COMPARACIÓN DE PREFERENCIAS

Ejercicio 1. Relación de problemas 1.

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

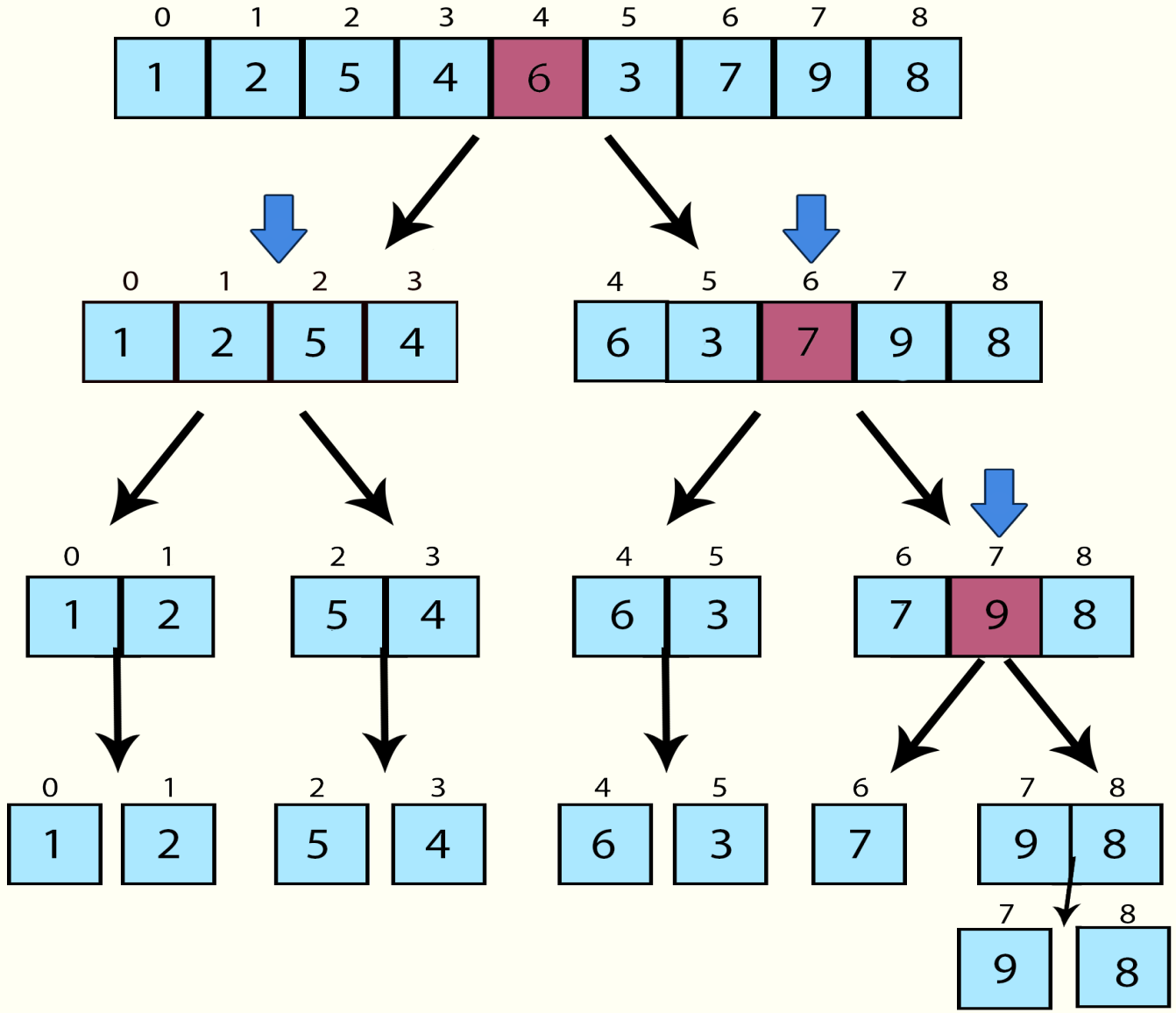
Descripción.

