Algorítmica

Práctica 1: Análisis de Eficiencia de Algoritmos

César Muñoz Reinoso

Curso 2018-2019

17 de marzo de 2019

Índice

1. Eficiencia teórica

1.1. Algoritmo Pivotar

```
1  int pivotar (double *v , const int ini , const int fin) {
2    double pivote = v[ini], aux ;
3    int i = ini + 1, j = fin ;
4
5    while (i<=j){
6      while (v[i] < pivote && i <= j) i++;
7      while (v[j] >= pivote && j >= i) j--;
8
9      if (i<j){
10       aux = v[i];
11       v[i] = v[j];
12       v[j] = aux;
13     }
14   }
15
16   if (j > ini) {
17     v[ini] = v[j] ;
18     v[j] = pivote ;
19   }
20   return j ;
21 }
```

Para hallar la eficiencia teórica, vamos a acotar por una constante K a la porción de codigo de las lineas 8-14 y la porción de codigo de las lineas 15-21 por otra constante M. El codigo que se encuentra dentro de un bucle while se ejecutará fin - ini + 1 veces. Dentro del while principal se encuentran 2 while, no anidados, por lo que tendran cada uno una eficiencia O(1). Tenemos entonces que la eficiencia de este algoritmo solo esta condicionada por el bucle while, por lo que es O(n).

1.2. Algoritmo Busqueda

```
1  int Busqueda (int *v ,  int n ,  int elem) {
2
3    int inicio, fin, centro;
4
5    inicio = 0;
6    fin = n-1;
7    centro = (inicio + fin)/2;
8    while ((inicio <= fin) && (v[centro] != elem)){
9      if (elem < v[centro])
10       fin = centro - 1;
11     else
```

```
12        inicio = centro + 1;
13
14      centro = (inicio + fin)/2;
15    }
16    if (inicio > fin)
17      return - 1;
18
19
20    return centro;
21  }
```

Para este algoritmo, podemos ver que dentro del bucle while la eficiencia es O(1), y observamos que el algoritmo va reduciendo el vector de busqueda a la mitad en cada iteración del bucle while. Tenemos también un algoritmo condicionado por el bucle while, de orden O($log_2 n$)

1.3. Algoritmo EliminaRepetidos

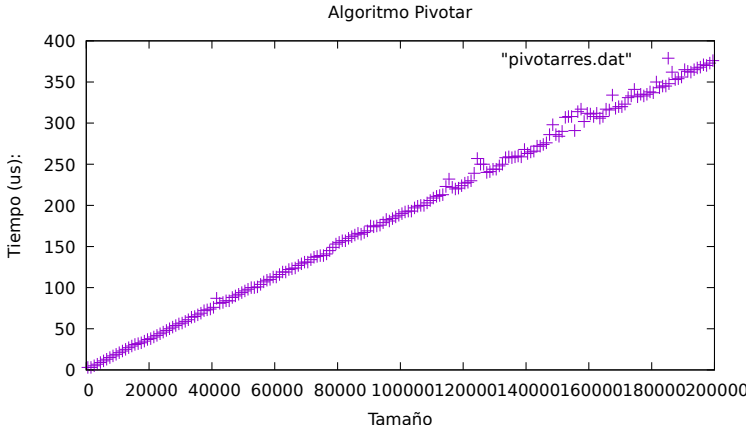
```
1  void EliminaRepetidos(double original[], int &nOriginal) {
2
3    int i, j, k;
4
5    for (i = 0; i < nOriginal; i++) {
6      j = i + 1;
7      do{
8        if(original[j] == original[i]){
9          for (k = j + 1; k < nOriginal; k++)
10           original[k - 1] = original[k];
11         nOriginal--;
12       }else
13       j++;
14     } while (j < nOriginal) ;
15   }
16 }
```

Este algoritmo presenta un bucle for anidado con un bucle while, donde el for y el while iteran nOriginal veces. El bucle for dentro del while es de orden O(1). Tenemos un orden O(n^2)

