UGR

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones
Grado en Ingeniería Informática

Asignatura: Algorítmica

Práctica 2

Autores:
Míriam Mengíbar Rodríguez
Néstor Rodríguez Vico
Ángel Píñar Rivas

Granada, 2 de abril de 2016

Índice:

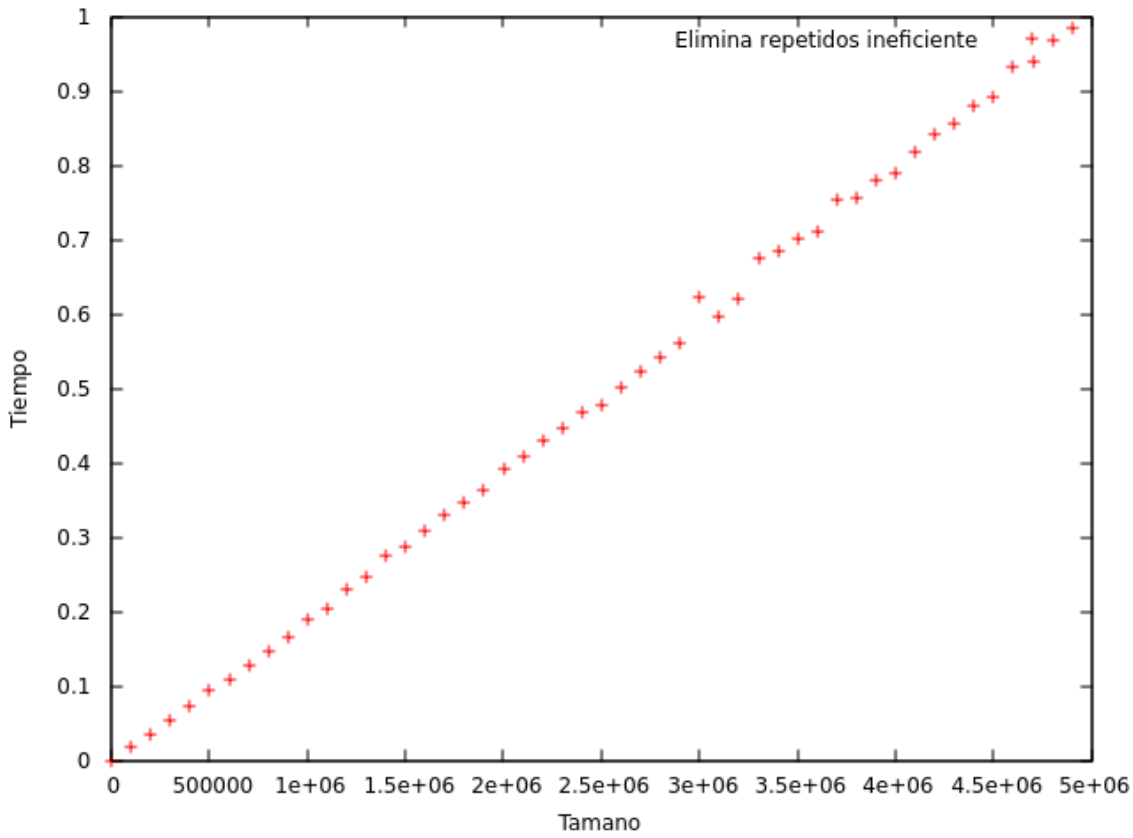
1. Elimina repetidos sin Divide y Vencerás.

