SERIE UNIMODAL DE NUMEROS

Ejercicio 5. Relación de problemas 1.

Jose Antonio Ruiz Millán
Adrián Peláez Vegas
Julio Antonio Fresneda García
Alejandro Rodríguez Muñoz

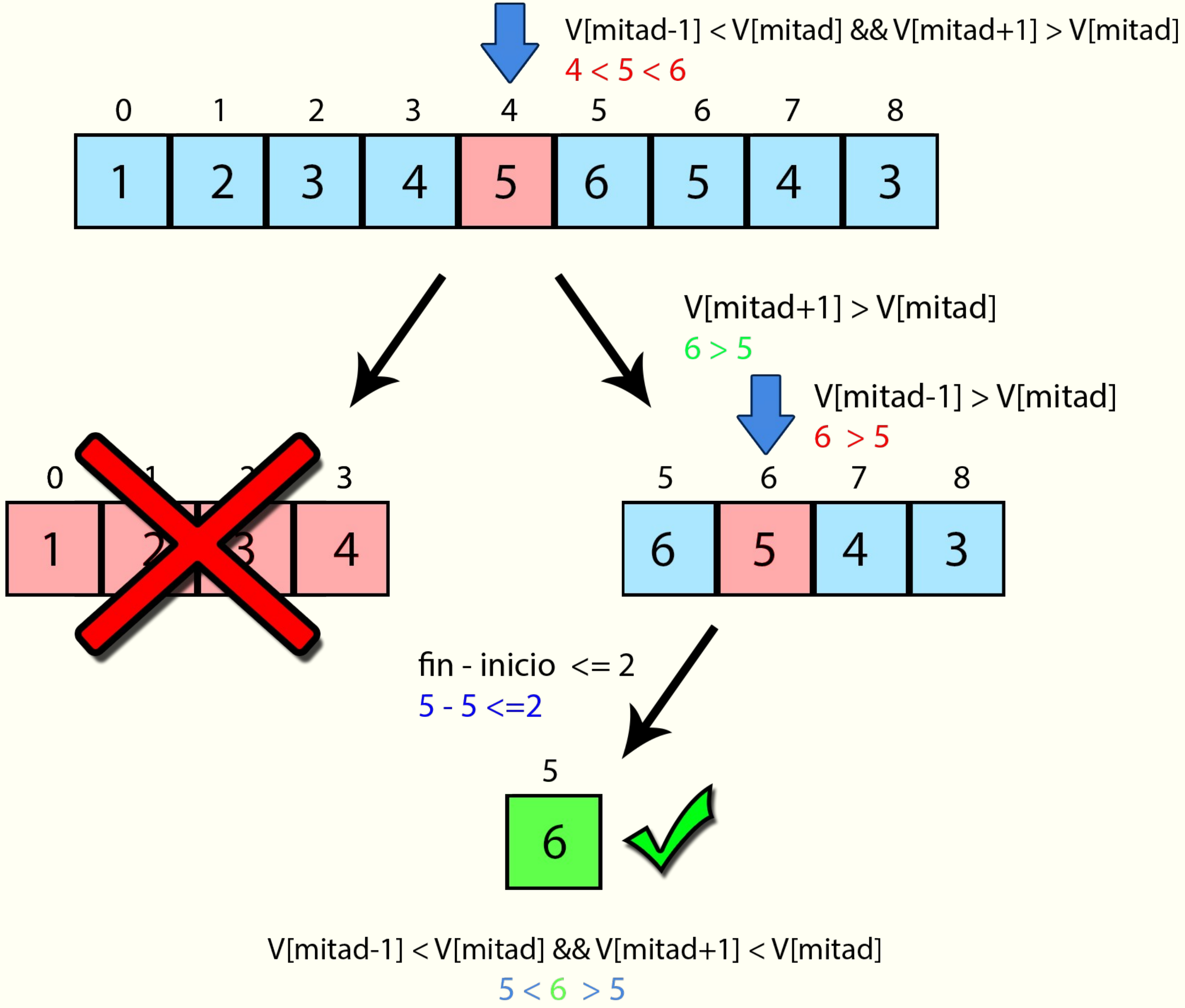
Descripción.

- Este problema consiste en dado un vector sin elementos repetidos, encontrar el elemento que tiene todos los elementos tanto a su izquierda como a su derecha menores que él.
- El algoritmo básico va comparando cada elemento anterior y posterior al que tenemos seleccionado y verificando si se cumple la condición. Por lo tanto tenemos que este algoritmo tendrá un orden O(n).
- Nuestra **solución** permite pasar del orden O(n) a O(log(n)) mejorando las prestaciones y eficiencia del calculo que tenemos que realizar, utilizando como algoritmo **Divide y Vencerás.**

Algoritmo “fuerza bruta”.

```
int unimodalFB(const vector<int> &v){
  for(int i = 1; i < v.size()-1; i++){
    if(v[i-1] < v[i] && v[i+1] < v[i]){
      return v[i];
    }
  }
}
```

Solución al problema.



