Práctica 2: Algoritmos Divide y Vencerás

Alejandro Rodríguez López
alexrodriguezlop@correo.ugr.es

15 de abril de 2018

Índice general

Capítulo 0

Entorno y automatización

0.1. Hardware

0.1.1. CPU

Datos obtenidos mediante la ejecución de `$ lscpu` en la máquina virtual donde han sido realizadas las ejecuciones.
Todos los ejercicios han sido ejecutados en dicho hardware.

Máquina virtual:
La máquina virtual tiene asignados dos núcleos del procesador.
Arquitectura: `x86_64`
modo(s) de operación de las CPUs: `32-bit`, `64-bit`
Orden de bytes: Little Endian
CPU(s): 2
CPU MHz: 2594.004
Caché L1d: 32K
Caché L1i: 32K
Caché L2: 256K
Caché L3: 6144K

Sistema Anfitrión:
El sistema anfitrión dispone de un microprocesador I7 4720HQ 2.60GHz

0.1.2. Memoria ram

Máquina virtual:
Tiene asignados 2GB de memoria RAM

Sistema Anfitrión:
Dispone de 16GB de memoria RAM

0.2. Software

0.2.1. Compilador

G++ versión 4.8

Opciones de compilado: g++ fichero.cpp o ejecutable -std=gnu++0x

0.2.2. Sistema operativo

Ubuntu 14.04.5 LTS

0.3. Medición usando chrono

Todas las mediciones se han realizado usando la libreria chrono de STL.

```
int main(int argc, char * argv[])
{
  if (argc != 2)
    {
      cerr << "Formato " << argv[0] << " <num_elem>" << endl;
      return -1;
    }

  int n = atoi(argv[1]);

  int * T = new int[n];

  assert(T);

  srandom(time(0));

  for (int i = 0; i < n; i++)
    {
      T[i] = random();
    };

  high_resolution_clock::time_point tantes, tdespues;
  duration<double> transcurrido;

  tantes = high_resolution_clock::now();

  burbuja(T, n);

  tdespues = high_resolution_clock::now();

  transcurrido = duration_cast<duration<double>>(tdespues - tantes);
  cout << n << " "  << transcurrido.count() << endl;

  delete [] T;

  return 0;
};
```

Capítulo 1

Análisis de eficiencia empírica e híbrida

Se han llevado a cabo determinadas mediciones sobre los distintos algoritmos, a continuación se detallan los resultados obtenidos de dichas mediciones para cada uno de los algoritmos.

1.1. Algoritmos basados en divide y vencerás

1.1.1. Vectores sin elementos repetidos

Eficiencia empírica

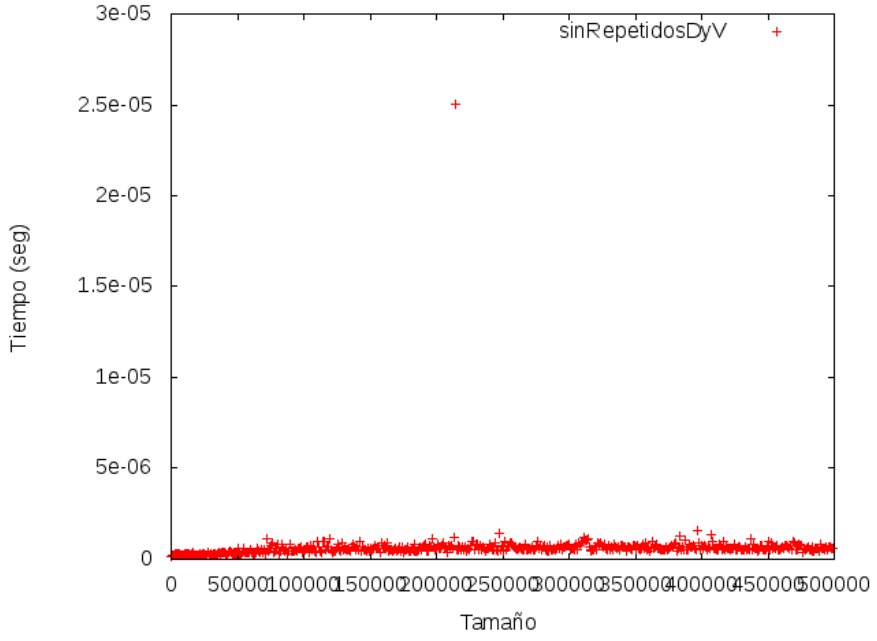


Figura 1.1: Como podemos observar el algoritmo que emplea la técnica divide y vencerás sobre vectores previamente ordenados de forma no decraciente nos da como resultado unos tiempos muy buenos.

