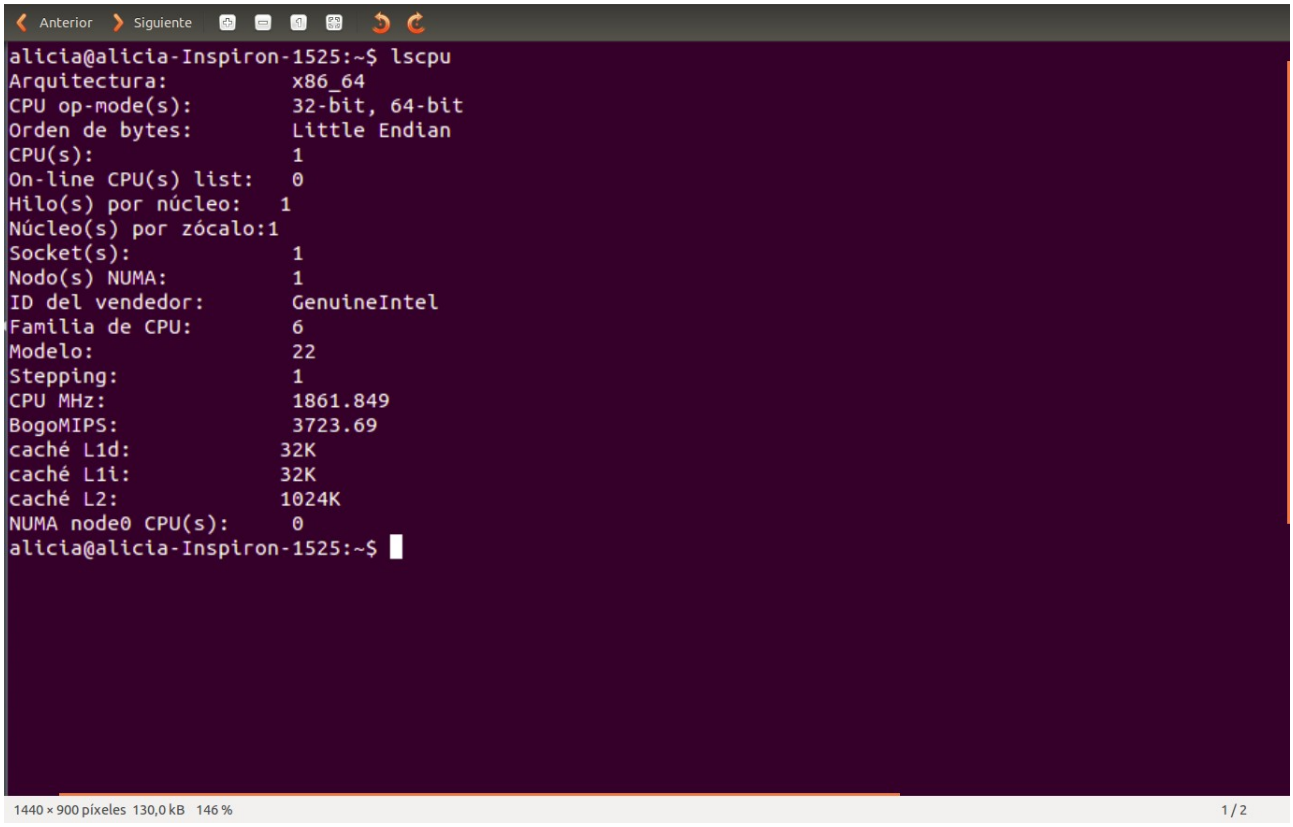


Ejercicio 3: Problemas de precisión

Junto con este guión se le ha suministrado un fichero ejercicio_desc.cpp. En él se ha implementado un algoritmo. Se pide que:

- **Explique qué hace este algoritmo.**
- **Calcule su eficiencia teórica.**
- **Calcule su eficiencia empírica.**

Si visualiza la eficiencia empírica debería notar algo anormal. Explíquelo y proponga una solución. Compruebe que su solución es correcta. Una vez resuelto el problema realice la regresión para ajustar la curva teórica a la empírica.

