Instituto Politécnico Nacional

Escuela Superior de Cómputo

Ingeniería en Inteligencia Artificial, Análisis y Diseño de Algoritmos
Sem: 2024-1, 3BV1, Práctica 1, 14 de septimebre de 2023

Práctica 1: Determinación experimental de la complejidad temporal de un algoritmo

Catonga Tecla Daniel Isaí 1, Olguin Castillo Victor Manuel 2.

$daniel9513importantes@gmail.com_1, manuelevansipn@gmail.com_2$

Resumen: Redactar de manera breve y concisa de que trata el trabajo presentado.

Palabras Clave: Algoritmo, C++, big O, Complejidad Temporal

1. Introducción

Un algoritmo es una serie de pasos o instrucciones que deben ser precisos, ordenados y finitos. La finalidad de un algoritmo es para resolver problemas como para optimizar procesos y es de suma importancia ya que los algoritmos ayudan a la resolución de problemas muy grandes o complejos de forma que el problema se descompone en problemas más pequeños haciendo referencia a "divide y vencerás", es importante mencionar que un algoritmo consta de 3 partes que es la entrada, proceso y salida con esto tenemos un orden para que sea claro y compresivo el algoritmo que se desarrolla y así poder mejorar este mismo si es necesario.

Los algoritmos han sido esenciales para la resolución de problemas matemáticos y científicos por lo que los algoritmos son fundamentales en varias áreas de las ciencias o ingeniería por lo antes mencionado.

El análisis de algoritmos es el estudio del diseño de estos, en complejidad temporal (número de operaciones o pasos ejecutados) y complejidad espacial (recurso de memoria que utiliza). Esto es de suma importancia debido a que en cuestión de complejidad temporal algunos algoritmos pueden requerir años en resolver un solo problema, esto no es para nada eficiente en la práctica por lo tanto es importante conocer la complejidad temporal de un algoritmo antes de aplicarlo en la vida real. Por otra parte, sabemos que una maquina no cuenta con recursos infinitos de memoria o espacio de almacenamiento, por lo tanto, el algoritmo diseñado puede consumir más memoria de la que el equipo dispone causando problemas por lo que se tiene que evaluar su complejidad espacial para evitar este tipo de problemas y no causar daños al equipo.

Por lo tanto, el análisis de algoritmos es importante ya que con esto se puede evaluar el comportamiento de un algoritmo en específico y así ver si es eficiente o no, con esto podemos reducir costos, optimizar recursos, reducir los cuellos de botella, etc.

Y todo esto nos lleva a evaluar la complejidad temporal de los algoritmos de la práctica para ver su comportamiento que tienen haciendo uso de gráficas para ver el mejor de los casos, peor de los casos y casos promedios, observando la eficiencia que tienen los algoritmos llegando a una conclusión con respecto a lo llevado a cabo.

2. Experimentación y Resultados

Ejercicio 1

Con el código desarrollado en C++, se considera la experimentación con el mejor caso, peor caso y caso promedio. Por lo tanto para el mejor caso, hacemos que nuestro arreglo contenga el mismo valor en todo el arreglo, con esto nuestro algoritmo encontrará el mismo valor en la primera iteración. Ejecutando nuestro programa obtenemos la siguiente tabla:

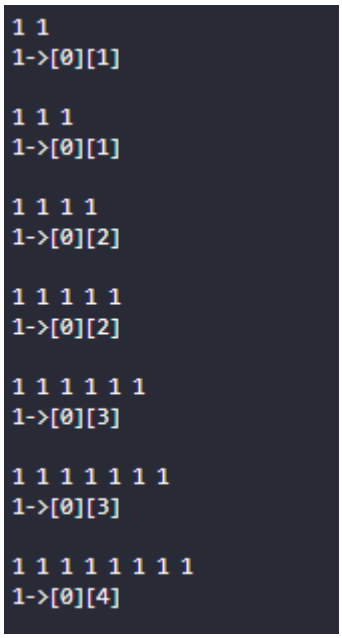

Figura 1. Resultado de la ejecución del algoritmo (mejor caso).

Valor de n	**# de pasos**
2	11
3	11
4	11
5	11
6	11
7	11
8	11
9	11
10	11
11	11
12	11
13	11
14	11
15	11
16	11
⋮	⋮

Tabla 1. Datos obtenidos (mejor caso).