Una primera aproximación a la solución de nuestro problema sería la siguiente:

```
vector<int> EliminaRepetidos(int T[], int n){
   quicksort(T, n);
   int auxiliar = T[0];
   vector<int> salida;
   salida.push_back(auxiliar);
   for(int i = 1; i < n; i++){
      if(T[i] != auxiliar){
         salida.push_back(T[i]);
         auxiliar = T[i];
      }
   }

   return salida;
}
```

Lo primero que hacemos es ordenarlo, una vez que esta ordenado el vector, lo vamos recorriendo comparando la posición en la que estamos con la anterior, en caso de que sean distintos, lo metemos en el vector salida, que es nuestro vector sin los elementos repetidos.
Teóricamente podemos ver que este algoritmo es O(nlogn):
Supongamos, en primer lugar, que T1(n) y T2(n) son los tiempos de ejecución de dos segmentos de programa, P1 y P2 . Suponiendo que P1 es la parte de la llamada al algoritmo quicksort y que P2 es el resto de código, sabemos que T1(n) es O(nlogn) y T2(n) es O(n). Entonces el tiempo de ejecución de P1 seguido de P2 , es decir T1(n)+T2(n) es O(max(f(n), g(n))) = O(max(nlogn, n)) = O(nlogn).

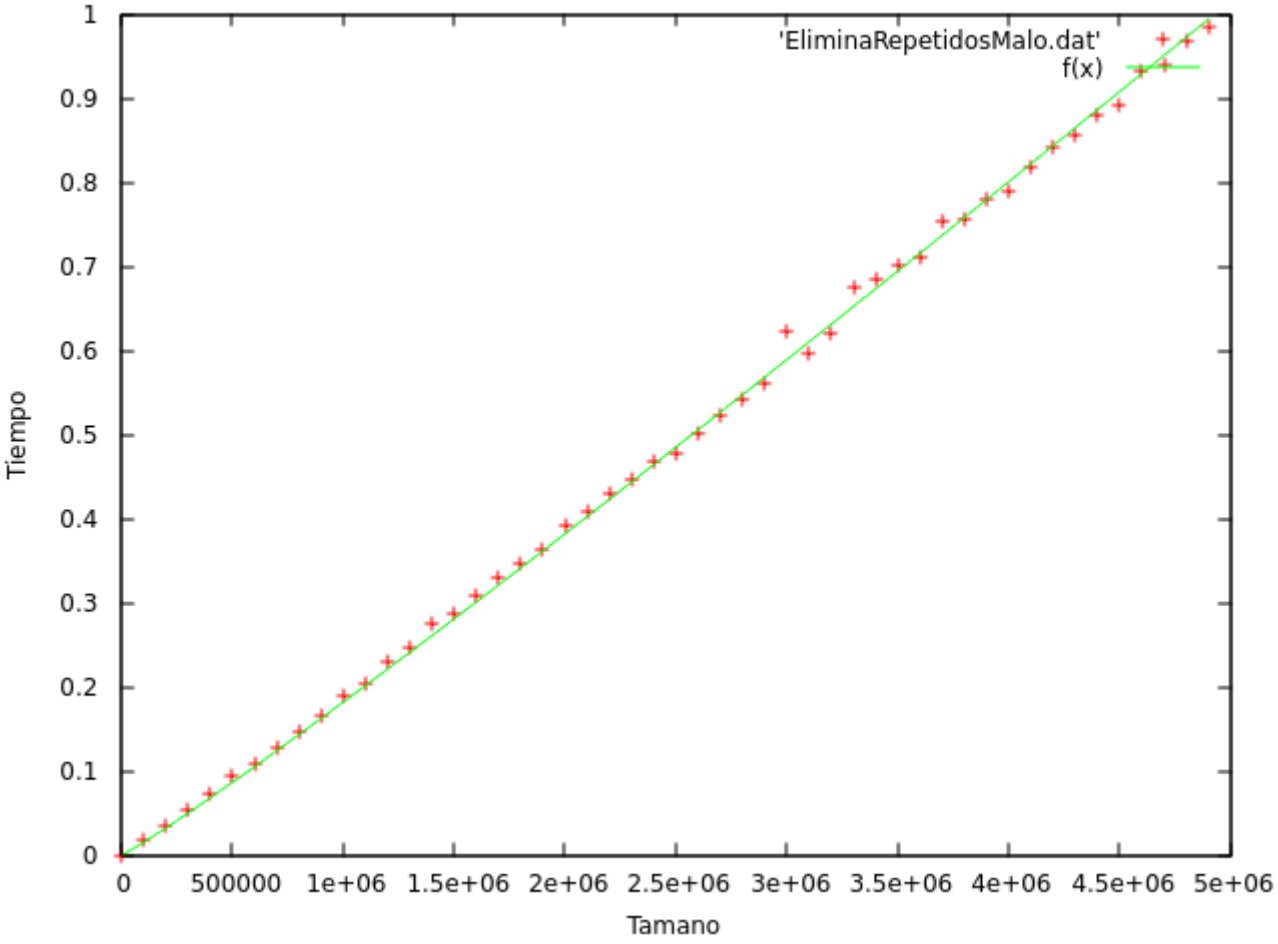
El análisis práctico nos da los siguientes datos:



Para comprobar que se ajusta correctamente con nuestro análisis teórico vamos a ajustar a la función f(x)=a0*x*logx mediante gnuplot.
La constante oculta obtenida es:

a0 = 3.29807e-15 +/- 1.214e-16 (3.681%)

Una vez hemos obtenido la constante, podemos pintar f(x) junto con los datos obtenidos empíricamente:



Así podemos ver que nuestro análisis práctico se ajusta correctamente a nuestro análisis teórico.

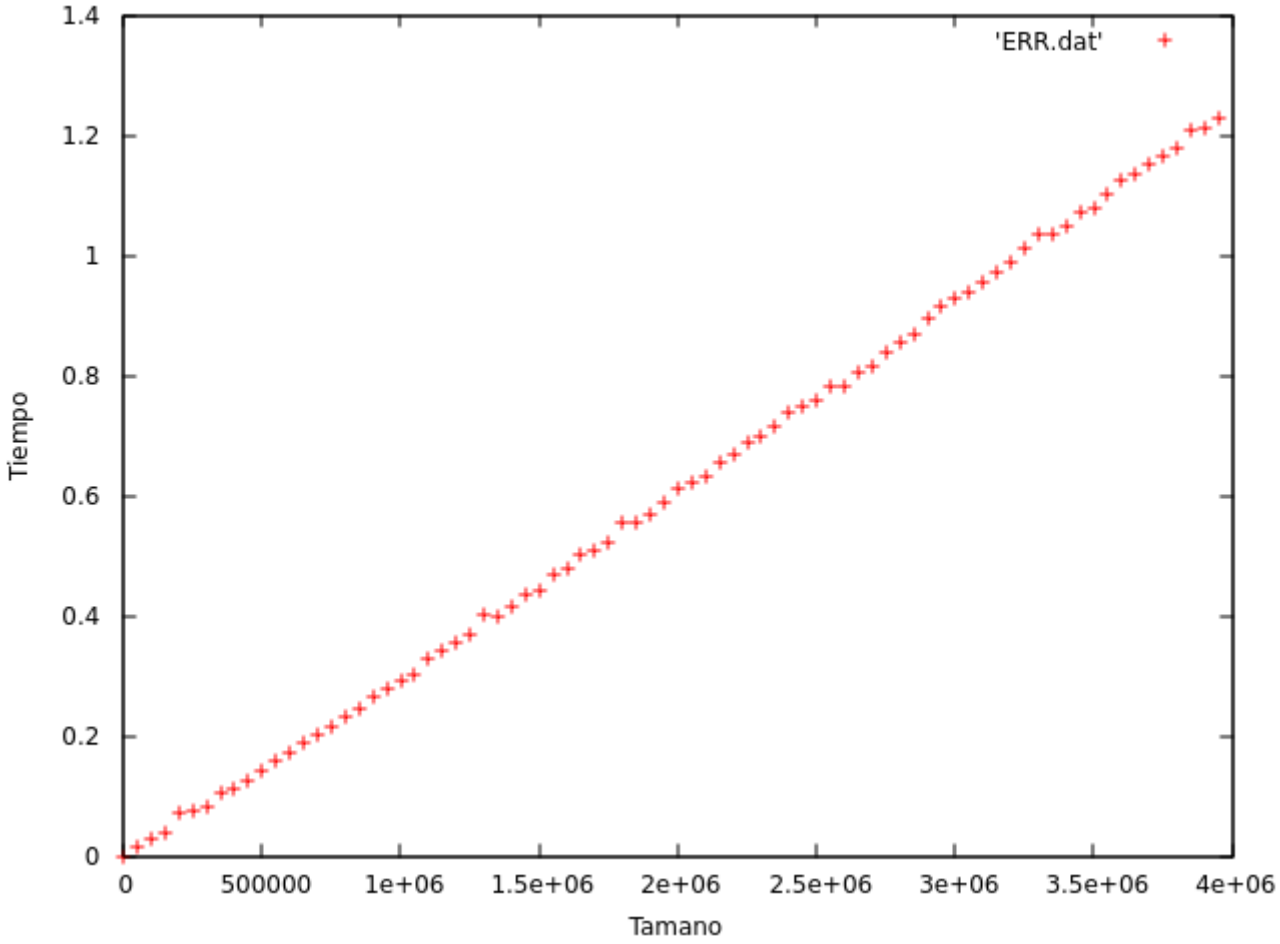
2. Elimina repetidos con Divide y Vencerás.

