EJERCICIO 3 : Problemas de precisión

Autor : Jesús Ruiz Castellano, 76439001-L

1.- Código fuente : ejercicio_desc.cpp

```cpp
#include <iostream>
#include <ctime>    // Recursos para medir tiempos
#include <cstdlib>  // Para generación de números pseudoaleatorios

using namespace std;

// Algoritmo de busqueda con pivote

int operacion(int *v, int n, int x, int inf, int sup) {
  int med;
  bool enc=false;
  while ((inf<sup) && (!enc)) {
    med = (inf+sup)/2;
    if (v[med]==x)
      enc = true;
    else if (v[med] < x)
      inf = med+1;
    else
      sup = med-1;
  }
  if (enc)
    return med;
  else
    return -1;
}

void sintaxis()
{
  cerr << "Sintaxis:" << endl;
  cerr << "  TAM: Tamaño del vector (>0)" << endl;
  cerr << "Se genera un vector de tamaño TAM con elementos aleatorios" << endl;
  exit(EXIT_FAILURE);
}
int main(int argc, char * argv[])
{

  // Tamaño del vector
  int tam=atoi(argv[1]);

  if (tam<=0)
    sintaxis();

  // Generación del vector aleatorio
  int *v=new int[tam];       // Reserva de memoria
  srand(time(0));            // Inicialización del generador de números pseudoaleatorios
  for (int i=0; i<tam; i++)  // Recorrer vector
    v[i] = rand() % tam;

  clock_t tini;    // Anotamos el tiempo de inicio
  tini=clock();

  // Algoritmo a evaluar
  for ( int i = 0; i < 10000 ; i++ )
     operacion(v,tam,tam+1,0,tam-1);

  clock_t tfin;    // Anotamos el tiempo de finalización
  tfin=clock();

  // Mostramos resultados
  cout << tam << "\t" << ((tfin-tini)/(double)CLOCKS_PER_SEC) / 10000.0 << endl;

  delete [] v;     // Liberamos memoria dinámica
}
```

EJERCICIO 3 : Problemas de precisión

2.- Hardware usado:

2.1- CPU

vendor_id : GenuineIntel
model name : Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz
cpu MHz : 933.000

2.2- Velocidad de Reloj

Versión : hwclock de util-linux 2.20.1
mar 11 oct 2016 23:45:36 CEST -0.573198 segundos

2.3- Memoria RAM

MemTotal : 3907668 kB
SwapTotal : 4049916 kB

3.- Sistema Operativo

Ubuntu 14.04.3 LTS
Arquitectura : x86_64 (64 bits)

4.- Compilador usado y opciones de compilación

gcc - GNU project C and C++ compiler
Opción de compilación : g++ -o <nombre_ejecutable> <ejecutable.cpp>
g++ -o ejercicio_desc ejercicio_desc.cpp

