Práctica 2: Algoritmos Divide y Vencerás

Daniel Bolaños Martínez, José María Borrás Serrano,
Santiago De Diego De Diego, Fernando De la Hoz Moreno

ETSIIT

Introducción

Nos ha tocado resolver el ejercicio serie unimodal de números.

Para ello, 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 $O(n)$.

Desarrollo de la Práctica

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.

Código Divide y Vencerás

Listing 1: Función unimodal DyV

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

        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;
}
```

Listing 2: Función main DyV

```
int main(int argc, char* argv[])
{
        vector<int> array;
        int valor = -1;
        double suma=0;

        int v_size = atoi(argv[1]);
        array.resize(v_size);

        for(int i=0; i<100; ++i)
        {
                int p = 1 + rand() %(v_size-2);
                array.at(p) = v_size-1;
                for (int i=0; i<p; i++)
                        array.at(i)=i;
                for (int i=p+1; i<v_size; i++)
                        array.at(i)=v_size-1-i+p;

                clock_t tantes;
                clock_t tdespues;
                tantes=clock();
                valor = unimodal(array);
                tdespues=clock();
                suma += (double)(tdespues - tantes) / CLOCKS_PER_SEC;
        }
        cout << v_size <<" "<< suma/100 << endl;
}
```

Código secuencial

Listing 3: Función unimodal Secuencial

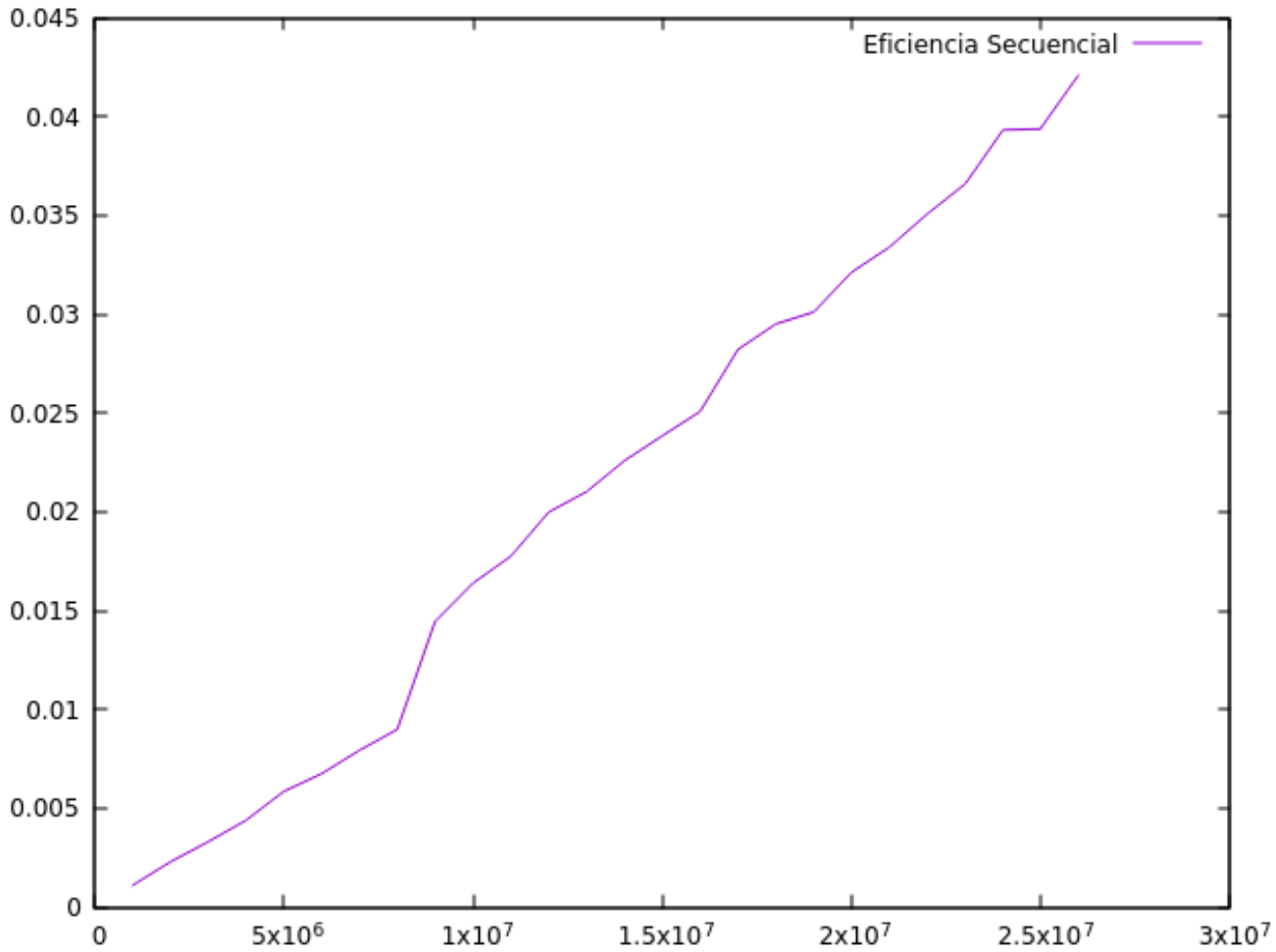
```
int unimodal_secuencial(vector<int> v)
{
        bool fin=false;
        int indice=1;

