Práctica 1 ALG

Alfonso Soto López y Juan Anaya Ortiz

March 6, 2016

Ejercicio 5.1

Ordenación por Inserción:

```
void inserc(int &V[], int tam){
    int j, index;
    for(int i=1; i < tam; i++){
        index = V[i];
        j = i-1;
        while(j >= 0 && V[j] > index){
            V[j+1] = V[j];
            j--;
        }
        V[j+1] = index;
    }
}
```

Línea 2: 2 OE (Reserva Memoria)
Línea 3: 3 OE (Asignación, Comparación, Incremento)
Línea 4: 2 OE (Acceso al elemento V[i], Asignación)
Línea 5: 2 OE (Decremento, Asignación)
Línea 6: 3 OE (Comparación j>=0, Acceso V[j], Comparación V[j] > index)
Línea 7: 3 OE (Acceso elemento V[j+1], Acceso elemento V[j]), Asignación)
Línea 8: 2 (Decremento, Asignación)
Línea 10: 2 (Acceso V[j+1], Asignación)

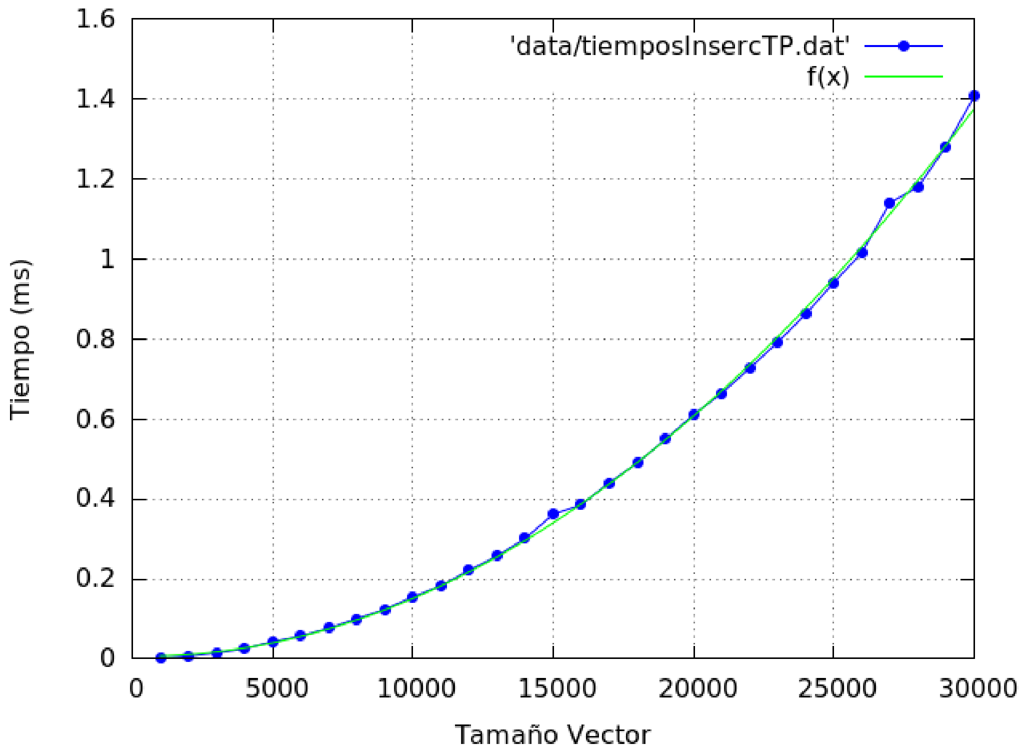
Inserción - Peor Caso:

El peor caso se da con el vector ordenado de mayor a menor.

$$T_P(n) = 2 + 1 + \sum_{i=1}^{n-1} \left(2 + 2 + 2 + \sum_{0}^{i-1} (3+2) + 2 \right) =$$

$$= 3 + \left(8(n-1) + \frac{5}{2}(n-1)n \right) = \frac{5}{2}n^{\mathbf{2}} + \left(\frac{11}{2}\right) n - 5 \in \mathbf{O}(\mathbf{n^2})$$

Tiempo Empírico:



$$f(x) = an^{\mathbf{2}} + bn - c; \left\{ \begin{smallmatrix} a=1.5611e-09 \\ b=-1.20986e-06 \\ c=-0.00660574 \end{smallmatrix} \right\}$$

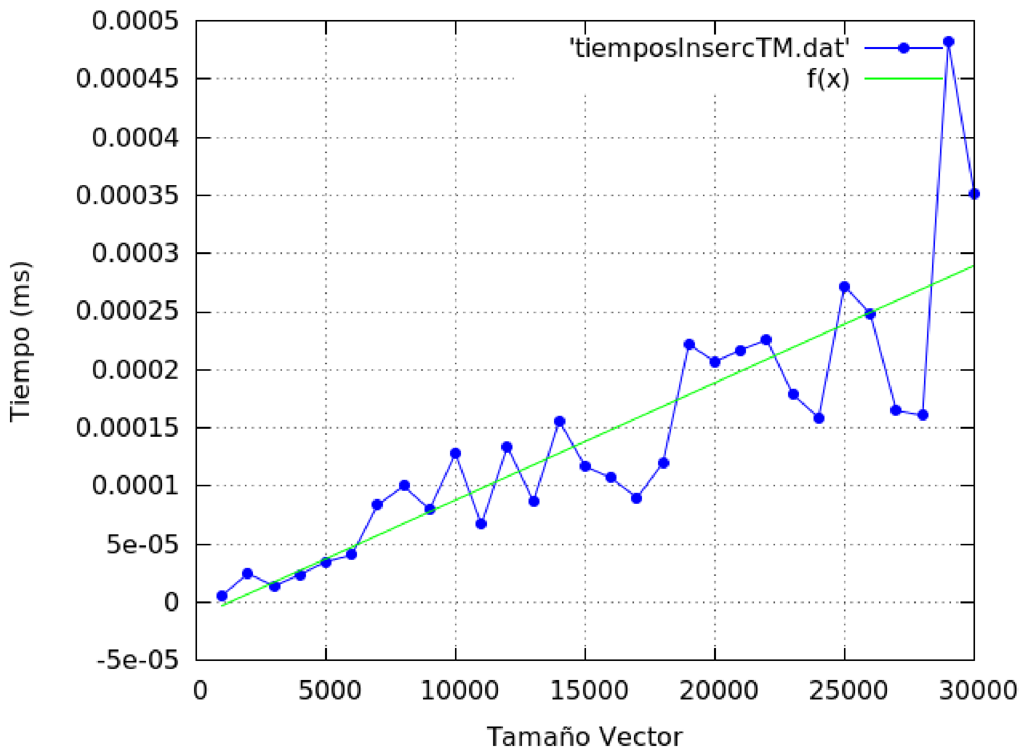
Inserción - Mejor Caso:

El mejor caso se da con el vector totalmente ordenado de menor a mayor.

El bucle de la linea 6 se sale a la primera iteración ya que el primer elemento buscado siempre será mayor.

$$T_m = 2 + 1 + \sum_{i=1}^{n-1} (2 + 2 + 2 + 3 + 2 + 2) = 13n - 10 \in \mathbf{O(n)}$$

Tiempo Empírico:



$$f(x) = an - b; \left\{ {a=1.0083e-08 \atop b=1.27862e-05} \right\}$$

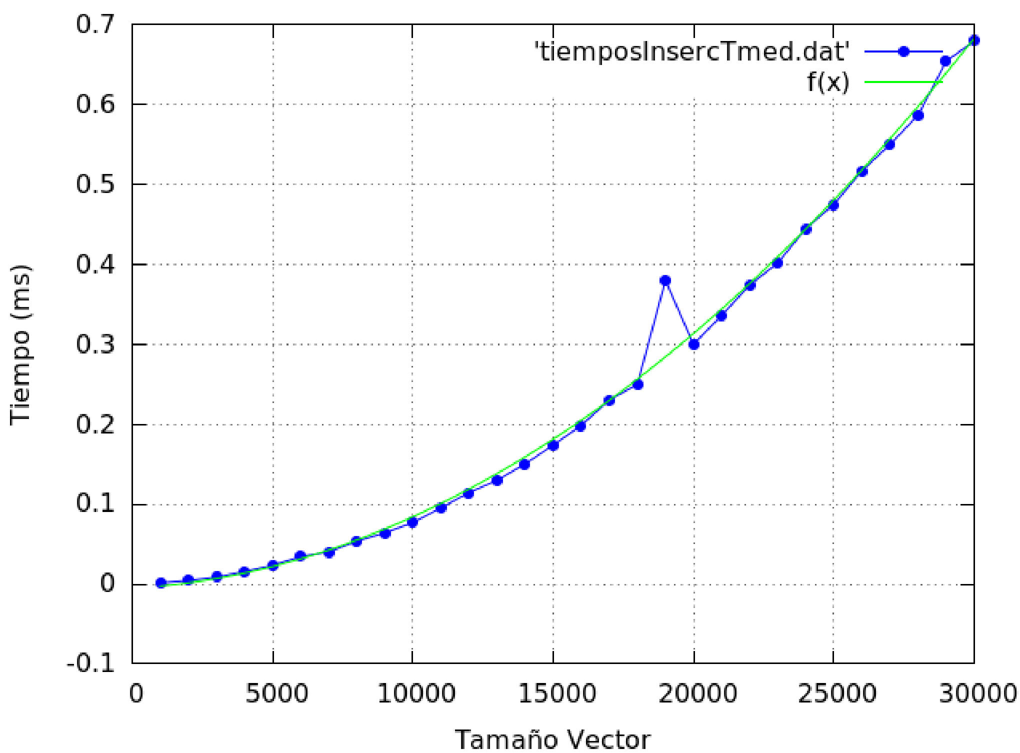
Inserción - Caso Promedio:

En el caso promedio el bucle while de la línea 6 se ejecutará una razón de n/2.

$$T_{\frac{1}{2}}(n) = 2 + 1 + \sum_{i=1}^{n-1}\left(2 + 2 + 2 + \sum_{0}^{(i-1)/2}(3+2) + 2\right) =$$

$$= 3 + 8(n-1) + \frac{5}{4}(n-1)n \in \mathbf{O}(\mathbf{n^2})$$

Tiempo Empírico:



$$f(x) = an^{\mathbf{2}} + bn - c; \left\{\begin{smallmatrix} a=6.99947e-10 \\ b=1.92869e-06 \\ c=0.00541029 \end{smallmatrix}\right\}$$

Ordenación por Selección:

```
void selec(int &V[], int tam){

    int min=0
    for(int i=0 ; i<tam-1 ; i++){
        minimo=i;
        for(int j=i+1 ; j<tam ; j++)
            if (V[min] > V[j])
            min=j;
        swap( V[min], V[i] );
    }

}
```

```
inline void swap(int &x, int &y){

    int aux = x;
    x = y;
    y = aux;

}
```