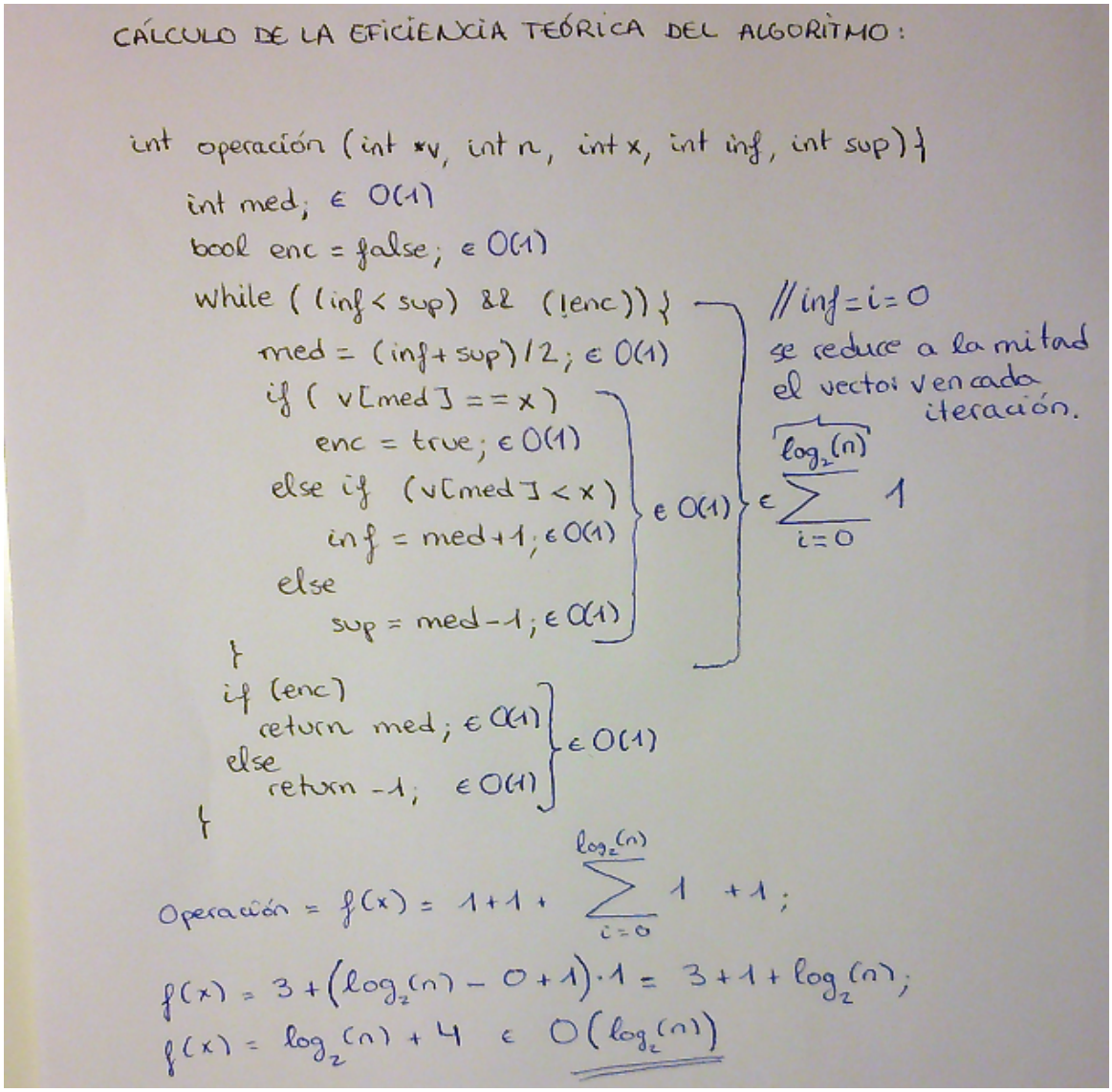
EJERCICIO 3 : Problemas de precisión

5.- Descripción del algoritmo

Se trata del algoritmo de Búsqueda binaria. Partimos de la base de un vector ya ordenado, de menor a mayor.
Éste algoritmo utiliza una variable como pivote, llamada "med". Ésta variable hace referencia al punto medio del vector (med = (sup + inf)/2), siendo "sup" la posición superior del vector (v[n-1]) e "inf" la posición inferior (v[0]) del vector, v.
"med" se utiliza tanto para buscar en el vector a un lado y a otro suya, y en caso de encontrar el valor "x", que buscamos, devolvemos med, que indicará la posición de v donde se encuentra el elemento buscado.

