



UNIVERSIDAD DE GRANADA

Práctica 2: Divide y Vencerás

Análisis de algoritmos Divide y Vencerás

José María Gómez García
Fernando Lojano Mayaguari
Valentino Lugli
Carlos Mulero Haro

Universidad de Granada
Granada
Abril 2020

Índice

1	Introducción	2
2	Problema común: Matriz traspuesta	2
2.1	Descripción de caso de ejecución	2
2.1.1	Fuerza Bruta	2
2.1.2	Divide y Vencerás	3
2.2	Análisis teórico	4
2.2.1	Fuerza Bruta	4
2.2.2	Divide y Vencerás	4
2.3	Análisis empírico	4
2.3.1	Fuerza Bruta	5
2.3.2	Divide y Vencerás	5
2.4	Análisis híbrido	6
2.4.1	Fuerza Bruta	6
2.4.2	Divide y Vencerás	7
2.5	Obtención del umbral	8
3	Problema asignado: Eliminar elementos repetidos en vector	10
3.1	Descripción de caso de ejecución	10
3.1.1	Fuerza bruta	10
3.1.2	Divide y Vencerás	11
3.2	Análisis teórico	12
3.2.1	Fuerza Bruta	12
3.2.2	Divide y Vencerás	12
3.3	Análisis empírico	12
3.3.1	Fuerza Bruta	13
3.3.2	Divide y Vencerás	13
3.4	Análisis híbrido	15
3.4.1	Fuerza Bruta	15
3.4.2	Divide y Vencerás	16
3.5	Obtención del umbral	16
4	Conclusión	17

1 Introducción

En esta práctica se nos pide analizar de diferentes formas dos tipos de problemas. En nuestro caso vamos a tratar el problema de la matriz traspuesta (común a todos los grupos) y la eliminación de elementos repetidos en un vector. El análisis consiste en realizar para cada problema dos versiones de algoritmos: la primera se realizará por “fuerza bruta” y la segunda con el método divide y vencerás. Para cada una de las versiones creadas se realizará lo siguiente:

- Descripción de un caso de ejecución, el cálculo de la eficiencia teórica, análisis empírico e híbrido de los algoritmos y finalmente el análisis del umbral.
- El ordenador que se utilizó en esta práctica para realizar los análisis empíricos e híbridos posee un **procesador** Intel Core i3-4000M, de **memoria RAM** posee 16 GB y utilizando el **SO** Ubuntu 19.10
- Para compilar los programas solo se requiere de utilizar la orden **make**

2 Problema común: Matriz traspuesta

2.1 Descripción de caso de ejecución

2.1.1 Fuerza Bruta

La función que traspone una matriz cuadrada de dimensión 2^k , mediante el método de “fuerza bruta” lleva a cabo esta tarea colocando el elemento de la posición (i,j), en la posición (j,i) de una matriz auxiliar. Finalmente asignamos los valores de la matriz auxiliar a la matriz original. De esta forma conseguimos modificar la matriz original ya que dicha matriz se pasa como parámetro por referencia en la función. A continuación se muestra un ejemplo de ejecución:

```
[CarlosMuleroHaro carlos@carlos-HP-EliteBook-840-G5:~/Escritorio/Universidad/ALG/P2] 2020-04-02 jueves
$ ./ejecprueba 4
Matriz original:

9 8 0 3
2 9 4 6
4 9 4 5
8 6 5 4

Pasos para trasponer la matriz:
Paso 1: ponemos la columna 0 de la matriz inicial como la fila 0 de la traspuesta

9 2 4 8

Paso 2: ponemos la columna 1 de la matriz inicial como la fila 1 de la traspuesta

9 2 4 8
8 9 9 6

Paso 3: ponemos la columna 2 de la matriz inicial como la fila 2 de la traspuesta

9 2 4 8
8 9 9 6
0 4 4 5

Paso 4: ponemos la columna 3 de la matriz inicial como la fila 3 de la traspuesta

9 2 4 8
8 9 9 6
0 4 4 5
3 6 5 4

Resultado:

9 2 4 8
8 9 9 6
0 4 4 5
3 6 5 4
```

2.1.2 Divide y Vencerás

La función que traspone una matriz cuadrada de dimensión 2^k , mediante el método de “divide y vencerás” establece como caso base la obtención de una matriz de 2×2 , ya que de esta forma lo único que se haría para transponerla sería intercambiar los elementos $(1,2)$ y $(2,1)$. Cuando no se está en el caso base, se realiza una división de la matriz en 4 submatrices de tamaño $\frac{n}{2}$ las cuales utilizarán 4 llamadas recursivas: una para cuadrante de la original. Cabe destacar que la “división” no implica crear 4 nuevas submatrices, en realidad se utiliza la misma matriz todo el tiempo pero en las llamadas recursivas se restringen las filas y columnas a un cuadrante dado, esto se repetirá hasta alcanzar el caso base. Tras realizarse completamente el caso base, se realiza el paso final de juntar las soluciones mediante dos bucles anidados. Este paso consiste, en esencia, en lo mismo que realiza el caso base, pero ahora adaptado a las 4 submatrices iniciales, de forma que se intercambian la submatriz de la esquina superior derecha con la de la esquina inferior izquierda, quedando así completamente traspuesta la matriz inicial. A continuación se muestra un ejemplo de ejecución:

```
/cygdrive/c/Users/chemi/Desktop/Algoritmica-master/P2-DyV/Codigo
chemi@LAPTOP-VF95F98N /cygdrive/c/Users/chemi/Desktop/Algoritmica-master/P2-DyV/Codigo
$ ./m 4
Matriz inicial:
3 4 9 2
5 2 3 4
4 4 4 9
9 6 6 6
Llamada al método DyV
¿Es el caso base (2x2)? No
Divido la matriz en cuatro submatrices:
3 4 9 2
5 2 3 4
4 4 4 9
9 6 6 6
Hago una llamada al método DyV con la primera submatriz:
¿Es el caso base (2x2)? Si
Realizamos el intercambio en el caso de la matriz base:
3 4
5 2
Cambiamos '5' por '4':
3 5
4 2
Hago una llamada al método DyV con la segunda submatriz:
¿Es el caso base (2x2)? Si
Realizamos el intercambio en el caso de la matriz base:
9 2
3 4
Cambiamos '3' por '2':
9 3
2 4
Hago una llamada al método DyV con la tercera submatriz:
¿Es el caso base (2x2)? Si
Realizamos el intercambio en el caso de la matriz base:
4 4
9 6
Cambiamos '9' por '4':
4 9
4 6
Realizamos el intercambio en el caso de la matriz base:
9 2
3 4
Cambiamos '3' por '2':
9 3
2 4
Hago una llamada al método DyV con la tercera submatriz:
¿Es el caso base (2x2)? Si
Realizamos el intercambio en el caso de la matriz base:
4 4
9 6
Cambiamos '9' por '4':
4 9
4 6
Hago una llamada al método DyV con la cuarta submatriz:
¿Es el caso base (2x2)? Si
Realizamos el intercambio en el caso de la matriz base:
4 9
6 6
Cambiamos '6' por '9':
4 6
9 6
Resultado tras llamar a DyV con cada submatriz:
3 5 9 3
4 2 2 4
4 9 4 6
4 6 9 6
Intercambio la segunda submatriz con la tercera submatriz:
3 5 4 9
4 2 4 6
9 3 4 6
2 4 9 6
chemi@LAPTOP-VF95F98N /cygdrive/c/Users/chemi/Desktop/Algoritmica-master/P2-DyV/Codigo
$
```

