Estructura de datos (2016-2017)
Grado en Ingeniería Informática
Universidad de Granada

Práctica I: Eficiencia de los algoritmos

Elena María Gómez Ríos

16 de octubre de 2016

Índice

1. Realizar el análisis de eficiencia cuando consideramos el código en ocurrencias.cpp.

Vamos a obtener la eficiencia teórica del código de `ocurrencias.cpp`. Como se puede observar la mayor parte del tiempo de ejecución se emplea en el método `contar_hasta`, el cual es de orden $0(n)$.

```
int contar_hasta( vector<string> & V, int ini, int fin, string & s) {
   int cuantos = 0;
   for (int i=ini; i< fin ; i++)
      if (V[i]==s) {
         cuantos ++;
      }
   return cuantos;
}
```

Para realizar el cálculo empírico de la eficiencia del algoritmo, hemos ejecutado el programa en un ordenador cuyas prestaciones son: procesador Intel Core i7 con 6GB de RAM, y el compilador empleado es 'g++'.

1.1. Considerando como entrada el texto del fichero quijote.txt.

Compilamos el código `ocurrencias.cpp` leyendo y contando las palabras del fichero `quijote.txt`:

```
g++ -std=c++11 -o ./bin/ocurrencias ./src/ocurrencias.cpp
```

Guardamos la salida de la ejecución en un fichero llamado `ocurrenciasQuijote.txt`, donde la primera columna representa el tamaño del conjunto de palabras y la segunda el tiempo necesario para buscar la palabra "hidalgo":

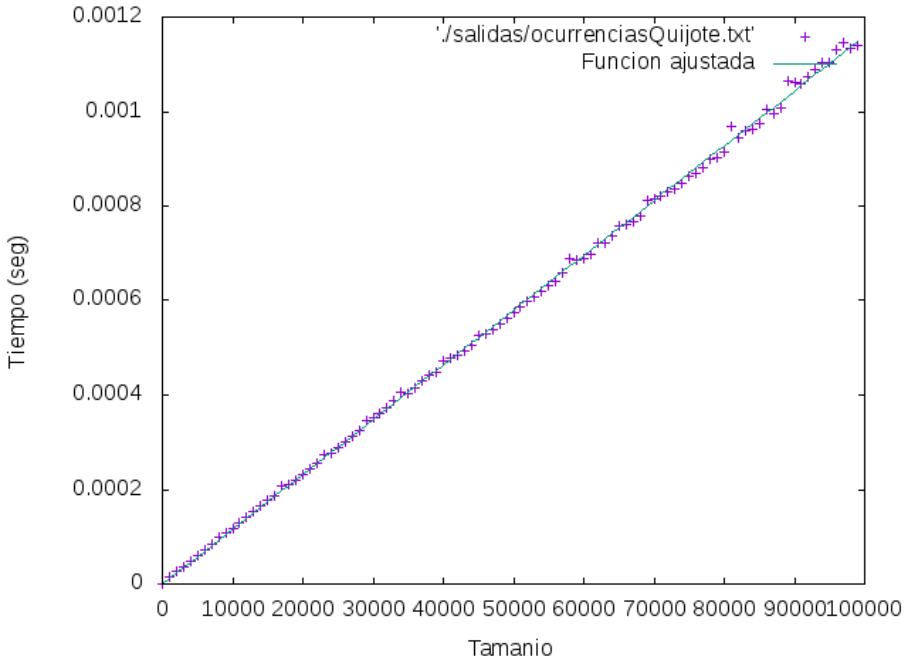
```
./bin/ocurrencias > ./salidas/ocurrenciasQuijote.txt
```

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x$, ya que el algoritmo tiene orden de eficiencia $O(n)$.

```
gnuplot> plot './salidas/ocurrenciasQuijote.txt' title 'Eficiencia ocurrencia Quijote'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x
gnuplot> fit f(x) './salidas/ocurrenciasQuijote.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 1.15928e-08      +/- 1.467e-11    (0.1265%)

gnuplot> plot './salidas/ocurrenciasQuijote.txt', f(x) title 'Funcion ajustada'
```



1.2. Considerando como entrada el texto del fichero lema.txt.

Compilamos el código `ocurrencias.cpp` modificando el código para que lea y cuente las palabras del fichero `lema.txt`:

```
g++ -std=c++11 -o ./bin/ocurrencias ./src/ocurrencias.cpp
```

