

Universidad Santiago de Chile  
Facultad de Ciencia  
Licenciatura en Ciencia de la Computación

## COMPLEJIDAD DE ALGORITMOS

“Algoritmos de búsqueda y ordenamiento”

2º Laboratorio

**Datos Informe**

**Nombre alumnos:** Sergio Salinas

Danilo Abellá

**Nombre profesor:** Nicolas Thériault

## Contents

<b>1</b>	<b>Problema 1</b>	<b>2</b>
1.1	Solución Encontrada . . . . .	2
1.2	Estrategia . . . . .	2
1.3	Justificación . . . . .	2
<b>2</b>	<b>Problema 2</b>	<b>3</b>
2.1	Solución Encontrada . . . . .	3
2.2	Estrategia . . . . .	5
2.3	Justificación . . . . .	5
<b>3</b>	<b>Problema 3</b>	<b>6</b>
3.1	Solución Encontrada . . . . .	6
3.2	Estrategia . . . . .	6
3.3	Justificación . . . . .	6

# 1 Problema 1

## 1.1 Solución Encontrada

La solución que se encontró es que el número que más se repite es 5896 con 16 repeticiones.

## 1.2 Estrategia

El algoritmo guarda todos los caracteres del archivo tex en un arreglo A y crea otro arreglo B de lleno de ceros que tiene 10000 espacios de memoria y se usa para almacenar las frecuencias que tiene cada número.

En un ciclo, se obtienen los primeros 4 dígitos de A y se usan como índice para acceder al espacio de memoria que corresponde a ese índice y aumenta la frecuencia de aparición de ese número en 1, luego se quita el primer elemento de A aumentando la posición del puntero en 1 y se vuelve al inicio del ciclo. Esto se repite hasta que se recorra A por completo.

Además hay otra variable que se encarga de guardar la mayor frecuencia durante el ciclo, para mostrar el número que más repite solo se busca en B el índice que tenga esa frecuencia.

## 1.3 Justificación

Se utiliza este método por su efectividad, en lo que es la memoria, el arreglo A es de largo  $n$  y el arreglo B siempre es constante ya que siempre tendrá 10000 espacios de memoria que son los necesarios para guardar la frecuencia de los números del 0 al 9999, por lo la memoria total utilizada es de  $n + 10000$ , por lo que es de orden  $O(n)$ .

En lo que es la complejidad del algoritmo, es de orden  $O(n)$ , esto es debido a que en el ciclo de aumentar la frecuencia de B dado un índice obtenido de A su operación básica se repite  $n$  veces, y en el ciclo de buscar el mayor número la operación comparar se repite 10000 veces, por lo que se tiene que la complejidad es  $n + 10000$ .

## 2 Problema 2

### 2.1 Solución Encontrada

Como solución se tienen las siguientes secuencias de 4 “dígitos” (hexadigitos) consecutivos que se repiten en el número hexadecimal dado:

Secuencia Repetida - Frecuencia.

