

ESTRUCTURA DE DATOS

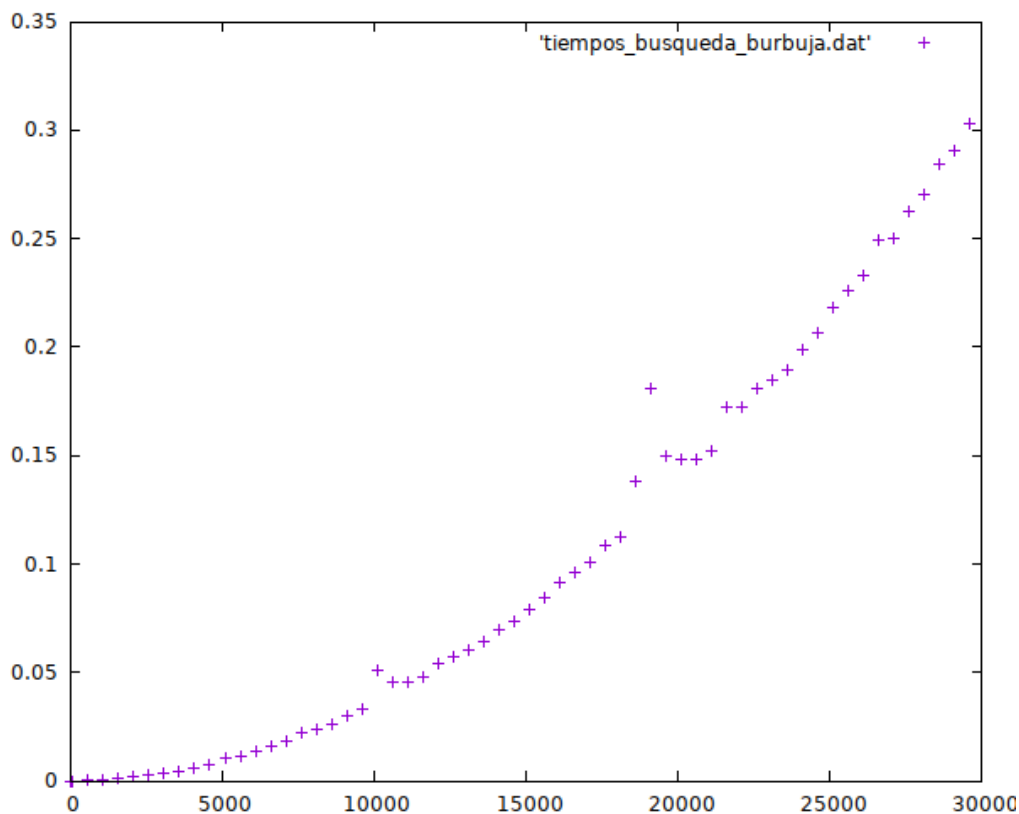
GRADO INGENIERIA INFORMATICA (2016 – 2017)

UNIVERSIDAD DE GRANADA

Cuaderno Practica 1: Eficiencia

Ejercicio 1: *Calcular la eficiencia del algoritmo burbuja, un fichero ordenación, un script para las ejecuciones y usar el gnuplot para dibujar los datos obtenidos.*

Los archivos están en la carpeta del ejercicio1. La grafica optenida es:



