

EFICIENCIA TEÓRICA

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

Diagram illustrating the complexity analysis of the bubble sort algorithm:

- The innermost loop (if statement) has a complexity of $O(1)$.
- The middle loop (for j) has a complexity of $O(n-i)$.
- The outer loop (for i) has a complexity of $O(n)$.
- The overall complexity is $O(n^2)$.

SCRIPT DE EJECUCIÓN DE C-SHELL

```
ordenacion.cpp x ejecuciones.csh x
1  #!/bin/csh
2  @ inicio = 100
3  @ fin = 30000
4  @ incremento = 500
5  @ i = $inicio
6
7  g++ ordenacion.cpp -o ordenacion
8
9  echo > tiempos2.dat
10
11 while ( $i <= $fin )
12     echo Ejecución tam = $i
13     echo `./ordenacion $i 10000` >> tiempos2.dat
14     @ i += $incremento
15 end
16
```

ORDENACION.CPP

```
ordenacion.cpp x ejecuciones.csh x
1 #include <iostream>
2 #include <ctime> // Recursos para medir tiempos
3 #include <cstdlib> // Para generación de números pseudoaleatorios
4
5 using namespace std;
6
7 void ordenar(int* v, int n)
8 {
9     for(int i = 0; i < n-1 ; i++)
10     {
11         for(int j = 0 ; j < n-i-1 ; j++)
12         {
13             if (v[j] > v[j+1])
14             {
15                 int aux = v[j];
16                 v[j] = v[j+1];
17                 v[j+1] = aux;
18             }
19         }
20     }
21 }
22
23 void sintaxis()
24 {
25     cerr << "Sintaxis:" << endl;
26     cerr << " TAM: Tamaño del vector (>0)" << endl;
27     cerr << " VMAX: Valor máximo (>0)" << endl;
28     cerr << "Se genera un vector de tamaño TAM con elementos aleatorios en [0,VMAX[" << endl;
29     exit(EXIT_FAILURE);
30 }
31
32 int main(int argc, char * argv[])
33 {
34     // Lectura de parámetros
35     if (argc!=3)
36         sintaxis();
37     int tam=atoi(argv[1]); // Tamaño del vector
38     int vmax=atoi(argv[2]); // Valor máximo
39     if (tam<=0 || vmax<=0)
40         sintaxis();
41
42     // Generación del vector aleatorio
43     int *v=new int[tam]; // Reserva de memoria
44     srand(time(0)); // Inicialización del generador de números pseudoaleatorios
45     for (int i=0; i<tam; i++) // Recorrer vector
46         v[i] = rand() % vmax; // Generar aleatorio [0,vmax[
47
48     clock_t tini; // Anotamos el tiempo de inicio
49     tini=clock();
50
51     ordenar(v, tam); // Ordenación por burbuja
52
53     clock_t tfin; // Anotamos el tiempo de finalización
54     tfin=clock();
55
56     // Mostramos resultados
57     cout << tam << "\t" << (tfin-tini)/(double)CLOCKS_PER_SEC << endl;
58
59     delete [] v; // Liberamos memoria dinámica
60 }
61
```

REPRESENTACIÓN DE LOS DATOS EMPÍRICOS EN GNUPLOT