Para realizar nuestro algoritmo nos hemos basado en el mergesort. Hemos modificado ligeramente el mergesort para que, en el momento en el que introducimos los datos en el array principal, comprobamos a la vez si el dato a introducir coincide con el ultimo dato introducido, para así no añadirlo de nuevo. Con esta idea, ya que el vector está ordenado, hemos logrado eliminar los repetidos de forma correcta.

```
void EliminaRepetidos(int array[], int size);
int merge(int list1[ ], int size1, int list2[ ], int size2, int list3[ ], int size);

int merge(int list1[ ], int size1, int list2[ ], int size2, int list3[ ], int size) {
   int i1, i2, i3;
   if (size1 + size2 > size) {
      return false;
   }
   i1 = 0;
   i2 = 0;
   i3 = 0;
   while (i1 < size1 && i2 < size2) {
      if (list1[i1] < list2[i2]) {
         list3[i3++] = list1[i1++];
      } else {
         list3[i3++] = list2[i2++];
      }
   }
   while (i1 < size1) {
      list3[i3++] = list1[i1++];
   }
   while (i2 < size2) {
      list3[i3++] = list2[i2++];
   }
}

void EliminaRepetidos(int array[], int size, int sin[]) {
   int temp[size];
   int mid, i;

   if (size < 2) {
      return;
   } else {
      mid = size / 2;
      EliminaRepetidos(array, mid, sin);
      EliminaRepetidos(array + mid, size - mid, sin);
      merge(array, mid, array + mid, size - mid, temp, size);
      sin[0] = temp[0];
      int leer = 0;
      for (i = 0; i < size; i++) {
         array[i] = temp[i];
         if (temp[i] != sin[leer]) {
            sin[leer+1] = temp[i];
            leer++;
         }
      }

   }
}
```

La ecuación de recurrencia de nuestro algoritmo es $t(n) = 2t(n/2) + n \Rightarrow t(n) \in O(n\log n)$, ya que descomponemos el trabajo en 2 subproblemas de tamaño la mitad y luego tardamos del O(n) en combinar la solución, por lo tanto nuestro algoritmo es O(nlogn).