Después de ver que obtenemos un resultado constante decimos que el mejor de los casos nuestro algoritmo es $\Omega(1)$ por lo que si gráficamos nuestros datos obtenemos el siguiente resultado:

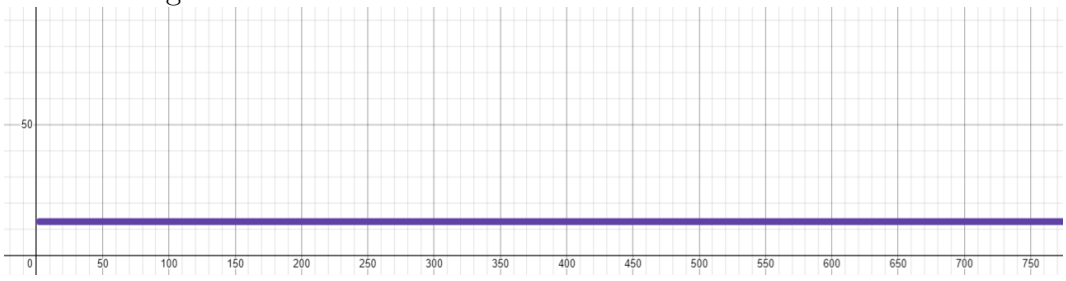

Figura 2. Grafica "mejor caso".

Sabiendo que es lineal entonces tiene relación con la gráfica ya que es lineal. Por otra parte tenemos que evaluar nuestro algoritmo en el peor de los casos, para esto ningún valor se debe de repetir en el vector, obteniendo lo siguiente:

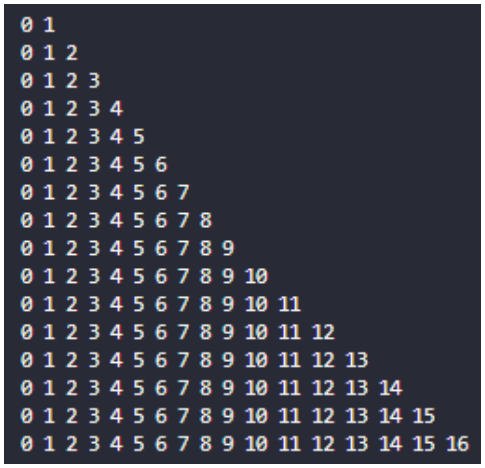

Figura 3. Resultado de la ejecución del algoritmo (peor caso).

Valor de n	**# de pasos**
2	11
3	14
4	24
5	30
6	43
7	52
8	68
9	80
10	99
11	114
12	136
13	154
14	179
15	200
16	228
⋮	⋮

Tabla 2. Datos obtenidos (peor caso).

Con los datos obtenidos observamos que mientras n aumenta, el número de pasos va aumentando rapidamente, graficando se muestra el comportamiento. Graficamos los datos sobre la misma grafica donde se encuentra el mejor de los casos, obtenemos lo siguiente:

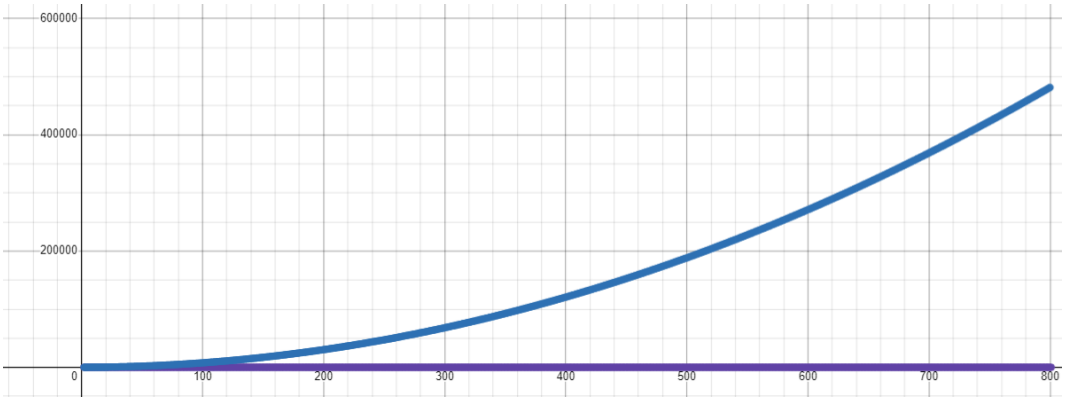

Figura 4. Grafica mejor caso y peor caso.

Observando la gráfica podemos ver que para el peor caso tiene comportamiento de una función cuadratica, siendo este el de los puntos azules que parece una línea, es por esto que decimos que para el peor caso nuestro algoritmo es $O(n^2)$.

