ESTRUCTURAS DE DATOS

PRACTICA 1

Jose Manuel Martínez de la Insua

3) Ejercicio_desc.cpp

```
1       int operacion(int *v, int n, int x, int inf, int sup) {
2               int med;
3               bool enc=false;
4               while ((inf<sup) && (!enc)) {
5                       med = (inf+sup)/2;
6                       if (v[med]==x)
7                               enc = true;
8                       else if (v[med] < x)
9                               inf = med+1;
10                      else
11                              sup = med-1;
12              }
13              if (enc)
14                      return med;
15              else
16                       return -1;
17      }
```

El algoritmo trata de buscar un entero en un vector de enteros. El método empleado para ello es preguntar si el valor se encuentra justo en la mitad de dicho vector, en caso de no encontrarlo, divide el vector en dos partes y se queda con la mitad en la cual se encuentre x y repite operación reiteradamente hasta encontrar x o recorrer por completo el vector (entendemos pues que el vector debe estar ordenado, en caso contrario éste algoritmo no sería efectivo).

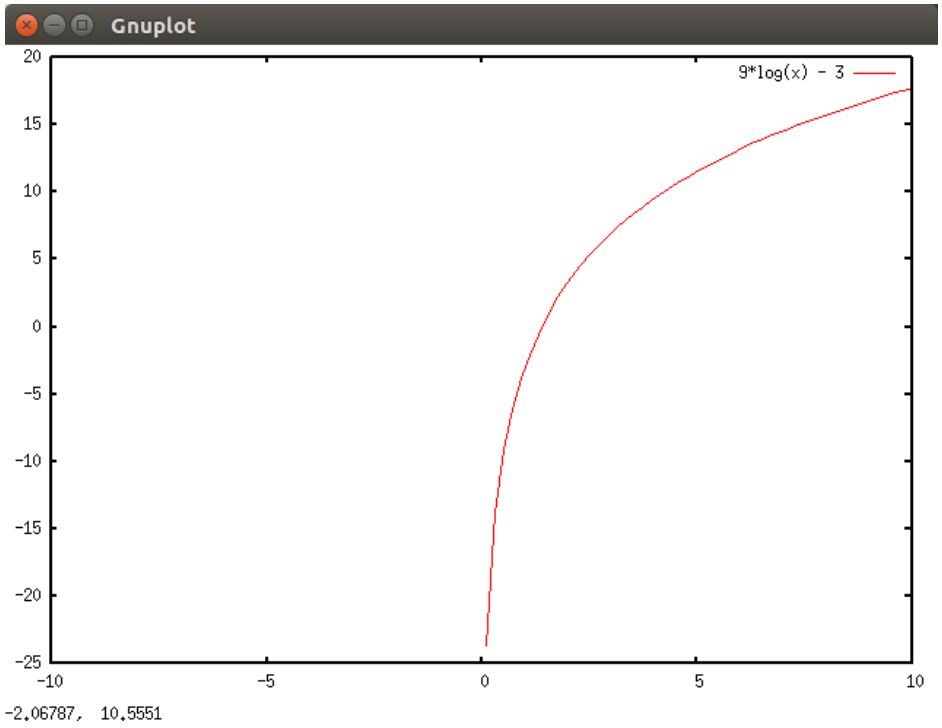
<u>Eficiencia teórica:</u>

- Línea 2: 1OE : declaración
- Línea 3: 2OE: declaración y asignación.
- Línea 4: 2OE: dos evaluaciones de condición
- Línea 5: 3OE: suma , división y asignación.
- Línea 6: 2OE: indexación y evaluación de condición.
- Línea 7: 1OE: asignación.
- Línea 8: 2OE: indexación y evaluación de condición.
- Línea 9: 2OE: suma y asignación.
- Línea 11: 2OE: resta y asignación.
- Línea 13: 1OE: evaluación de condición.
- Línea 14: 1OE: retorno de valor.
- Línea 16: 1OE: retorno de valor.

