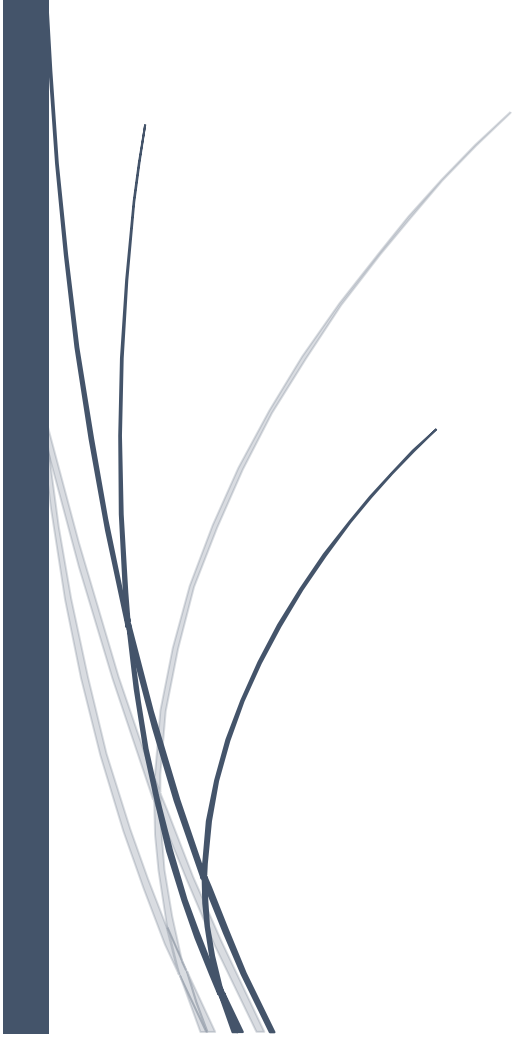
7-4-2017

Práctica 2

Algoritmos Divide y Vencerás



Pablo Álvarez Cabrera
Johanna Capote Robayna
Cristina de la Torre Valverde
Guille Galindo Ortuño
Adolfo Soto Werner

Índice

- Introducción
- Algoritmo de Fuerza Bruta
- Algoritmo Divide y vencerás
- Análisis de la eficiencia teórica, empírica e híbrida

Introducción.

En esta práctica vamos a realizar el primero de los ejercicios propuestos, llamado “Comparación de preferencias”.

Este ejercicio consiste en realizar dos algoritmos diferentes, uno basado en la “Fuerza Bruta” y otro aplicando la filosofía “Divide y Vencerás”. Este último soluciona el problema de forma recursiva y de forma más rápida que el de “Fuerza Bruta”, que soluciona el problema de forma más intuitiva y menos eficiente.

El problema planteado consiste en comparar dos vectores midiendo la similitud que existe entre ellos, haciendo hincapié en el número de inversiones de elementos del primer vector existentes en el segundo. Con inversiones nos referimos al siguiente concepto:

> Sean dos vectores v_1 y v_2 . Decimos que los elementos *i* y *j* están invertidos en los vectores si en el vector v_1 el elemento *i* se encuentra a la derecha del elemento *j* mientras que en el vector v_2 el elemento *i* se encuentra a la izquierda del elemento *j*

v_1

1	2	…	i	…	j	…	n

1	2	…	j	…	i	…	n

v_2

Suponiendo que estos vectores representan dos rankings de elementos de dos personas distintas, podemos utilizar este algoritmo para “medir” la similitud en las preferencias de estos usuarios. Para ello consideramos que si un elemento *i* está más cerca de la posición 0 del vector que otro elemento *j*, o dicho de otro modo, el elemento *i* está a la derecha del elemento *j,* esto significa que el usuario prefiere el elemento *i* al *j.* Por tanto, a mayor número de inversiones existentes en los dos vectores, menor grado de similitud existe entre las preferencias de los dos usuarios.

En este ejercicio medimos la similitud entre los vectores midiendo sólo el número de inversiones, es decir, no nos fijamos en la cantidad de elementos del vector que existen entre los elementos invertidos, lo que haría esta medición mucho más precisa, pues cuanto más separados estén los elementos menor grado de similitud existe entre los vectores.

Tras implementar los dos algoritmos mencionados anteriormente, compararemos los tiempos de ejecución de los dos algoritmos y veremos que la filosofía Divide y Vencerás es mucho más eficiente que la reflejada en el algoritmo “de Fuerza Bruta”.

Algoritmo de Fuerza Bruta.