Guardamos la salida de la ejecución en un fichero llamado `ocurrenciasDicc.txt`, donde la primera columna representa el tamaño del conjunto de palabras y la segunda el tiempo necesario para buscar la palabra "hidalgo":

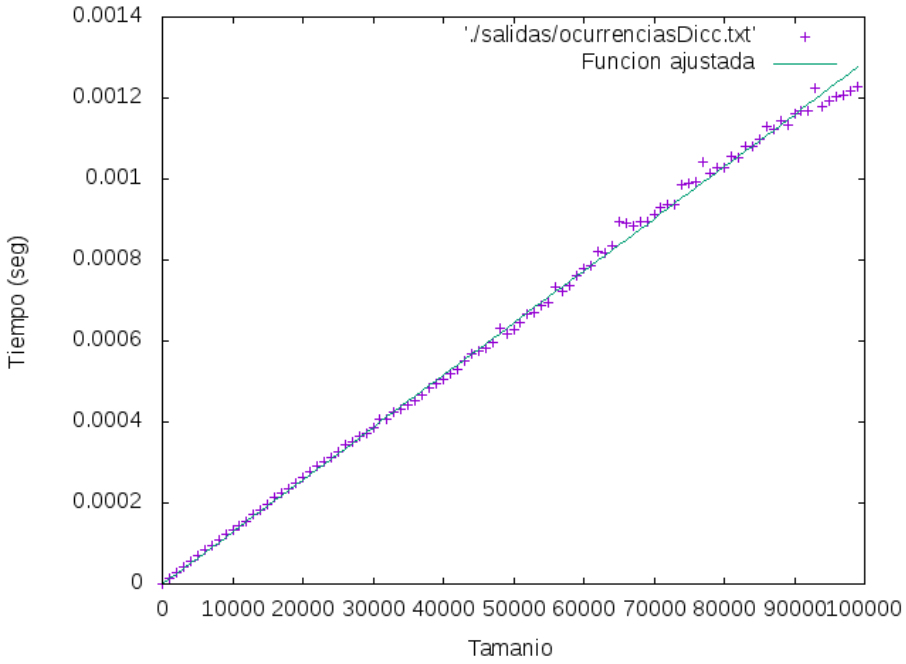
```
./bin/ocurrencias > ./salidas/ocurrenciasDicc.txt
```

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x$, ya que el algoritmo tiene orden de eficiencia $O(n)$.

```
gnuplot> plot './salidas/ocurrenciasDicc.txt' title 'Eficiencia ocurrencia Diccionario'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x
gnuplot> fit f(x) './salidas/ocurrenciasDicc.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 1.28826e-08      +/- 2.827e-11    (0.2194%)

gnuplot> plot './salidas/ocurrenciasDicc.txt', f(x) title 'Funcion ajustada'
```



2. Análisis de eficiencia del código en frecuencias.cpp, considerando como entradas el texto del fichero quijote.txt.

Para realizar el cálculo empírico de la eficiencia del algoritmo, hemos ejecutado el programa en un ordenador cuyas prestaciones son: procesador Intel Core i7 con 6GB de RAM, y el compilador empleado es 'g++'.

Para poder realizar bien el ejercicio modificamos el código para poder guardar en un fichero distinto las salidas de las distintas versiones de `contar_frecuencias`. Para compilar el código utilizamos:

```
g++ -std=c++11 -o ./bin/frecuencias ./src/frecuencias.cpp
```

Guardamos las distintas salidas de las ejecuciones en ficheros llamados `frecuenciasV1.txt`, `frecuenciasV2.txt`, `frecuenciasV3.txt`, `frecuenciasV4.txt`, donde la primera columna representa el tamaño del conjunto de palabras y la segunda el tiempo transcurrido:

```
./bin/ocurrencias > ./salidas/frecuencias.txt
```

2.1. Análisis de eficiencia del código con la estructura de datos de la versión 1.

```
int contar_hasta( vector<string> & V, int ini, int fin, string & s) {
   int cuantos = 0;
   for (int i=ini; i< fin ; i++)
      if (V[i]==s) {
         cuantos ++;
      }
   return cuantos;
}
```

```
void contar_frecuencias_V1( vector<string> & libro, int ini, int fin,
                            vector<string> &pal, vector<int> & frec ){
   int cuantas;
   for (int i = ini; i<fin; i++){

      cuantas = contar_hasta(libro,ini,fin,libro[i]); //O(n)
      pal.push_back(libro[i]); //tiempo amortizado O(1)
      frec.push_back(cuantas); //tiempo amortizado O(1)
   }
}
```