6.- Desarrollo completo del cálculo de la Eficiencia teórica

EFICIENCIA TEÓRICA

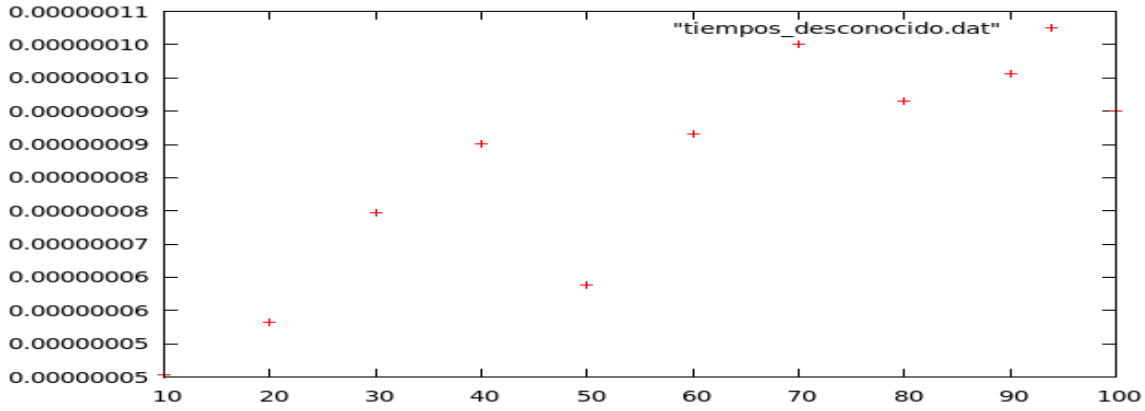


EJERCICIO 3 : Problemas de precisión

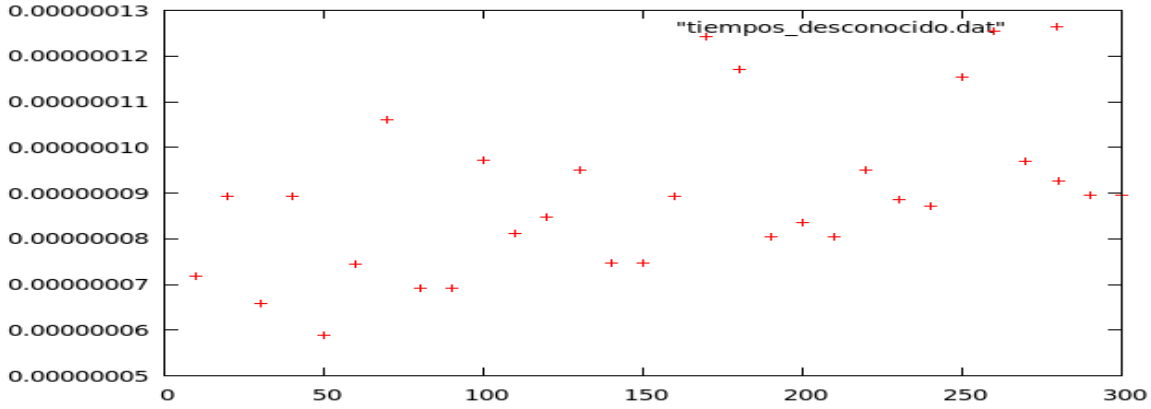
7.- Parámetros usados para el cálculo de la eficiencia empírica y gráfica

Para ésta parte he ejecutado el programa con los siguientes valores para tamaño máximo del vector :

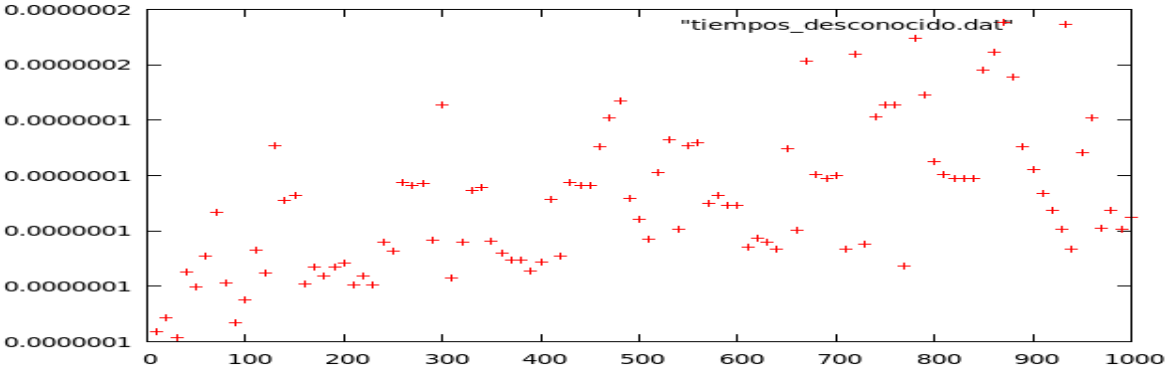
6.1- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 100