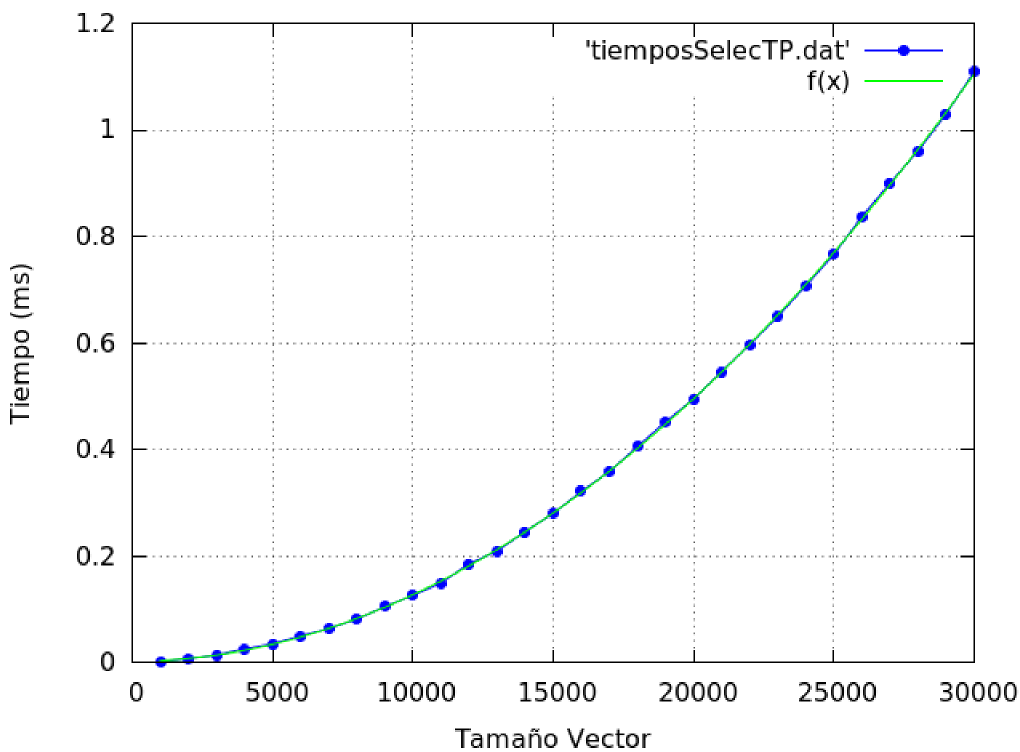
Línea 1: 0
Línea 2: 1 OE (Asignación)
Línea 3 (1º for): 3 OE (Asignación, Comparación $j < tam$, Incremento j++)
Línea 4: 1 OE (Asignación)
Línea 5 (2º for): 3 OE (Asignación, Comparación, Incremento)
Línea 6 (if): 3 OE (Acceso V[min] y V[j], Comparación $V[min] > V[j]$)
Línea 7: 1 OE Asignación ($min = j$)
*(Ignoro la llamada a la función ya que se usa "inline")
Línea 8: 2 OE (Acceso a V[min] y V[i])
Línea 9: 1 OE (Asignación $aux = x$)
Línea 10: 1 OE (Asignación $x = y$)
Línea 11 1 OE (Asignación $y = aux$)

Selección - Peor Caso:

$$T_p(n) = 1 + \sum_{i=0}^{n-1} \left(1 + \sum_{i+1}^{n} (3 + max(0,1)_{=1}) + 2 + 1 + 1 + 1 \right) =$$

$$= 1 + \frac{1}{2} n^{\mathbf{2}} 4 + (n-1)\,6 = 2n^{\mathbf{2}} + 6n + 7 \in \mathbf{O(n^2)}$$

Tiempo Empírico:



$$f(x) = an^{\mathbf{2}} + bn - c; \left\{ \begin{smallmatrix} a=1.21108e-09 \\ b=4.35477e-07 \\ c=-0.0021317 \end{smallmatrix} \right\}$$

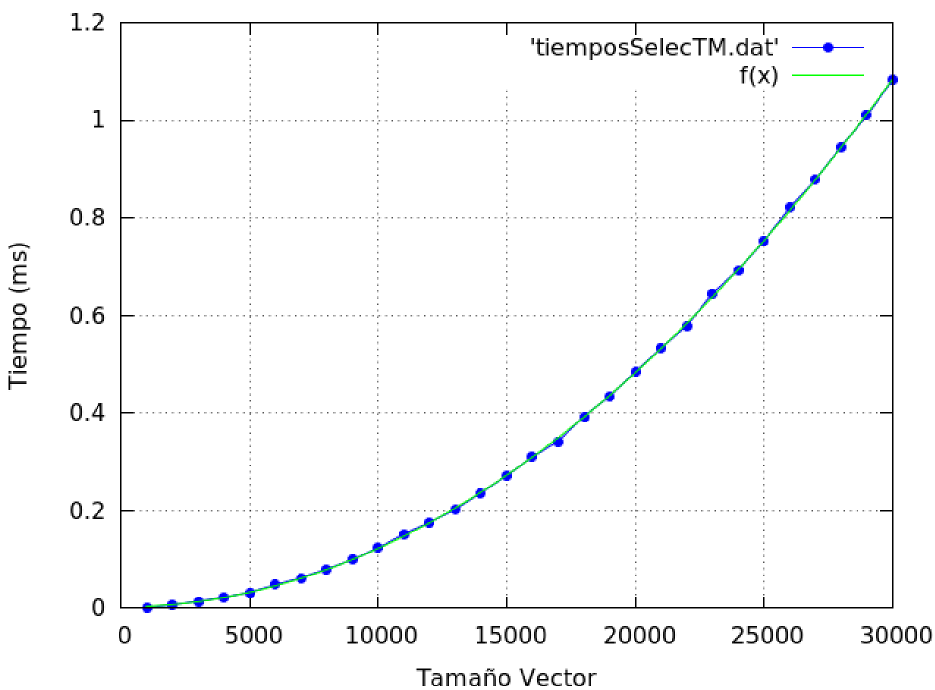
Selección - Mejor Caso:

En el mejor caso (Vector ordenado) lo único que cambia es el número de accesos al "if" de la línea 6, en el que nunca encontrará un elemento menor a V[min], por lo que nos ahorraremos un solo OE. (max(0,1) = 0)

$$T_m(n) = 1 + \sum_{i=0}^{n-1} \left(1 + \sum_{i+1}^{n} (3 + max(0,1)_{=0}) + 2 + 1 + 1 + 1 \right) =$$

$$= 1 + \frac{1}{2} n^{\mathbf{2}} 3 + (n-1)\, 6 = \frac{3}{2} \mathrm{n}^{\mathbf{2}} + 6\mathrm{n} + 7 \in \mathbf{O}(\mathbf{n^2})$$