Antonio Miguel Morillo Chica
77379021Y

Ejercicio 2: Replique el experimento de ajuste por regresión a los resultados obtenidos en el ejercicio 1 que calculaba la ediciencia del algoritmo de ordenación burbuja. Para ell considere $f(x)$ es de la forma $ax^2 + bx + c$

```
gnuplot> fit f(x) 'tiempos_busqueda_burbuja.dat' using 1:2 via a, b, c
iter      chisq      delta/lim  lambda  a          b          c
0 9.4779954273e+18  0.00e+00  2.28e+08  1.000000e+00  1.000000e+00  1.000000e+00
1 2.7995135097e+14 -3.39e+09  2.28e+07  5.393130e-03  9.999583e-01  1.000000e+00
2 1.1087725752e+09 -2.52e+10  2.28e+06 -4.158266e-05  9.999560e-01  1.000000e+00
3 1.1074628991e+09 -1.18e+02  2.28e+05 -4.187068e-05  9.997421e-01  1.000000e+00
4 1.0615637635e+09 -4.32e+03  2.28e+04 -4.099374e-05  9.788028e-01  9.999972e-01
5 1.0773748134e+08 -8.85e+05  2.28e+03 -1.305677e-05  3.117303e-01  9.999077e-01
6 2.3408892926e+03 -4.60e+09  2.28e+02 -5.671196e-08  1.317850e-03  9.998646e-01
7 7.9723291283e+00 -2.93e+07  2.28e+01  4.061315e-09 -1.332733e-04  9.997109e-01
8 7.7337031847e+00 -3.09e+03  2.28e+00  4.007573e-09 -1.313146e-04  9.845903e-01
9 1.2095929022e+00 -5.39e+05  2.28e-01  1.772605e-09 -5.126901e-05  3.873441e-01
10 3.6133486742e-03 -3.34e+07  2.28e-02  3.229399e-10  6.508622e-07 -4.697946e-05
11 3.5626086388e-03 -1.42e+03  2.28e-03  3.134762e-10  9.898060e-07 -2.575950e-03
12 3.5626086386e-03 -6.07e-06  2.28e-04  3.134756e-10  9.898281e-07 -2.576115e-03
iter      chisq      delta/lim  lambda  a          b          c
```

After 12 iterations the fit converged.
final sum of squares of residuals : 0.00356261
rel. change during last iteration : -6.06961e-11

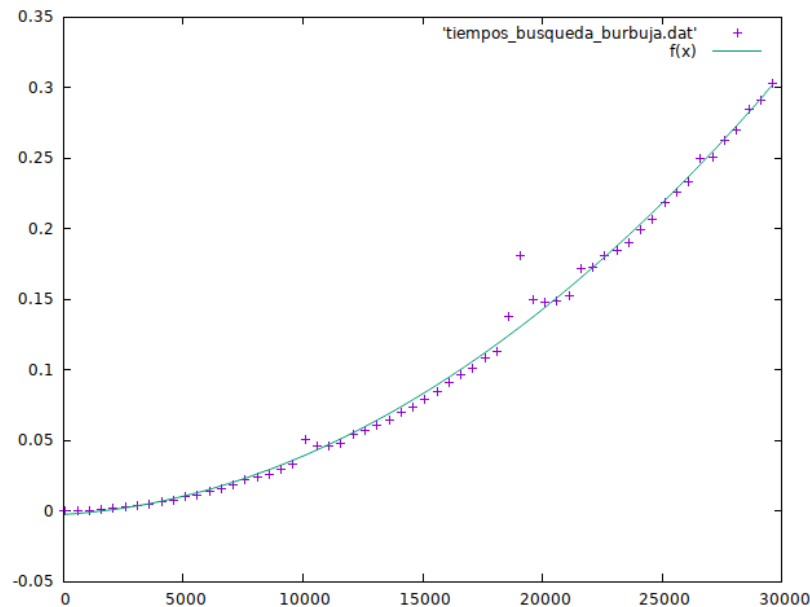
degrees of freedom (FIT_NDF) : 58
rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 0.00783736
variance of residuals (reduced chisquare) = WSSR/ndf : 6.14243e-05

| Final set of parameters | Asymptotic Standard Error |
|-------------------------|---------------------------|
| a = 3.13476e-10 | +/- 1.457e-11 (4.647%) |
| b = 9.89828e-07 | +/- 4.411e-07 (44.56%) |
| c = -0.00257611 | +/- 0.002783 (108%) |

correlation matrix of the fit parameters:

| | a | b | c |
|---|--------|--------|-------|
| a | 1.000 | | |
| b | -0.966 | 1.000 | |
| c | 0.715 | -0.846 | 1.000 |

```
gnuplot> plot 'tiempos_busqueda_burbuja.dat', f(x)
```



Antonio Miguel Morillo Chica
77379021Y

Ejercicio3: Explicar algoritmo de ejercicio_desc.cpp, calcular su eficiencia teorica y empirica. Al visualizar su eficiencia empírica debería notar algo anormal. Explíquelo y ponga una solución. Una vez resuelto realice la regresión para ajustar la curva teórica a la empírica.

