454
50 pct	16368
80 pct	19657
large	19661

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se puede observar, a medida que la entrada crece, la complejidad temporal aumenta. Esto, se debe a que, al tener una complejidad linearítmica (nlogN) o lineal (n) (dependiendo del caso), tiene que recorrer y ordenar una mayor cantidad de elementos.