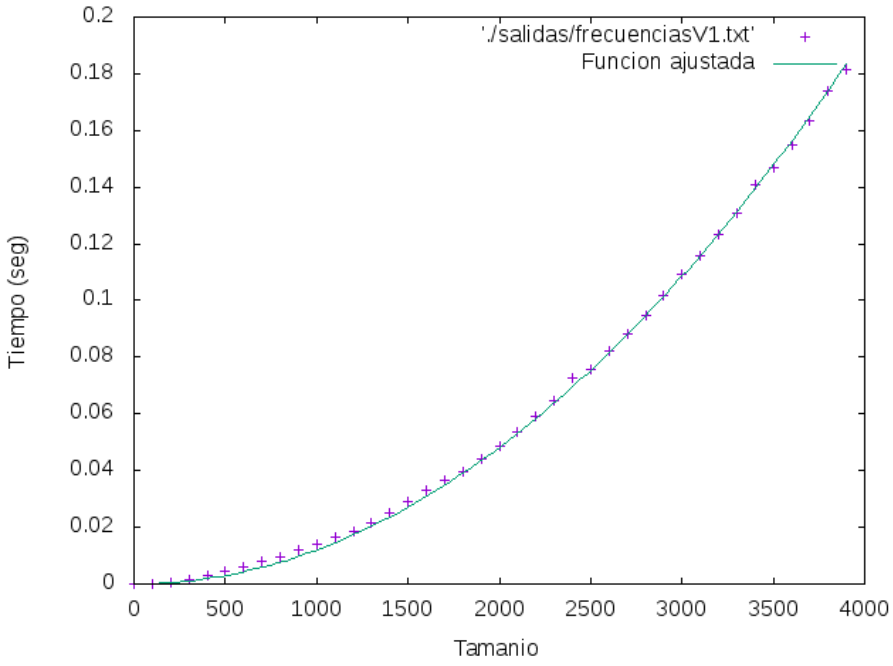
El método `contar_hasta` es del orden $O(n)$ y el método `push_back` del vector es 0(1), por lo tanto `contar_frecuencias_V1` es del orden $O(n^2)$.

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x * x$, ya que el algoritmo tiene orden de eficiencia $O(n^2)$.

```
gnuplot> plot './salidas/frecuenciasV1.txt' title 'Eficiencia frecuenciasV1'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x * x
gnuplot> fit f(x) './salidas/frecuenciasV1.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 1.2061e-08       +/- 2.999e-11    (0.2487%)

gnuplot> plot './salidas/frecuenciasV1.txt', f(x) title 'Funcion ajustada'
```



2.2. Análisis de eficiencia del código con la estructura de datos de la versión 2.

```
int buscar( vector<string> & V, string & s) {
   bool enc= false;
   int pos = POS_NULA;
   for (int i=0; i< V.size() && !enc; i++)
      if (V[i]==s) {
         enc = true;
         pos = i;
      }
   return pos;
}


void contar_frecuencias_V2( vector<string> & libro, int ini, int fin,
                        vector<string> &pal, vector<int> & frec ){
   int pos;
   for (int i = ini; i<fin; i++){

      pos = buscar(pal, libro[i]); // O(n)
      if (pos==POS_NULA) {
         pal.push_back(libro[i]);    // Análisis amortizado O(1)
         frec.push_back(1);    // Análisis amortizado O(1)
      }
      else {
         frec[pos]++;
      }
```

```
    }
}
```