Solución al problema con un ejemplo.

Algoritmo Divide y Vencerás.

```
int unimodal(const vector<int> &v,int ini, int fin){

  int p = 0;
  vector<int> v1;
  int mitad;
  mitad = (ini+fin)/2;

  if((fin-ini) > 2){
    if( v[mitad-1] < v[mitad] && v[mitad+1] > v[mitad] ){
      p = unimodal(v,mitad+1,fin);
    }
    else if( v[mitad-1] > v[mitad] && v[mitad+1] < v[mitad] ){
      p = unimodal(v,ini,mitad-1);
    }
    else if(v[mitad-1] < v[mitad] && v[mitad+1] < v[mitad] ){
      p = v[mitad];
    }
  }
  else if( v[mitad-1] < v[mitad] && v[mitad+1] < v[mitad] ){
    p = v[mitad];
  }

  return p;
}
```

Solución al problema con un ejemplo.

Algoritmo Divide y Vencerás.

$$T(n) \begin{cases} c_1 & si\ n=1 \\ T(n/2)+c_2 & si\ n>1,\ n=2^k \end{cases}$$

Para saber la eficiencia, vamos a usar expansión

$T(n) = T(n/2) + c_2$

$T(n/2) = T(n/4) + c_2$

Es decir:

$T(n) = T(n/4) + 2c_2$

o

$T(n) = T(n/8) + 3c_2$

En general:

$T(n) = T(n/2^i) + ic_2$, siendo i el número de llamadas recursivas.

Cuando $i = k$, quiere decir que no habrá más llamadas recursivas, es d en la parte derecha hay $T(1)$.

La fórmula quedaría:

$T(n) = T(1) + kc_2$

Como $2^k = n$, $k = log_2(n)$.

La fórmula quedaría:

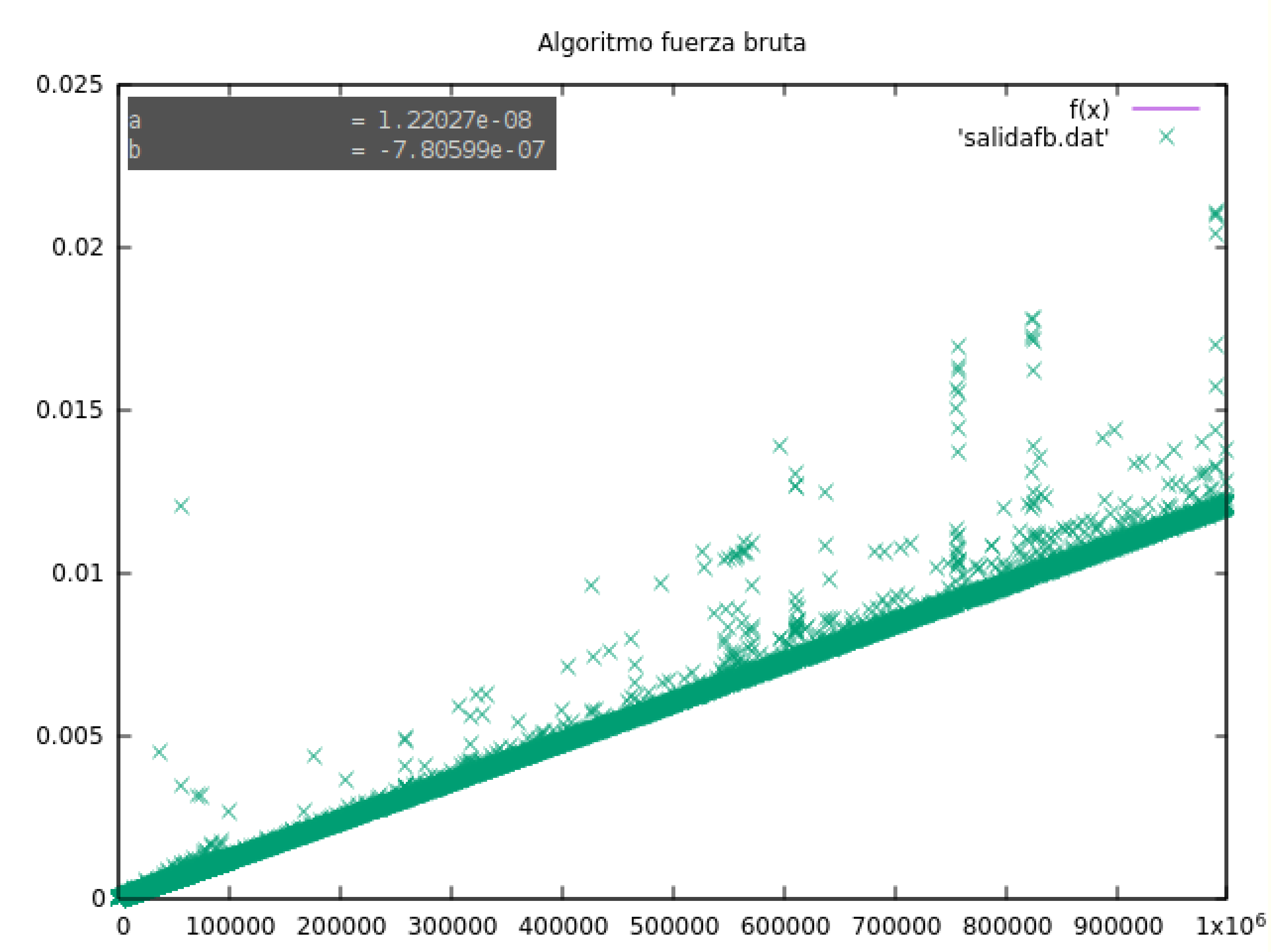
$T(n) = T(1) + c_2 log_2(n)$

Podemos tomar $T(1) = c_1$

Por lo que la fórmula quedaría $T(n) = c_1 + c_2 log_2(n)$, por lo que la eficiencia sería $O(log(n))$.

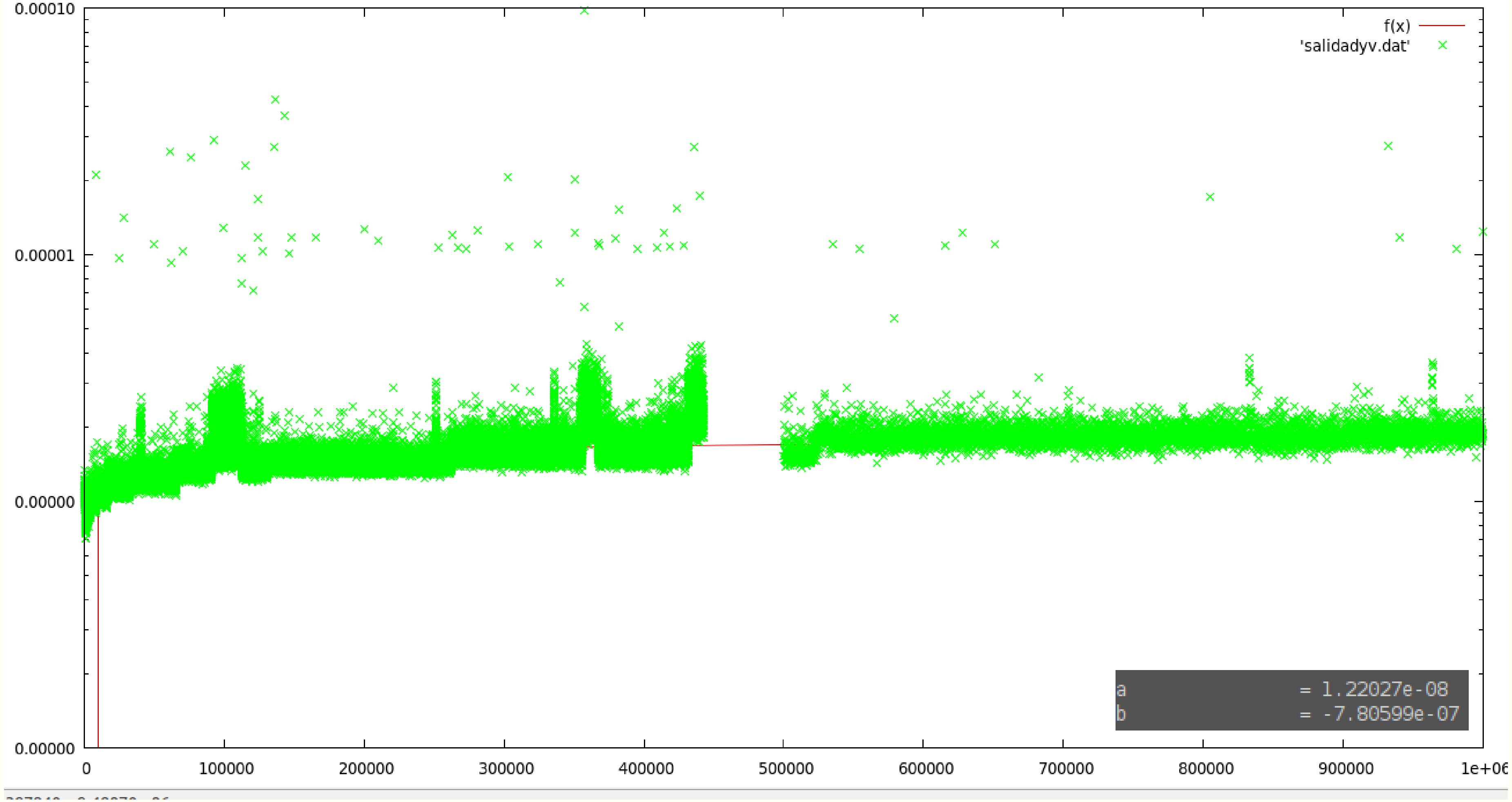
Gráficas comparativas.