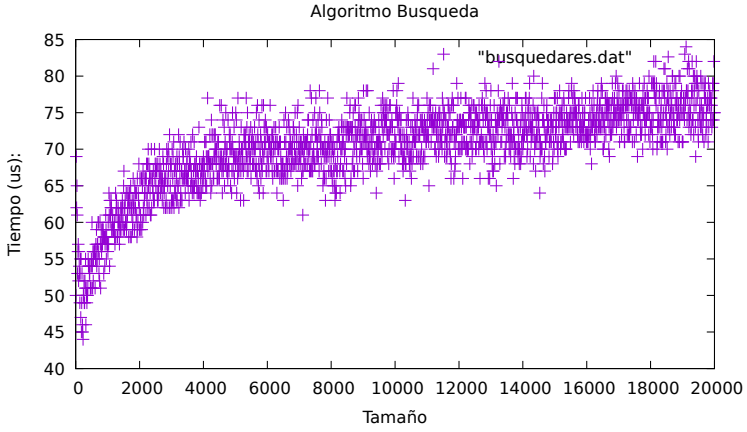
2. Eficiencia empírica

Tomando los valores adecuados para cada algoritmo, he realizado la ejecuccion de los mismos para corroborar empíricamente que dichos algoritmos presentan el orden de eficiencia antes calculado.

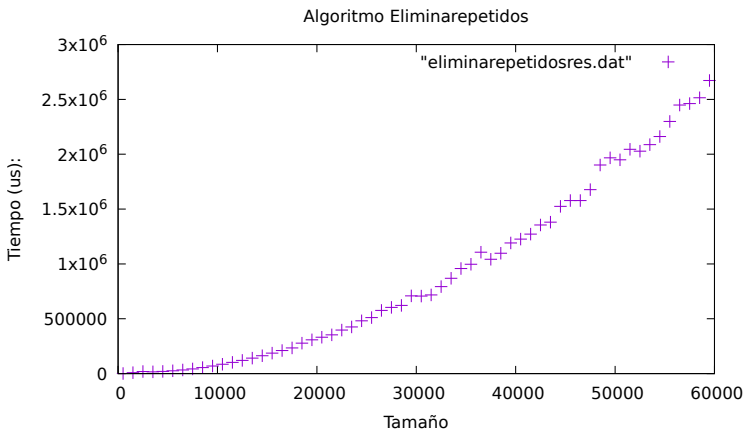
2.1. Algoritmo Pivotar



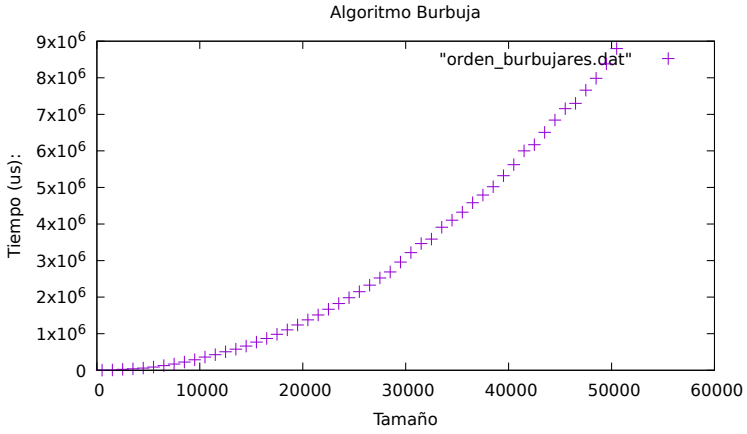
2.2. Algoritmo Busqueda



2.3. Algoritmo EliminaRepetidos



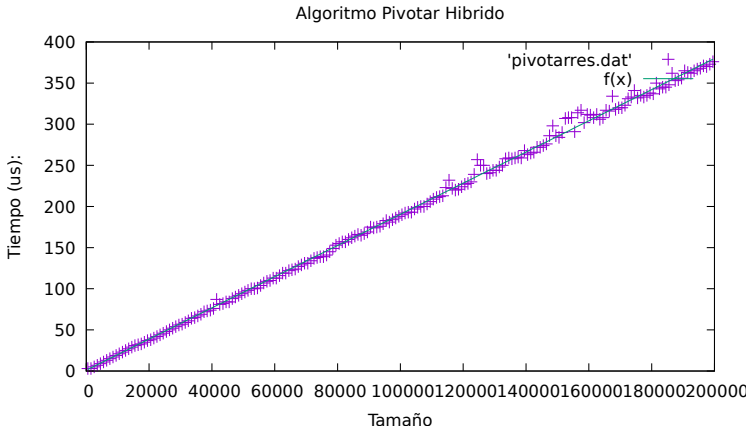
2.4. Algoritmo Burbuja



3. Eficiencia híbrida

Realizando una regresión con su correspondiente función, tenemos los resultados:

3.1. Algoritmo Pivotar

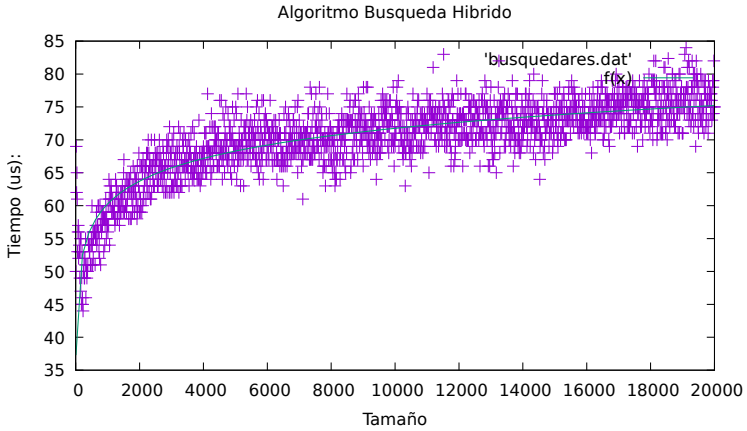


Hemos ajustado la nube 200 puntos en el intervalo [0,200000] con la funcion $f(x) = a0 * x + a1$. Los valores de las constantes ocultas son :

a0 = 0.00189478

a1 = 0.999905

3.2. Algoritmo Busqueda



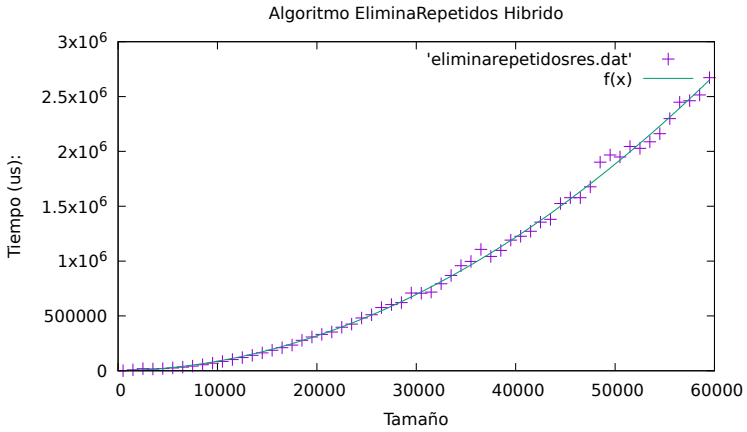
Hemos ajustado la nube de 2000 puntos en el intervalo [0,20000] con la funcion $f(x) = a0 *$

$log_2(x) + a1$. Los valores de las constantes ocultas son :

a0 = 3.4584

a1 = 25.7965

3.3. Algoritmo EliminaRepetidos



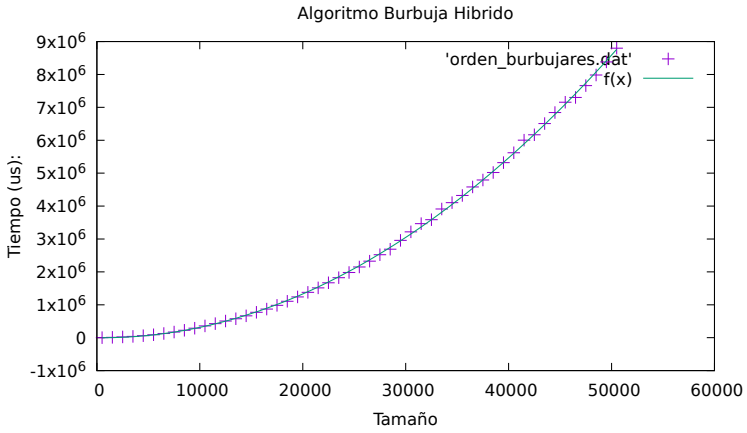
Hemos ajustado la nube 60 puntos en el intervalo [0,60000] con la funcion $f(x) = a0 * x^2 + a1 * x + a2$. Los valores de las constantes ocultas son :

a0 = 0.00072425

a1 = 1.46784

a2 = 0.94093

3.4. Algoritmo Burbuja



Hemos ajustado la nube 51 puntos en el intervalo [0,60000] con la funcion $f(x) = a0 * x^2 + a1 * x + a2$. Los valores de las constantes ocultas son :

a0 = 0.00350886

a1 = -3.69285

a2 = -5.44292