- Explicación del código:

Lo que hace la función operación es hacer una búsqueda de un valor concreto x en el vector, para ello lo parte por la mitad y actualiza el valor inf o sup dependiendo si el dato se encuentra en uno de los dos extremos, así repetidas veces hasta que lo encuentra. Es como coger una regla de 20 cm, partir por los 10 cm y ver si el dato es menor o mayor de 10, si es mayor inf pasa de 0 a 11 y sup se queda igual (30) así varias veces hasta que se encuentra.

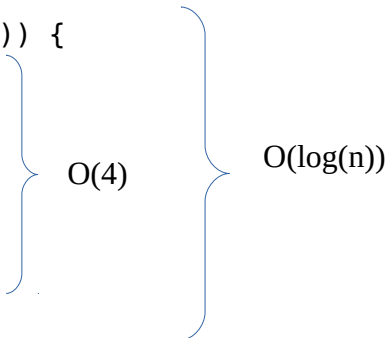
El problema viene en que este algoritmo solo funciona bien si el vector está ordenado en caso contrario esta técnica no tiene sentido. Tras buscar información he encontrado que este algoritmo se llama búsqueda binaria.

- Eficiencia teorica:

```
int operacion(int *v, int n, int x, int inf, int sup) {
    int med;
    bool enc=false;

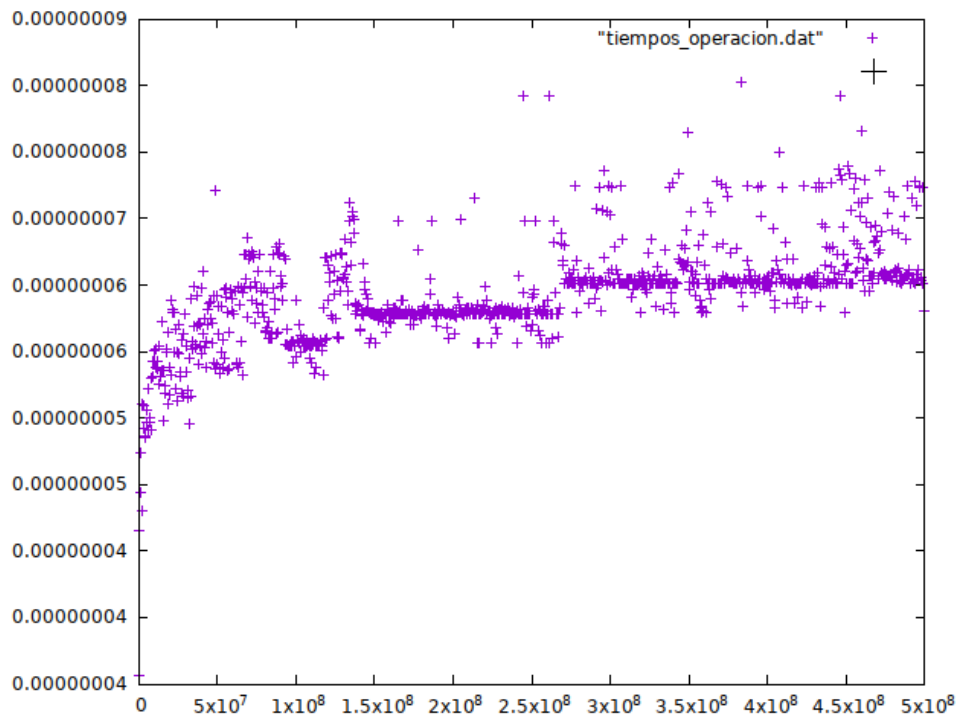
    while ((inf<sup) && (!enc)) {
        med = (inf+sup)/2;
        if (v[med]==x)
            enc = true;
        else if (v[med] < x)
            sup = med+1;
        else
            inf = med-1 ;
    }

    if (enc)
        return med;
    else
        return -1;
}
```