0006 - 2	1548 - 2	2821 - 2	3E89 - 2	56E1 - 2
0068 - 2	15D6 - 2	2850 - 2	3FD6 - 2	5748 - 3
00BB - 2	1612 - 2	2851 - 2	4000 - 2	5770 - 2
00DF - 2	1625 - 2	2858 - 2	4152 - 2	586E - 2
0115 - 2	1636 - 2	2895 - 2	4183 - 2	5882 - 2
011A - 3	16DF - 2	28AB - 2	42EF - 2	593E - 2
013C - 2	1811 - 2	299B - 2	42F5 - 2	59CB - 2
01AB - 2	183B - 2	299F - 2	42F6 - 2	5AA5 - 2
01AD - 2	1856 - 2	2AB3 - 2	4324 - 3	5BDD - 2
03A1 - 2	18B1 - 2	2ADA - 2	442E - 2	5C02 - 2
0402 - 2	1936 - 2	2B89 - 2	4574 - 2	5CB0 - 2
04C0 - 2	1948 - 2	2B8B - 2	4595 - 2	5CFA - 3
056A - 2	19EE - 2	2D38 - 2	4614 - 2	5D88 - 2
0584 - 2	1BD3 - 2	2D51 - 2	46DB - 2	5EE3 - 2
061B - 2	1C20 - 2	2DFD - 2	46F6 - 2	5F01 - 2
0734 - 2	1C36 - 2	2E13 - 2	46FC - 2	5F79 - 2
07EF - 2	1C90 - 2	2E28 - 2	479B - 2	5F95 - 2
0810 - 2	1CE6 - 2	2E9C - 2	4898 - 2	5FEC - 2
085F - 2	1D00 - 2	2EB3 - 2	48E4 - 2	602A - 2
08BA - 2	1D39 - 2	2EF1 - 2	48FD - 2	6078 - 2
0A12 - 2	1DC2 - 2	2EF6 - 2	49F1 - 2	6085 - 2
0A47 - 2	1DDF - 2	2F32 - 2	4A41 - 2	612A - 2
0BA6 - 2	1E0A - 2	30DC - 2	4A99 - 2	614E - 2
0BCB - 2	1E63 - 2	314E - 2	4B27 - 2	618B - 2
0DC7 - 2	1E79 - 2	317C - 2	4B47 - 2	6282 - 3
0DF0 - 2	1E8A - 2	31EA - 2	4BA3 - 2	6293 - 2
0F28 - 2	1EAF - 2	31F6 - 3	4BFB - 2	62E9 - 2
1018 - 2	1EBD - 2	320F - 2	4CDD - 2	6369 - 2
105D - 2	1F2E - 2	33A3 - 2	4E3D - 2	6382 - 2
10B3 - 2	1F6C - 2	33E8 - 2	4E6C - 2	6629 - 3
1133 - 2	1F9B - 2	3604 - 2	501A - 3	6636 - 2
1156 - 2	202E - 2	361D - 2	5056 - 2	6645 - 2
1159 - 2	20AD - 2	369B - 2	5094 - 2	66A0 - 2
1183 - 2	20C8 - 2	3707 - 2	5133 - 2	67BC - 2
11A1 - 2	2105 - 2	3820 - 2	525F - 2	6842 - 2
1290 - 3	21E7 - 2	3822 - 2	5282 - 2	698D - 2
12B4 - 2	2299 - 2	38D8 - 2	52DF - 2	6A36 - 2
12DC - 2	22B6 - 2	392E - 2	52EC - 2	6A37 - 2
133A - 2	2463 - 2	396A - 2	532E - 2	6A51 - 2
13E0 - 2	2466 - 2	3A3A - 2	542F - 2	6B2A - 2
1421 - 2	2469 - 2	3AE1 - 2	5470 - 2	6B6A - 2
1462 - 2	2547 - 2	3B12 - 2	5546 - 2	6DFF - 2
14BA - 2	25BD - 2	3B3E - 2	55AB - 2	6E2F - 2
14CD - 2	2623 - 2	3C6F - 2	55FD - 2	6E37 - 2
14D9 - 2	2690 - 2	3D5A - 2	5605 - 2	6EA6 - 2
14E5 - 3	2796 - 2	3D7C - 2	562E - 2	6F47 - 2
153C - 2	27A1 - 2	3E81 - 2	5664 - 2	6FB2 - 2