2.2 Análisis teórico

2.2.1 Fuerza Bruta

El enfoque fuerza bruta de la matriz traspuesta es uno bastante directo y obvio: se hace uso de dos bucles `for` que intercambian las filas por las columnas de la matriz, para generar la traspuesta, esto supone, por lo tanto que si se tiene una matriz con dimensión n , este par de bucles anidados reemplazarán n^2 elementos, que son todos los de la matriz, por lo tanto la eficiencia de la versión fuerza bruta es $O(n^2)$.

2.2.2 Divide y Vencerás

Para la eficiencia teórica de la versión Divide y Vencerás puramente recursiva, comenzamos por el caso base del algoritmo: esto es cuando la matriz posee una dimensión de 2×2 , aquí lo que se realiza es un intercambio entre dos elementos, como es una operación básica, su eficiencia es $O(1)$.

Cuando no se ha llegado al caso base, se divide la matriz en 4 partes con dimensión $\frac{n}{2}$ y se realizan 4 llamadas recursivas respectivamente. El tiempo que se tarda en la división del problema es constante ya que nuestra implementación utiliza la misma matriz para la recursividad pero lo que se modifica son los índices de inicio y fin de las columnas y filas, por lo que en cada llamada se trabaja en un cuadrante diferente.

Para combinar las soluciones, se tienen dos bucles `for` anidados que realizan el intercambio entre los dos cuadrantes que se encuentran por debajo y por encima de la diagonal principal, por lo tanto, realiza $\frac{n^2}{2}$ operaciones, ya que recorre un cuadrante en su totalidad, por lo que su eficiencia es entonces $O(n^2)$.

Juntando todo lo anterior, obtenemos la siguiente ecuación de recurrencia.

$$T(n) = \begin{cases} T(1), & \text{si } n = 2 \\ 4T(\frac{n}{2}) + n^2, & \text{si } n > 2 \end{cases}$$

Partiendo de esta ecuación de recurrencia, obtendremos la ecuación característica; primero tomamos la parte que nos interesa.

$$T(n) = 4T(\frac{n}{2}) + n^2$$

Se realiza un cambio de variable $n = 2^k$

$$\begin{aligned} T(2^k) &= 4T(2^{k-1}) + (2^k)^2 \\ T(2^k) - 4T(2^{k-1}) &= 4^k \end{aligned}$$

Ahora por comodidad, reescribimos $T(2^k)$ por $T(k)$.

$$T(k) - 4T(k-1) = 4^k$$

Y obtenemos la ecuación característica:

$$\begin{aligned} (x-4)(x-4) &= 0 \implies (x-4)^2 = 0 \\ T(k) &= C_1 4^k + C_2 k 4^k \end{aligned}$$

Devolvemos el cambio de variable, ahora $k = \log_2(n)$

$$\begin{aligned} T(n) &= C_1 4^{\log_2(n)} + C_2 4^{\log_2(n)} \log_2(n) \\ T(n) &= C_1 n^2 + C_2 n^2 \log_2(n) \end{aligned}$$

Por lo tanto, la versión Divide y Vencerás posee una eficiencia de $O(n^2 \log_2(n))$.

2.3 Análisis empírico

Como hemos visto en el análisis teórico calculado, la eficiencia del problema de la matriz traspuesta será $O(n^2)$ para el enfoque fuerza bruta y $O(n^2 \log_2(n))$ para el enfoque divide y vencerás. A continuación observaremos si dichos cálculos pueden corresponder a los datos obtenidos de forma empírica. Cabe destacar que no se pudieron realizar casos mayores a 2^{14} ya que el uso de memoria RAM sobrepasaba la capacidad del ordenador en ambas versiones.

2.3.1 Fuerza Bruta

Tabla de datos

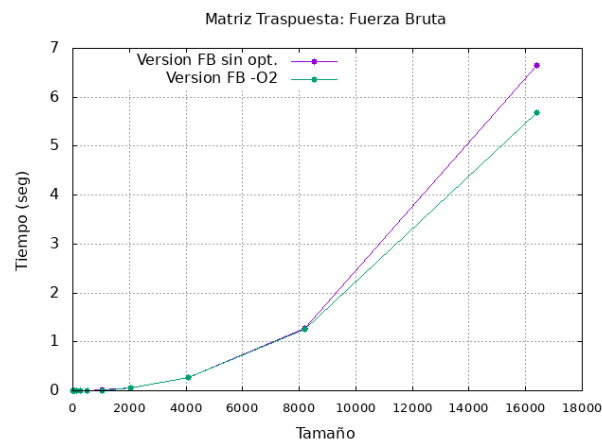
Hemos creado esta tabla de datos a partir del programa fuerza bruta sin optimización y con optimización (-O2) para ello hemos medido el tiempo de ejecución del programa en 14 casos aumentando el tamaño del problema de la forma 2^x empezando con $x = 1$ y acabando con $x = 14$, sumándole 1 a x en cada nueva muestra:

Tamaño	Tiempo sin opt. (seg)	Tiempo -O2 (seg)
2	0.000020	0.000002
4	0.000005	0.000004
8	0.000002	0.000002
16	0.000003	0.000002
32	0.000008	0.000005
64	0.000026	0.000015
128	0.000095	0.000046
256	0.000386	0.000199
512	0.001809	0.000964
1024	0.011081	0.005407
2048	0.056303	0.052396
4096	0.277671	0.264035
8192	1.276680	1.256180
16384	6.650840	5.676110

A simple vista podemos observar como a medida que el tamaño del problema aumenta el tiempo de esta aumenta a su vez. Además, podemos considerar que la diferencia entre el tiempo de ejecución sin optimización y con optimización es casi un segundo para tamaños "grandes" por lo que no es una gran optimización.