6.2- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 300

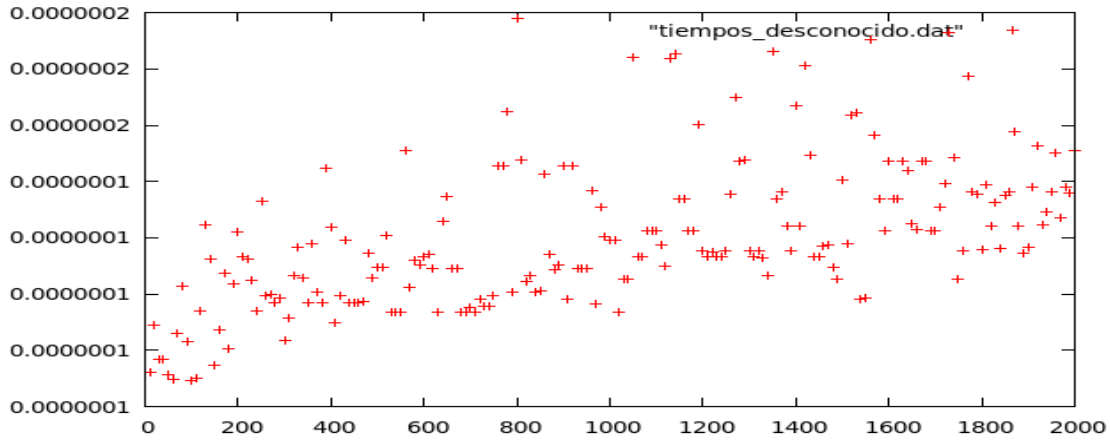


6.3- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 1000

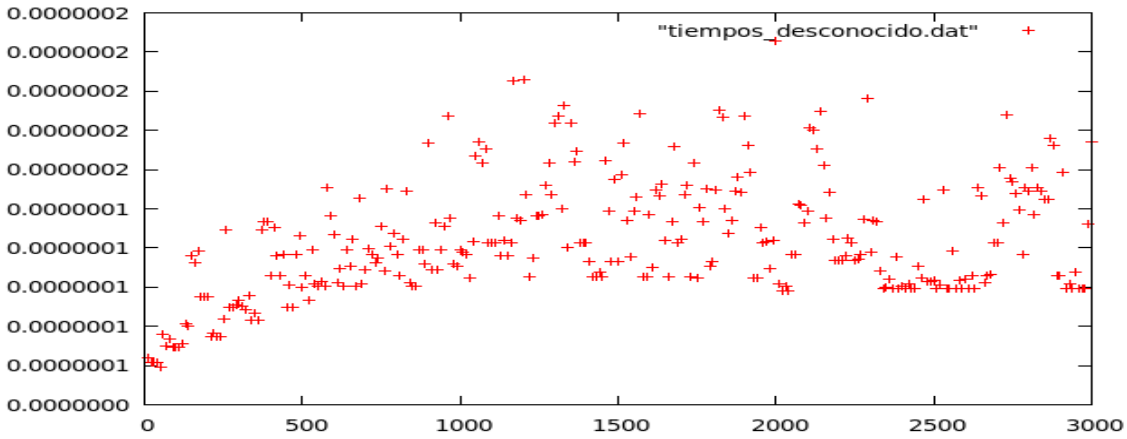


EJERCICIO 3 : Problemas de precisión

6.4- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 2000



6.5- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 3000



6.6- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 6000