- Este problema consiste en comparar las preferencias de dos personas en un ranking para cada persona, obteniendo como resultado la relación que hay entre ellos y así poder estimar los gustos parecidos entre diversas personas.
- El problema lo tenemos cuando el vector de preferencias empieza ha ser elevado en elementos, ya que el algoritmo por “fuerza bruta” tiene un orden de $O(n^2)$ ya que para cada elemento del vector, debemos comprobar todos los elementos que le siguen en el vector.
- Nuestra **solución** permite pasar del orden $O(n^2)$ a $O(n \log(n))$ mejorando las prestaciones y eficiencia del calculo que tenemos que realizar, utilizando como algoritmo **Divide y Vencerás**.

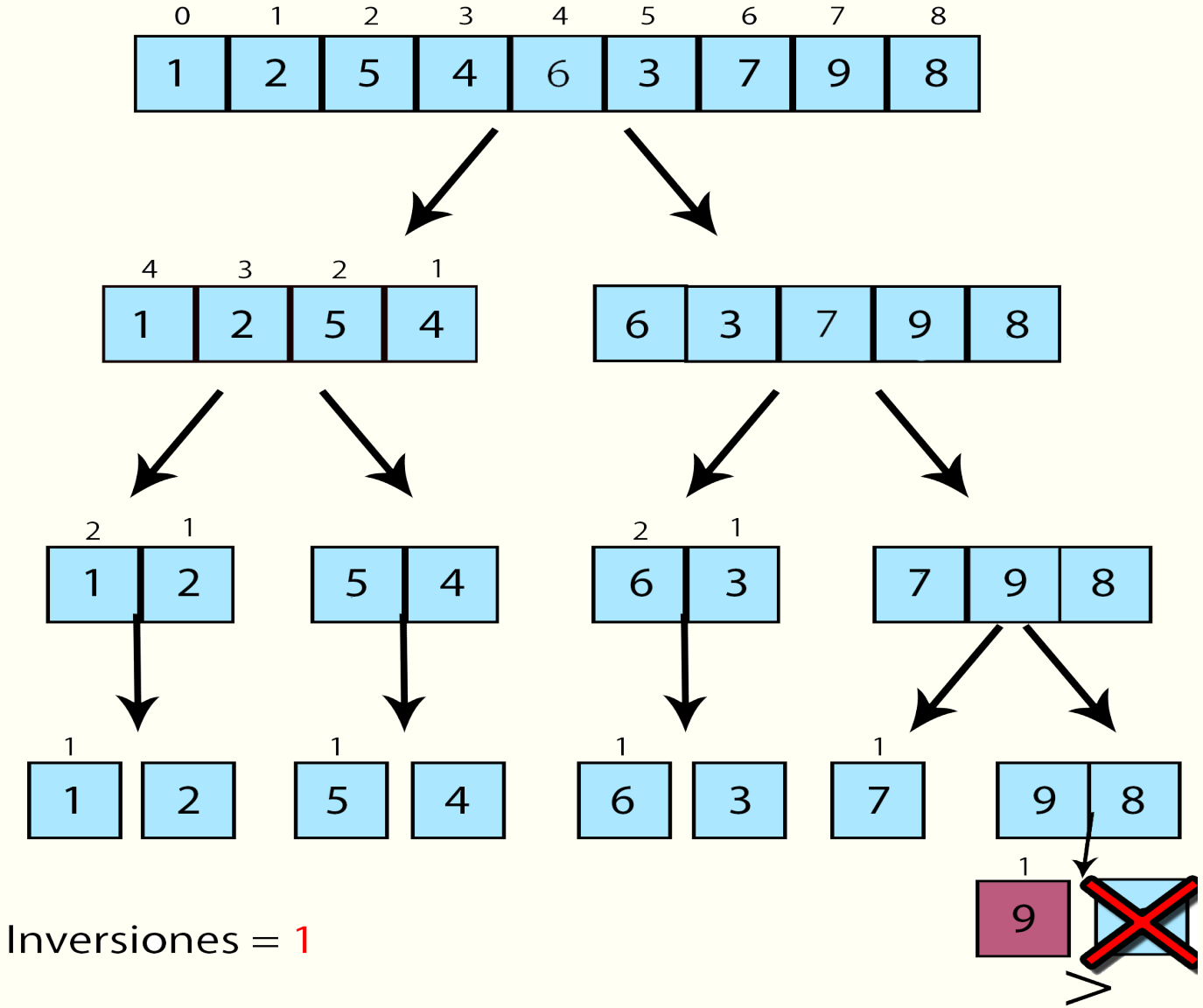
Algoritmo “fuerza bruta”.

```
int FuerzaBruta(vector<int> &v){
  int inv = 0;
  for(int i = 0; i < v.size()-1; i++)
    for(int j = i+1; j < v.size(); j++)
      if(v[i] > v[j]) inv++;
  return inv;
}
```