Representación de datos

Gracias a las muestras obtenidas anteriormente podemos graficar el tiempo de ejecución de la siguiente manera:



Según el resultado obtenido en el cálculo teórico de la eficiencia la gráfica debería mostrar un comportamiento de orden cuadrático; y como podemos ver así es.

2.3.2 Divide y Vencerás

Tabla de datos

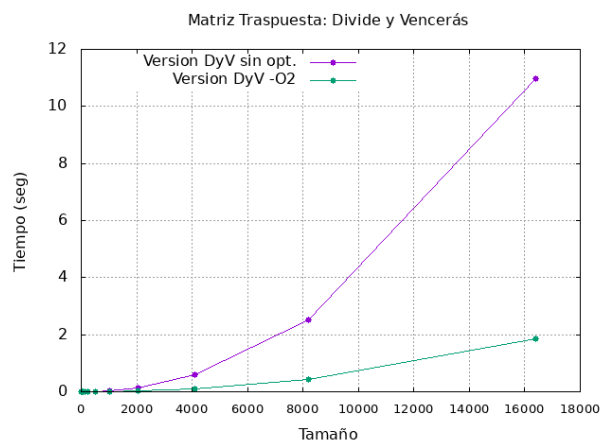
Creamos una nueva tabla para el enfoque divide y vencerás del problema con los mismo tamaños del problema que para fuerza bruta y estos fueron los resultados:

Como era de esperar, a medida que el tamaño del problema aumenta también lo hace a su vez el tiempo de ejecución; sin embargo, en el enfoque divide y vencerás, la optimización generada por el compilador ha mejorado significativamente el tiempo de ejecución siendo casi 10 veces mejor el programa compilado con optimización.

Tamaño	Tiempo sin opt. (seg)	Tiempo -O2 (seg)
2	0.000002	0.000001
4	0.000002	0.000001
8	0.000002	0.000002
16	0.000005	0.000003
32	0.000018	0.000005
64	0.000076	0.000022
128	0.000340	0.000060
256	0.001583	0.000258
512	0.007259	0.001097
1024	0.033319	0.008318
2048	0.136567	0.040976
4096	0.601853	0.098468
8192	2.535520	0.417563
16384	10.956500	1.858030

Representación de datos

Graficamos los datos obtenidos en el apartado anterior tal que:

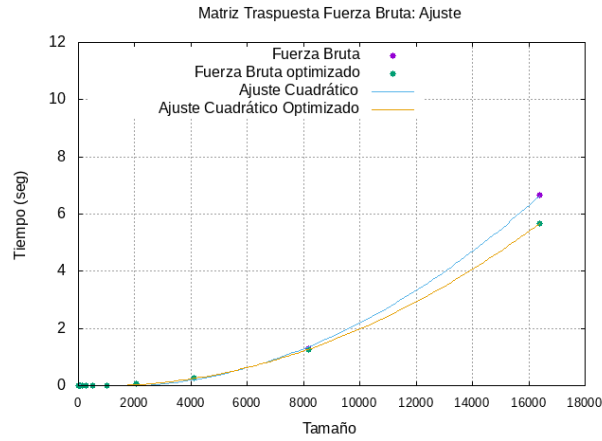


En comparación con la gráfica anterior del enfoque de fuerza bruta podemos ver que el tiempo de ejecución sin optimización tiene una curva bastante parecida; en ambos casos podemos decir que estamos hablando de funciones cuadráticas. El tiempo de ejecución sin embargo muestra una curva mucho más aplanada. Ambos tiempos deberían de estar acotados por $n^2 \log_2(n)$ suponiendo que el análisis teórico es el correcto.

2.4 Análisis híbrido

2.4.1 Fuerza Bruta

El algoritmo que realiza la traspuesta de la manera más directa se puede notar que describe una curva cuadrática, esto además queda plasmado en el análisis teórico, por lo tanto se realizará un ajuste cuadrático.



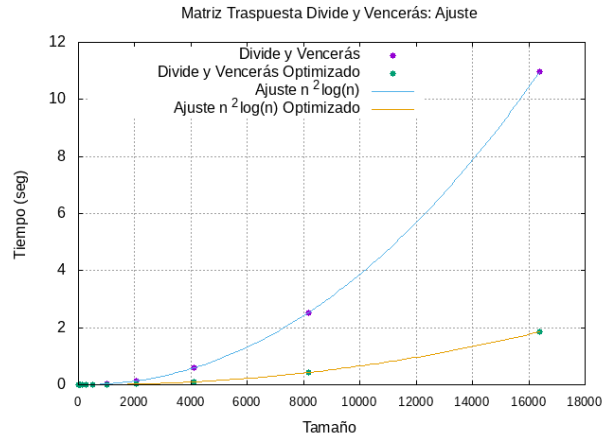
$$f(x) = ax^2 + bx + c$$

	Fuerza Bruta		Fuerza Bruta Optimizado	
	Valor	Error	Valor	Error
a	2.91512×10^{-8}	2.108%	2.32635×10^{-8}	0.6732%
b	-7.38925×10^{-8}	12.91%	-3.53478×10^{-7}	6.875%
c	0.0187478	78.3%	0.00623189	60.02%

Como podemos ver, efectivamente el algoritmo es de orden $O(n^2)$ y se puede también notar que la optimización 02 no ha hecho mucho efecto en la mejora del tiempo.