Eficiencia híbrida

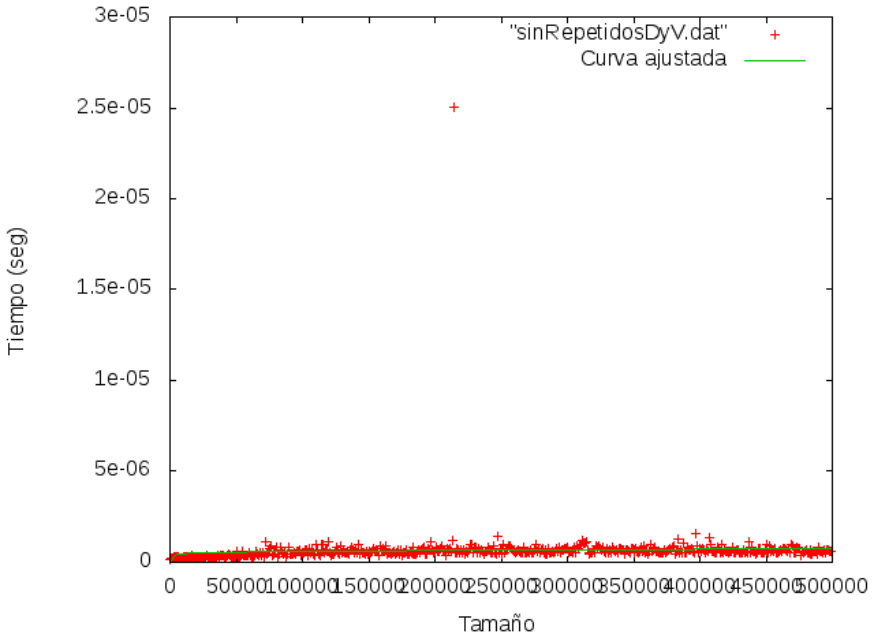


Figura 1.2: La eficiencia del algoritmo es de $\mathbf{O(\log(n))}$, la gráfica coincide con el ajuste.

```
*******************************************************************************
Sat Apr 14 20:27:52 2018


FIT:    data read from 'sinRepetidosDyV.dat'
        format = z
        #datapoints = 500
        residuals are weighted equally (unit weight)

function used for fitting: f(x)
fitted parameters initialized with current variable values



 Iteration 0
 WSSR        : 4.15451e+13       delta(WSSR)/WSSR   : 0
 delta(WSSR) : 0                 limit for stopping : 1e-05
 lambda          : 203819

initial set of free parameter values

a               = 1
b               = 1
```

```
After 8 iterations the fit converged.
final sum of squares of residuals : 6.09826e-10
rel. change during last iteration : -7.59761e-09

degrees of freedom    (FIT_NDF)                        : 498
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 1.10659e-06
variance of residuals (reduced chisquare) = WSSR/ndf   : 1.22455e-12

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a               = 1.54535e-13      +/- 3.916e-13    (253.4%)
b               = 5.06389e-08      +/- 9.287e-09    (18.34%)


correlation matrix of the fit parameters:

               a      b
a               1.000
b              -0.899  1.000
```

1.1.2. Vectores con elementos repetidos

Eficiencia empírica

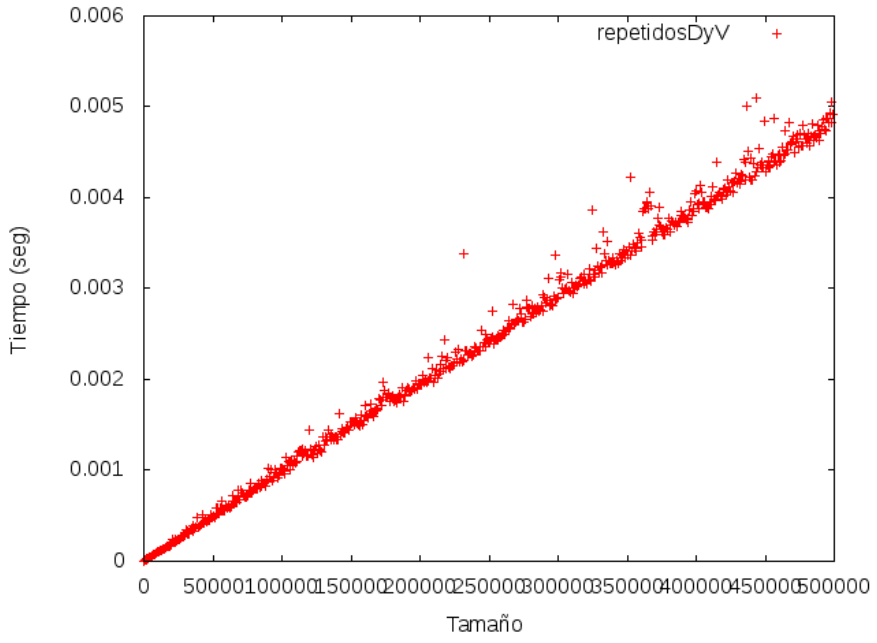
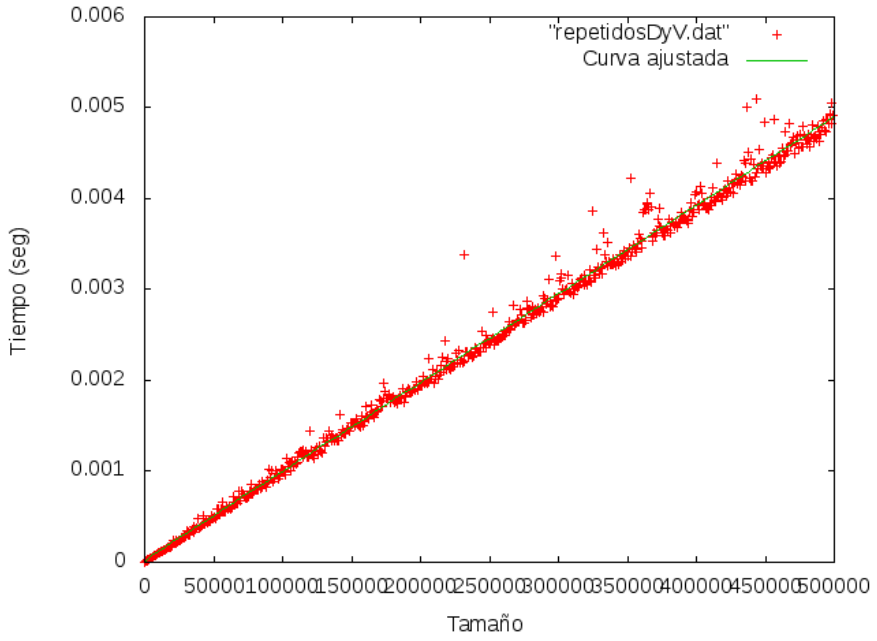