Tiempo Empírico:



$$f(x) = an^{\mathbf{2}} + bn - c; \left\{ \begin{smallmatrix} a=1.20314e-09 \\ b=-6.88768e-08 \\ c=-0.00306035 \end{smallmatrix} \right\}$$

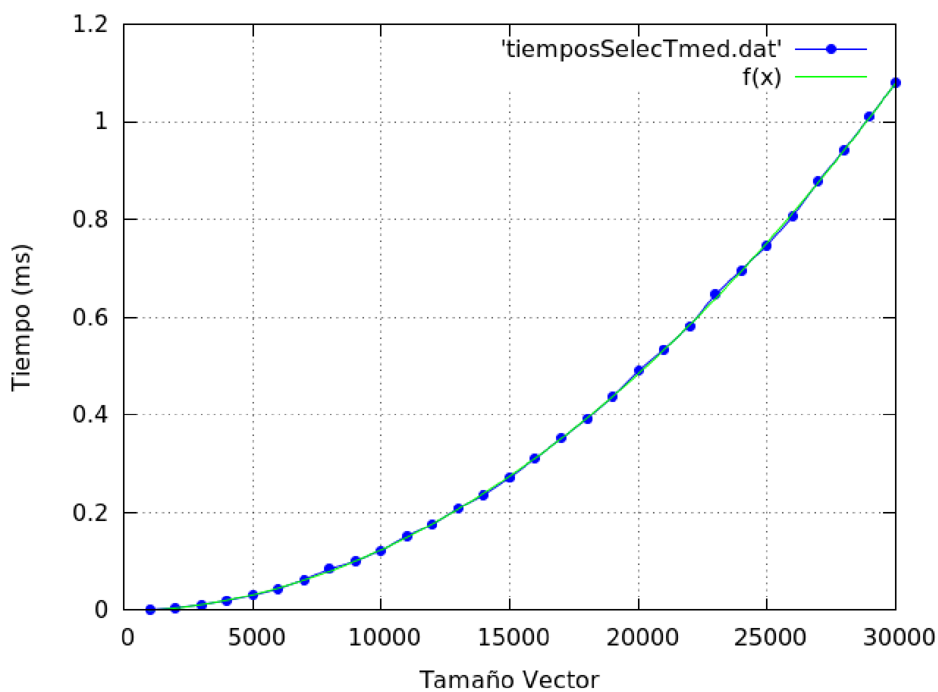
Selección - Caso Promedio:

En el caso promedio en número de accesos aciertos en la condicion “if” de la línea 6 será de una razón a n/2.

$$T_m(n) = 1 + \sum_{i=0}^{n-1} \left(1 + \sum_{i+1}^{n} (3 + max(0,1)_{=0.5}) + 2 + 1 + 1 + 1 \right) =$$

$$= 1 + \frac{1}{2} n^2 \frac{\mathbf{3}}{\mathbf{2}} + (n-1)\,6 = \frac{\mathbf{3}}{\mathbf{4}} \mathbf{n^2} + \mathbf{6n} + \mathbf{7} \in \mathbf{O(n^2)}$$

Tiempo Empírico:



$$f(x) = an^2 + bn - c; \left\{ \begin{matrix} a{=}1.17479e{-}09 \\ b{=}7.92897e{-}07 \\ c{=}0.00166957 \end{matrix} \right\}$$

Ordenación por Burbuja:

```
void burbuja(int *V, int tam){

    for(int i=0; i < tam-1; i++){
        for(int j=tam-1; j > i; j--){
            if( V[j] < V[j-1] )
                swap( V[j], V[j-1] );
        }
    }

}
```

```
inline void swap(int &x, int &y){

    int aux = x;
    x = y;
    y = aux;

}
```

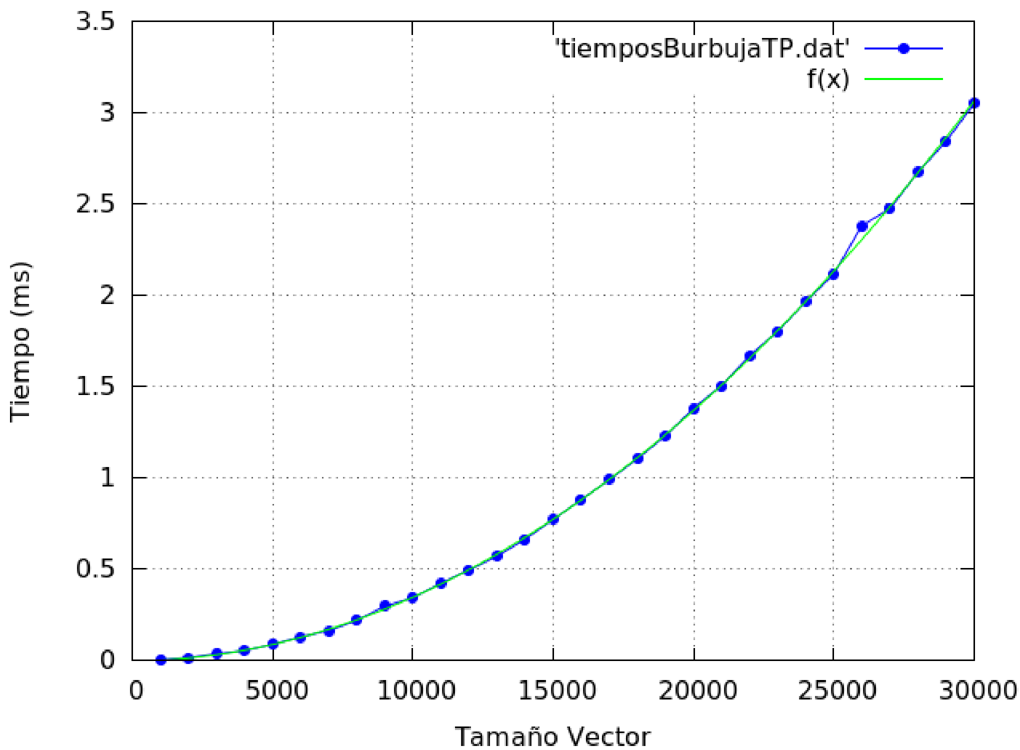
Línea 1 (1º “for”): 3 OE (Asignación, comparación, incremento)
Línea 2 (2º “for”): 4 OE (Resta, asignación, comparación, decremento)
Línea 3 (“if”): 4 OE (Decremento (j-1), acceso variables V[j] y V[j-1], comparación)
*(Ignoro la llamada a la función ya que se usa “inline”)
Línea 4: 3 OE (Resta (j-1), Acceso a V[j] y V[j-1])
Línea 5: 1 OE (Asignación aux = x)
Línea 6: 1 OE (Asignación x = y)
Línea 7: 1 OE (Asignación y = aux)