6FF3 - 2	9045 - 2	B023 - 2	CA92 - 2	E799 - 2
7061 - 2	924A - 2	B03A - 2	CB06 - 2	E8A3 - 2
7080 - 2	9317 - 2	B155 - 2	CBEE - 2	E8AE - 2
7157 - 2	9361 - 2	B1DC - 2	CCC8 - 2	E8EF - 2
7216 - 2	93D5 - 2	B21A - 2	CE0B - 2	E9DB - 2
724D - 2	93E8 - 2	B266 - 3	CE77 - 2	EB26 - 2
74E6 - 2	94B7 - 3	B277 - 2	CF0B - 2	EB65 - 2
7509 - 2	94C2 - 2	B27B - 2	D04C - 2	EC6E - 2
7560 - 2	9586 - 2	B285 - 2	D1B5 - 2	ECC8 - 2
75B1 - 2	9662 - 2	B2F3 - 2	D1CF - 2	ECF1 - 2
7634 - 2	96A2 - 2	B331 - 2	D1D0 - 2	ED93 - 2
7711 - 2	9802 - 2	B38E - 2	D519 - 2	EDFA - 2
771F - 2	991B - 2	B3B1 - 2	D586 - 2	EE30 - 2
7732 - 2	993B - 2	B3EE - 2	D65F - 2	EE60 - 2
7808 - 2	9A45 - 2	B457 - 2	D6A1 - 2	EECC - 2
78C1 - 2	9A53 - 2	B513 - 2	D6EB - 2	EF6A - 2
7960 - 2	9B04 - 2	B557 - 2	D83D - 2	EF7D - 2
7A58 - 2	9B07 - 2	B5FA - 2	D8FE - 2	F011 - 2
7B8E - 2	9B6F - 2	B6A3 - 2	D915 - 2	F017 - 2
7B9F - 2	9C2B - 2	B6C1 - 2	D993 - 2	F01C - 2
7CCF - 2	9C60 - 2	B750 - 2	D9B9 - 2	F050 - 2
7CD3 - 2	9D65 - 2	B7D9 - 2	DA2F - 2	F06A - 2
7D9C - 2	9DCB - 2	B7E3 - 2	DAE9 - 2	F092 - 2
7DA8 - 2	9E15 - 2	B812 - 2	DC09 - 2	F0BD - 2
7DD1 - 2	9E86 - 2	B820 - 2	DC14 - 2	F0CA - 2
7E6A - 2	9F25 - 2	B8B5 - 2	DC25 - 2	F0F7 - 2
800B - 2	9F8D - 2	B93D - 2	DD57 - 2	F19B - 2
80BB - 2	9FB4 - 2	B9D3 - 2	DDF8 - 2	F2B8 - 2
8128 - 2	A0C1 - 2	BA33 - 2	DF01 - 2	F442 - 2
8129 - 2	A0E2 - 2	BA99 - 2	DFA1 - 2	F5EB - 2
8162 - 2	A124 - 2	BA9B - 3	DFA6 - 2	F708 - 2
82EF - 2	A14D - 2	BB3A - 2	DFF8 - 2	F724 - 2
8346 - 2	A161 - 2	BB56 - 2	E03A - 2	F728 - 2
83B5 - 2	A1D4 - 2	BBCA - 2	E0E1 - 2	F74B - 2
84A5 - 2	A285 - 2	BC2A - 2	E14E - 2	F74E - 2
8507 - 2	A323 - 2	BC9B - 2	E153 - 2	F7DA - 2
8509 - 2	A366 - 2	BD31 - 2	E169 - 2	F845 - 2
8556 - 2	A37F - 2	BE32 - 2	E1B0 - 2	F89E - 2
858E - 2	A3CB - 2	BF00 - 2	E1E7 - 2	F8E7 - 2
85A3 - 2	A476 - 2	BF2C - 2	E1F9 - 2	F8E8 - 2
85F0 - 2	A58C - 2	C115 - 2	E305 - 2	F953 - 2
86E3 - 2	A65C - 2	C25A - 2	E39F - 2	F95E - 2
8740 - 2	A7A9 - 2	C277 - 3	E3D0 - 2	FA3D - 2
8839 - 2	A835 - 2	C2DA - 2	E3D3 - 2	FAD5 - 2
8986 - 2	A995 - 2	C2DD - 2	E4C6 - 2	FBC9 - 2
8A0B - 2	A9BE - 2	C372 - 2	E4D6 - 2	FCED - 2
8B1D - 2	ABEA - 2	C4BF - 2	E593 - 2	FD2C - 2
8B38 - 2	ABFC - 2	C511 - 2	E60B - 2	FE6E - 2
8D01 - 2	AC71 - 2	C6E8 - 2	E647 - 2	FF8E - 2
8D40 - 2	ACE1 - 2	C902 - 2	E69F - 2	FFA3 - 2
8DB3 - 3	ADA3 - 3	C904 - 2	E6AD - 2	FFEA - 2
8F48 - 2	AE4D - 2	C946 - 2	E6BA - 2	
8FD2 - 2	AFD6 - 2	C9E0 - 2	E6C5 - 2	