Figura 1.3: La técnica divide vencerás no nos aporta un comportamiento tan rápido en cuando se trata de un vector sin los elementos ordenados.

Eficiencia híbrida



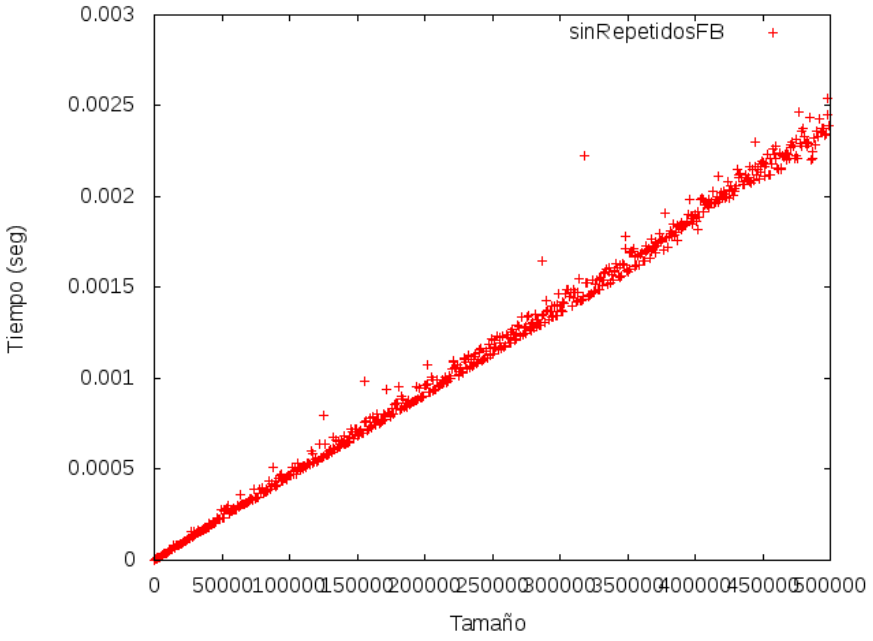

```
******************************************************************************
Sat Apr 14 20:28:20 2018


FIT:    data read from 'repetidosDyV.dat'
        format = z
        #datapoints = 625
        residuals are weighted equally (unit weight)

function used for fitting: f(x)
fitted parameters initialized with current variable values



 Iteration 0
 WSSR        : 5.19626e+13       delta(WSSR)/WSSR   : 0
 delta(WSSR) : 0                 limit for stopping : 1e-05
 lambda          : 203880

initial set of free parameter values

a               = 1
b               = 1

After 8 iterations the fit converged.
final sum of squares of residuals : 7.94266e-06
rel. change during last iteration : -1.71056e-13
```

```
degrees of freedom    (FIT_NDF)                        : 623
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.000112912
variance of residuals (reduced chisquare) = WSSR/ndf   : 1.27491e-08

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a                = 9.74192e-09     +/- 3.573e-11    (0.3667%)
b                = 2.45396e-06     +/- 8.473e-07    (34.53%)


correlation matrix of the fit parameters:

                a      b
a                1.000
b               -0.899  1.000
```