2.4.2 Divide y Vencerás

En vista de lo obtenido en el análisis teórico así como las gráficas obtenidas del análisis empírico, se realizó el ajuste a los datos con la función obtenida de manera teórica, que se le podría llamar cuadrática-logarítmica.



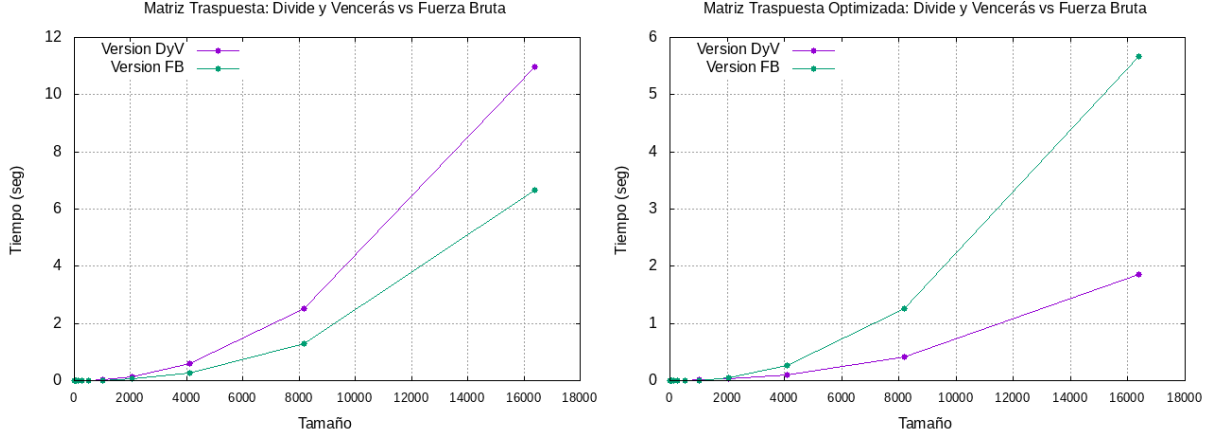
$$f(x) = ax^2 \log(x) + b$$

	Divide y Vencerás		Divide y Vencerás Optimizado	
	Valor	Error	Valor	Error
a	4.20521×10^{-9}	0.04571%	7.11795×10^{-10}	0.3647%
b	0.00108531	126.8%	0.000817628	227.2%

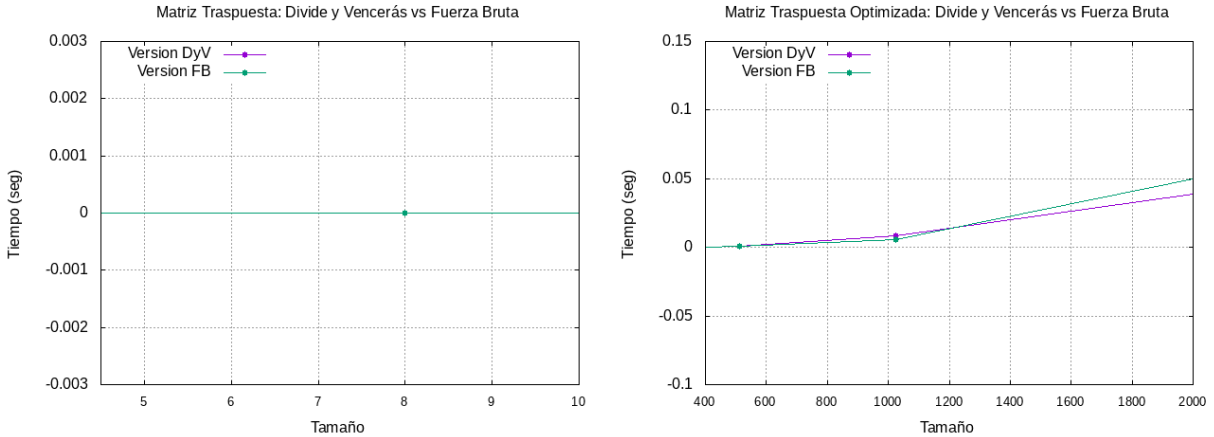
El ajuste es muy bueno para ambas versiones del algoritmo, resulta interesante que la versión con la optimización 02 rebaja de una gran manera las constantes ocultas, a tal nivel que logra ser más eficiente empíricamente que fuerza bruta, esto se puede deber a que el algoritmo de Divide y Vencerás puede obtener optimizaciones en el uso de la memoria al realizar llamadas recursivas que van decrementando en tamaño.

2.5 Obtención del umbral

Para la obtención del umbral, en primer lugar, la versión sin optimizar no tiene mucho sentido hablar del umbral ya que la función Divide y Vencerás posee una eficiencia peor tanto teórica como empírica, aún así se puede obtener el punto en el que empieza a ser más lento que fuerza bruta, pero, en cambio, en la versión optimizada si tiene sentido que se le obtenga un umbral, ya que la optimización logra rebajar considerablemente las constantes ocultas y por lo tanto, sus ejecuciones son más rápidas que la versión fuerza bruta.



Ahora bien, el umbral se encuentra entre los primeros valores, por lo tanto, se necesita ver más de cerca.



La versión original empieza a diferir a tan solo $n = 8$, que además es un valor tan ínfimo que apenas y se logra ver en la gráfica. En este caso ese sería el umbral para la versión sin optimizar de Divide y Vencerás en el que se debe cambiar a la versión Fuerza Bruta.

Pero si se trata de la versión optimizada, entonces, de manera inversa se tiene que para valores menores que $n_0 = 1024$ es mejor utilizar únicamente la versión fuerza bruta.

Por otro lado si se realiza por la forma teórica el umbral, esto es, igualando las funciones de las versiones optimizadas se tiene que:

$$ax^2 \log(x) + b = cx^2 + dx + e$$

Siendo a, b, c, d y e los valores obtenidos por la eficiencia híbrida

$$7.11795 \times 10^{-10} x^2 \log(x) + 0.000817628 = 2.32635 \times 10^{-8} x^2 + -3.53478 \times 10^{-7} x + 0.00623189$$

Se obtiene el valor de x que nos interesa, utilizándose el método de bisección para obtener el punto en que se tocan las funciones:

$$x = 1804.44$$

El cual es relativamente cercano al que se obtuvo empíricamente, se encuentra entre los dos posibles valores que puede tomar el algoritmo, por lo tanto podemos concluir que $n_0 = 1024$ para la versión optimizada y $n_0 = 8$ para la versión original.