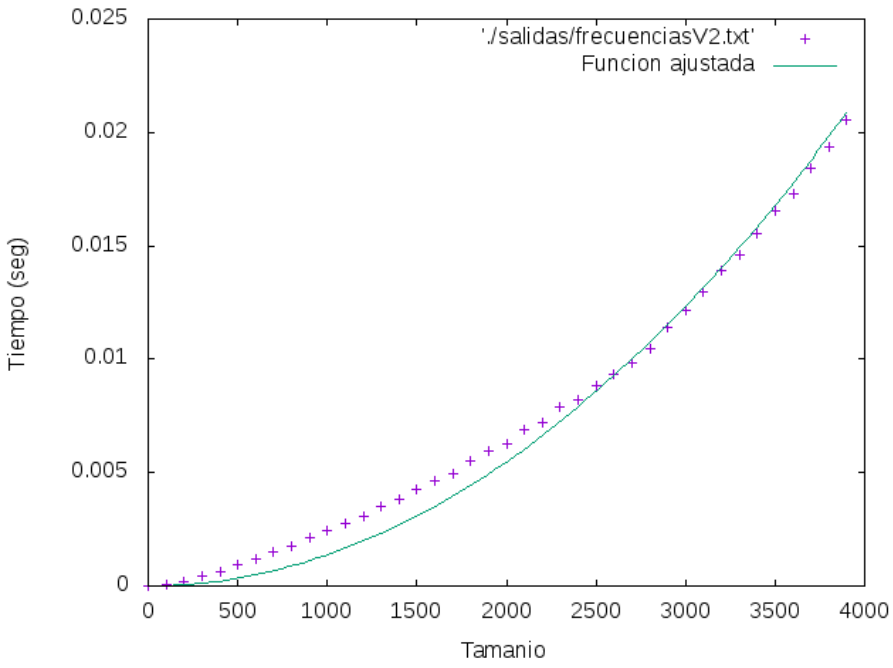
El método `buscar` es del orden $O(n)$ y el método `push_back` del vector es 0(1), por lo tanto `contar_frecuencias_V2` es del orden $O(n^2)$.

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x * x$, ya que el algoritmo tiene orden de eficiencia $O(n^2)$.

```
gnuplot> plot './salidas/frecuenciasV2.txt' title 'Eficiencia frecuenciasV2'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x * x
gnuplot> fit f(x) './salidas/frecuenciasV2.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 1.37174e-09      +/- 1.585e-11    (1.156%)

gnuplot> plot './salidas/frecuenciasV2.txt', f(x) title 'Funcion ajustada'
```



2.3. Análisis de eficiencia del código con la estructura de datos de la versión 3.

```
void contar_frecuencias_V3( vector<string> & libro, int ini, int fin,
                            vector<string> &pal, vector<int> & frec ){
   vector<string>::iterator pos;
   for (int i = ini; i<fin; i++){
```

```
      pos = lower_bound(pal.begin(), pal.end(), libro[i]);  // O(log (n) )
                                                  // n tama del vector  Busqueda Binaria
      if ((pos==pal.end()) || (*pos!=libro[i])) {
         frec.insert(frec.begin() + (pos-pal.begin()), 1);   //O(n)
         pal.insert(pos, libro[i]);            //O (n)
      }
      else {
         frec[pos-pal.begin()]++; // O(1)
      }

   }
}
```

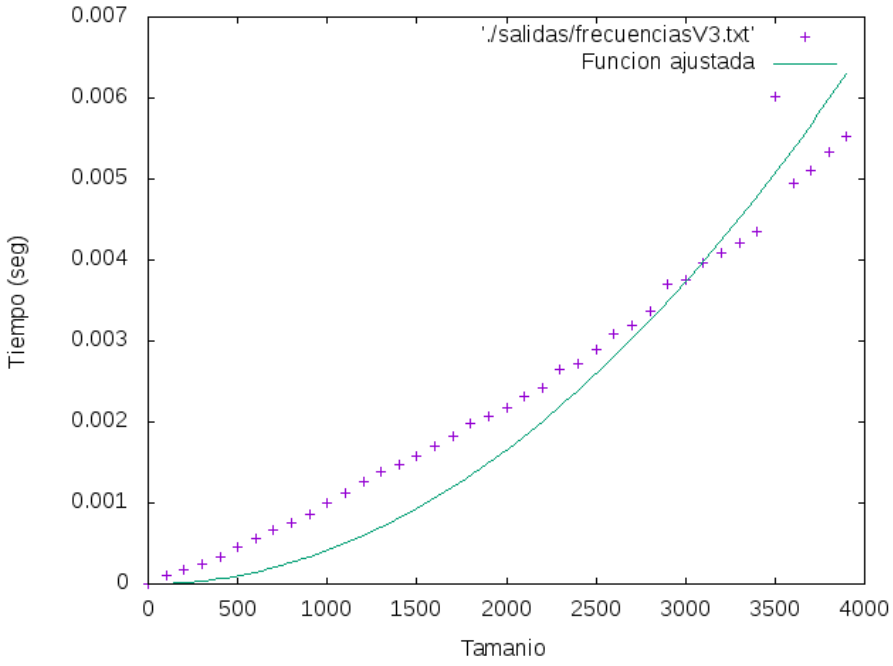
La eficiencia en el interior del bucle for es $O(log(n) + n) = O(n)$, por lo tanto el método `contar_frecuencias_V3` es del orden $O(n^2)$.