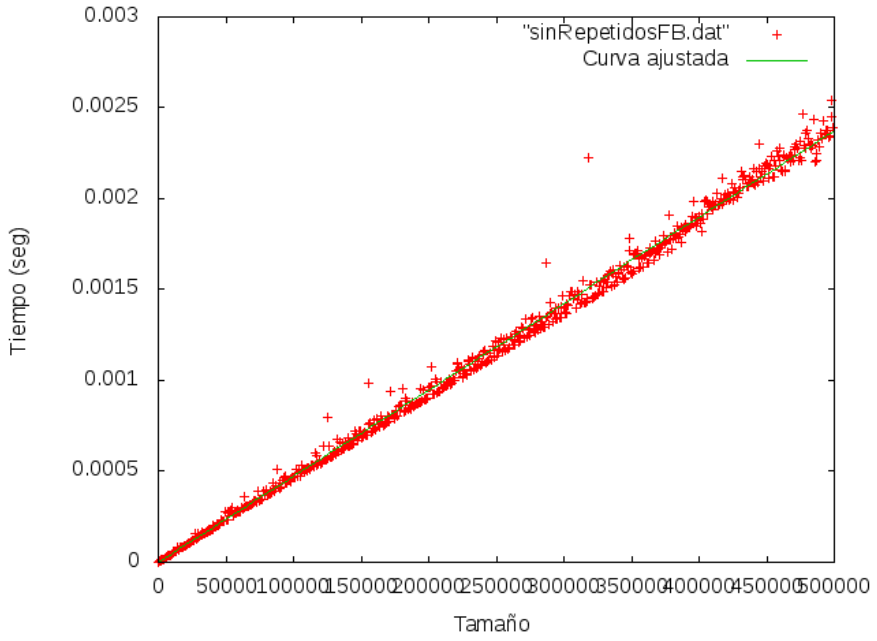
1.2. Algoritmos basados en fuerza bruta

1.2.1. Vectores sin elementos repetidos

Eficiencia empírica



Eficiencia híbrida

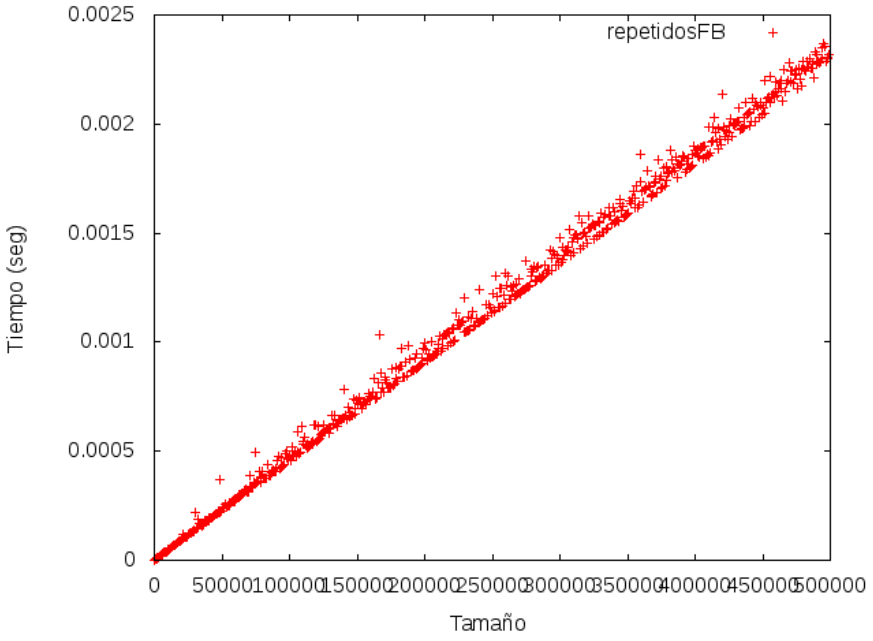



```
*******************************************************************************
Sat Apr 14 20:28:10 2018


FIT:    data read from 'sinRepetidosFB.dat'
        format = z
        #datapoints = 625
        residuals are weighted equally (unit weight)

function used for fitting: f(x)
fitted parameters initialized with current variable values



 Iteration 0
 WSSR        : 5.19587e+13       delta(WSSR)/WSSR   : 0
 delta(WSSR) : 0                 limit for stopping : 1e-05
 lambda          : 288330

initial set of free parameter values

a               = 1

After 4 iterations the fit converged.
final sum of squares of residuals : 1.68059e-06
rel. change during last iteration : -5.17026e-10
```

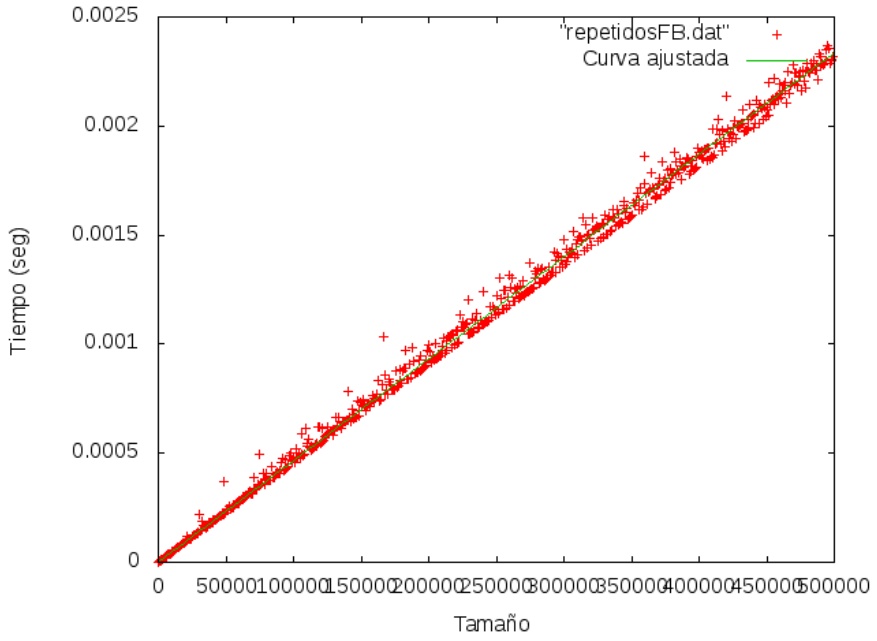
```
degrees of freedom    (FIT_NDF)                        : 624
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 5.18966e-05
variance of residuals (reduced chisquare) = WSSR/ndf   : 2.69326e-09

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a               = 4.74345e-09      +/- 7.2e-12      (0.1518%)


correlation matrix of the fit parameters:

               a
a               1.000
```