Por último se ejecuta el algoritmo para ver su comportamiento en los casos promedios, obteniendo los siguientes, resultados:

```
2 1
8 4 1
1 3 2 4
8 10 5 0 9
1 8 8 14 3 5
10 12 3 4 11 19 21
8 24 16 14 18 21 9 11
16 22 21 8 17 13 3 26 9
13 28 4 12 25 21 12 12 4 26
4->[2][8]

16 14 31 8 15 27 17 0 7 18 14
14->[1][10]

28 12 19 15 34 13 22 31 9 16 31 28
28->[0][11]

8 34 21 24 33 1 39 6 1 20 28 25 8
8->[0][12]
```

Figura 5. Resultado de la ejecución del algoritmo (casos promedio).

Valor de n	**# de pasos**
2	11
3	14
4	24
5	30
6	43
7	52
8	68
9	80
10	58
11	48
12	26
13	29
14	129
15	57
16	57
⋮	⋮

Tabla 3. Datos obtenidos (casos promedio).

En la tabla se puede observar que en los primeros casos son iguales al peor caso, pero conforme n es mayor ya se nota un cambio, se puede observar desde que $n \geq 10$ los "# de pasos"disminuyen, si graficamos todo en la misma gráfica obtenemos:

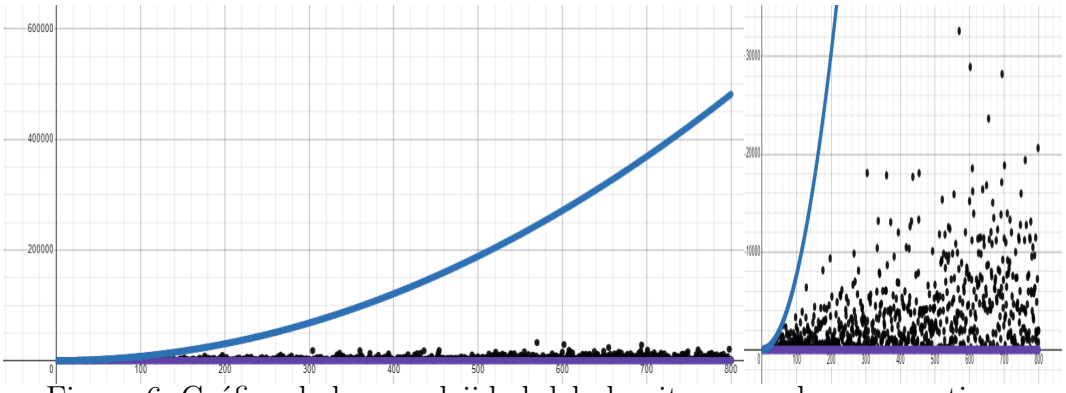

Figura 6. Gráfica de la complejidad del algoritmo con dos perspectivas.

Se puede observar que en la gráfica se tiene diversos puntos pero todos están entre el mejor caso y peor caso, con esto observamos que en verdad nuestros casos promedios no superan al peor de los casos, pero también hay una particulairdad es que casi ningun caso promedio se acerca al peor caso. Por otra parte si ponemos una función tal que $g(n) = n^2$ y que $g(n) = 10$ tendremos lo siguiente:

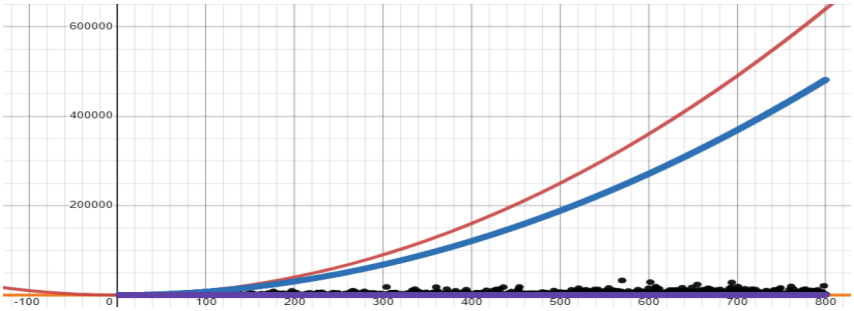

Figura 7. Gráfica con límites.

Con esto, nuestra n_0 para el ajuste asíntotico sería cuando $n_0 \geq 250$ ya que desde la gráfica podemos ver que desde ese punto ya se nota que $f(n)$ no supera a n^2 que para nuestro algoritmo como resultado tenemos que es $O(n^2)$, y tenemos que para nuestro Ω tenemos que es $\Omega(1)$.