Burbuja - Peor Caso:

$$T_p(n) = 1 + \sum_{i=0}^{n-1}\left(1 + \sum_{n-1}^{i+1}(3 + max(0,3)_{=3})\right) =$$

$$= 1 + \frac{1}{2}n^2 6 + (n-1) = 3n^2 + (n-1) + 1 \in \mathbf{O(n^2)}$$

Tiempo Empírico:



$$f(x) = an^2 + bn - c; \left\{ \begin{smallmatrix} a=3.36417e-09 \\ b=1.28487e-06 \\ c=0.0014478 \end{smallmatrix} \right\}$$

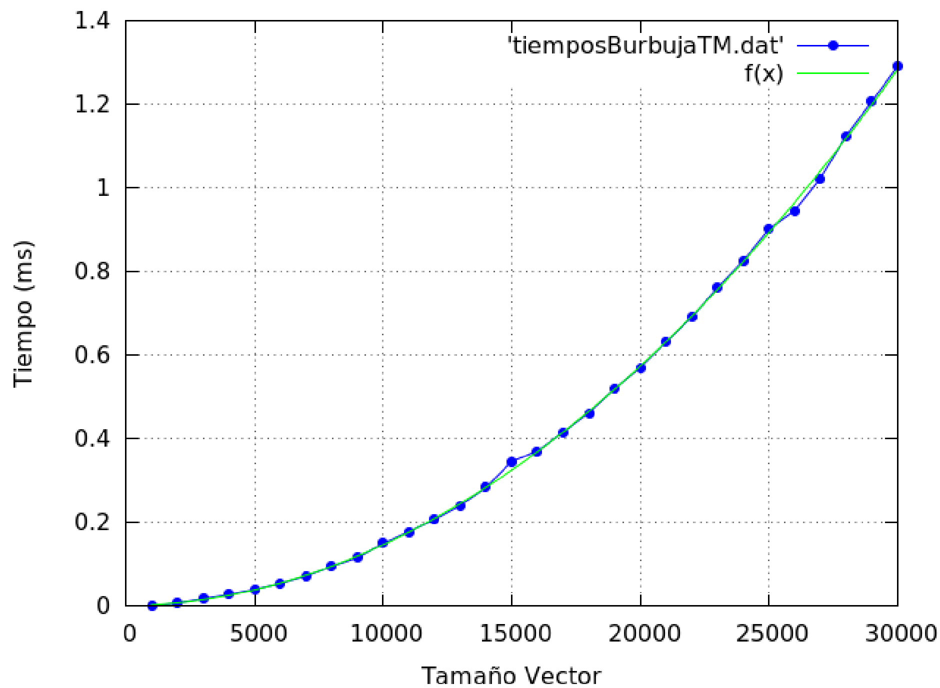
Burbuja - Mejor Caso:

En el mejor caso (Vector ordenado) no ejecutará nunca el condicional “if” por lo que el max(0,3)=0 siempre.

$$T_p(n) = 1 + \sum_{i=0}^{n-1}\left(1 + \sum_{n-1}^{i+1}(3 + max(0,3)_{=0})\right) =$$

$$= 1 + \frac{1}{2}n^{\mathbf{2}}\mathbf{3} + (n-1) = \frac{\mathbf{3}}{\mathbf{2}}n^{\mathbf{2}} + (n-1) + 1 \in \mathbf{O(n^2)}$$

Tiempo Empírico:



$$f(x) = an^{\mathbf{2}} + bn - c; \left\{\begin{smallmatrix} a=1.4085e-09 \\ b=4.40067e-07 \\ c=-0.00199238 \end{smallmatrix}\right\}$$

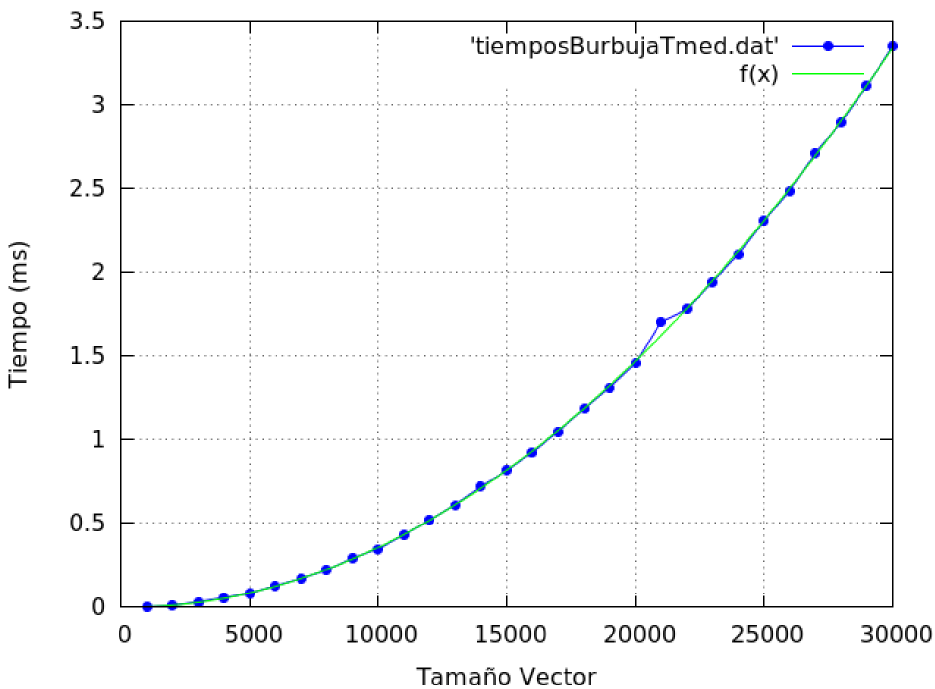
Burbuja - Caso Promedio:

En el mejor promedio (Vector aleatorio) ejecutará el condicional "if" (linea 3) la mitad de las ocasiones a una razón de "n/2" veces, lo que se puede traducir como max(0,3) = (n/2)*0 + (n/2)*3; o lo que es lo mismo, max(0,3) = n*1.5;

$$T_p(n) = 1 + \sum_{i=0}^{n-1} \left(1 + \sum_{n-1}^{i+1} (3 + max(0,3)_{=1.5}) \right) =$$

$$= 1 + \frac{1}{2}\frac{3}{2}n^{\mathbf{2}} + (n-1) = \frac{\mathbf{3}}{\mathbf{4}}n^{\mathbf{2}} + (n-1) + 1 \in \mathbf{O}(\mathbf{n^2})$$

Tiempo Empírico:



$$f(x) = an^{\mathbf{2}} + bn - c; \left\{ \substack{a=3.78822e-09 \\ b=-2.43326e-06 \\ c=-0.00285813} \right\}$$

Ejercicio 5.4

Ordenación por Mezcla:

Para calcular la eficiencia teórica del algoritmo, calculo por separado las eficiencias de los siguientes algoritmos:

Mergesort: Para el cálculo de "Mergesort" me baso en el caso general y desarrollo la recurrencia.

Tanto en el mejor caso, como en el peor y en el caso promedio la complejidad del algoritmo será la misma.

$$T(n) \begin{cases} 1, & n = 1 \\ 2T\left(\frac{n}{2}\right) + n, & n > 1 \end{cases}$$

$$t_i = T(2^i) \; t_i = 2t_{i-1} + 2^i \;\Rightarrow\; t_i = c_1 2^i + c_2 i 2^i$$

$$T(n) = c_1 2^{\log_2(n)} + c_2 \log_2(n) 2^{\log_2(n)} = c_1 n + c_2 n \log_2(n) \in O(n \log_2 n)$$

Inserción: Usaremos los resultados del cálculo del algoritmo ya resueltos en el ejercicio 5.1.

Mezcla - Peor Caso:

Para el cálculo de la eficiencia partimos del caso base y le restamos la eficiencia de la parte de mergesort que no se realiza, es decir, la parte del algoritmo donde los subvectores son de menor tamaño que UMBRAL_MS.

Si llamamos a UMBRAL_MS como "m" y el número de subvectores con valor igual o inferior a UMBRAL_MS se corresponde con $\left(\frac{n}{m}\right)$:

$$n \log_2(n) = Eficiencia\, Mergesort$$

$$\left(\frac{n}{m}\right) m \log_2(m) = Eficiencia\, de\, la\, parte\, no\, ejecutada$$

$$n \log_2(n) - \left(\frac{n}{m}\right) m \log_2(m)$$

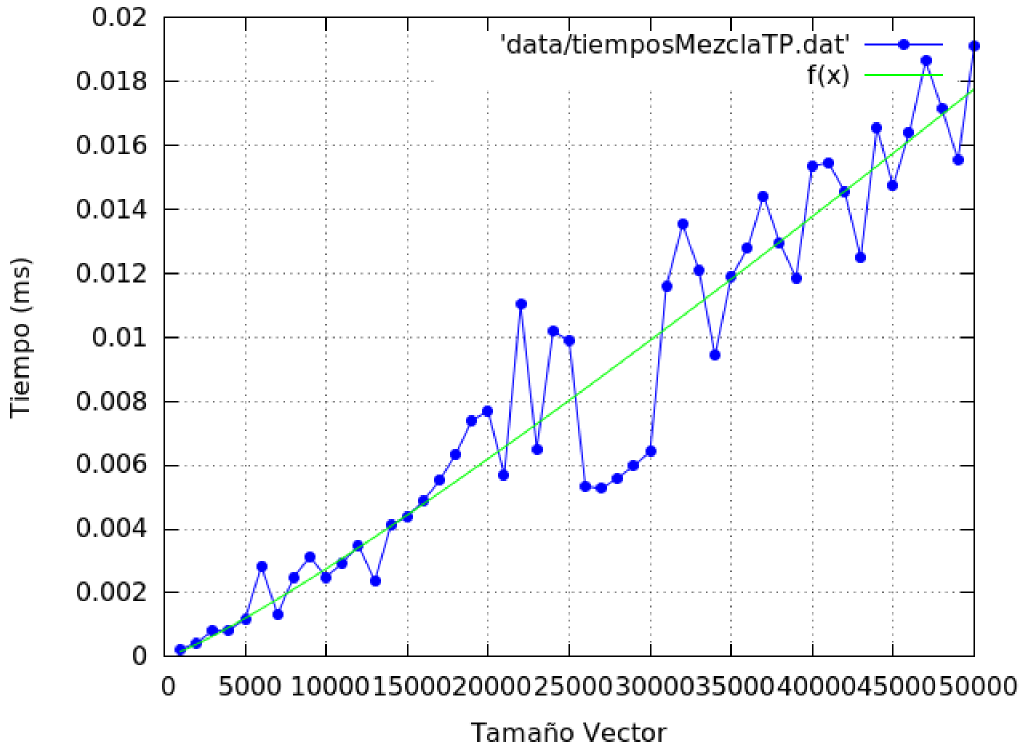
Ahora solo faltaría sumar la eficiencia de aplicar Inserción a todos esos subvectores de tamaño UMBRAL_MS.