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

        return indice;
}
```

Listing 4: Función main Secuencial

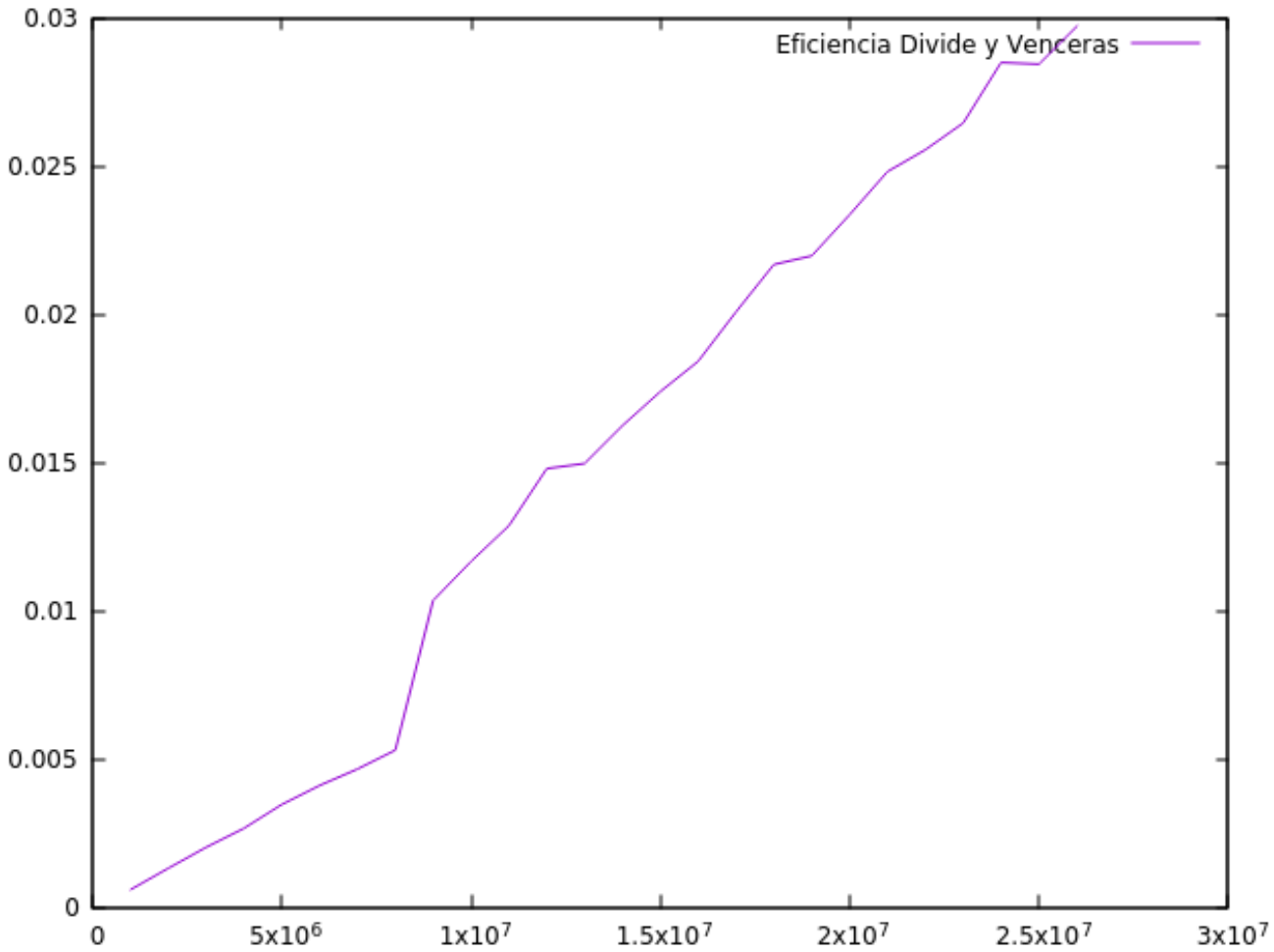
```
int main(int argc, char* argv[])
{
        vector<int> array;
        int valor = -1;
        double suma=0;

