PRÁCTICA 2:

ALGORITMOS DIVIDE Y VENCERÁS
Serie Unimodal de Números

Algorítmica
2016-2017

Componentes del Grupo:
Daniel Bolaños Martínez
José María Borrás Serrano
Santiago De Diego De Diego
Fernando De la Hoz Moreno

Índice:

Introducción y Presentación del problema:

Sea un vector v de números de tamaño n, todos distintos, de forma que existe un índice p (que no es ni el primero ni el último) tal que a la izquierda de p los números están ordenados de forma creciente y a la derecha de p están ordenados de forma decreciente.

Hemos diseñado un algoritmo basado en “Divide y Vencerás” el cual tiene como objetivo encontrar el valor máximo de una serie unimodal. El orden de eficiencia de este algoritmo es O(log(n)) y lo hemos comparado con el algoritmo trivial para este problema que es de orden O(n).

Para la comparación hemos obtenido unas tablas en las que se muestran el tiempo de ejecución según distintos número de elementos en los vectores, hemos representado los datos en una gráfica y hemos ajustado estos datos a la función obtenida por la eficiencia teórica por el ajuste de mínimos cuadrados.

Ejercicio 1:
Algoritmo Divide y Vencerás para la resolución del problema:

Función Algoritmo unimodal Divide y Vencerás:

```
int unimodal(vector<int> v)
{
        bool fin=false;
        int maximo=v.size()-1;
        int indice=maximo/2;
        int minimo=0;

        while(!fin)
        {
        if(v.at(indice-1)<v.at(indice))
           if(v.at(indice+1)<v.at(indice))
                           fin=true;
                   else
                   {
                               minimo=indice;
                               indice=indice+((maximo-indice)/2);
                       }
                else
                {
                        maximo=indice;
                        indice=minimo+((indice-minimo)/2);
                }
        }
        return indice;
}
```

El algoritmo consiste en tomar el elemento que se encuentra en mitad del vector y comprobar si es un máximo viendo si es mayor que el elemento de la izquierda y menor que el de la derecha. Si es así se ha terminado el algoritmo pues ya hemos encontrado el máximo. Si no es así vemos si el elemento esta en la zona creciente o decreciente del vector. En el caso de que este en la zona creciente el máximo se situara en la mitad de la derecha del vector y si se encuentra en la decreciente en la mitad izquierda. En este punto se vuelve a aplicar el algoritmo sobre la mitad del vector donde se encuentre el máximo y se repite el proceso hasta que se encuentre el máximo.

Como en cada iteración lo que se hace es dividir el vector por la mitad y buscar el máximo en una mitad el número máximo de iteraciones hasta encontrar el máximo es de log(n) siendo n el tamaño del vector. Como todo las comprobaciones realizadas en cada iteración son O(1) el algoritmo es O(log(n)).

Función Algoritmo unimodal Secuencial:

```
int unimodal_secuencial(vector<int> v)
{
        bool fin=false;
        int indice=1;

        while(!fin)
        {
        if(v.at(indice+1)<v.at(indice))
                fin=true;
        else
                indice++;
        }

        return indice;
}
```

En este caso lo único que se hace es recorrer el vector hasta ver que empieza a ser decreciente. En el peor caso puede empezar a ser decreciente en el penúltimo elemento por lo que habría que recorrer todo el vector, de manera que la eficiencia de este algoritmo es O(n).

Ejercicio 2:

Datos Algoritmos DyV y Secuencial:

Tamaño Vectores	Tiempo Divide y Vencerás
1048576	7.796e-05
2097152	0.00016308
4194304	0.00038871
8388608	0.00117717
16777216	0.00227126
33554432	0.00456919
67108864	0.00894183
134217728	0.0170173
268435456	0.0335588
536870912	0.0668834

Tamaño Vectores	Tiempo Secuencial
1000000	0.00169148
2000000	0.00341387
3000000	0.00515229
4000000	0.00688878
5000000	0.00583811
6000000	0.0102687
7000000	0.0119547
8000000	0.013579
9000000	0.0157071
10000000	0.017487
11000000	0.0192033
12000000	0.0209426
13000000	0.022794
14000000	0.0245116
15000000	0.0260875