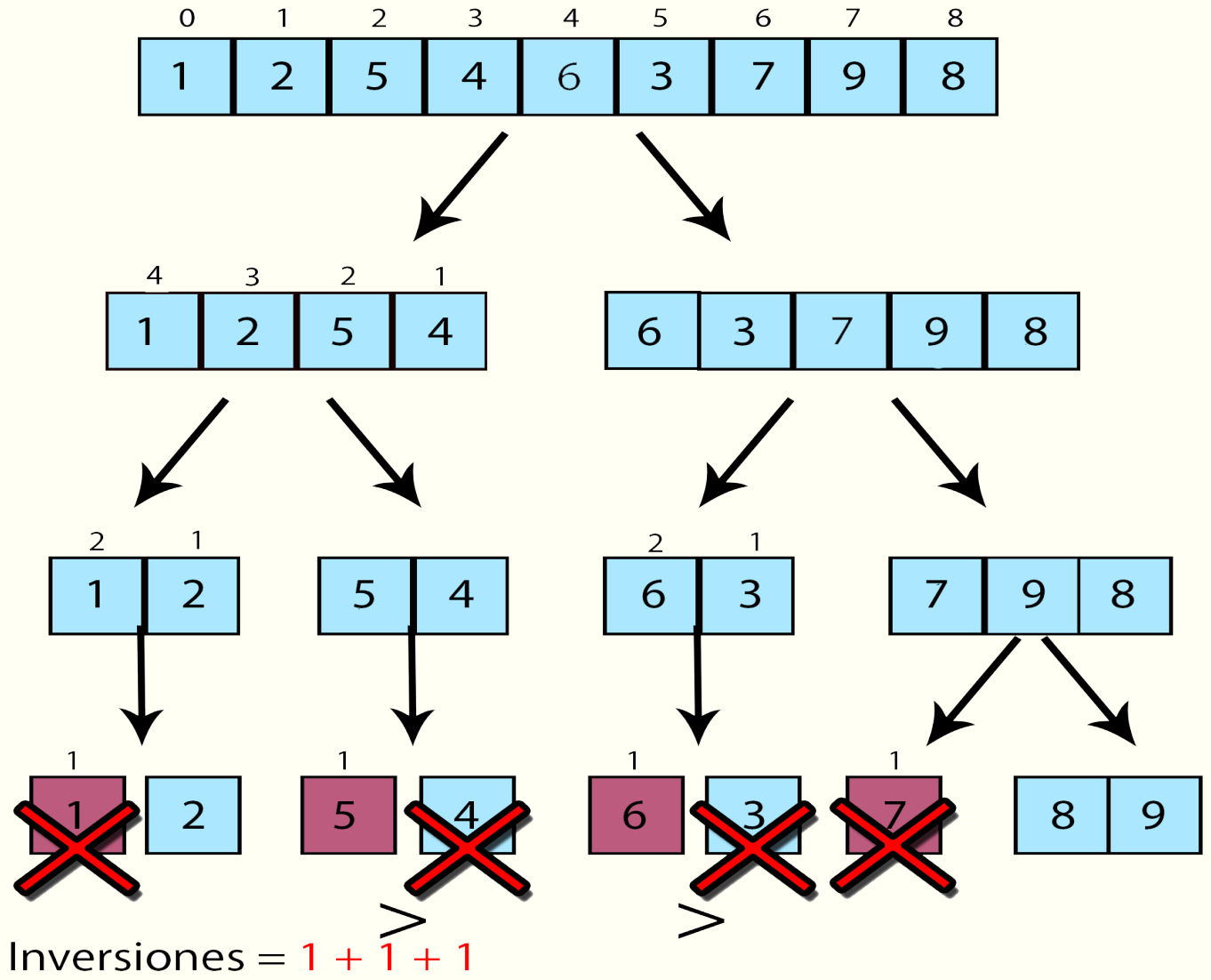
Solución al problema.