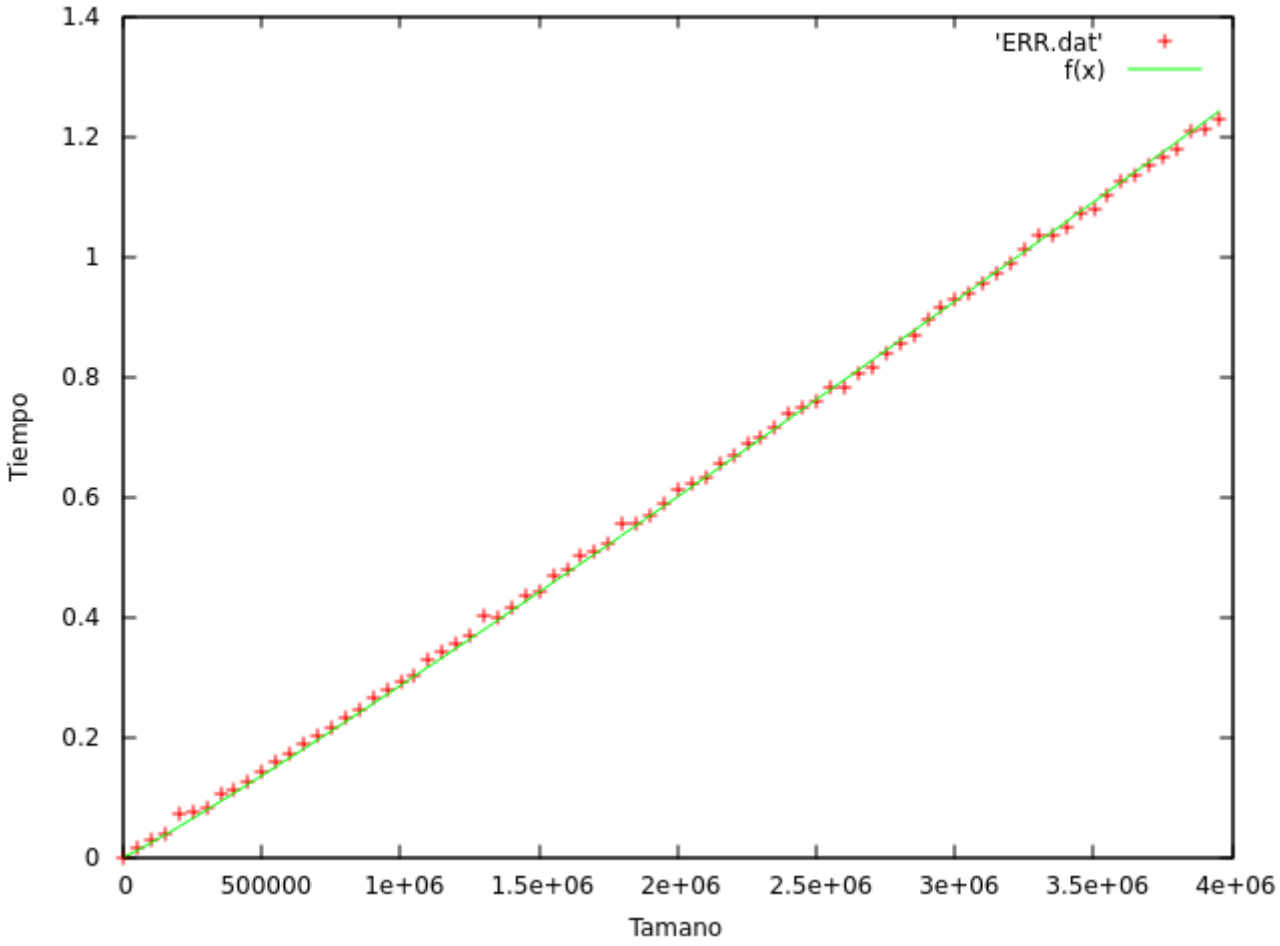
El análisis práctico nos da los siguientes datos:



Para comprobar que se ajusta correctamente con nuestro análisis teórico vamos a ajustar a la función f(x)=a0*x*logx mediante gnuplot.
La constante oculta obtenida es:

a0 = 2.06919e-08 +/- 2.668e-11 (0.129%)

Una vez hemos obtenido la constante, podemos pintar f(x) junto con los datos obtenidos empíricamente:



Así podemos ver que nuestro análisis práctico se ajusta correctamente a nuestro análisis teórico.