1.2.2. Vectores con elementos repetidos

Eficiencia empírica



Eficiencia híbrida




```
******************************************************************************
Sat Apr 14 20:28:27 2018


FIT:    data read from 'repetidosFB.dat'
        format = z
        #datapoints = 625
        residuals are weighted equally (unit weight)

function used for fitting: f(x)
fitted parameters initialized with current variable values



 Iteration 0
 WSSR        : 5.19587e+13       delta(WSSR)/WSSR   : 0
 delta(WSSR) : 0                 limit for stopping : 1e-05
 lambda          : 288330

initial set of free parameter values

a               = 1

After 4 iterations the fit converged.
final sum of squares of residuals : 8.85731e-07
rel. change during last iteration : -9.8101e-10
```

```
degrees of freedom    (FIT_NDF)                        : 624
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 3.76755e-05
variance of residuals (reduced chisquare) = WSSR/ndf   : 1.41944e-09

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a               = 4.67223e-09      +/- 5.227e-12    (0.1119%)


correlation matrix of the fit parameters:

               a
a               1.000
```

Como hemos podido observar, los resultados de las pruebas de eficiencia híbridas determinan que los resultados empíricos se ajustan correctamente con los teóricos mostrando una total semejanza.

Capítulo 2

Comparación de eficiencias

2.0.1. Algoritmos basados en fuerza bruta

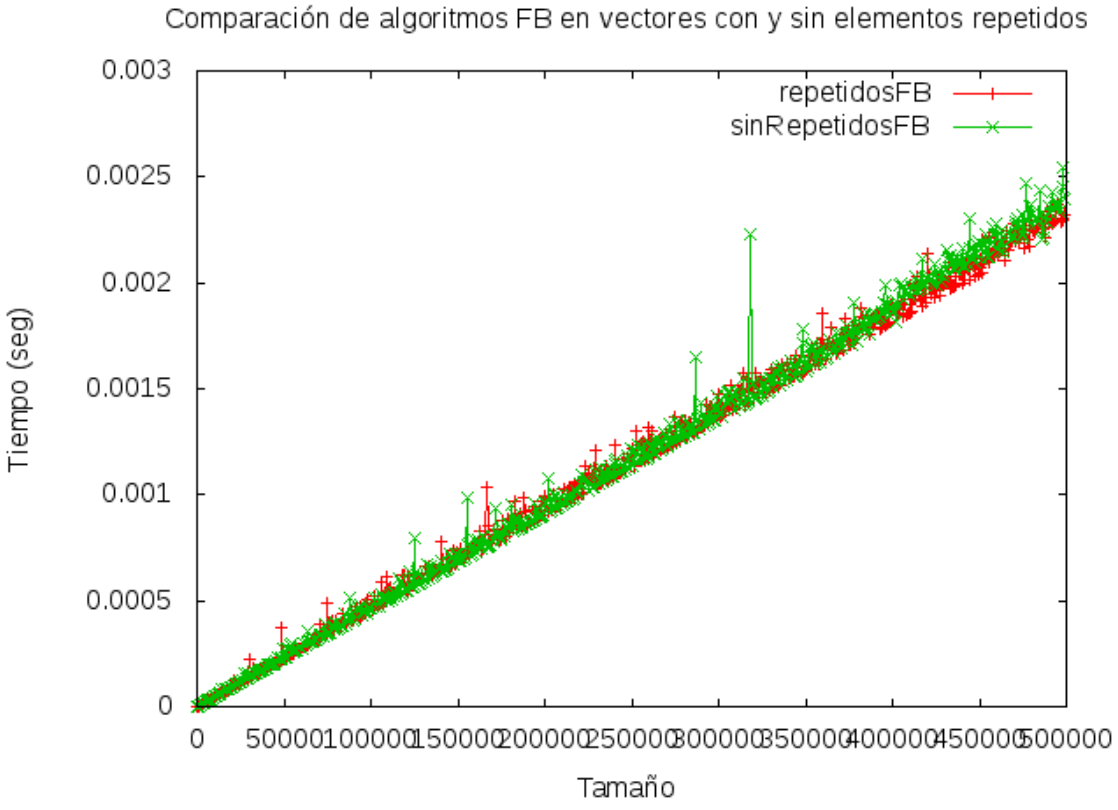


Figura 2.1: Como podemos observar en la gráfica, la eficiencia de los algoritmos no se ve alterada frente a la posibilidad de encontrar elementos repetidos dentro del vector. El algoritmo soluciona el problema en ambos escenarios con la misma eficiencia.