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x * x$, ya que el algoritmo tiene orden de eficiencia $O(n^2)$.

```
gnuplot> plot './salidas/frecuenciasV3.txt' title 'Eficiencia frecuenciasV3'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x * x
gnuplot> fit f(x) './salidas/frecuenciasV3.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 4.14655e-10      +/- 1.11e-11     (2.676%)

gnuplot> plot './salidas/frecuenciasV3.txt', f(x) title 'Funcion ajustada'
```



2.4. Análisis de eficiencia del código con la estructura de datos de la versión 4.

```
void contar_frecuencias_V4( vector<string> & libro, int ini, int fin,
                            vector<string> &pal, vector<int> & frec ){

   map<string,int> M;
   for (int i = ini; i<fin; i++)
      M[libro[i]]++;          // O( log(n) )

   map<string,int>::iterator it;
   for (it = M.begin(); it!= M.end(); ++ it){ // Bucle O(k log k) siendo k el
                                              //numero de palabras distintas
      pal.push_back( (*it).first ); //O(1)
      frec.push_back( (*it).second ); //O(1)
   }
}
```

La eficiencia del primer bucle for es $O(nlog(n))$,la eficiencia del segundo bucle for es $O(nlog(n))$, por lo tanto el método `contar_frecuencias_V4` es del orden $O(nlog(n))$.

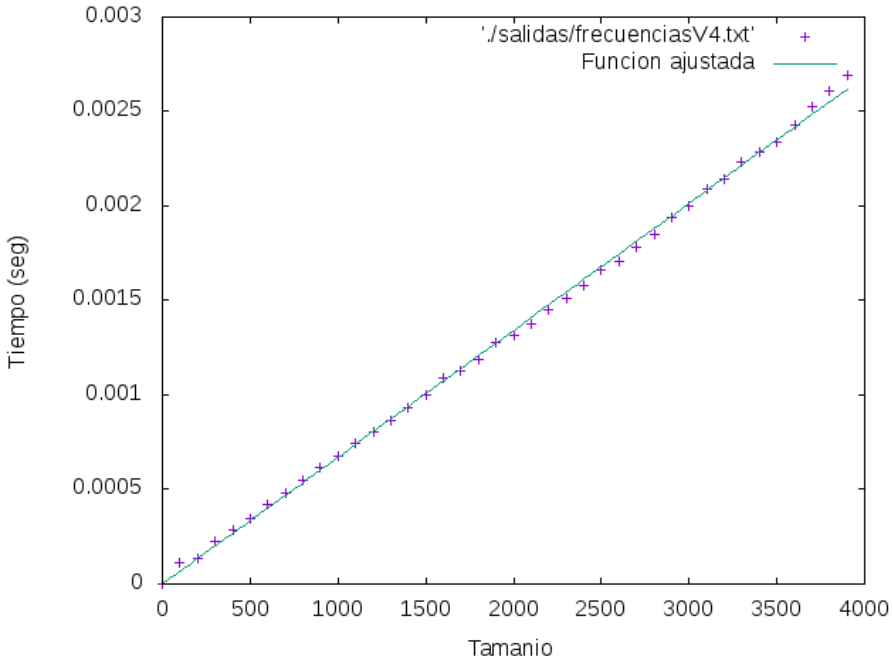
Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a*x$, aunque teóricamente el algoritmo tiene orden de eficiencia $O(nlog(n))$, se puede observar que el ajuste a $f(x) = a * x$ es mucho más exacto.

```
gnuplot> plot ’./salidas/frecuenciasV4.txt’ title ’Eficiencia frecuenciasV4’
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x
```

```
gnuplot> fit f(x) './salidas/frecuenciasV4.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 6.71595e-07      +/- 1.725e-09    (0.2569%)

gnuplot> plot './salidas/frecuenciasV4.txt', f(x) title 'Funcion ajustada'
```



3. Realizar el análisis de eficiencia teórico y práctico con los algoritmos de ordenación que se conocen (burbuja, inserción y selección).