¿Qué hace este algoritmo?:

1) Se ha creado un vector te tamaño “tam” con elementos aleatorios entre 0 y “tam”. Luego no está ordenado.
2) Desde en main se llama a la función con los siguientes parametros:
operacion(v,tam,tam+1,0,tam-1);
3)El algoritmo es:

```
int operacion(int *v, int n, int x, int inf, int sup) {  //x es el elemento a buscar
    int med;
    bool enc=false;
    while ((inf<sup) && (!enc)) {
        med = (inf+sup)/2; //punto medio
        if (v[med]==x)          ← Este if no se va a cumplir nunca, porque x=tam+1 y
                                  el vector solo contiene numeros del 0 hasta “tam”
            enc = true;
        else if (v[med] < x)
            inf = med+1;
        else
            sup = med-1;
    }
    if (enc)
        return med;
    else
        return -1;
}
```

Este algoritmo es el algoritmo de busqueda binaria, el cual consiste en que el elemento que estamos buscando se compara con el elemento que ocupa la mitad del vector, si coinciden se termina la búsqueda, si no, se determina la mitad del vector en la que puede estar el elemento y se repite el proceso. Para ello el vector debe estar ordenado,cosa que no esta porque esta creado con elementos aleatorios entre 0 y tam.

Eficiencia Teórica:

$$T(n) = 1+2+3+\sum_{inf=0}^{sup-1}(3+2+2+3)+1+1 = 8+\sum_{inf=0}^{sup-1}10 = 8+10sup \approx 8+10\,n \in \mathbf{O(n)}$$

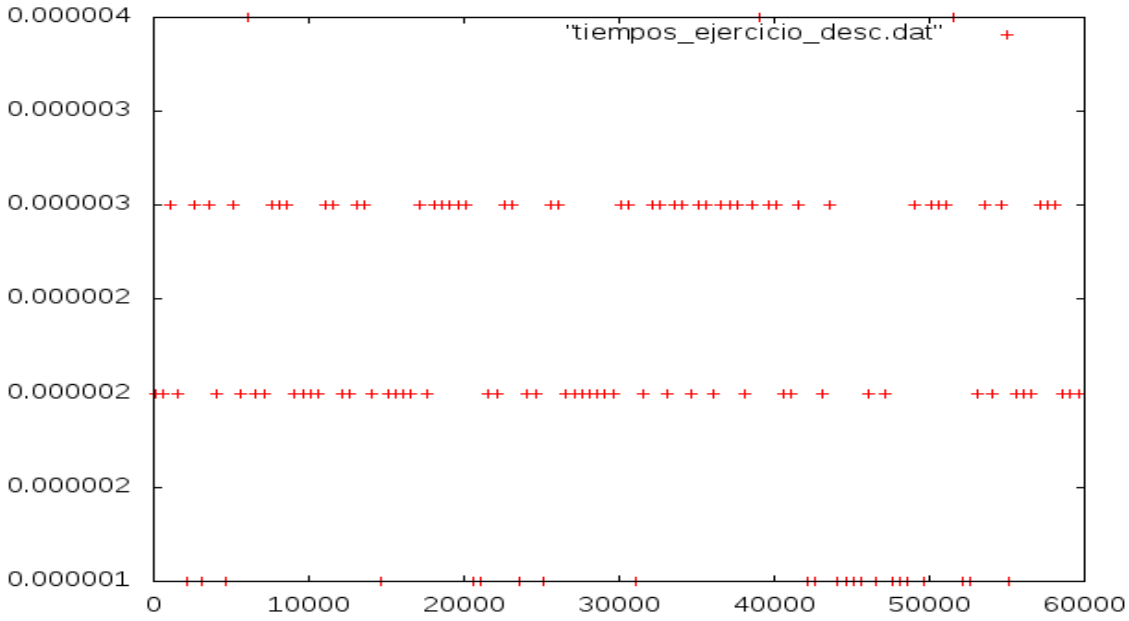
Eficiencia Empírica:

1) Creamos el ejecutable con : **g++ ejercicio_desc.cpp -o ejercicio_desc**

2) Creo el script **ejecuciones_ejercicio_desc.csh** para ejecutar varias veces el programa anterior para tamaños del vector 100, 600, 1000, ..., 60000 y generar un fichero con los datos obtenidos.