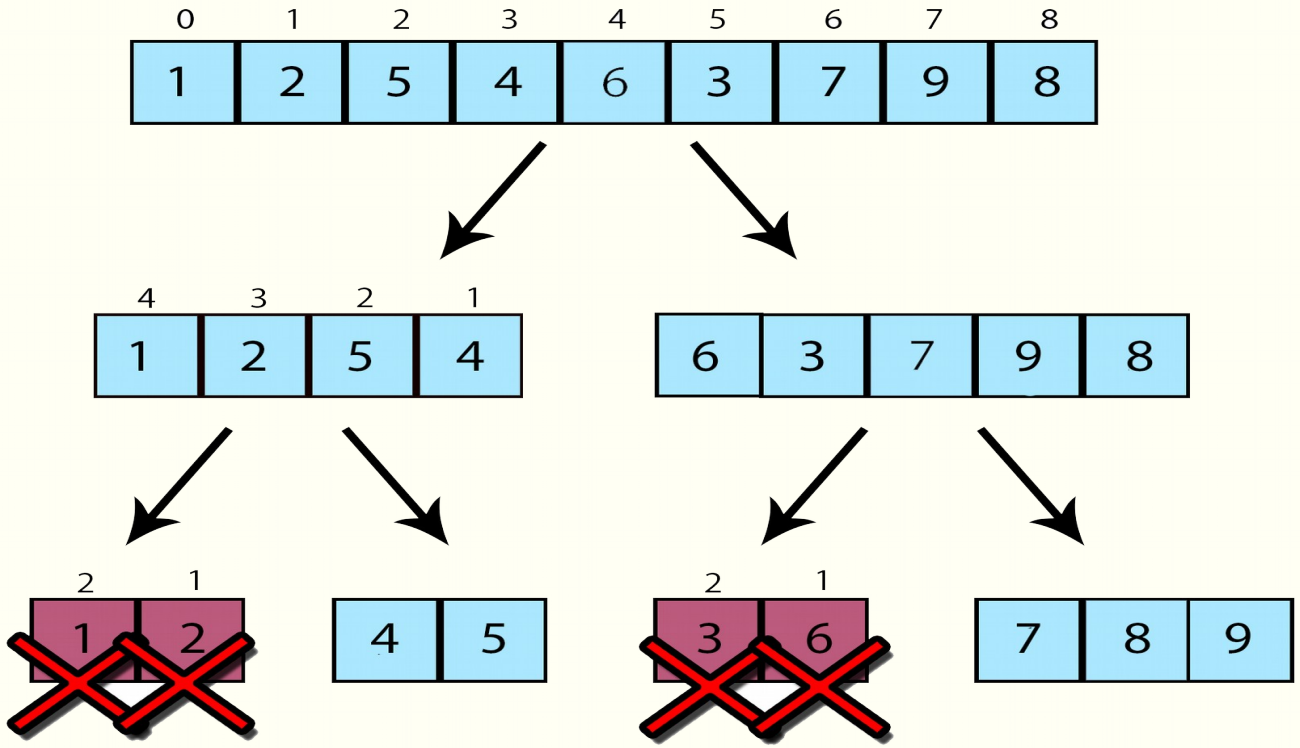
Solución al problema.



Solución al problema.

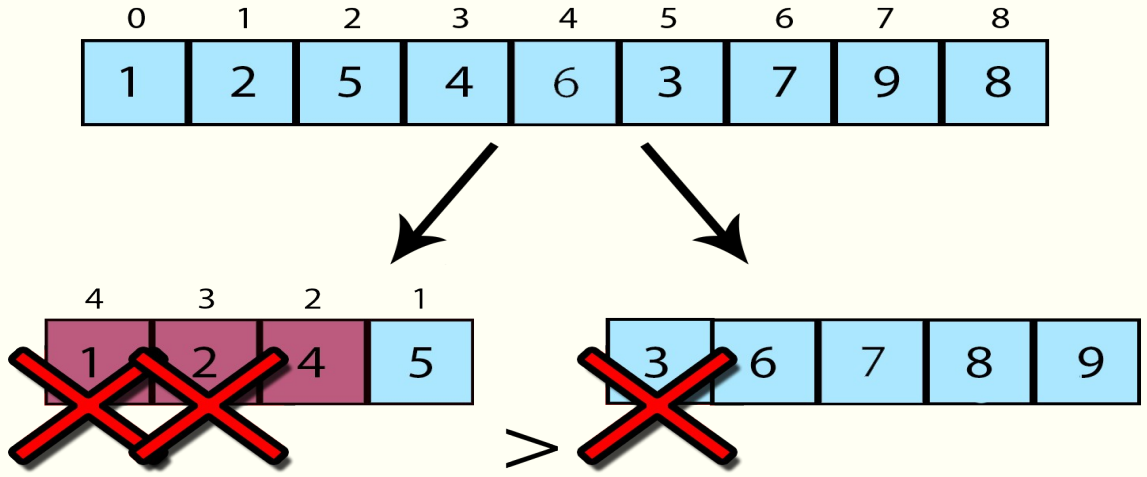


Solución al problema.