Grafica de algoritmo por “fuerza bruta”; O(n):

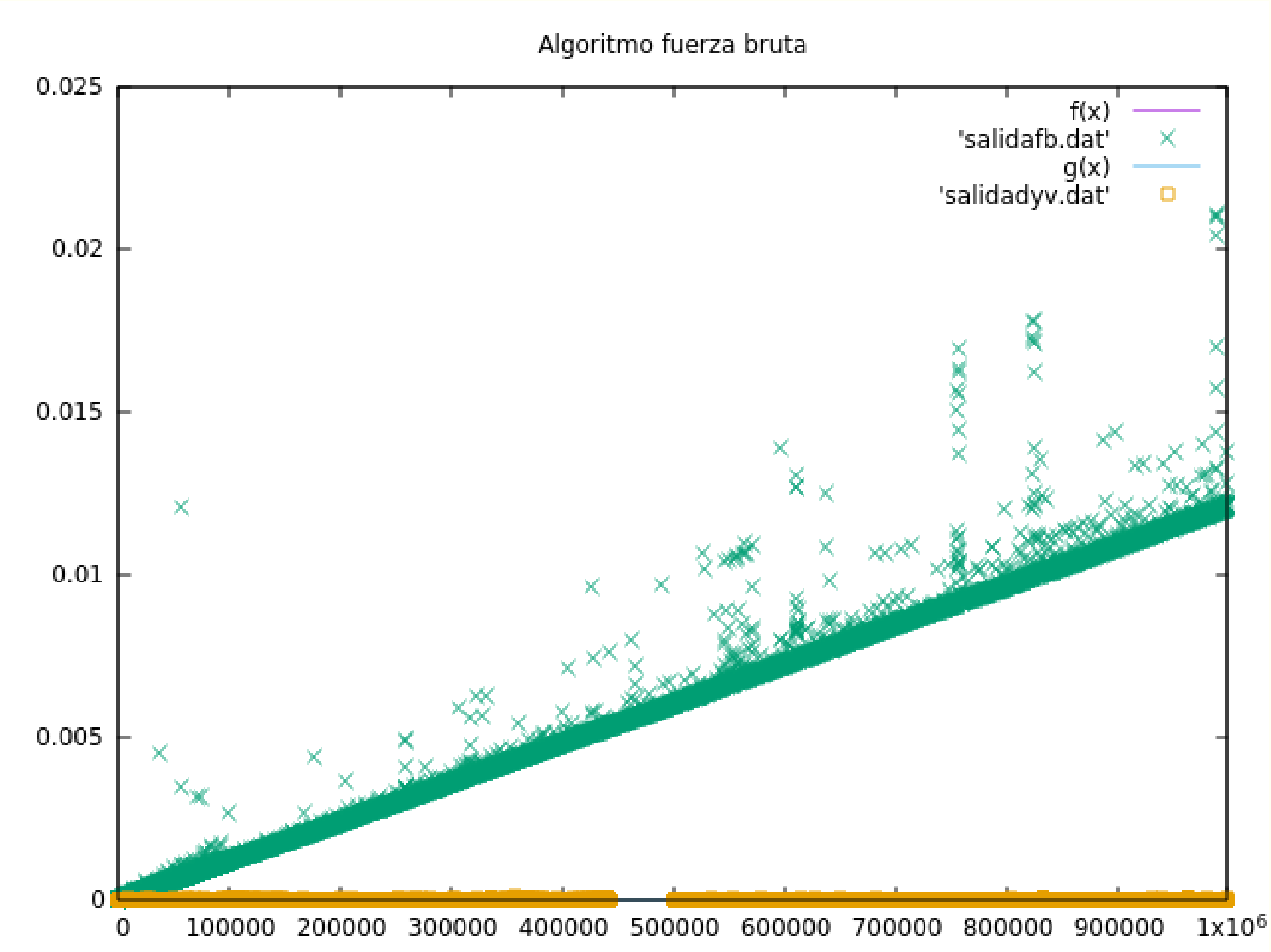


Gráficas comparativas.

Grafica de algoritmo **Divide y Vencerás**; O(log(n)):



Gráficas comparativas.



Gráficas comparativas.