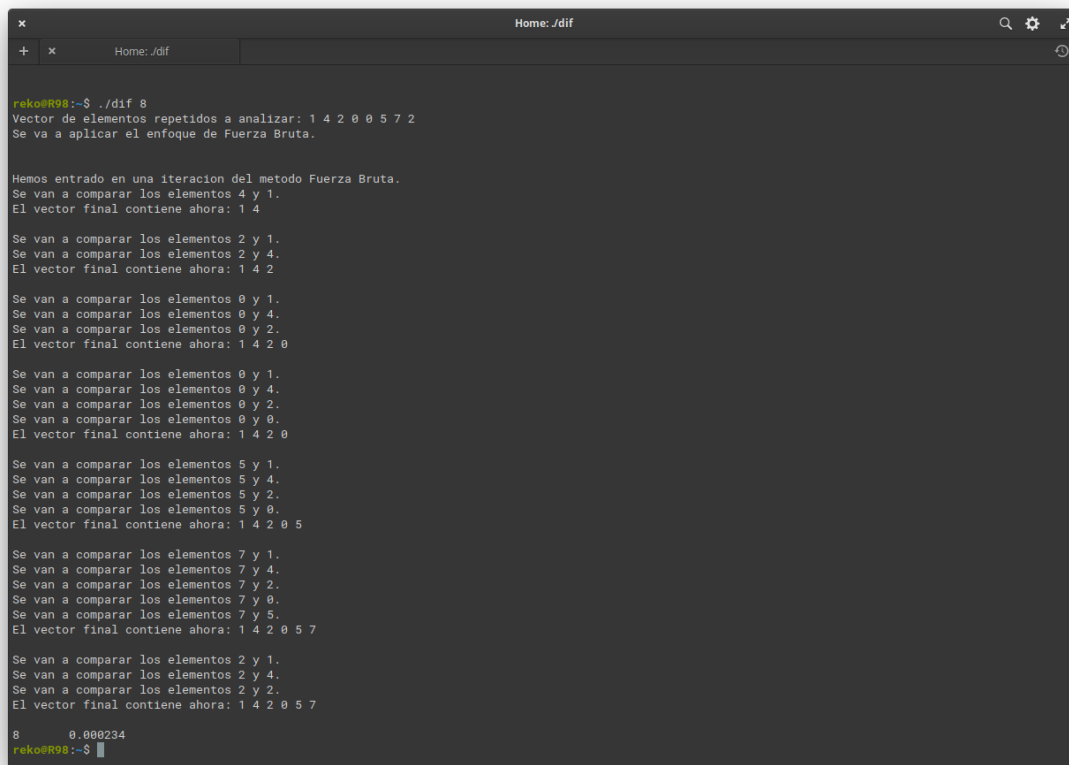
3 Problema asignado: Eliminar elementos repetidos en vector

3.1 Descripción de caso de ejecución

Para poder tener una idea básica sobre el algoritmo, vamos a presentar una pequeña explicación sobre el funcionamiento del algoritmo tanto en el enfoque de fuerza bruta, como en el de divide y vencerás. No obstante podemos explicar la función principal del programa de forma conjunta, pues ambos casos de Eliminar elementos tienen el mismo objetivo: Dado un vector de n elementos, obtendremos otro vector en el cuál se mantendrán únicamente los valores no repetidos de cada elemento, reduciendo así el tamaño del vector original. La única diferencia notable entre ambos casos es que en Fuerza bruta hemos supuesto que el orden de los valores no tiene importancia y por tanto el vector resultante obtenido está desordenado. Esto no pasa en divide y vencerás pues para en este caso se ordena el vector de forma ascendente.

3.1.1 Fuerza bruta

En primer lugar, explicaremos la ejecución de Elementos Repetidos con el enfoque de fuerza bruta. Para esto, nos ayudaremos de una captura de pantalla.



```
rekodR98@:~$ ./dif 8
Vector de elementos repetidos a analizar: 1 4 2 0 0 5 7 2
Se va a aplicar el enfoque de Fuerza Bruta.

Hemos entrado en una iteración del metodo Fuerza Bruta.
Se van a comparar los elementos 4 y 1.
El vector final contiene ahora: 1 4

Se van a comparar los elementos 2 y 1.
Se van a comparar los elementos 2 y 4.
El vector final contiene ahora: 1 4 2

Se van a comparar los elementos 0 y 1.
Se van a comparar los elementos 0 y 4.
Se van a comparar los elementos 0 y 2.
El vector final contiene ahora: 1 4 2 0

Se van a comparar los elementos 0 y 1.
Se van a comparar los elementos 0 y 4.
Se van a comparar los elementos 0 y 2.
Se van a comparar los elementos 0 y 0.
El vector final contiene ahora: 1 4 2 0

Se van a comparar los elementos 5 y 1.
Se van a comparar los elementos 5 y 4.
Se van a comparar los elementos 5 y 2.
Se van a comparar los elementos 5 y 0.
El vector final contiene ahora: 1 4 2 0 5

Se van a comparar los elementos 7 y 1.
Se van a comparar los elementos 7 y 4.
Se van a comparar los elementos 7 y 2.
Se van a comparar los elementos 7 y 0.
Se van a comparar los elementos 7 y 5.
El vector final contiene ahora: 1 4 2 0 5 7

Se van a comparar los elementos 2 y 1.
Se van a comparar los elementos 2 y 4.
Se van a comparar los elementos 2 y 2.
El vector final contiene ahora: 1 4 2 0 5 7

8      0.000234
rekodR98@:~$
```

Como podemos ver, el tamaño del vector escogido en esta ejecución es 8. Los elementos que rellenan este vector se seleccionan aleatoriamente mediante un método auxiliar proporcionado para facilitar el trabajo. En este caso, los elementos repetidos que podemos apreciar en el vector son el 2 y el 0. ¿Cómo funciona entonces el método en fuerza bruta? Su forma de actuar es bastante trivial, en primer lugar, en un vector auxiliar se va a añadir el primer elemento del vector original y después se van a recorrer mediante un bucle for los elementos del vector original y se irán comparando con aquellos que ya hemos incluido en el vector auxiliar.