Usando la notación O-grande que hemos visto en teoría sabemos que su eficiencia es logarítmica.

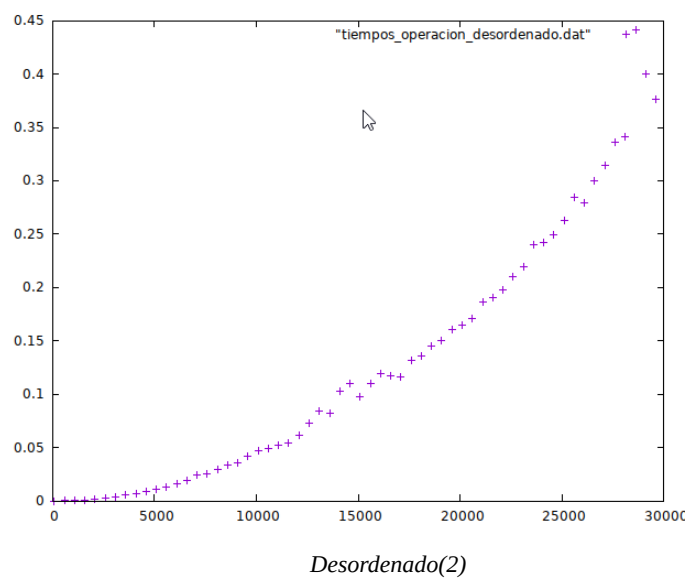
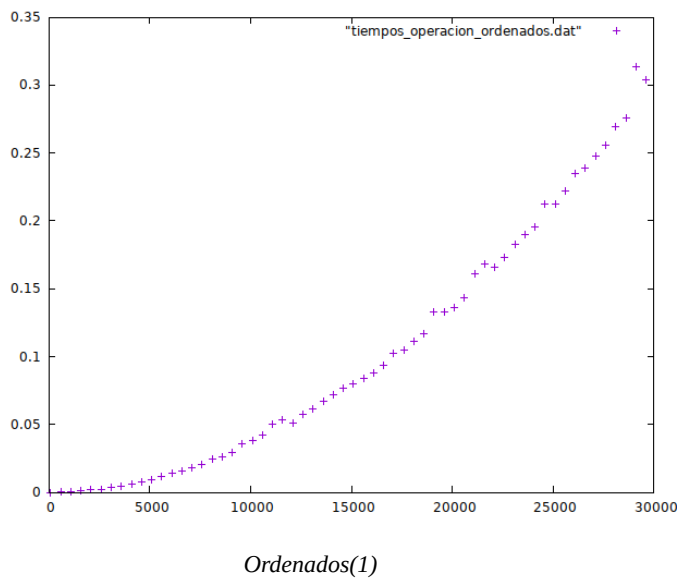
- Eficiencia:



Ejercicio 4: Retomar el ejercicio anterior y hacer unas modificaciones:

- Para el mejor caso posible.
- Para el peor caso posible.