$$\left(\frac{n}{m}\right)(n^2 + n) = Eficiencia\, Inserción$$

$$n\log_2(n) - \left(\frac{n}{m}\right) m\log_2(m) + \left(\frac{n}{m}\right)(m^2 + m) =$$

$$n\log_2 n - n\log_2 m + nm + n$$

$$n\left(\log_2(n) - \log_2(m) + m + 1\right) \in O(nm)$$



$$f(x) = x(a\log_2(x) - b\log_2(100) + 100c + d); \begin{cases} a = & -2.2541e-09 \\ b = & 1.06241 \\ c = & 0.0606787 \\ d = & 0.990607 \end{cases}$$

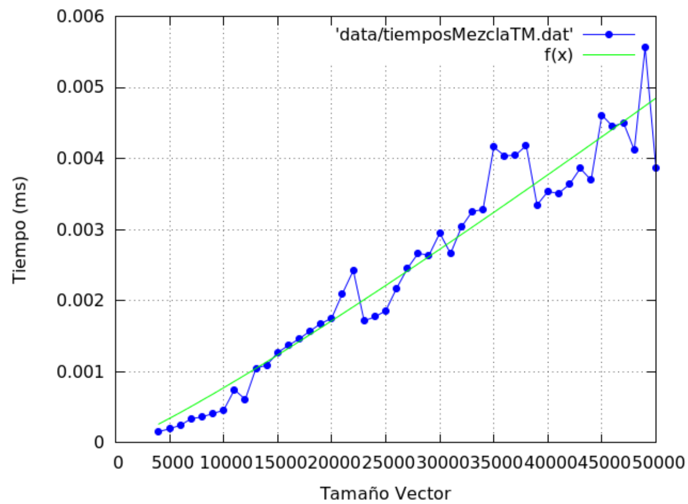
Mezcla - Mejor Caso:

Ya que la eficiencia de Mergesort no varía, solo hay que cambiar la del algoritmo de inserción.

$$n\log_2(n) - \left(\frac{n}{m}\right) m\log_2(m) + \left(\frac{n}{m}\right)(m) =$$

$$n\log_2(n) - n\log_2(m) + n =$$

$$n\left(\log_2(n) - \log_2(m) + 1\right) \in O\left(n\log(n)\right)$$



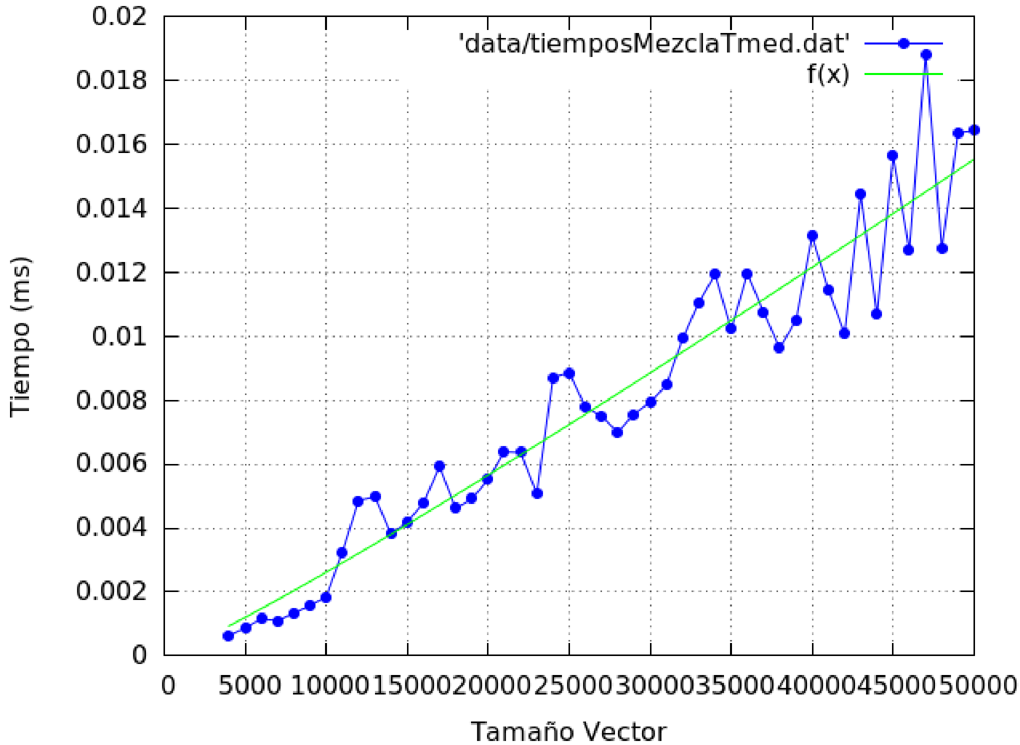
$$f(x) = x(a\log_2(x) - b\log_2(100) + c); \begin{cases} a = & 8.42324e-09 \\ b = & 0.169334 \\ c = & 1.12503 \end{cases}$$

Mezcla - Caso Promedio:

Al igual que en el mejor caso, la eficiencia de Mergesort no varía, pero si la de inserción que ya teníamos calculada.

$$n\log_2(n) - \left(\frac{n}{m}\right) m\log_2(m) + \left(\frac{n}{m}\right)(m^2 + m) = Identica\,al\,peor\,caso$$

$$n\left(\log_2(n) - \log_2(m) + m + 1\right) \in \begin{cases} n > 2^m & \in O(nm) \\ n \leq 2^m & \in O(n\log_2(n)) \end{cases}$$



$$f(x) = x(a \log_2(x) - b \log_2(100) + 100c + d); \begin{cases} a = & 2.12614e-08 \\ b = & 1.06241 \\ c = & 0.0606787 \\ d = & 0.990607 \end{cases}$$