Durante esta comparación se pueden dar dos casos:

El primer caso es que al comparar los elementos, estos sean iguales y por tanto hemos encontrado un valor repetido. Si nos encontramos ante esta situación, omitiremos el valor analizado del vector original y pasaremos a analizar el siguiente. Este caso se puede observar en la cuarta comparación de la cuarta iteración, cuando analizamos los

elementos 0 y 0.

El siguiente caso que se puede dar es que los elementos analizados no sean iguales. Si esto sucede y ya hemos realizado la comparación con todos los elementos del vector auxiliar, podemos asumir que nos hemos encontrado con un elemento único y por tanto, lo añadimos al vector auxiliar, de forma que en las comparaciones siguientes tendremos un nuevo valor para analizar. Esta acción se puede ver reflejada en la primera comparación que realizamos.

3.1.2 Divide y Vencerás

Una vez explicado el enfoque de fuerza bruta, pasaremos a explicar Divide y Vencerás. Y, como en el caso anterior, también nos ayudaremos de una imagen para este proceso.

```
rekodR98@:~$ ./div 4
Vector de elementos repetidos a analizar: 1 2 3 2
Se va a aplicar el enfoque de divide y venceras.

Hemos entrado en una nueva iteracion del metodo divide y venceras.
El tamaño del vector es: 4
Se han asignado en dos vectores la parte derecha e izquierda del vector original.
El primer vector contiene: 1 2
El segundo vector contiene: 3 2
Aplicamos de forma recursiva divide y venceras sobre los nuevos vectores creados.

Hemos entrado en una nueva iteracion del metodo divide y venceras.
El tamaño del vector es: 2
Se han asignado en dos vectores la parte derecha e izquierda del vector original.
El primer vector contiene: 1
El segundo vector contiene: 2
Aplicamos de forma recursiva divide y venceras sobre los nuevos vectores creados.

Hemos entrado en una nueva iteracion del metodo divide y venceras.

Hemos entrado en una nueva iteracion del metodo divide y venceras.
Ahora vamos a unir los vectores separados con el metodo joinVector
Los elementos a analizar son 1 y 2.

Hemos entrado en una nueva iteracion del metodo divide y venceras.
El tamaño del vector es: 2
Se han asignado en dos vectores la parte derecha e izquierda del vector original.
El primer vector contiene: 3
El segundo vector contiene: 2
Aplicamos de forma recursiva divide y venceras sobre los nuevos vectores creados.

Hemos entrado en una nueva iteracion del metodo divide y venceras.

Hemos entrado en una nueva iteracion del metodo divide y venceras.
Ahora vamos a unir los vectores separados con el metodo joinVector
Los elementos a analizar son 3 y 2.
Ahora vamos a unir los vectores separados con el metodo joinVector
Los elementos a analizar son 1 y 2.
Los elementos a analizar son 2 y 2.
4      6.2e-05
rekodR98@:~$
```

Para su análisis y no alargar demasiado el proceso, en este caso hemos seleccionado un tamaño de vector menor que en el caso anterior, 4. Podemos ver que el único elemento que se repite es 2.

¿Cómo procede este algoritmo?

Este método va dividiendo el vector en 2 hasta dejar solo subvectores compuestos por un único elemento. Una vez llegado a este punto, vamos a analizar los valores de dos en dos, pudiéndose dar tres posibles casos:

- El elemento de la izquierda es menor que el de la derecha: En este caso y como estamos ordenando el vector con un orden ascendente, el elemento de la izquierda se pone antes que el de la derecha.
- El elemento de la izquierda es mayor que el de la derecha: Ahora, como el vector resultante debe tener la misma propiedad con respecto al orden, colocamos en el vector final primero los elementos del vector de la derecha.
- Ambos elementos sean iguales: Como ambos elementos son iguales, en este caso no importa cual de ellos coloquemos en el vector final. No obstante hay que destacar que, como el objetivo principal es eliminar elementos repetidos, SÓLO debemos colocar uno de los elementos, descartando el otro.

Este proceso se va repitiendo hasta que hemos unido todos los elementos que hemos separado previamente y eliminado los que se repiten, obteniendo como resultado un vector reducido y completamente ordenado.

3.2 Análisis teórico

3.2.1 Fuerza Bruta

Realizaremos ahora un análisis teórico del caso Elementos Repetidos con el enfoque de fuerza bruta. Observando el código utilizado en la función principal que se encarga de realizar todo el proceso de selección y omisión de elementos repetidos, podemos ver que hay varias sentencias declarativas, de asignación, de lectura y de escritura, que ocupan un tiempo $O(1)$. No obstante, podemos ver que hay dos bloques destacables, un bucle **for** y dentro del mismo, existe un bucle **while** anidado. Ambos tienen un tiempo $O(n)$ cada uno. Al estar anidados, ambos tiempos se multiplicarán formando un tiempo total de $O(n^2)$. Por lo que podemos afirmar que el orden de eficiencia de este enfoque es cuadrático.

3.2.2 Divide y Vencerás

Para el análisis teórico del enfoque divide y vencerás, debemos tener en cuenta de que dicho enfoque consiste en ir dividiendo los elementos del objeto que se nos da, de forma que en su última instancia, solo hay que dos vectores con un único elemento cada vez.

En el caso base, si nos encontramos ante un vector con $n=1$, únicamente realizamos una llamada al mismo elemento, una operación básica con orden de eficiencia $O(1)$.