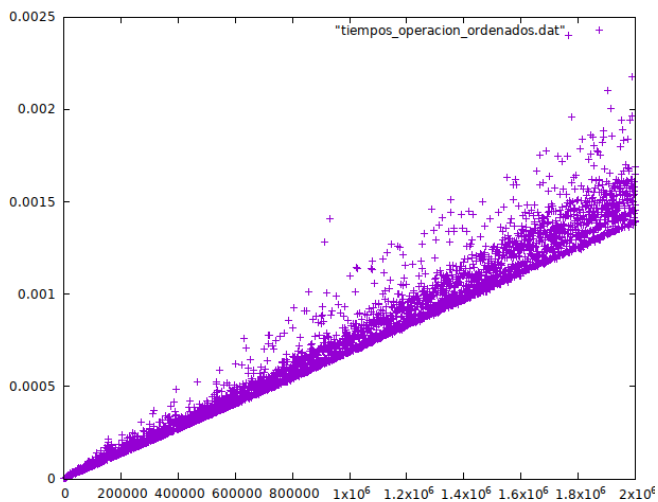
Calcular la eficiencia empirica en ambos casos.



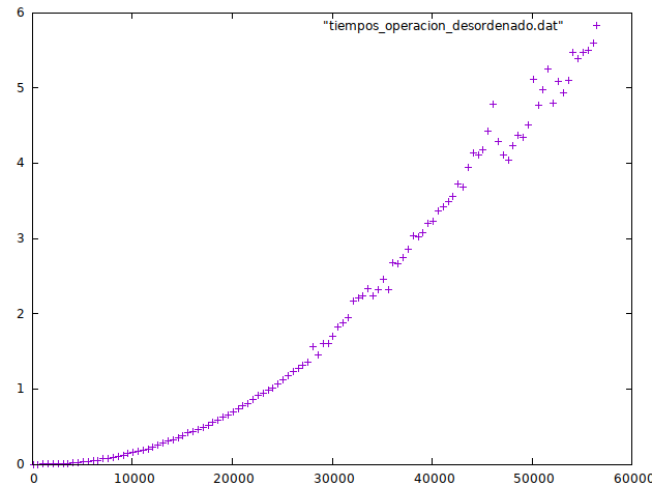
Como vemos Los tiempos obtenidos son siempre semejantes. Esto es debido a que en la operación ordenar siempre se ejecutan todas las iteraciones de los bucles, es decir no tenemos forma de saber si estamos ante el mejor o el peor caso. El algoritmo no contempla este hecho por eso el promedio, el caso peor y el mejor siempre es el mismo.

***Nota:** Si los tiempos del ejercicio 1 son mayores es por no haber compilado con -O2.

Ejercicio 5: Considere el código burbuja dado, que está modificado para que no ocurra el problema del ejercicio 4. Calcule pues, el mejor y el peor caso de nuevo.



Ordenado(1)



Desordenado(2)

Como vemos en la eficiencia empírica el crecimiento en el vector ordenado sería $O(n)$ en cambio en el del desordenado sería n^2 o n^3 . Ahora vamos a compararlo con la eficiencia teórica:

- Mejor caso:

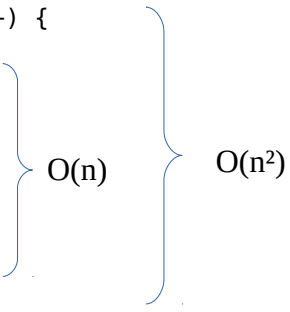
```
void ordenar(int *v, int n){
    bool cambio=true;
    for (int i=0; i<n-1 && cambio; i++) {
        cambio=false;
        for (int j=0; j<n-i-1; j++) {
            if (v[j]>v[j+1]) {
                cambio = true;
                int aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
            }
        }
    }
}
```

$O(5)$ $O(n+5)$ $O(n)$

Solo se ejecuta n veces las comparaciones de todo el vector, se recorre una vez completamente.

- Peor caso:

```
void ordenar(int *v, int n){
    bool cambio=true;
    for (int i = 0; i < n-1 && cambio; i++) {
        cambio = false;
        for (int j = 0; j < n-i-1; j++) {
            if (v[j]>v[j+1]) {
                cambio = true;
                int aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
            }
        }
    }
}
```



Ejecutas todas las iteraciones de todos los bucles.

Conclusión: Tanto la eficiencia teorica como la empírica se ajustan a nuestra previsión-.