Este algoritmo es el menos eficiente de los dos que hemos implementado. Su funcionamiento se basa en tres bucles. Los dos mayores son los que se encargan de establecer las relaciones que hay entre los elementos del vector original (el vector 0, 1, 2, 3, …, n). Es el tercer bucle el que se encarga de buscar los elementos que queremos comparar en el segundo vector.

Cuando el tercer bucle encuentra alguna de las dos coincidencias (o *v[k] == i* o *v[k] == j*), almacena la posición en la que se encuentra ese elemento y busca la siguiente coincidencia. Cuando ha localizado ambas coincidencias compara las posiciones en las que se encuentran, y si la posición en la que se encuentra el elemento *j* está a la derecha de la posición en la que se encuentra el elemento *i,* se incrementa el contador de inversiones.

Con un ejemplo se entenderá mejor:

Pasamos como parámetro el siguiente vector

3	1	0	2

y al empezar a ejecutar el algoritmo las variables *i, j* y *k* tienen como valores iniciales 0, 1 y 0 respectivamente, luego en la primera iteración del bucle *k* hace dos comparaciones:

Primera iteración del bucle k *i = 0, j = 1, k = 0*	Segunda iteración del bucle k *i = 0, j = 1, k = 1*	Tercera iteración del bucle k *i = 0, j = 1, k = 2*
1º) ¿es *v[0] (=3)* igual a *0*? ***NO*** 2º) ¿es *v[0] (=3)* igual a 1? ***NO*** **Siguiente iteración**	1º) ¿es *v[1] (=1)* igual a *0*? ***NO*** 2ª) ¿es *v[1] (=1)* igual a 1? ***SI*** 3º) Cambio el valor de *encontrado1* a *true* y almaceno la posición de esta primera coincidencia con *j* **Siguiente iteración**	1º) ¿es *v[2] (=0)* igual a *0*? ***SI*** 2º) Cambio el valor de *encontrado2* a *true* y almaceno la posición de esta segunda coincidencia con *i.* 3º) ¿es *v[2] (=0)* igual a 1? ***NO*** ***Se para el bucle ya que ha encontrado los dos elementos***

Como ya se han encontrado las dos coincidencias paso a comparar las posiciones. ¿Es la posición en la que he encontrado a *i* mayor que la posición en la que he encontrado a *j*? Si, entonces hay una inversión, la del elemento 1 (valor de *j*) con el 0 (valor de *i*). Esto se ve claramente enfrentando los dos vectores:

i *j*

0	1	2	3

j *i*

3	1	0	2

Algoritmo Divide y Vencerás.

Este algoritmo se trata de un algoritmo recursivo, en el que se utiliza la estrategia de "divide y vencerás". Su funcionamiento se basa en dividir el vector en dos partes y mediante dos bucles comparar los elementos de la izquierda con los de la derecha, si estos últimos son más pequeños que los de la izquierda, sumaremos uno al contador de inversiones. Tras esto, se llama de nuevo a la función esta vez pasándole las dos partes.

Lo primero que se hace al entrar en la función es dividir el vector en dos partes (0 – n/2 y n/2 - n). El primer bucle controla la parte izquierda, es decir desde el principio del vector hasta la mitad; y por cada y iteración de éste hay un segundo bucle que recorre la segunda parte del vector (n/2 – n), comparando cada vez, si el elemento que indica el primero bucle es mayor que el que indica el seguno bucle, si ocurre esto, el contador de inversiones aumenta.

Veamos el ejemplo anterior:

Pasamos como parámetro el siguiente vector

3	1	0	2

y lo dividimos en dos partes. La primera parte va desde el 3 hasta el 1, y la segunda parte desde el 0 hasta el 2. En el primero bucle el índice i=3, y en el seguno bucle el indice j=0.

Primera iteración del bucle i:

Primera iteración del bucle j *i = 3, j = 0*	Segunda iteración del bucle k *i = 3, j = 2*
1º) ¿es *v[i] (=3) > v[j] (=0)*? ***SI, sumamos uno al contador de inversiones: "inversiones ++"***	1º) ¿es *v[i] (=3) > v[j] (=2)*? ***SI, sumamos uno al contador de inversiones: "inversiones ++"***

Segunda iteración del bucle i:

Primera iteración del bucle j *i = 1, j = 0*	Segunda iteración del bucle k *i = 1, j = 2*
1º) ¿es *v[i] (=1) > v[j] (=0)*? ***SI, sumamos uno al contador de inversiones: "inversiones ++"***	1º) ¿es *v[i] (=1) > v[j] (=2)*? ***NO***

Y a continuación volvemos a llamar al método dos veces, una con la primera parte del vector (desde el 3 hasta el 1) y otra con la segunda parte del vector (desde el 0 hasta el 2).