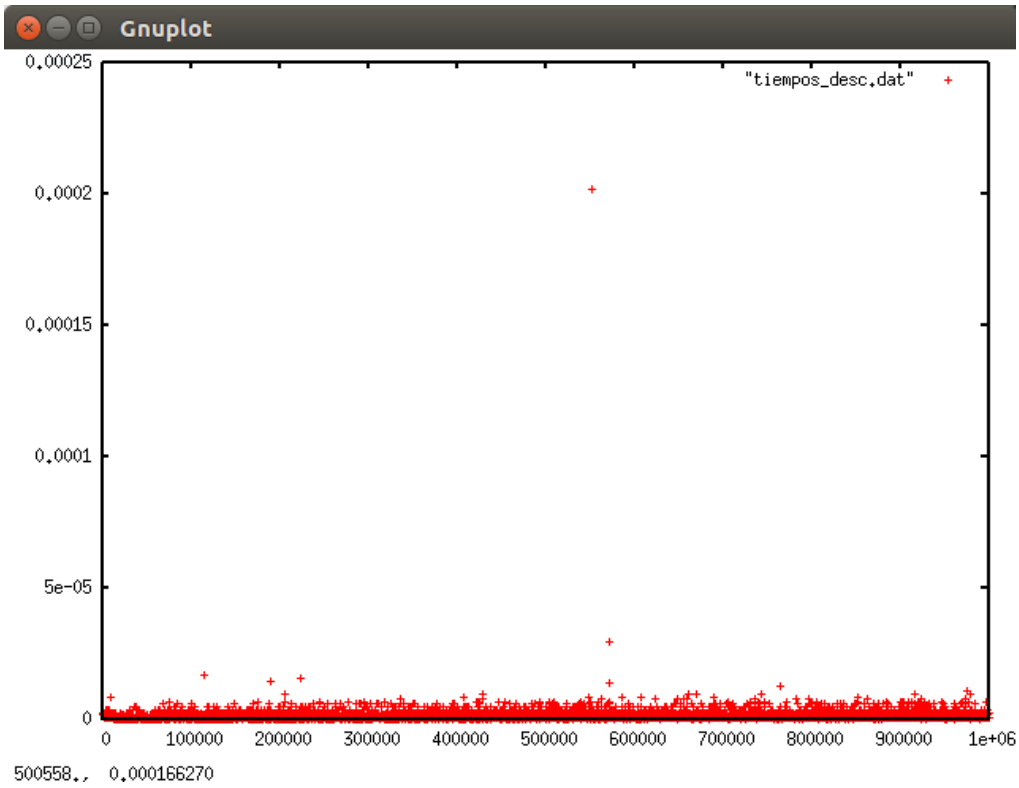
$$1+2+2+\left(\sum_{i=0}^{\log(n)} 2+3+2+max(1,2,2)\right)+max(1,1) \quad = \quad 5 + (\log_2(n) - 1) \text{ x } 9 + 1$$

$$= \quad \mathbf{9 \text{ x } \log_2(n) - 3 \ \in \ O(\log(n))}$$

Eficiencia teórica:

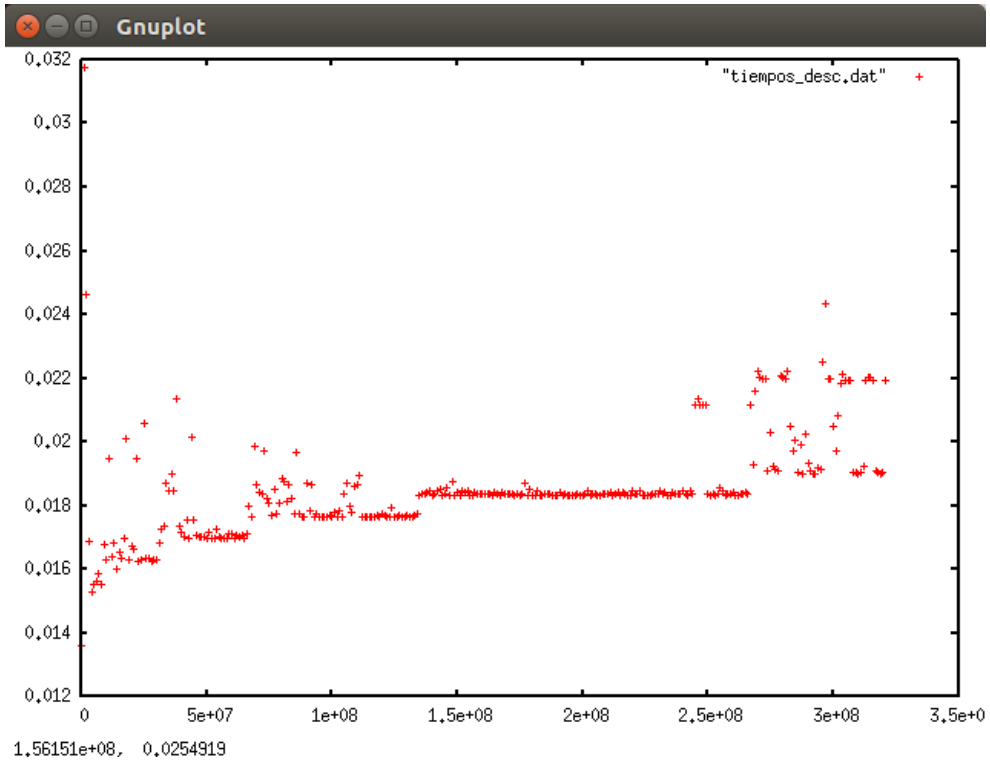


Eficiencia empírica:



Está sucediendo que el algoritmo que estamos empleando es muy rápido " O(log(n)) " y los tiempos son muy cercanos a 0 o incluso 0 ya que ctime no tiene tanta precisión como sería deseable para medir el tiempo en este caso. Una posible solución sería ejecutar el algoritmo muchas veces y dividir el tiempo obtenido entre el número de ejecuciones.

Ejecutando el algoritmo 100.000 veces para cada tamaño del vector obtenemos la siguiente gráfica:



Ya se puede apreciar la forma logarítmica.

<u>Regresión y ajuste:</u>