2.0.2. Algoritmos basados en divide y vencerás

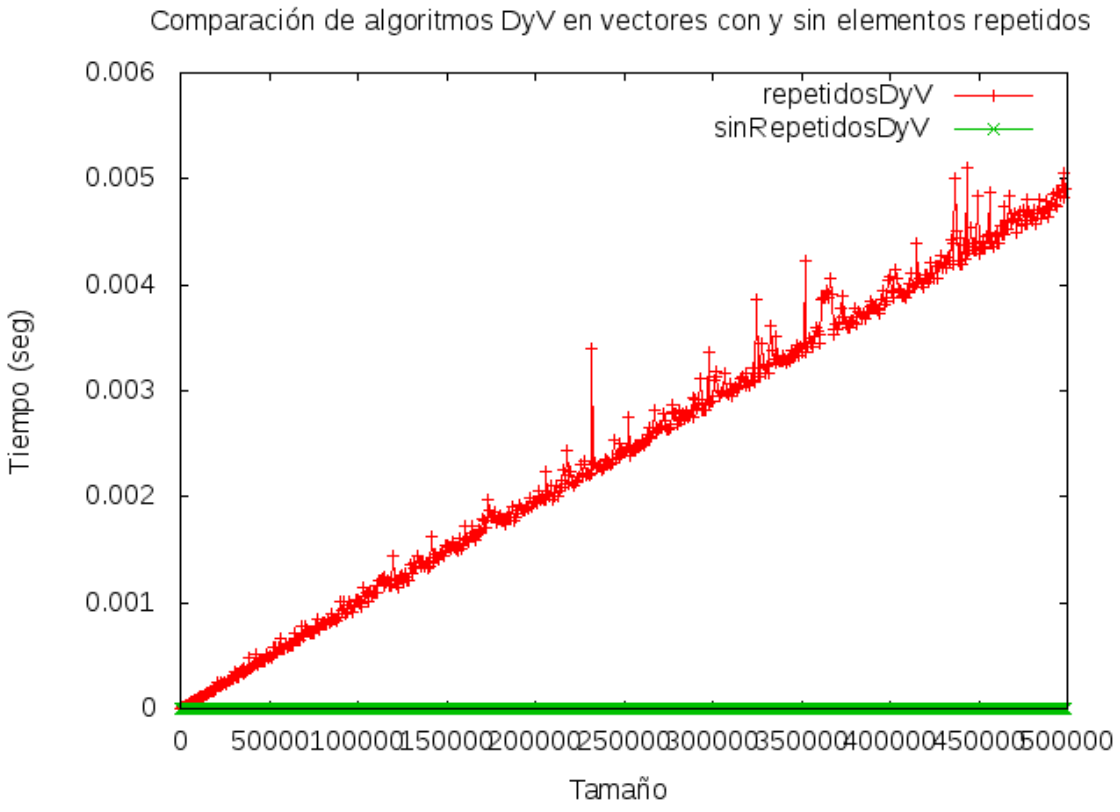


Figura 2.2: Como podemos observar el algoritmo desarrollado haciendo uso de la técnica divide y vencerás ve comprometida su eficiencia en los escenarios donde el vector presenta elementos repetidos, quedando claramente visible las diferencias de eficiencia en ambos casos.