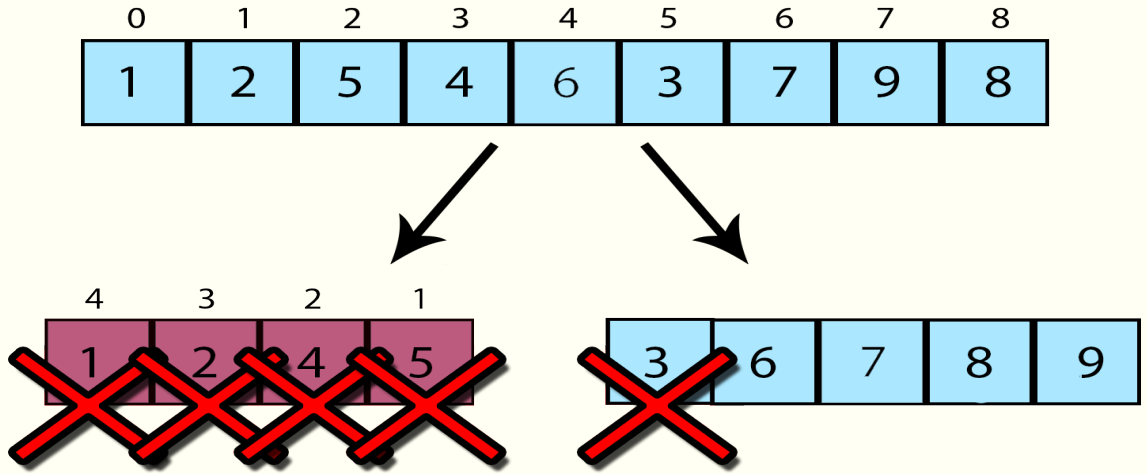
Inversiones = 1 + 1 + 1

Solución al problema.



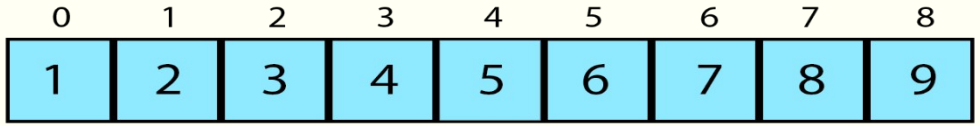
Inversiones = 1 + 1 + 1 + 2

Solución al problema.



Inversiones = 1 + 1 + 1 + 2

Solución al problema.



Inversiones = 1 + 1 + 1 + 2 = 5.

Algoritmo Divide y Vencerás.

```
void mezcla(vector<int> &v)
{
    vector<int> vector1;
    vector<int> vector2;
    int n1, n2,i,j;

    if (v.size() > 1)
    {
        if (v.size()%2 == 0)
            n1=n2=(int) v.size() / 2;
        else
        {
            n1=(int) v.size() / 2;
            n2=n1+1;
        }
        for(i=0;i<n1;i++)
            vector1.push_back(v[i]);
        for(j=0;j<n2;i++,j++)
            vector2.push_back(v[i]);
        v.clear();
        mezcla(vector1);
        mezcla(vector2);
        combinar(vector1, vector2, v);
    }
}
```

```
int inv = 0;

void combinar(const vector<int> &arreglo1,const vector<int> &arreglo2,
vector<int> &arreglo3)
{
    int x1=0, x2=0 , indice = arreglo1.size();

    while (x1<arreglo1.size() && x2<arreglo2.size()) {
        if (arreglo1[x1]<arreglo2[x2]) {
            arreglo3.push_back(arreglo1[x1]);
            x1++;
            indice--;
        }
        else {
            arreglo3.push_back(arreglo2[x2]);
            x2++;
            inv+=indice;
        }
    }
    while (x1<arreglo1.size()) {
        arreglo3.push_back(arreglo1[x1]);
        x1++;
    }
    while (x2<arreglo2.size()) {
        arreglo3.push_back(arreglo2[x2]);
        x2++;
    }
}
```

Algoritmo Divide y Vencerás.