Para realizar el cálculo empírico de la eficiencia del algoritmo, hemos ejecutado el programa en un ordenador cuyas prestaciones son: procesador Intel Core i7 con 6GB de RAM, y el compilador empleado es 'g++'.

3.1. Burbuja

Para compilar el código utilizamos:

```
g++ -std=c++11 -o ./bin/BurbujaEficiencia ./src/BurbujaEficiencia_Lims.cpp
```

Guardamos la salida del programa en un fichero llamado `ordenacionBurbuja.txt`, donde la primera columna representa el tamaño del conjunto de palabras y la segunda el tiempo transcurrido:

```
./bin/BurbujaEficiencia 0 10000 100 > ./salidas/ordenacionBurbuja.txt
```

```
void burbuja(vector<int> & T, int inicial, int final)
{
 int i, j;
 int aux;
 for (i = inicial; i < final - 1; i++) //O(n^2)
   for (j = final - 1; j > i; j--) //O(n)
       if (T[j] < T[j-1])
         {
           aux = T[j];  //O(1)
           T[j] = T[j-1]; //O(1)
           T[j-1] = aux; //O(1)
         }
}
```

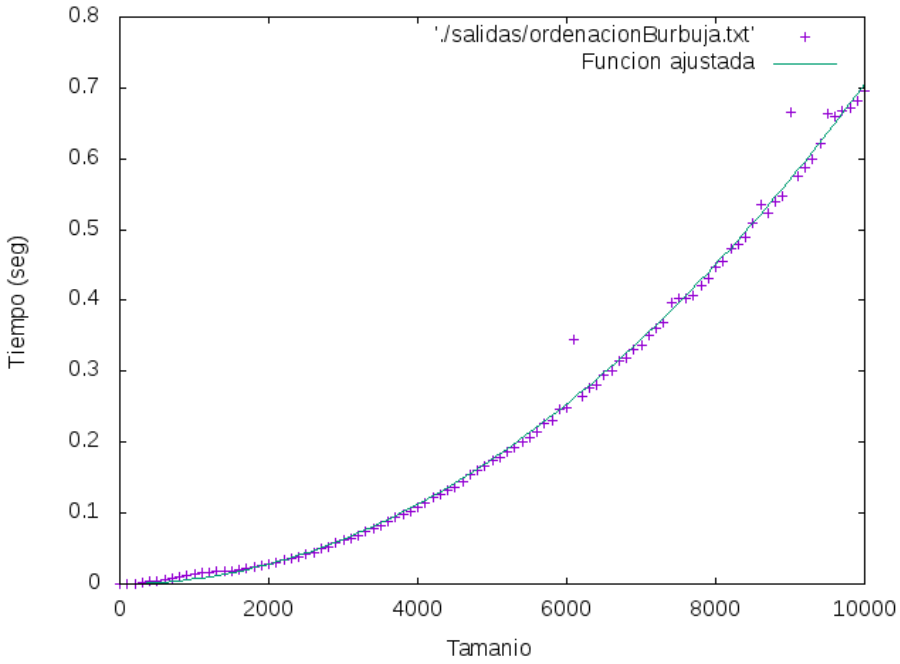
La eficiencia de ordenación mediante burbuja es $O(n^2)$.

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x * x$.

```
gnuplot> plot './salidas/ordenacionBurbuja.txt' title 'Eficiencia ordenacionBurbuja'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x * x
gnuplot> fit f(x) './salidas/ordenacionBurbuja.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 7.05041e-09      +/- 3.095e-11    (0.439%)

gnuplot> plot './salidas/ordenacionBurbuja.txt', f(x) title 'Funcion ajustada'
```



3.2. Selección

Para compilar el código utilizamos:

```
g++ -std=c++11 -o ./bin/SeleccionEficiencia ./src/SeleccionEficiencia.cpp
```

Guardamos la salida del programa en un fichero llamado `ordenacionSeleccion.txt`, donde la primera columna representa el tamaño del conjunto de palabras y la segunda el tiempo transcurrido:

```
./bin/SeleccionEficiencia > ./salidas/ordenacionSeleccion.txt
```

```
void seleccion(vector<int> & T, int inicial, int final)
{
   int i, j,min,aux;
   for (i = inicial; i < final - 1; i++){ // O(n^2)
      min = i;
      for (j = i + 1; j < final ; j++) // O(n)
         if (T[j] < T[min]) // O(1)
            min=j; // O(1)
      aux = T[min]; // O(1)
      T[min] = T[i]; // O(1)
      T[i] = aux; // O(1)
   }
}
```