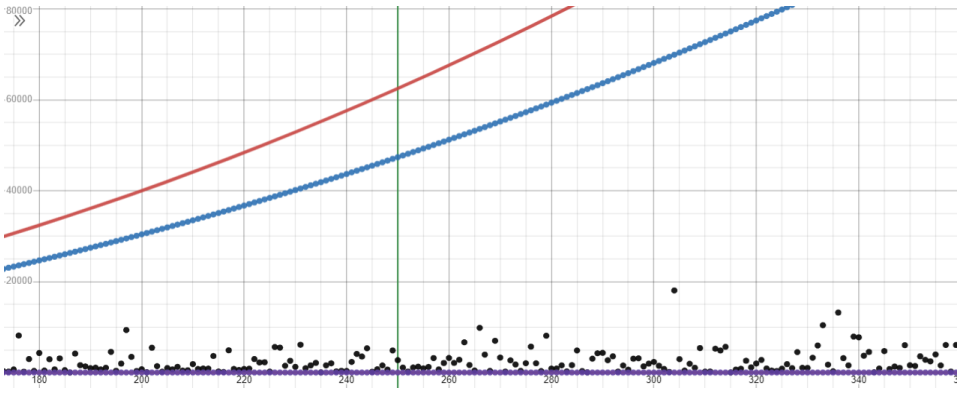

Figura 8. Resultado del primer ejercicio.

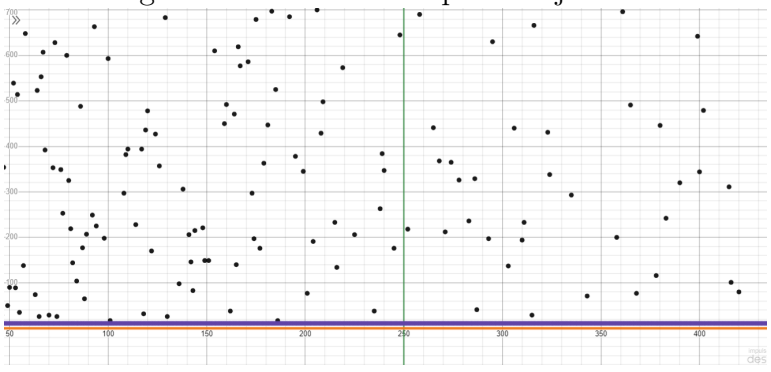

Figura 9. Resultado de una perspectiva más cerca.

3. Conclusiones

Se intentó encontrar el caso lineal del primer algoritmo pero se desconocía de la implementación del HashMap por lo que se optó por utilizar lo que es un caso cuadratico, en este caso se utilizó la fuerza bruta, por otra parte en el primer algoritmo se tuvo el problema de que estábamos sobrescribiendo los datos del .csv y se perdian los datos, la solución que implementamos fue que por cada caso se creó un nuevo archivo de .csv cambiando solo el nombre en el código de C++.

Conclusiones Catonga Tecla Daniel Isaí 1

Después de la experimentación, aprendí cómo se podría implementar el primer algoritmo con HashMap dejando de un lado la fuerza bruta. Por otra parte, con los resutados de la práctica pude observar en las dos gráficas de los algoritmos, que son muy distintos. Esto es interesante, ya que indica que cada algoritmo tiene su propia complejidad. En este caso, no se realizó el análisis a priori pero es interesante cómo graficamente, nos podemos dar cuenta de cómo se comportan los algoritmos y con ello me dió un panorama más a detalle de lo que se hace en el análisis a priori indirectamente, me gustó la práctica ya que es importante tener nociones de lo que es big O, dadp que se usa en la vida laboral. Aprendí cosas interesantes y relevantes en esta práctica.

Conclusiones Alumno 2: [Texto de las conclusiones del Alumno 2]

4. Anexo

En esta sección se anexarán la resolución de los problemas de tarea planteados utilizando **LaTeX** (**NO** fotos de los problemas realizados a mano).

5. Bibliografía

Mostrar referencias en **formato APA**. **Importante observación**: Si alguna práctica tiene al menos el 25 % de similitud respecto a otros trabajos, la calificación de esa práctica será igual a 0. Si se detecta alguna práctica plagiada en su totalidad (código o reporte), el equipo será acreedor a una sanción de la califición igual a 0 en toda la evaluación correspondiente. Para evitar el plagio en sus trabajos (reporte), se utilizará la plataforma de **turnitin**.