3)Para dibujar los datos obtenidos en el apartado anterior uso gnuplot:

gnuplot> plot "tiempos_ejercicio_desc.dat"



El resultado de la eficiencia empírica es una gráfica con varios valores repetidos en el eje de ordenadas. Para un mismo tamaño resultan tiempos diferentes.

Mi solución:

1) Crear un vector te tamaño “tam” con elementos ordenados de menor a mayor.

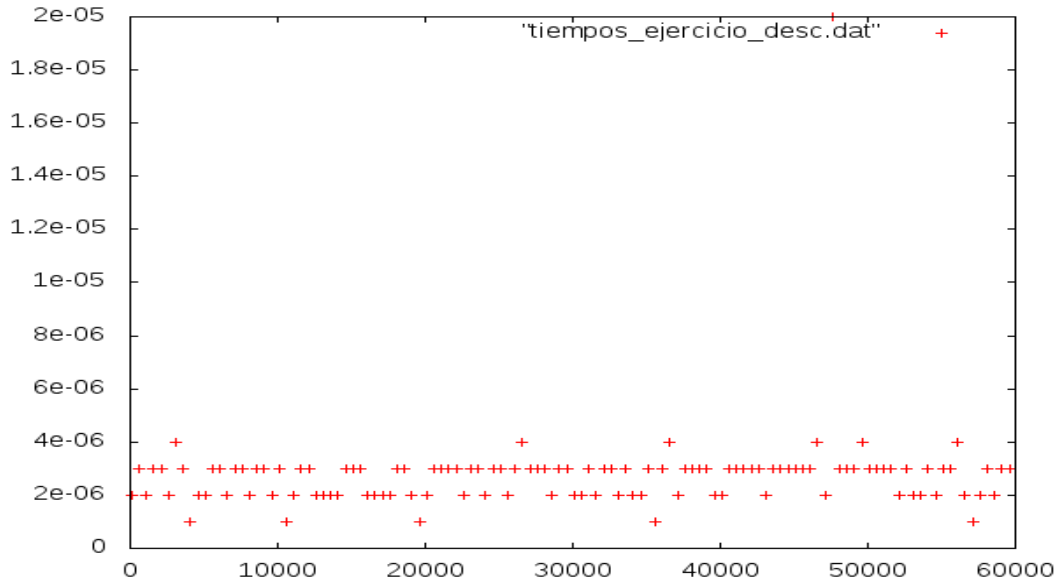
2) Desde en main llamar a la función con los siguientes parametros:
operacion(v,tam+1,0,tam-1);

3)El algoritmo es:

```
int operacion(int *v, int x, int inf, int sup) {  //x es el elemento a buscar
    int med;
    bool enc=false;
    while ((inf<=sup) && (!enc)) {
        med = (inf+sup)/2; //punto medio
        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;
```

```
}
```

Ahora la eficiencia empirica queda de la siguiente forma:



Regresión para ajustar la curva teórica a la empírica:

1) **gnuplot> f(x)=a*x+b**

2) **gnuplot> fit f(x) "tiempos_ejercicio_desc.dat" via a,b**

```
a               = 1.42371e-11   +/- 8.975e-12   (63.04%)
b               = 2.34169e-06   +/- 3.097e-07   (13.23%)
```

3)**gnuplot>plot "tiempos_ejercicio_desc.dat" , f(x)**