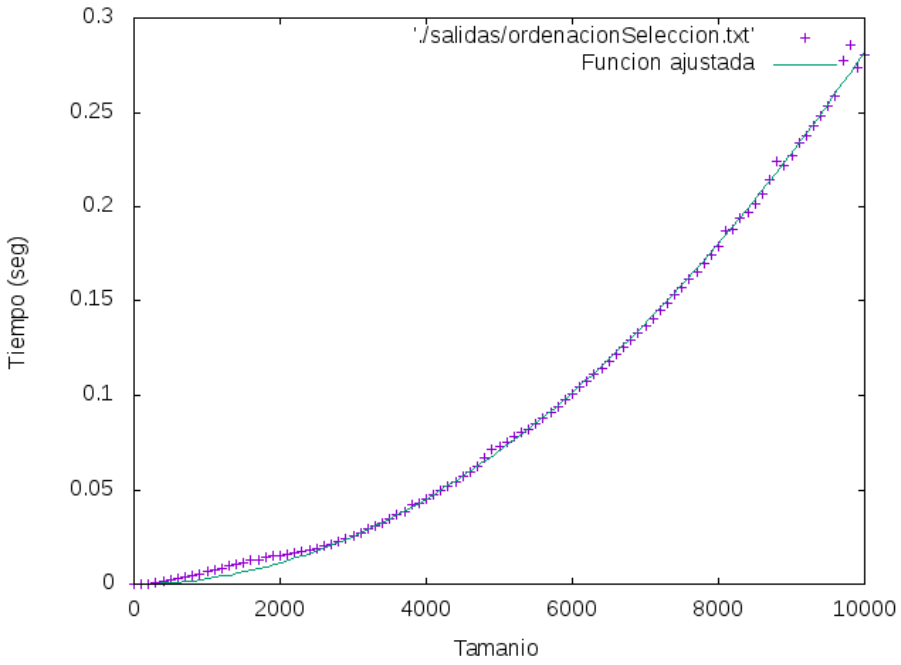
La eficiencia de ordenación mediante selección es $O(n^2)$.

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x * x$.

```
gnuplot> plot './salidas/ordenacionSeleccion.txt' title 'Eficiencia ordenacionSeleccion'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x * x
gnuplot> fit f(x) './salidas/ordenacionSeleccion.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 2.82297e-09      +/- 6.298e-12    (0.2231%)


gnuplot> plot './salidas/ordenacionSeleccion.txt', f(x) title 'Funcion ajustada'
```



3.3. Inserción

Para compilar el código utilizamos:

```
g++ -std=c++11 -o ./bin/InsercionEficiencia ./src/InsercionEficiencia.cpp
```

Guardamos la salida del programa en un fichero llamado `ordenacionInsercion.txt`, donde la primera columna representa el tamaño del conjunto de palabras y la segunda el tiempo transcurrido:

```
./bin/InsercionEficiencia 0 10000 100> ./salidas/ordenacionInsercion.txt
```

```
void insercion(vector<int> & T, int inicial, int final)
{
   int i, j,valor;
   for (i = inicial; i < final; i++){ O(n^2)
```

```
      valor = T[i];
      for (j = i ; j > 0 && T[j-1] > valor; j--) // O(n)
         T[j] = T[j-1];
      T[j] = valor;
   }
}
```

La eficiencia de ordenación mediante inserción es $O(n^2)$.

Realizamos el enfoque híbrido con el ajuste de los datos obtenidos. Para hacer la regresión con `gnuplot` he definido la función $f(x) = a * x * x$.

```
gnuplot> plot './salidas/ordenacionInsercion.txt' title 'Eficiencia ordenacionInsercion'
with points
gnuplot> set xlabel "Tamanio"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x) = a * x * x
gnuplot> fit f(x) './salidas/ordenacionInsercion.txt' via a

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 2.03009e-09      +/- 1.219e-11    (0.6006%)

gnuplot> plot './salidas/ordenacionInsercion.txt', f(x) title 'Funcion ajustada'
```