¿QUÉ HACE EL ALGORITMO?

Se trata del algoritmo de búsqueda binaria.
Se debe partir de un vector ordenado.
Divide el vector inicial en dos, compara el valor central truncado con el elemento que buscamos y descarta una de las mitades del vector (si el elemento a buscar es mayor que el central descartamos la mitad izquierda y viceversa). Cuando se encuentra el dato se devuelve su posición en el vector.

EFICIENCIA TEÓRICA

La eficiencia teórica del algoritmo de búsqueda binaria es $Log_2(x)$

EFICIENCIA EMPÍRICA

La eficiencia empírica de este algoritmo es altísima y su tiempo de ejecución es muy próximo a 0,0s.

suma.cpp

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


using namespace std;

void sintaxis()
{
  cerr << "Sintaxis:" << endl;
  cerr << "  fs: Fichero del que se extraen los datos" << endl;
  cerr << "Se genera un vector con los datos del ficheto fs" << endl;
  exit(EXIT_FAILURE);
}

int main(int argc, char * argv[])
{
  // Lectura de parámetros
  if (argc!=2)
    sintaxis();

    ifstream fs(argv[1]); // Creacion de fichero

    double v[5000]; // creacion de vector con un máximo de 5000 elementos

    int c = 0, i = 0; // c: variable tamaño del vector; i: índice del vector
    while(!fs.eof()){ // leemos los datos del fichero hasta el final del mismo
        fs >> v[i];
        c++; i++;
    }


    double suma = 0.0; // inicializamos suma

    for (int j=0; j<c; j++){  // Recorrer vector
        //cout << v[j] << endl;
        suma += v[j];
    }

  // Mostramos resultados
  cout << endl << endl << suma/c << endl;
}
```

Para obtener un resultado mas preciso realizamos muchas iteraciones del proceso calculando su tiempo de ejecución. Sumamos todos estos tiempos y los dividimos entre el número total de iteraciones para hallar un tiempo medio.

COMO RESULTADO OBTENEMOS:
6.52381e-07 sec