Solución al problema con un ejemplo.

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

Para saber la eficiencia, podemos usar expansión:
$T(n) = 2T(n/2) + c_2 n$
$T(n/2) = 2T(n/4) + c_2 n/2$
Es decir,
$T(n) = 4T(n/4) + 2c_2 n$ o
$T(n) = 8T(n/8) + 3c_2 n$
En general:

$$T(n) = 2^i T(n/2^i) + ic_2 n$$

Tomando $n = 2^k$, la expansión termina cuando llegamos a $T(1)$ en el lado de la derecha, lo que ocurre cuando $i = k$

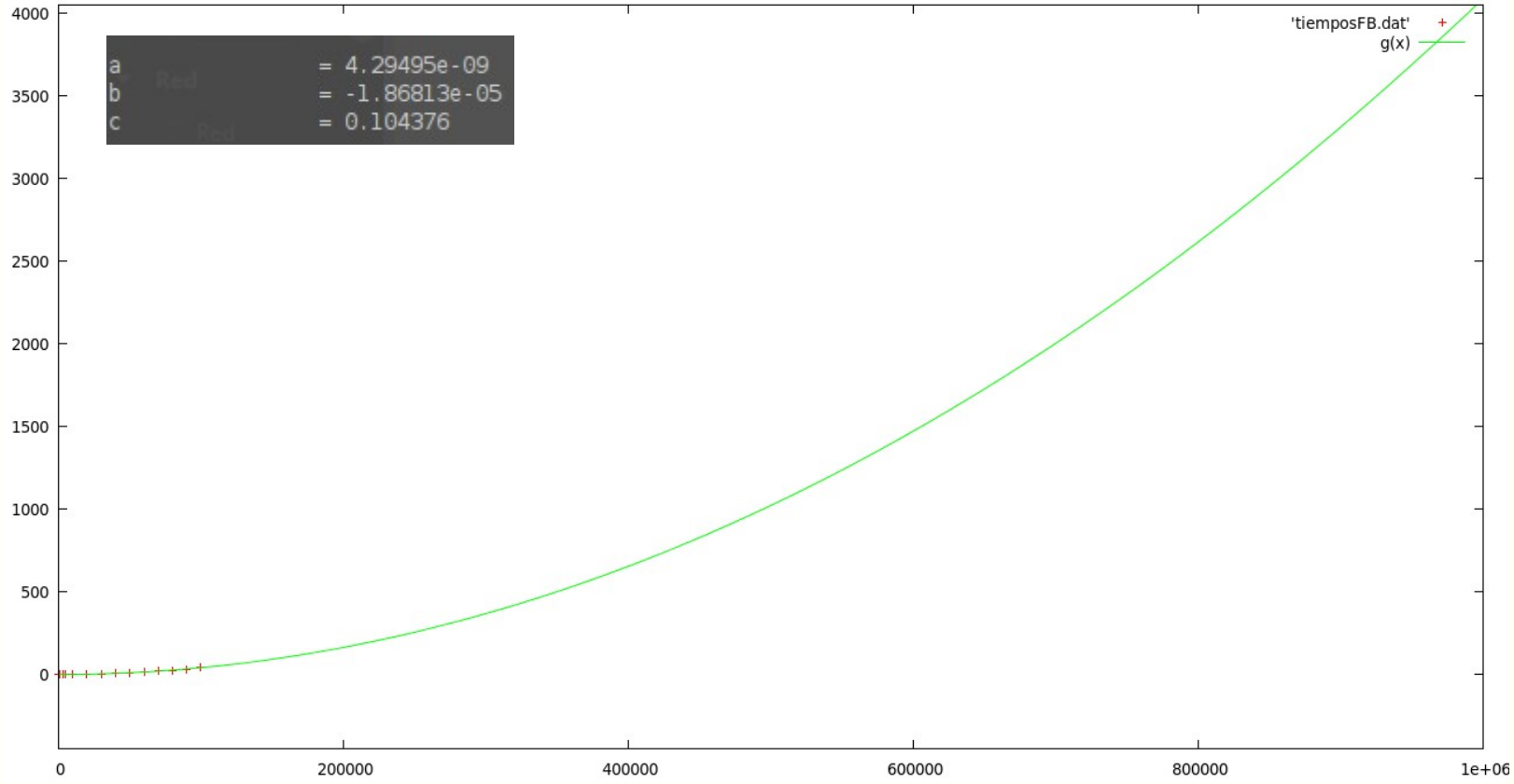
$T(n) = 2^k T(1) + kc_2 n$
Ya que $2^k = n$, operando, $k = log(n)$;
Además, $T(1) = c_1$, por lo que tenemos:

$$T(n) = c_1 n + c_2 n log(n)$$

Por tanto el tiempo del algoritmo es O(nlog(n))