16000000	0.0278383
17000000	0.0296462
18000000	0.0314487
19000000	0.033057
20000000	0.0348266
21000000	0.0367226
22000000	0.0383142
23000000	0.0401301
24000000	0.0418608
25000000	0.0434716
26000000	0.0455227

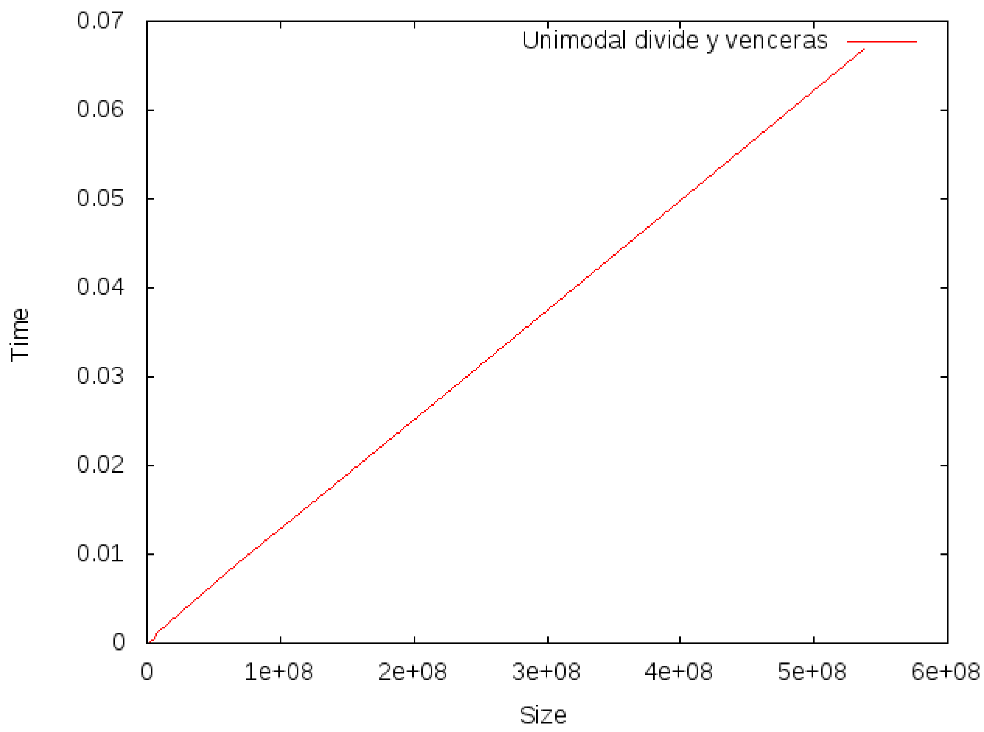
Como podemos observar los tiempos de divides y vencerás son mejores que los del secuencial.

A continuación podemos observar las gráficas que nos muestran los tiempos de ejecución en función del numero de elementos del vector.