Si no nos encontramos ante el caso base, debemos dividir el problema en dos subproblemas con tamaño $n/2$, el tiempo realizado para la división de las funciones tiene orden lineal $O(n)$ y el tiempo empleado para unir los elementos también podemos tener un orden de eficiencia $O(n)$. De esta forma obtenemos la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} T(1) & \text{si } n = 1. \\ 2T(\frac{n}{2}) + n & \text{si } n > 1. \end{cases}$$

Vamos a intentar demostrar que $T(n) = O(n \log n)$ partiendo de la ecuación de recurrencia obtenida, nos centraremos para esto en el segundo caso en el que tenemos $2T(\frac{n}{2}) + n$

Realizaremos un cambio de variable sobre el mismo, $n = 2^k$:

$$\begin{aligned} T(2^k) &= 2T(2^{k-1}) + 2^k \\ T(2^k) - 2T(2^{k-1}) &= 2^k \end{aligned}$$

Ahora haciendo uso de una de las propiedades de las recurrencias no homogéneas obtenemos la siguiente igualdad:

$$\begin{aligned} (x-2)(x-2) &= 0 \implies (x-2)^2 = 0 \\ T(k) &= C_1 2^k + C_2 k 2^k \end{aligned}$$

Deshacemos el cambio de variable, ahora $k = \log_2(n)$

$$\begin{aligned} T(n) &= (C_1) 2^{\log_2(n)} + (C_2) \log_2(n) 2^{\log_2(n)} \\ T(n) &= (C_1)n + (C_2)n \log_2(n) \end{aligned}$$

Como queda reflejado, hemos demostrado que el algoritmo posee un orden de $O(n \log_2(n))$

3.3 Análisis empírico

Como se ha visto en el análisis teórico anteriormente citado, la eficiencia del problema de la eliminación de elementos repetidos será $O(n^2)$ para la versión de fuerza bruta y $O(n \log(n))$ para la versión de divide y vencerás. Vamos a observar los cálculos empíricos con más detalle a continuación para comprobar si corresponden con los resultados obtenidos en el apartado anterior.

3.3.1 Fuerza Bruta

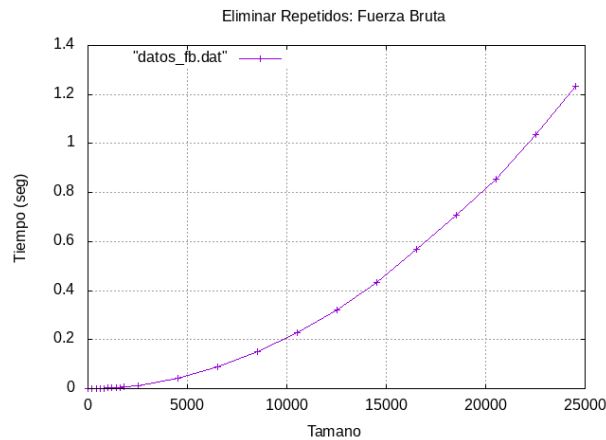
Tabla de datos

Creamos una tabla con 22 tamaños y el tiempo que tarda en ejecutarse el algoritmo para cada uno:

Tamaño	Tiempo (seg)
10	4e-06
210	9.7e-05
410	0.000371
610	0.000819
810	0.001419
1010	0.002179
1210	0.003112
1410	0.004173
1610	0.005509
1810	0.006831
2500	0.013145
4500	0.041497
6500	0.087849
8500	0.149674
10500	0.228615
12500	0.321288
14500	0.434299
16500	0.566833
18500	0.706975
20500	0.855647
22500	1.03649
24500	1.2326

Representación de datos

La gráfica obtenida de los datos mostrados arriba es la siguiente:



Comprobamos que efectivamente, la gráfica tiene tendencia cuadrática, tal y como hemos concluido en el análisis teórico.

3.3.2 Divide y Vencerás

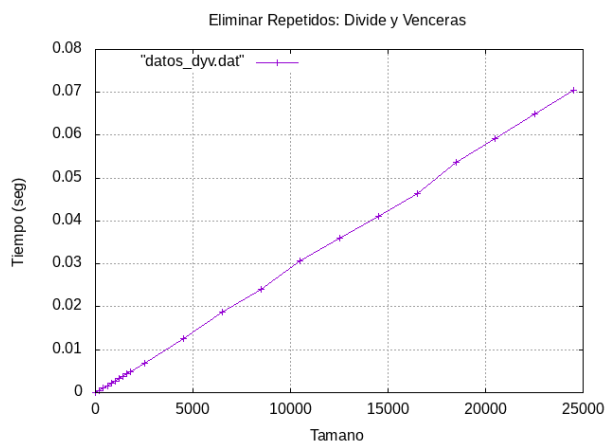
Tabla de datos

Creamos una nueva tabla para el enfoque divide y vencerás del problema con los mismo tamaños del problema que para fuerza bruta y estos fueron los resultados:

Tamaño	Tiempo (seg)
10	3e-05
210	0.000546
410	0.00107
610	0.001635
810	0.00216
1010	0.002641
1210	0.003283
1410	0.003849
1610	0.004395
1810	0.004925
2500	0.006949
4500	0.012543
6500	0.018707
8500	0.024141
10500	0.030779
12500	0.036108
14500	0.041089
16500	0.046361
18500	0.053661
20500	0.059133
22500	0.064877
24500	0.070598

Representación de datos

La gráfica obtenida de los datos mostrados arriba es la siguiente:



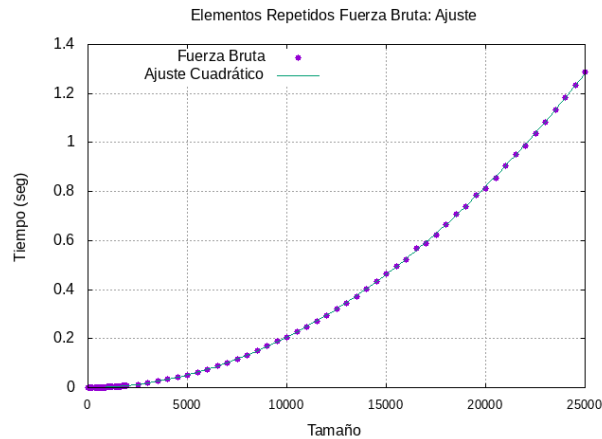
Podemos comprobar que el enfoque divide y vencerás es más eficiente ya que la gráfica obtenida se asemeja a una gráfica lineal, lo cual es coherente con lo obtenido teóricamente.

3.4 Análisis híbrido

A continuación, realizaremos un análisis híbrido de ambos enfoques de la misma forma en la que se hizo con el caso de la matriz traspuesta anteriormente.

3.4.1 Fuerza Bruta

En el caso de fuerza bruta, como ya hemos dicho antes, hemos usado un algoritmo con orden de eficiencia cuadrático, esto significa que si realizamos el ajuste del mismo con una ecuación de segundo grado, la trayectoria de ambas gráficas generadas debería ser similar. Vamos a comprobar dicha situación con el programa por terminal gnuplot. De esta forma, obtenemos la siguiente representación gráfica.