## 2.2 Estrategia

Se utiliza un arreglo "a" para guardar todos los dígitos del archivo tex y "hexarray" (solo con ceros y 10000 en espacio de memoria) para guardar las secuencias de 4 dígitos hexadecimales. Nota: Para trabajar con números hexadecimales se utilizó la función "strtol(char, NULL, 16)".

Luego se recorrió el arreglo "hexarray" ordenando todas las secuencias de 4 dígitos con el método de ordenamiento por inserción.

Una vez ordenado de menor a mayor, el arreglo es recorrido y se van mostrando las secuencias que se repiten al menos 1 vez con su respectiva frecuencia.

## 2.3 Justificación

Se decide crear una lista ordenada con todos los números de 4 dígitos del archivo tex debido a su facilidad de uso y porque el coste en memoria de este arreglo es de  $n-4$ . La complejidad de obtener todos los números de 4 dígitos del arreglo con los datos del archivo es de  $n-4$ .

Se decide usar el método de inserción dado a sus bajos requerimientos de memoria y estabilidad (nunca intercambia registros con claves iguales). Pese a su lento proceso y la numerosas comparaciones necesarias a realizar, muestra un comportamiento razonablemente bien en gran cantidad de situaciones, y esta no es la excepción, además que ser de fácil implementación.

El método de inserción tiene como complejidad en el peor de los casos  $O(n^2)$  y en el mejor de los casos  $O(n)$  (o sea, cuando el arreglo está ordenado), y el recorrido guardar las secuencias de 4 dígitos hexadecimales una es de  $n-4$ .

Por lo que la complejidad del algoritmo sería:

En el mejor caso:  $n + (n-4)$

En el peor caso:  $n^2 + (n-4)$

Y su uso de memoria total es de  $2n-4$ , debido a que  $n$  es lo que cuesta el arreglo con los dígitos de un archivo,  $n-4$  el arreglo con todos los números de 4 dígitos y el insertion sort solo requiere  $O(1)$  de espacio adicional.

### 3 Problema 3

#### 3.1 Solución Encontrada

*Archivo<sub>3</sub>.tex* no arroja ningún resultado, por lo que no hay ninguna secuencia de 6 dígitos repetidas según el corte que se le dio.

#### 3.2 Estrategia

Se uso un método similar al segundo ejercicio, se guarda la data del texto *Archivo<sub>3</sub>.tex* en un arreglo A de largo n, este arreglo se recorre  $\frac{n}{6}$  veces extrayendo en segmentos de 6 dígitos cada número del arreglo y guardandolos en un nuevo arreglo B en forma de INT.

Este arreglo B se ordena usando el método de ordenamiento quicksort, de esta forma el arreglo B tiene los elementos repetidos uno al lado del otro, por último se recorre el arreglo contando la frecuencia con la que aparece cada elemento repetido de manera seguida y los que tengan frecuencia mayor a uno se muestran.

#### 3.3 Justificación

Se utilizo este método debido a que su memoria es lineal, solo ocupa dos arreglos de largo n y  $\frac{n}{6}$ , el quicksort ocupa  $\log n$  de memoria, por lo que ocupa  $\frac{7n}{6} + \log n$  de memoria.

Además de su rapidez, dividir el arreglo en secciones de 6 tiene complejidad de  $\frac{n}{6}$  y ordenar el arreglo con el método quicksort tiene una complejidad de  $n \log n$ , por lo que su complejidad final es de  $\frac{n}{6} + n \log n$ , haciendo que el algoritmo sea de complejidad lineal logarítmica.

Se usa el algoritmo quicksort debido a su rapidez y que ser el largo del arreglo par es más seguro que el pivote elegido sea el correcto.