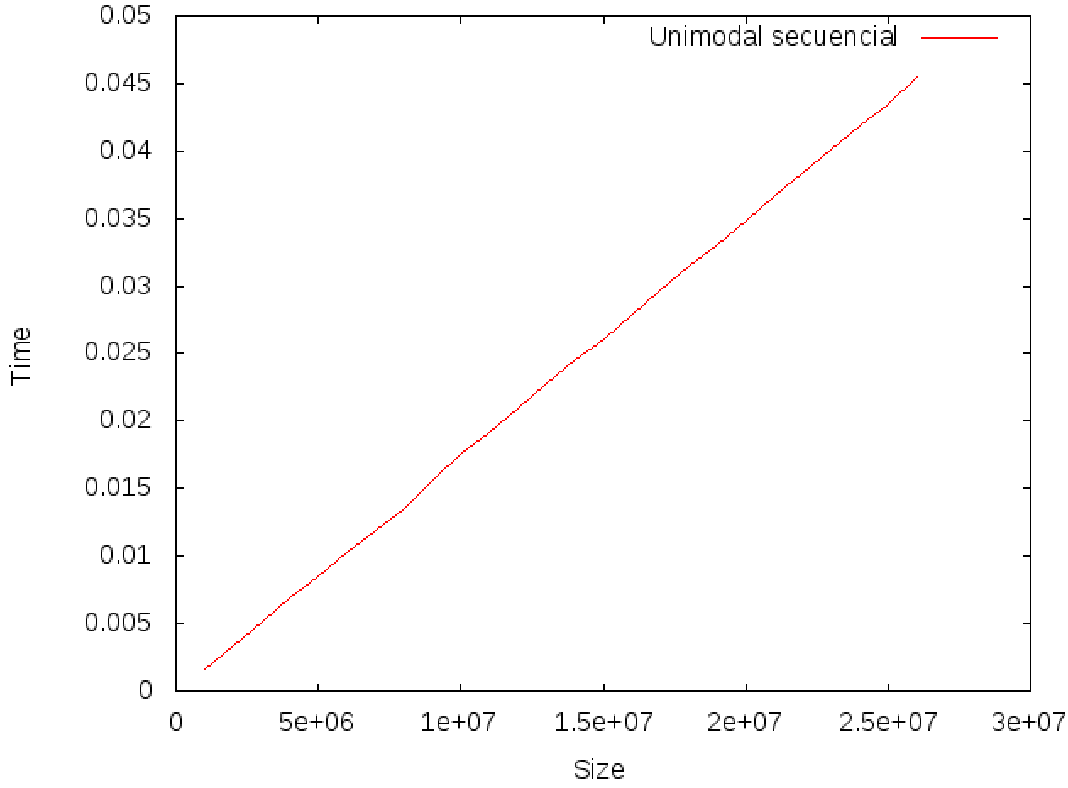
Ejercicio 3:

Eficiencia empírica (Gráficos):

Representación de los datos obtenidos por el algoritmo divide y vencerás para el problema planteado.

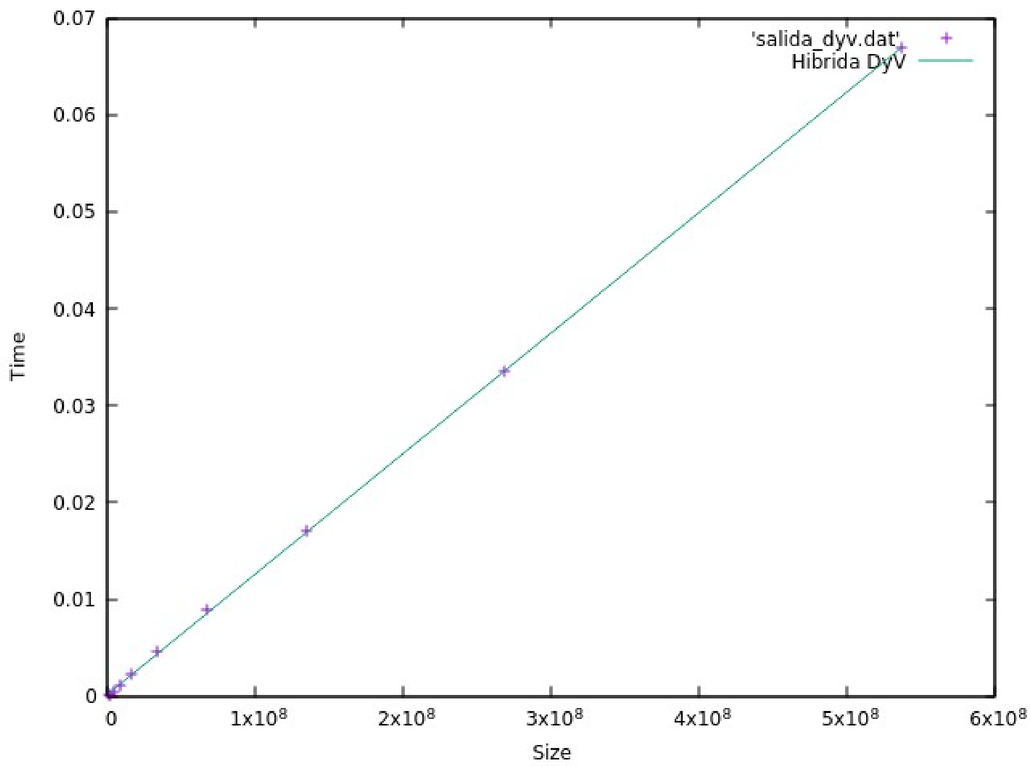


Representación de los datos obtenidos por el algoritmo secuencial para el problema planteado.