        int v_size = atoi(argv[1]);
        array.resize(v_size);

        for(int i=0; i<100; ++i)
        {
                int p = 1 + rand() %(v_size-2);
                array.at(p) = v_size-1;
                for (int i=0; i<p; i++)
                        array.at(i)=i;
                for (int i=p+1; i<v_size; i++)
                        array.at(i)=v_size-1-i+p;

                clock_t tantes;
                clock_t tdespues;
                tantes=clock();
                valor = unimodal_secuencial(array);
                tdespues=clock();
                suma += (double)(tdespues - tantes) / CLOCKS_PER_SEC;
        }
        cout << v_size <<" "<< suma/100 << endl;
}
```

Tabla Datos

Eficiencia en el caso secuencial



Eficiencia en el caso Divide y Vencerás



Ajuste híbrido en el caso secuencial

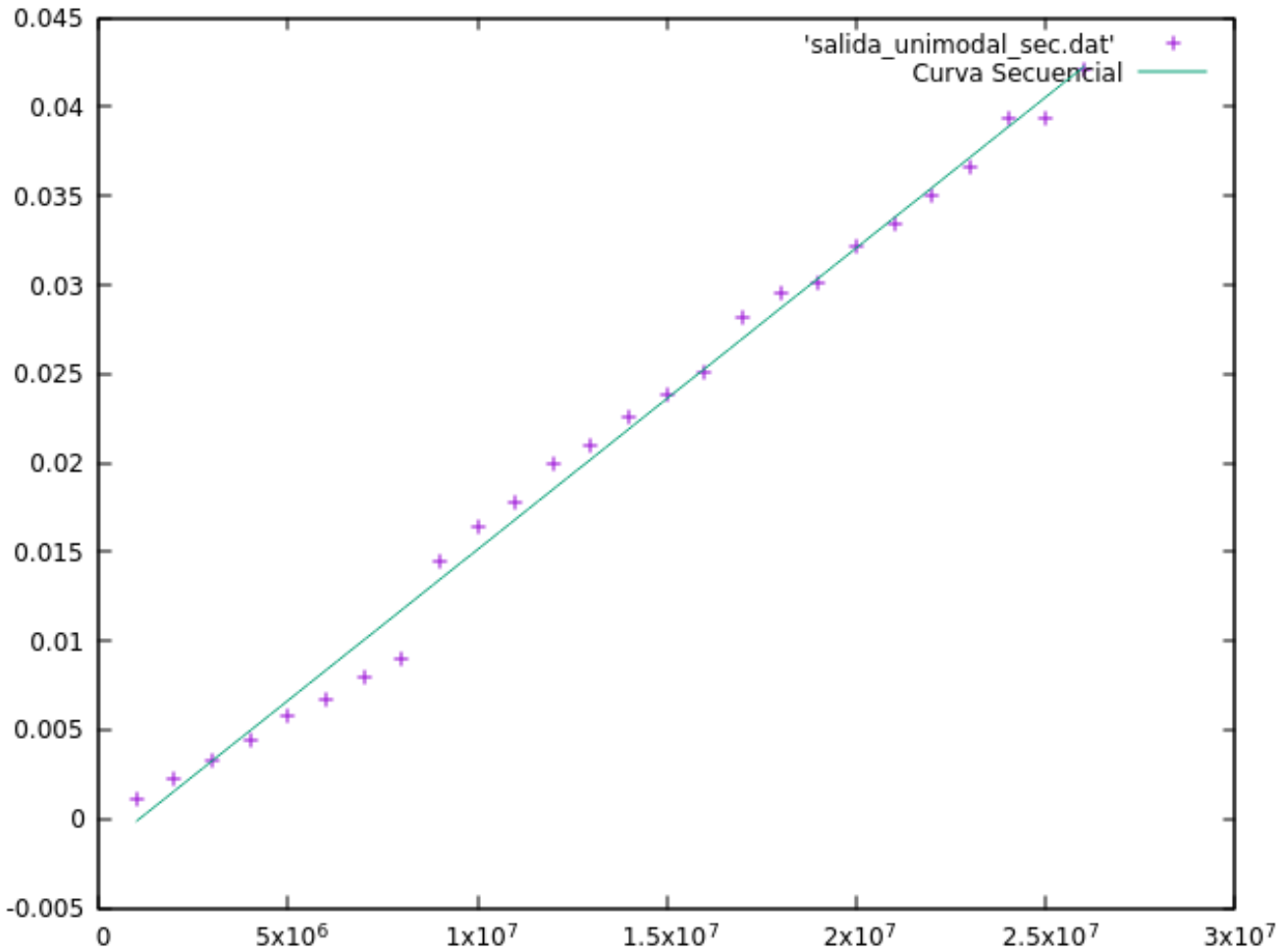


Figura: Ajustada a la función $f(x) = a_0 * x + a_1$

$$f(x) = a_0 * x + a_1$$
$$a_0 = 1{,}69539e - 09$$
$$a_1 = -0{,}00183396$$

Ajuste híbrido en el caso Divide y Vencerás

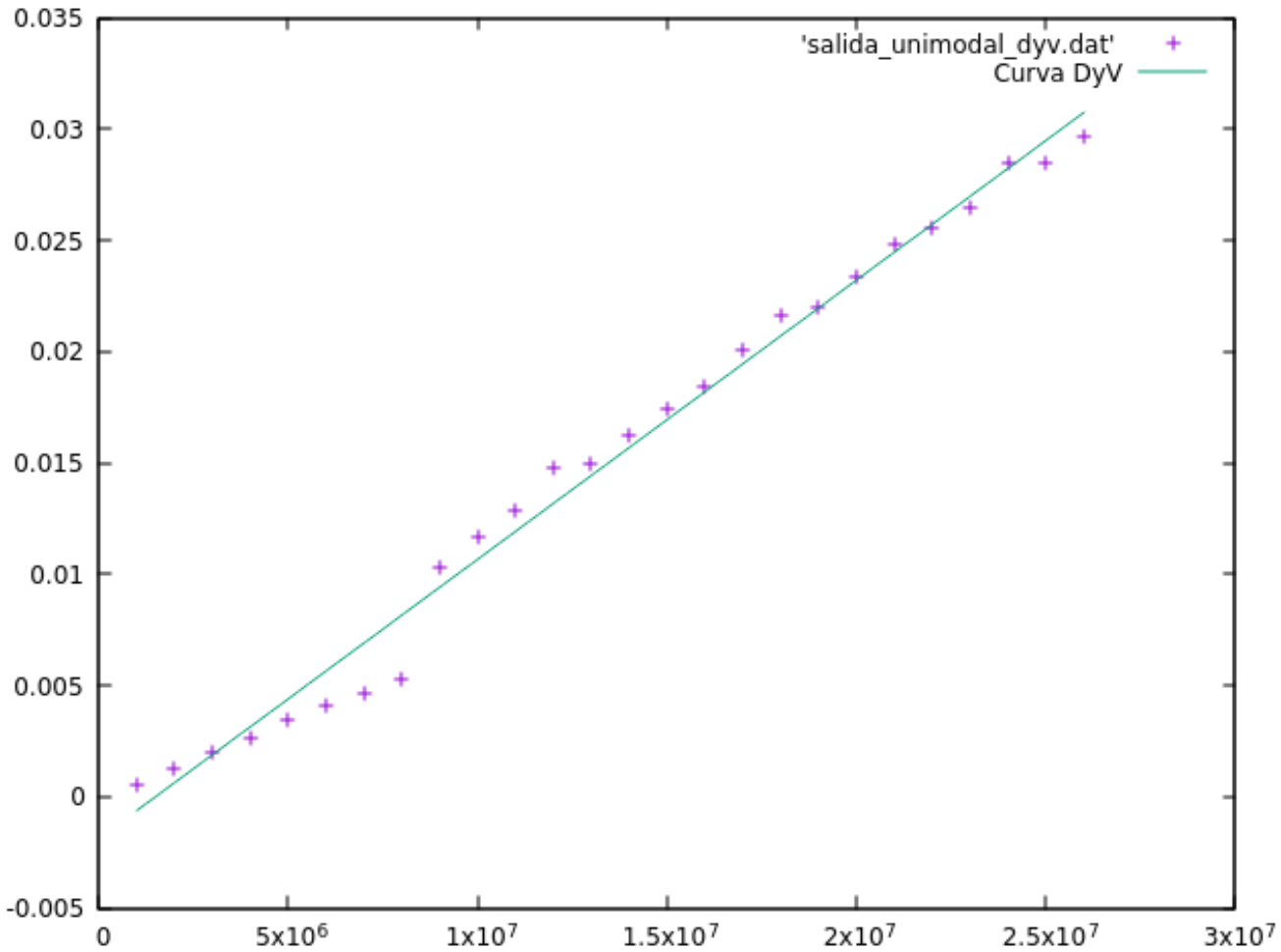


Figura: Ajustada a la función $f(x) = a_0 * log(x) + a_1 * x + a_2$

$$f(x) = a_0 * log(x) + a_1 * x + a_2$$
$$a_0 = 1{,}69539e-09$$
$$a_1 = 1{,}25523e-09$$
$$a_2 = -0{,}00189877$$

Conclusión

Escribir conclusión.