Mezcla - Estudio de UMBRAL_MS:

Primero vamos a comprobar que teóricamente en el caso de m=2 y m=n, recuperamos el total de las eficiencias por separado:

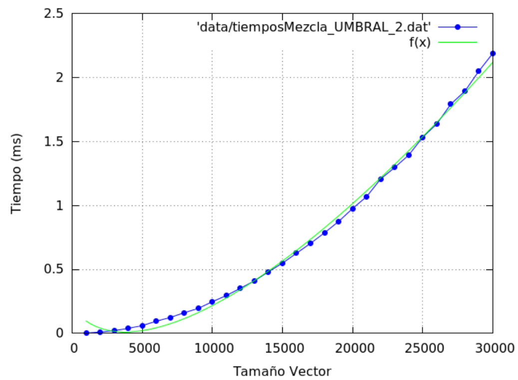
- Caso m=2: $n\left(\log_2(n) - \log_2(2) + 2 + 1\right) = n\left(\log_2(n) - 1 + 2 + 1\right) =$

$$=\text{nlog}_2\text{(n)+2n} \in \begin{cases} n > 2^m & O(nm) \\ n \leq 2^m & O(n\log_2(n)) \end{cases}$$

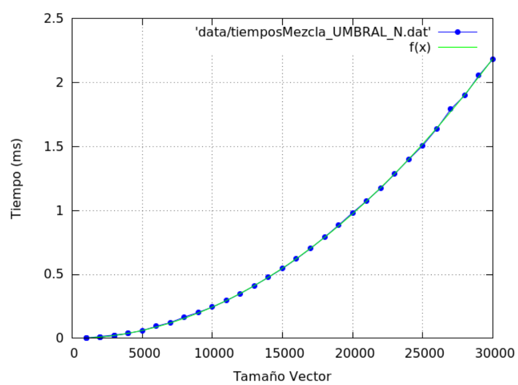
- Caso m=n: $n\left(\log_2(n) - \log_2(n) + n + 1\right) = n^{\mathbf{2}} + n \in O(n^{\mathbf{2}})$

Analizamos ahora los valores de forma empírica:

- Caso m=2:



- Caso m=n:



Ejercicio 5.3

Ordenación por Permutación:

Para la implementación he añadido una función "CheckOrden" en el "fichero pruebapermutacion.cpp" más el cálculo del tiempo.

```
bool CheckOrden(string &c,const Permutacion &P){

    const vector<unsigned int> s= (*(P.begin()));

    std::string::size_type sz;   // alias of size_t

    for (unsigned int i=0;i<s.size()-1;i++){
        if((int)c[s[i]-1] > (int)c[s[i]])
            return false;


    }
    return true;

}
```

Permutación - Peor Caso:

Para el cálculo del peor caso tendremos una serie de números ordenados de mayor a menor por lo que bastará con calcular el coste de crear todas las permutaciones posibles, que son las necesarias para llegar a la permutación (la última) que define el inverso de nuestro vector.

```
unsigned int Permutacion::NumeroPermutacionesPosibles()const{
  int total=1;
  int n= datos[0].size();
  for (int i=2;i<=n;i++)
    total*=i;
  return total;
}
```

Partiendo del cálculo ya implementado podemos obtener la eficiencia:

$$\prod_{i=2}^{n} ia = 2a * 3a * 4a * 5a * \ ... \ * na = n! \in O(n!)$$

Permutación - Mejor Caso:

El mejor caso se dará al ser el vector ordenado, ya que la primera permutación generada coincide con el orden del vector. Estaríamos hablando de una eficiencia de $O(1)$

Permutación - Caso Promedio:

Por último en el caso promedio tendremos que calcular la probabilidad de que la permutación correcta se encuentre en cualquier posición de nuestra lista de permutaciones.

Por lo tanto, si n es el número de permutaciones y 1/n es la probabilidad de acierto:

$$\sum_{i=1}^{n} \frac{1}{n} i = \frac{1}{n} \frac{n}{2} n = \frac{n}{2}$$

Sabiendo el número medio de veces que necesitaremos recorrer nuestra lista de permutaciones en busca de la correcta es de $\left(\frac{n}{2}\right)$:

$$\prod_{i=2}^{\frac{n}{2}} ia = 2a * 3a * 4a * 5a * \ ... \ * \frac{n}{2} a = \frac{n!}{2} \in O(n!)$$

DATOS PC:

00:00.0 Host bridge: Intel Corporation 3rd Gen Core processor DRAM Controller (rev 09) 00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor PCI Express Root Port (rev 09) 00:02.0 VGA compatible controller: Intel Corporation 3rd Gen Core processor Graphics Controller (rev 09) 00:14.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB xHCI Host Controller (rev 04) 00:16.0 Communication controller: Intel Corporation 7 Series/C210 Series Chipset Family MEI Controller #1 (rev 04) 00:1a.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB Enhanced Host Controller #2 (rev 04) 00:1b.0 Audio device: Intel Corporation 7 Series/C210 Series Chipset Family High Definition Audio Controller (rev 04) 00:1c.0 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 1 (rev c4) 00:1c.1 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 2 (rev c4) 00:1c.3 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 4 (rev c4) 00:1d.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB Enhanced Host Controller #1 (rev 04) 00:1f.0 ISA bridge: Intel Corporation HM76 Express Chipset LPC Controller (rev 04) 00:1f.2 SATA controller: Intel Corporation 7 Series Chipset Family 6-port SATA Controller [AHCI mode] (rev 04) 00:1f.3 SMBus: Intel Corporation 7 Series/C210 Series Chipset Family SMBus Controller (rev 04) 01:00.0 VGA compatible controller: NVIDIA Corporation GF108M [GeForce GT 630M] (rev ff) 03:00.0 Network controller: Intel Corporation Centrino Wireless-N 2230 (rev c4) 04:00.0 Ethernet controller: Qualcomm Atheros AR8161 Gigabit Ethernet (rev 08)