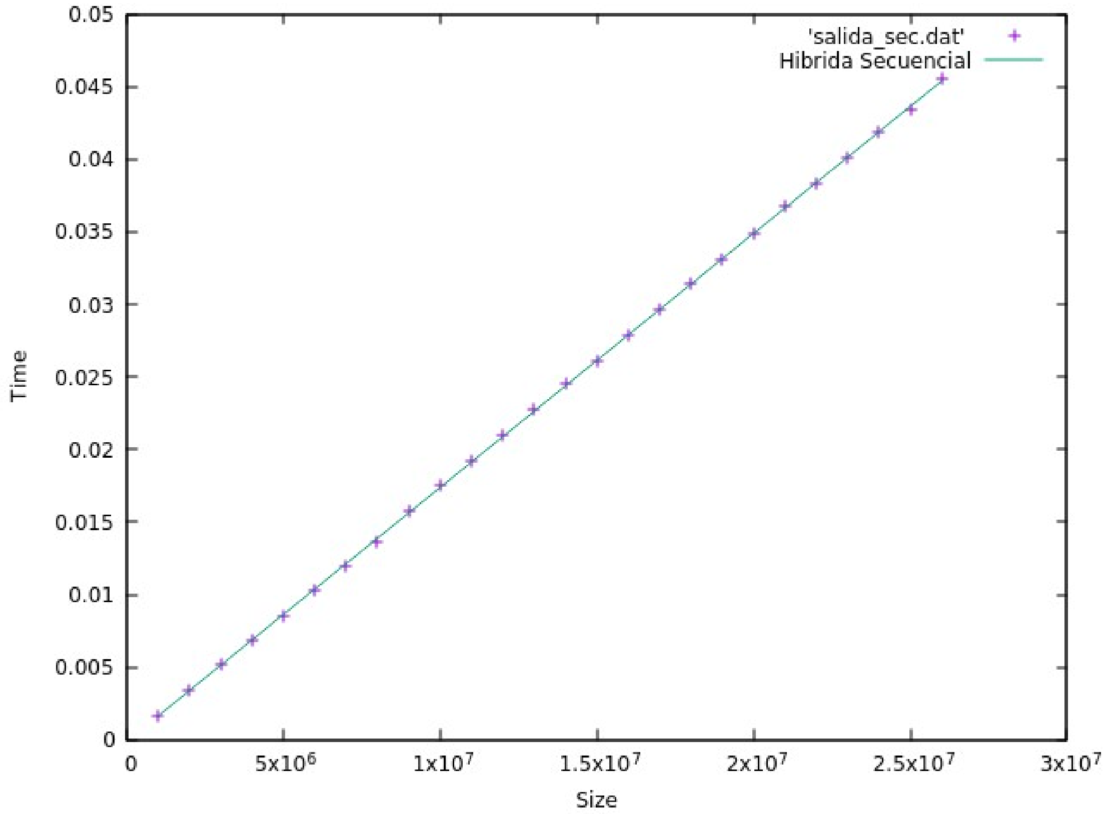


Ajustes híbridos:

A continuación, los ajustes de los datos a las expresiones obtenidas de la eficiencia teórica.



f(x)=a0*log(x)+a1*x+a2
a0=1.75072e-09
a1=1.24488e-10
a2= 0.000151147



f(x)=a0*x+a1
a0=1.75072e-09
a1=-0.000131396

<u>Correlación:</u>

- Unimodal secuencial:

Coeficiente de correlación en el caso lineal: 0,999967757
Coeficiente de correlación en el caso logarítmico: 0,999967634

- Unimodal Divide y Vencerás:

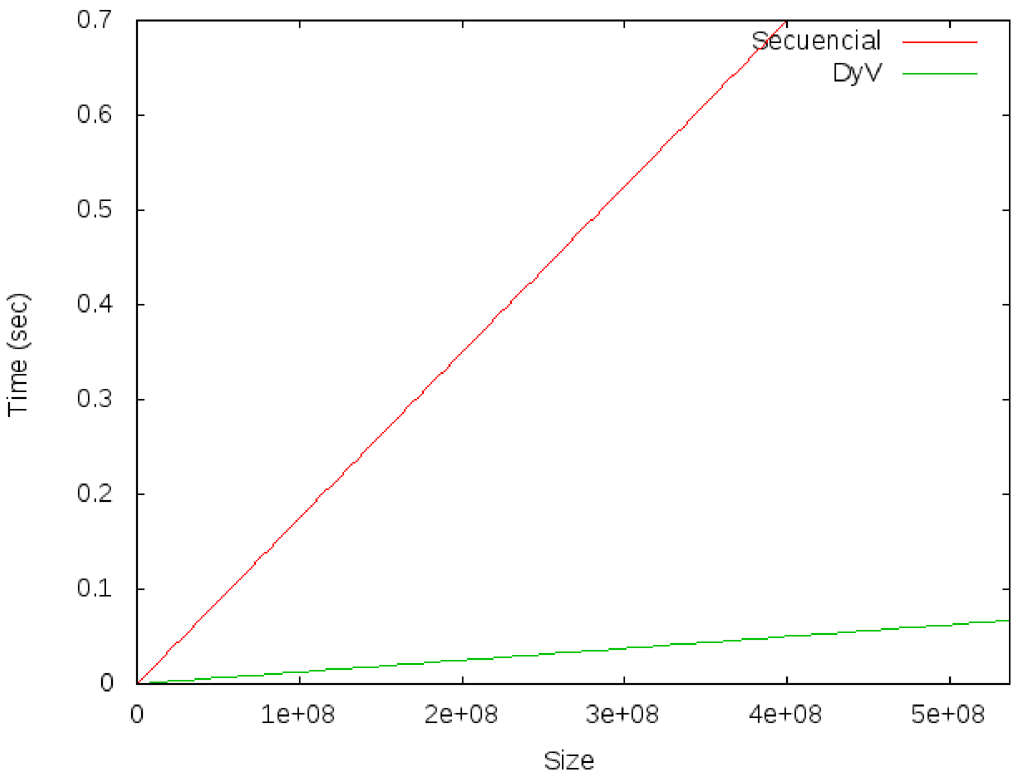
Coeficiente de correlación en el caso lineal: 0,993561274
Coeficiente de correlación en el caso logarítmico: 0,99566217

En el caso secuencial el ajuste lineal es mejor, mientras que en Divide y Vencerás el mejor ajuste es el logarítmico.

Umbral:

En algunos casos el algoritmo secuencial podría resultar más eficiente que el logarítmico, si tomásemos por ejemplo, tamaños pequeños para los vectores.

Por tanto sería interesante calcular el tamaño de los vectores a partir del cual es mejor usar una versión que otra y en el caso de que exista tal punto, añadir en el código que hasta ese valor, ejecutase el algoritmo que en cada caso fuese más eficiente.



La gráfica umbral obtenida representa los dos algoritmos muestra que para todos los valores, el algoritmo programado con la metodología de divide y vencerás siempre es más eficiente que el secuencial, por tanto obviaremos lo dicho anteriormente, puesto que siempre será mejor usar este programa para cualquier valor de vectores.

Conclusión:

Como podemos observar, el mismo problema se puede resolver de forma más rápida y eficiente si empleamos un algoritmo de tipo Divide y Vencerás que uno secuencial.

En este caso con Divide y Vencerás podemos conseguir que la eficiencia del algoritmo pase de ser O(n) a O(log n), por lo que somos capaces de procesar muchos más datos en un tiempo menor.

De esta forma se puede concluir que siempre será más eficiente resolver el problema utilizando nuestro algoritmo DV que por el método secuencial.