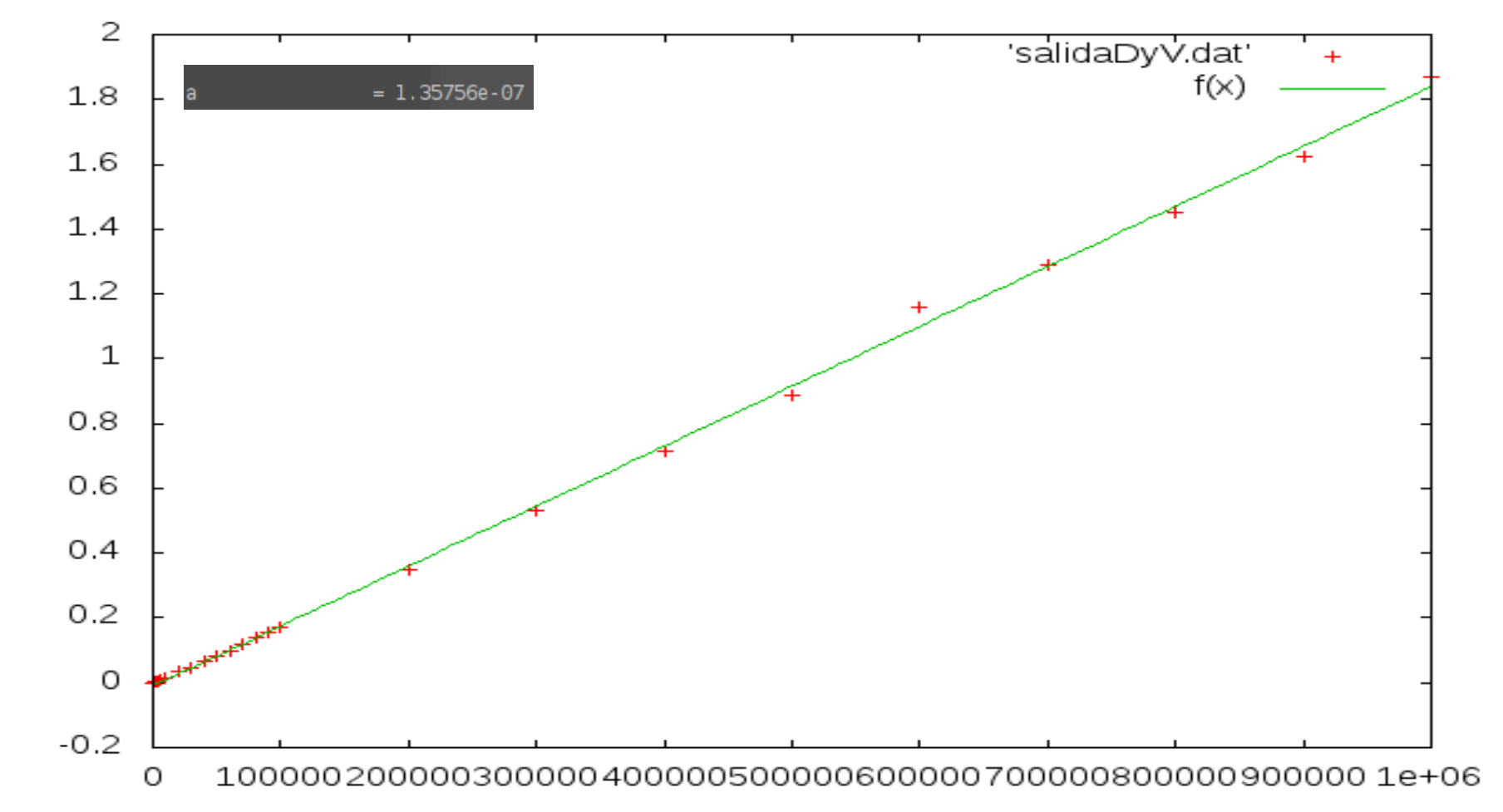
Gráficas comparativas.