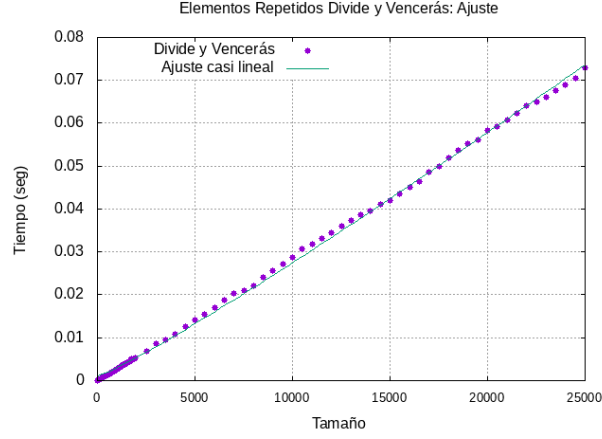
$$f(x) = ax^2 + bx + c$$

Fuerza Bruta		
	Valor	Error
a	2.04466×10^{-9}	0.2571%
b	1.31747×10^{-7}	89.07%
c	6.5883×10^{-5}	624.3%

Como podemos ver en la imagen, la ecuación de segundo grado utilizada (en este caso: $f(x) = ax^2 + bx + c$) se ajusta muy bien a los datos que hemos obtenido durante la prueba y ejecución de nuestro programa. De hecho, podemos ver que apenas presenta irregularidades. Los datos obtenidos para las variables junto con sus respectivos porcentajes de error se pueden ver en la tabla implementada justo después de la gráfica. Podemos afirmar por tanto de que, ciertamente, estamos ante un algoritmo de orden cuadrático.

3.4.2 Divide y Vencerás

Para Divide y vencerás vamos a realizar el ajuste con la ecuación $f(x) = ax \log(x) + b$. Deberíamos ser capaces de observar que ambas trayectorias son lineales, pues la orden de eficiencia de este algoritmo es casi lineal. De esta forma, contrastando ambas gráficas obtenidas en una sola obtenemos la siguiente imagen.

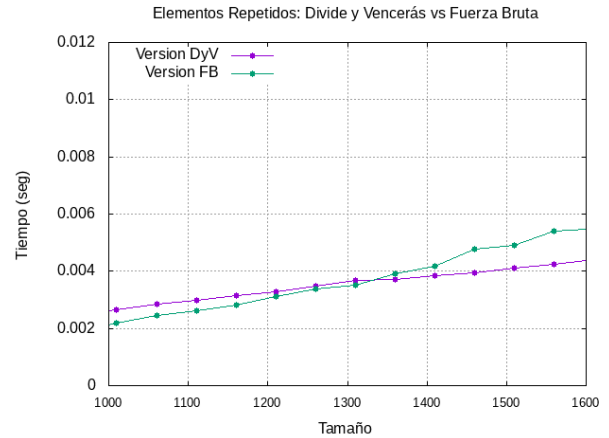
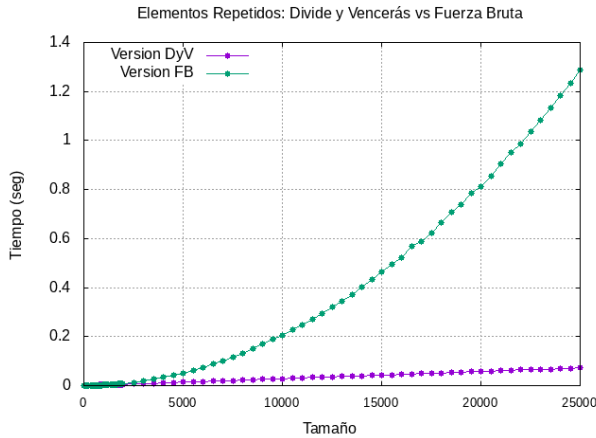


	$f(x) = ax \log(x) + b$	
	Divide y Vencerás	
	Valor	Error
a	2.86827×10^{-7}	0.3423%
b	0.00104065	10.35%

En este caso podemos ver que aunque se presenten ciertas irregularidades poco notables también tenemos trayectorias superpuestas en esta gráfica. Podemos afirmar entonces, que se trata de un algoritmo con orden de eficiencia casi lineal. Además en la tabla que se encuentra justo debajo de la gráfica podemos ver los valores obtenidos para cada variable y sus respectivos porcentajes de error.

3.5 Obtención del umbral

Para la obtención del umbral, igual que como se hizo en la matriz traspuesta, primero se realizaron las ejecuciones de los algoritmos por separado y se analizó la gráfica para detectar el momento en que las curvas se tocan.



Como se puede observar, el punto de corte se encuentra en el rango de 1300 a 1350, cabe notar que los valores muy similares entre sí, para afinar mejor el rango, se utilizó el método teórico, si igualamos las funciones entonces

$$ax^2 + bx + c = d \log(x) + e$$

Siendo a, b, c, d y e los valores obtenidos por la eficiencia híbrida

$$2.04466 \times 10^{-9}x^2 + 1.31747 \times 10^{-7}x + 6.5883 \times 10^{-5} = 2.86827 \times 10^{-7} \log(x) + 0.00104065$$

Se obtiene el valor de x que nos interesa, utilizándose el método de bisección para obtener el punto en que se tocan las funciones:

$$x = 1306.92$$

Que es muy cercano a lo obtenido experimentalmente, por lo tanto podríamos promediar el umbral a $n_0 = 1310$.

4 Conclusión

En conclusión sacamos que cualquier algoritmo tiene diversas formas de ser planteado, y este planteamiento es decisivo a la hora de querer ejecutar distintos tamaños de problema en un algoritmo específico. En nuestro caso el planteamiento de forma divide y vencerás ha sido el más óptimo en lo que al tiempo de ejecución se refiere, aunque para que eso sea así hemos usado la ayuda de la optimización del compilador. Divide y vencerás no es aplicable a todo tipo de algoritmos, solo aquellos en los cuales podemos dividir el problema en otros del mismo tipo, pero, si es posible, los hechos muestran que es la manera más eficiente de resolverlos.