Análisis de la eficiencia teórica, empírica e híbrida.

En este apartado se van a analizar la eficiencia teórica, empírica e híbrida de los distintos algoritmos implementados para resolver el problema (algoritmo de fuerza bruta y algoritmo divide y vencerás), comentando los resultados obtenidos.

EFICIENCIA TEÓRICA

- Algoritmo de fuerza bruta.

```
int fuerzabruta(int *v, int n){
  int inversiones = 0;
  bool encontrado1 = false;
  bool encontrado2 = false;
  int posi, posj;

  for(int i=0; i<n-1; i++){
    for(int j=i+1; j<n; j++){
      for(int k=0; k<n && (!encontrado1 || !encontrado2); k++){
        if(v[k] == i){
          posi = k;
          encontrado1 = true;
        }
        if(v[k] == j){
          posj = k;
          encontrado2 = true;
        }
      }
      encontrado2 = false;
      encontrado1 = false;
      if(posi > posj)
        inversiones ++;
    }
  }
  return inversiones;
}
```

La mayor parte del tiempo de ejecución de este algoritmo se emplea en los 3 bucles *for* que se encargan de buscar los elementos que se quieren comparar en el vector.
Si analizamos dichos bucles considerando que el tiempo de ejecución de lo que hay en su interior está acotado por una constante *a*, obtenemos:

$$\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\sum_{k=0}^{n-1} a = n\cdot\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} a = an\cdot\sum_{i=0}^{n-2}(n-i-1) = an\frac{n^2-n}{2} = \frac{a}{2}n^3 - \frac{a}{2}n^2$$

Como $\frac{a}{2}n^3 - \frac{a}{2}n^2 \in O(n^3)$, podemos decir que el algoritmo de fuerza bruta es de orden $O(n^3)$.

- Algoritmo divide y vencerás.

```
int DyV(int *v, int n){
    int inversiones = 0;

    if (n == 1){
        return inversiones;
    }

    int tam = n/2;

    for (int i = 0; i < tam; i++){
        for (int j = tam; j< n; j++){
            if (v[i] > v[j])
                inversiones++;
        }
    }

    inversiones += DyV(v, tam);
    inversiones += DyV(v+tam, n-tam);
    return inversiones;
}
```

Se trata de un algoritmo recursivo, por lo que notaremos como *T(n)* al tiempo que tarda una ejecución para *n* elementos.

Tenemos que $T(n) = \frac{n}{2} \cdot \frac{n}{2} \cdot T\left(\frac{n}{2}\right) = \frac{n^2}{4} \cdot T\left(\frac{n}{2}\right)$

Ahora realizamos el cambio de variable $n = 2^i$, obteniendo:

$$t_i = \left(\frac{2^i}{2}\right)^2 + 2t_{i-1} = \left(2^{i-1}\right)^2 + 2t_{i-1} = 2^{2(i-1)} + 2t_{i-1} = 4^{i-1} + 2t_{i-1}$$

Cambiamos *i* por *i+1* (no afecta a la ecuación en diferencias):

$$t_{i+1} = 4^i + 2t_i \Rightarrow t_i = C_1 2^i + C_2 4^i = C_1 2^i + C_2\left(2^i\right)^2$$

Por último, deshacemos el cambio de variable:

$$T(n) = C_1 n + C_2 n^2 \in O(n^2)$$

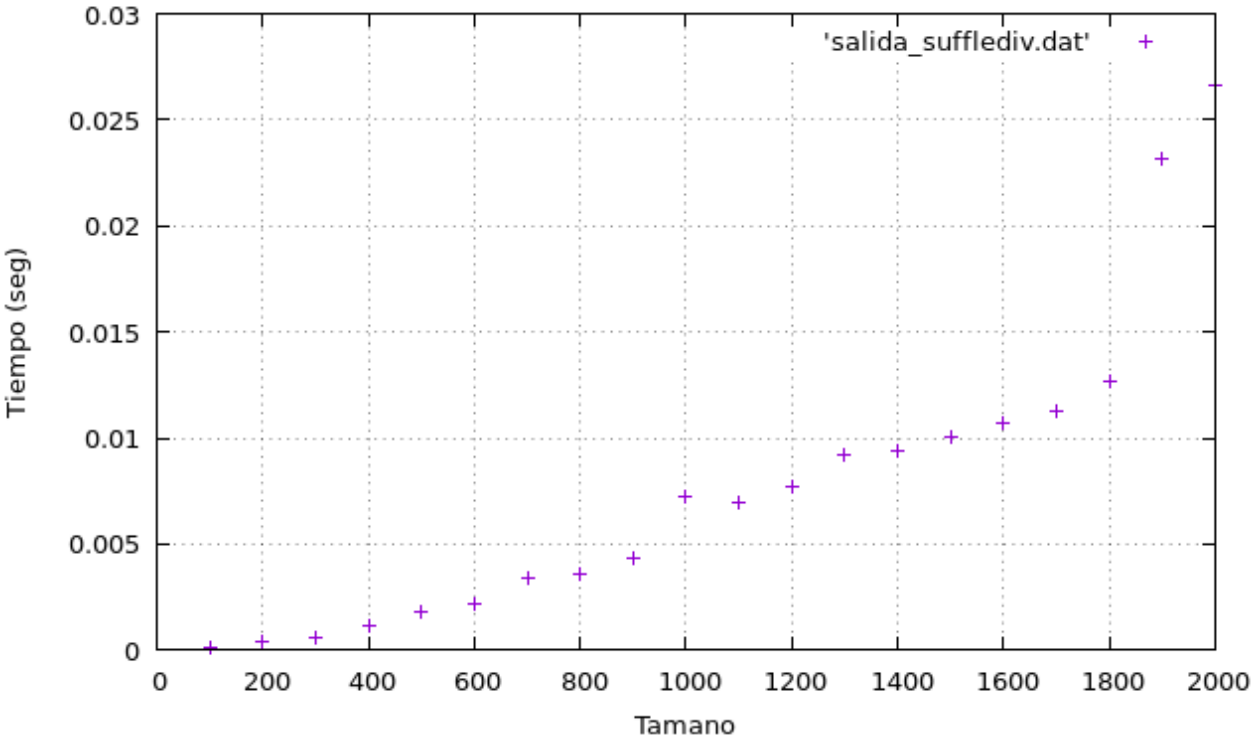
Hemos obtenido que el algoritmo divide y vencerás es de orden $O(n^2)$, es decir, es más rápido que el de fuerza bruta (como era de esperar).

EFICIENCIA EMPÍRICA

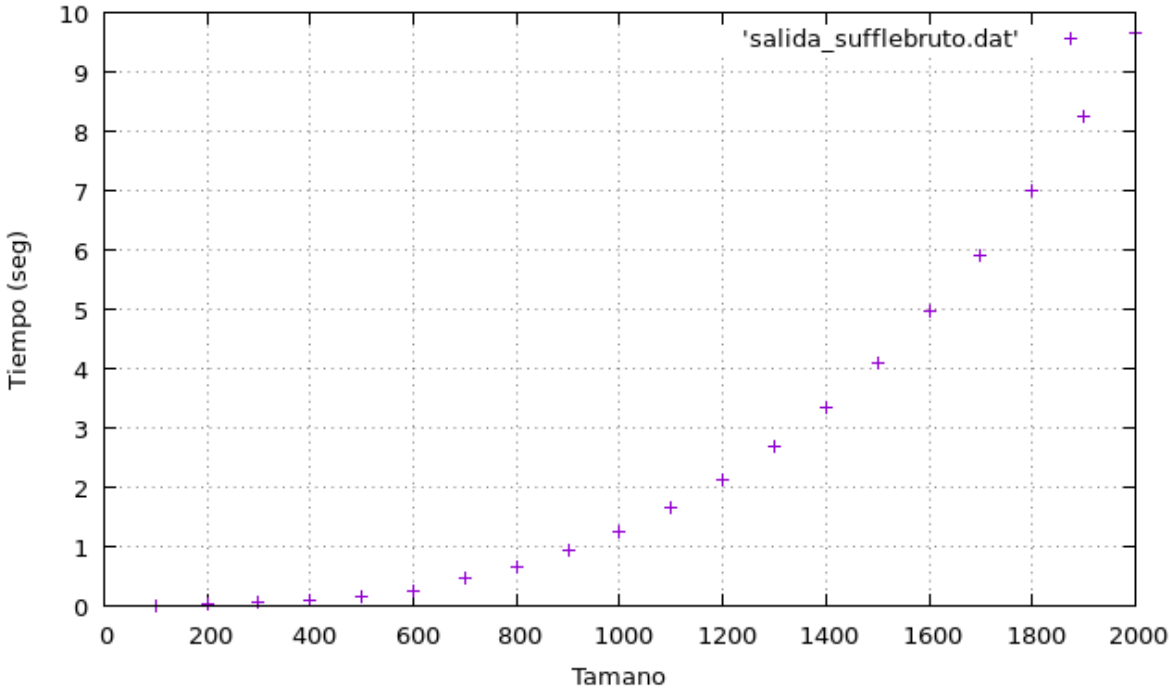
Para llevar a cabo este estudio se han realizado 20 mediciones del tiempo de ejecución de ambos algoritmos para distintos tamaños del vector, cuyos valores se muestran en la siguiente tabla:

Tamaño	Tiempo fuerza bruta	Tiempo DyV
100	0.003858	0.000133
200	0.023227	0.00035
300	0.07214	0.000625
400	0.098578	0.001162
500	0.153957	0.001769
600	0.265914	0.002217
700	0.46807	0.003343
800	0.673748	0.003592
900	0.927304	0.004294
1000	126.801	0.007216
1100	166.557	0.006904
1200	213.125	0.00774
1300	269.623	0.009182
1400	334.837	0.009391
1500	411.086	0.010059
1600	49.806	0.010724
1700	590.697	0.011286
1800	701.022	0.012643
1900	825.211	0.023121
2000	965.571	0.026647

A continuación mostramos las gráficas correspondientes a los datos obtenidos (la primera para el algoritmo de fuerza bruta y la segunda para el divide y vencerás).



Podemos observar la mejora en el tiempo que supone el algoritmo divide y vencerás, el cual llega a ser hasta unas 350 veces más rápido que el de fuerza bruta, para ejecuciones con 2000 elementos.

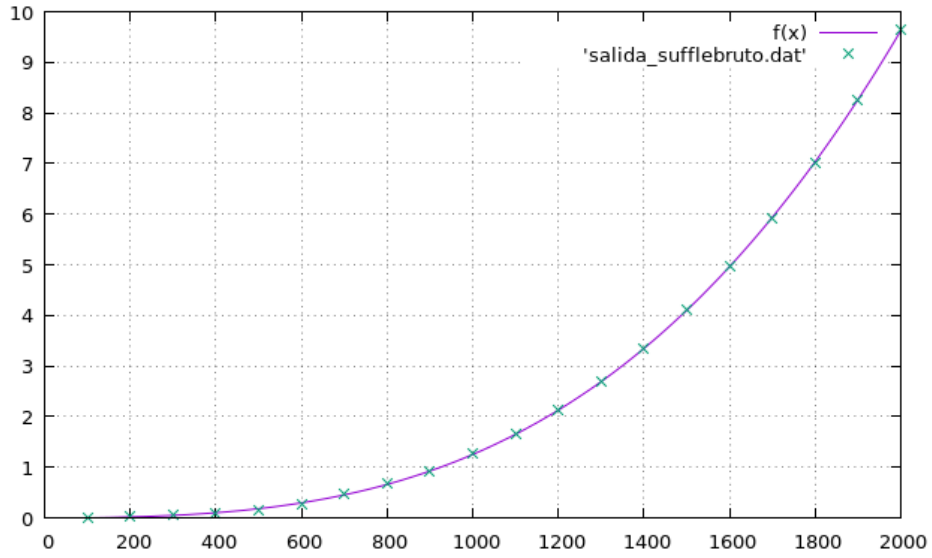


EFICIENCIA HÍBRIDA

Por último, vamos a estudiar la eficiencia híbrida, aproximando los datos obtenidos por una función del tipo que hemos determinado en el estudio de la eficiencia teórica.