2.0.3. Algoritmos aplicados sobre un vector con elementos repetidos

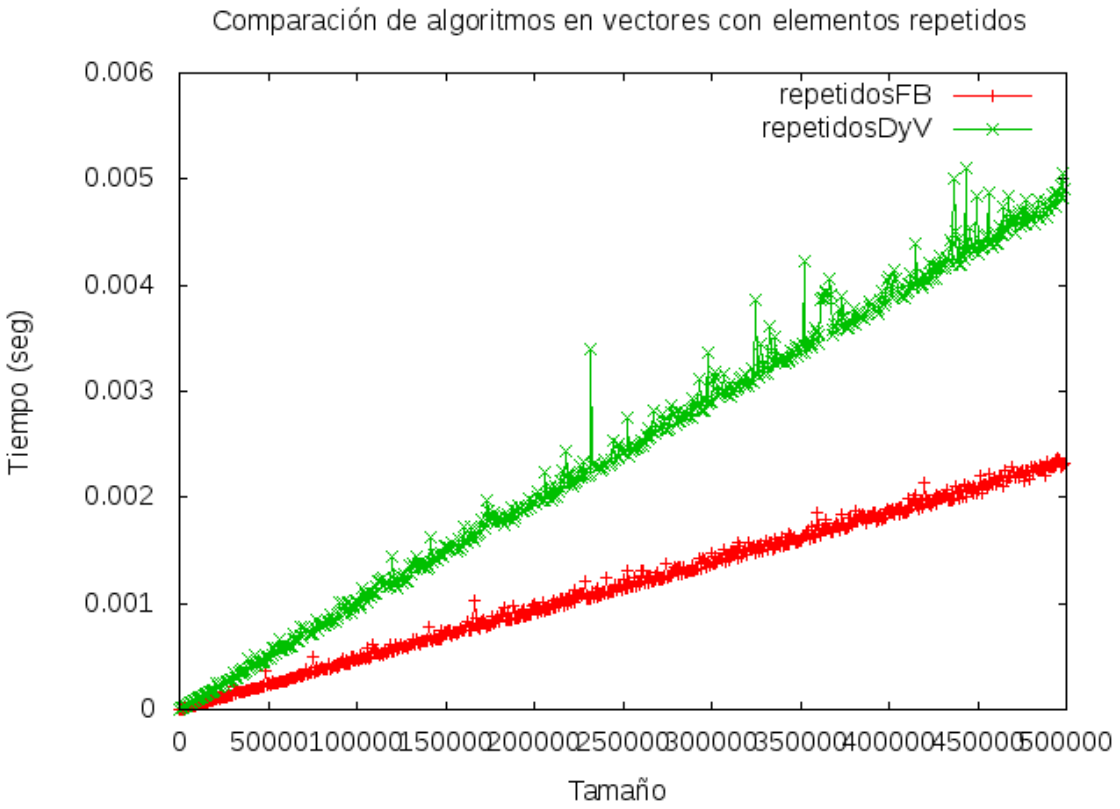


Figura 2.3: Como podemos ver ante un vector con elementos repetidos el algoritmo de fuerza bruta nos da mejores resultados.

2.0.4. Algoritmos aplicados sobre un vector sin elementos repetidos

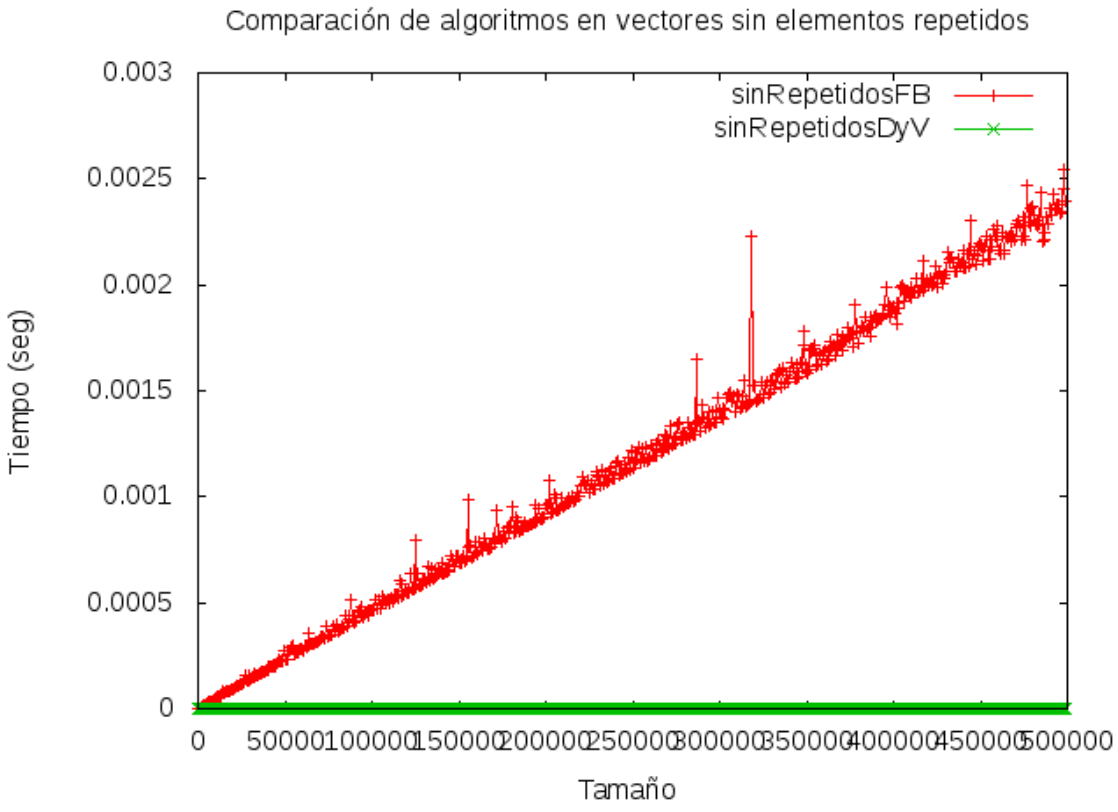


Figura 2.4: Al contrario que en la figura anterior, cuando el escenario presenta un vector sin elementos repetido el algoritmo desarrollado con la técnica divide y vencerás deja clara su eficiencia, siendo esta bastante mejor que la aportada por el algoritmo de fuerza bruta.