Grafica de algoritmo por “fuerza bruta”; $O(n^2)$:

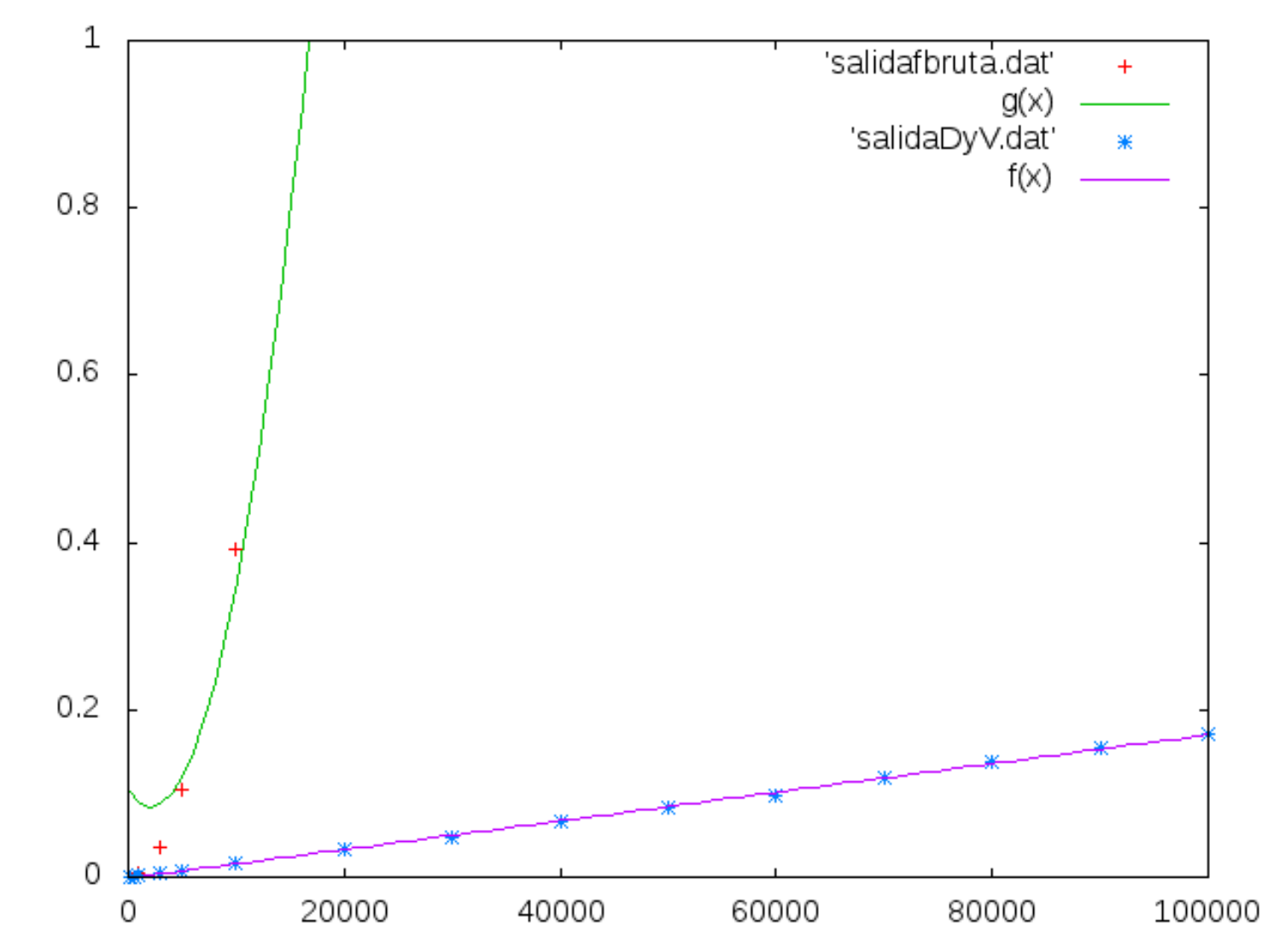


Gráficas comparativas.

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



Gráficas comparativas.



Gráficas comparativas.