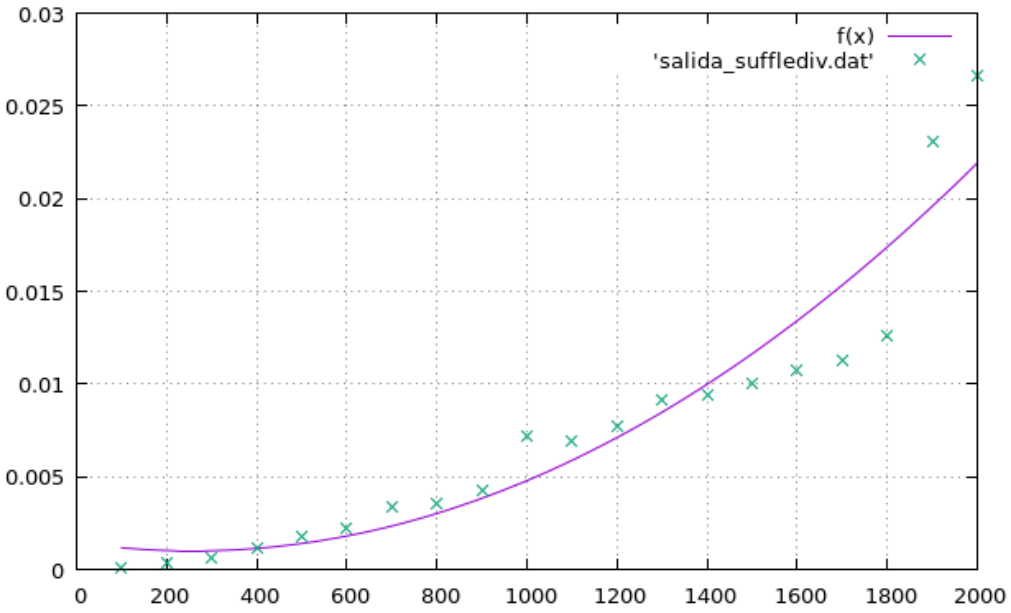
- Algoritmo de fuerza bruta.
  Para este algoritmo se ha usado una función cúbica, que ha resultado ser la siguiente:

$$f(x) = 1.19599 \cdot 10^{-9}x^3 - 3.52047 \cdot 10^{-8}x^2 + 0.000102819x - 0.00896225$$



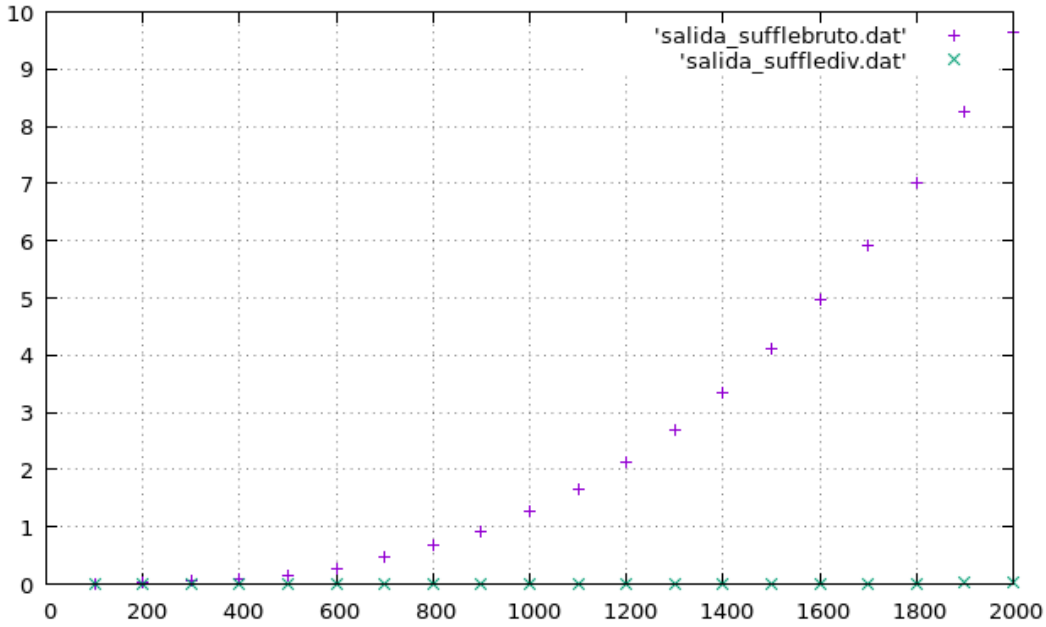
- Algoritmo divide y vencerás.
  En este caso se han aproximado los datos mediante una función cuadrática, dada por:

$$f(x) = 6.89583 \cdot 10^{-9}x^2 - 3.56282 \cdot 10^{-6}x + 0.00146535$$



Como podemos ver, el ajuste es menos preciso en el segundo caso, ya que los tiempos son mucho más pequeños que en el primero y, por tanto, las pequeñas perturbaciones que se producen al medir los tiempos son más notables.

Para finalizar esta sección, incluimos una gráfica con los tiempos de ambos algoritmos para ver la enorme diferencia que hay entre ambos.



Observamos que el tiempo que tarda usando el algoritmo divide y vencerás es tan pequeño que no se aprecia en la gráfica.