2.0.5. Comparativa global

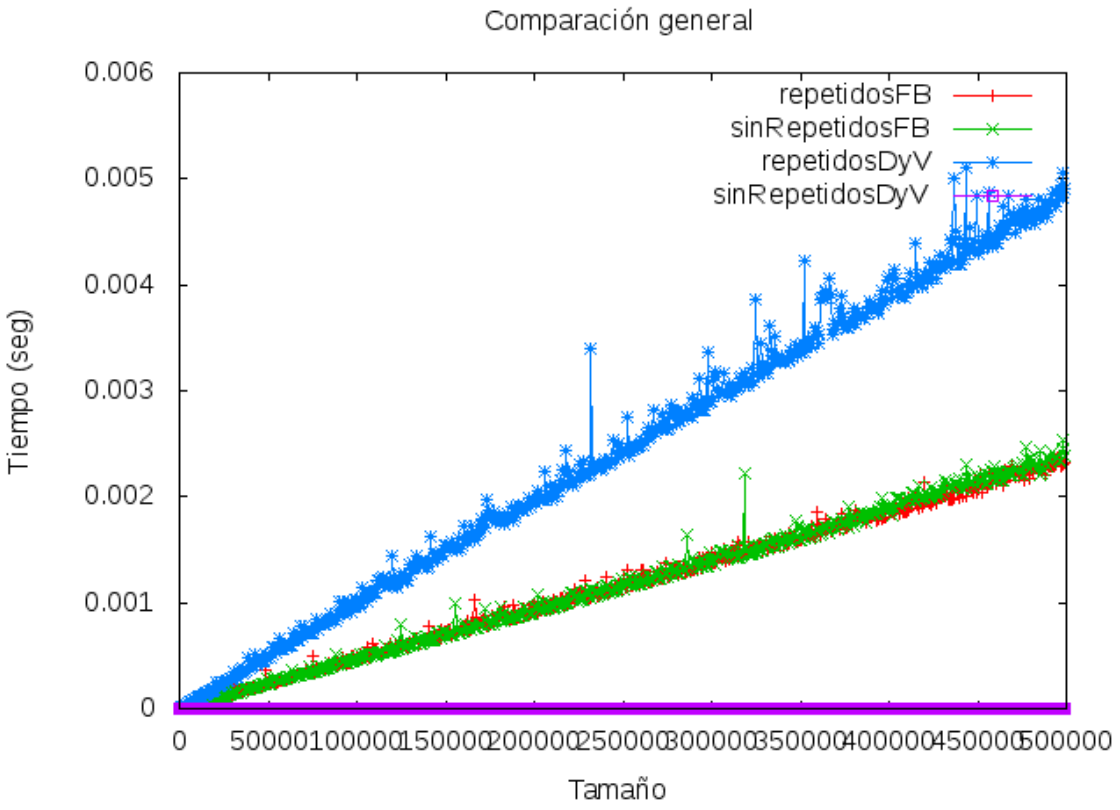


Figura 2.5: En esta figura podemos ver una comparativa global, donde se ve claramente las diferencias que existen a nivel de eficiencia entre las distintas técnicas aplicadas en la resolución del problema.

Capítulo 3

Código fuente

3.1. Algoritmos basados en divide y vencerás

3.1.1. Problema en un vector sin elementos repetidos

"Divide y vencerás sin elementos repetidos"

```
int Divide_y_venceras(vector<int> &v, int inicio, int fin) {
        int valor = -1;
        int mitad = (inicio + fin)/2;

        if(inicio <= fin){
                if(v[mitad]==mitad){
                        valor = mitad;
                }
                else{
                        if(v[mitad] > mitad)
                                valor =  Divide_y_venceras(v, inicio, mitad-1);
                        else
                                valor =  Divide_y_venceras(v, mitad+1, fin);
                }
        }
        return valor;
}
```

Figura 3.1: Solución planteada para resolver el problema usando la técnica divide y vencerás en vectores sin elementos repetidos.

3.1.2. Problema en un vector con elementos repetidos

"Divide y vencerás elementos repetidos"

```
int Divide_y_venceras(vector<int> &v, int inicio, int fin) {
        int valor = -1;
        int mitad = (inicio + fin)/2;

        if(inicio <= fin){
                if(v[mitad]==mitad){
                        valor = mitad;
                }
                else{
                        valor =  Divide_y_venceras(v, inicio, mitad-1);
                        if(valor == -1)
                                valor =  Divide_y_venceras(v, mitad+1, fin);
                }
        }
        return valor;
}
```

Figura 3.2: Solución planteada para resolver el problema usando la técnica divide y vencerás en vectores con elementos repetidos.

3.2. Algoritmos basados en fuerza bruta

"Fuerza bruta"

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

Figura 3.3: Solución planteada para resolver ambos problemas usando la técnica de fuerza bruta en vectores con y sin elementos repetidos.

Capítulo 4

Conclusiones

Como hemos podido apreciar en los datos mostrados a lo largo de este documento, hemos podido dar solución a las dos vertientes del problema haciendo uso de la técnica divide y vencerás. La complejidad de estas soluciones no es demasiado alta, aunque cabe destacar que la solución diseñada empleando fuerza bruta posee un nivel de dificultad bastante inferior frente a las soluciones planteadas haciendo uso de la técnica divide y vencerás. La solución

planteada para resolver el problema con vectores sin elementos repetidos mediante la aplicación de divide y vencerás no proporciona una solución valida al segundo escenario que se nos presenta, con lo cual para poder resolver el problema con vectores que permitan elementos repetidos es necesaria la modificación del algoritmo que da solución al primer problema adaptándolo a esté último. Aún cuando la adaptación permite aportar una solución correcta

al problema deja en evidencia que no merece la pena aplicar la técnica divide y vencerás para dicha solución, ya que la eficiencia disminuye notablemente. y la complejidad de la solución aumenta frente a la solución basada en fuerza bruta.