II Doble Grado Ingieniería Infomática y Matemáticas

Estructura de Datos

Práctica 1 - Eficiencia

Pedro Bonilla Nadal, Sofía Almeida Bruno,
Jesús Sánchez de Lechina Tejada

Ejercicio 1. El siguiente código realiza la ordenación mediante el algoritmo de la burbuja:

```
1. void ordenar(int *v, int n) {
2.       for (int i=0; i¡n-1; i++)
3.            for (int j=0; j < n-i-1; j++) {
4.                 if (v[j] > v[j+1]) {
5.                      int aux = v[j];
6.                      v[j] = v[j+1];
7.                      v[j+1] = aux;
8.                 }
9.            }
10. }
```

Calcule la eficiencia teórica de este algoritmo. A continuación replique el experimento que se ha hecho antes (búsqueda lineal) con este nuevo código. Debe:

- Crear un fichero ordenacion.cpp con el programa completo para realizar una ejecución del algoritmo.
- Crear un fichero ejecuciones_ordenación.csh que permite ejecutar varias veces el programa anterior y generar un fichero con los datos obtenidos.
- Usar gnuplot para dibujar los datos obtenidos en el apartado previo.

Los datos deben contener tiempos de ejecución para tamaños del vector 100, 600, 1100, ..., 30000. Pruebe a dibujar superpuestas la función con la eficiencia teórica y la empírica. ¿Qué sucede?

Solución.
El archivo ordenación.cpp y ejecuciones_ordenacion.csh se encuentran adjuntos en la práctica.

Eficiencia teórica:
Línea 2: 5OE (2 asignaciones, 3 operaciones aritmético-lógica).
Línea 3: 6OE (2 asignaciones, 4 operaciones aritmético-lógica).
Línea 4-7: 13OE (6 accesos a vector, 4 operaciones artimético-lógicas, 3 asignaciones).

Entonces:= $1 + \sum_{i=0}^{n-2}(5 + \sum_{j=0}^{n-i-2}(5+13)) = 1 + \sum_{i=0}^{n-2}(5 + ((n-i-2)*18)) =$
$1 + \sum_{i=0}^{n-2}(5) + \sum_{i=0}^{n-2}(18n) + \sum_{i=0}^{n-2}(18n) + \sum_{i=0}^{n-2}(18i) + \sum_{i=0}^{n-2}(18) = -4 + 4n + 9n^2$.

Nota: debemos considerar que para ordenar un vector debe tener al menos 2 elementos.

Ejercicio 5. Dependencia de la implementación:

```
1. void ordenar(int *v, int n) {
2.       bool cambio=true;
3.       for (int i=0; i<n-1 && cambio; i++) {
4.            cambio=false; {
5.            for (int j=0; j<n-i-1; j++){
6.                 if (v[j]>v[j+1]) {
7.                      cambio=true;
8.                      int aux = v[j];
9.                      v[j] = v[j+1];
10.                     v[j+1] = aux;
11.                }
12.           }
13.      }
14. }
```

En ella se ha introducido una variable que permite saber si, en una de las iteraciones del bucle externo no se ha modificado el vector. Si esto ocurre significa que ya está ordenado y no hay que continuar.

Considere ahora la situación del mejor caso posible en la que el vector de entrada ya está ordenado. ¿Cuál sería la eficiencia teórica en ese mejor caso? Muestre la gráfica con la eficiencia empírica y compruebe si se ajusta a la previsión.

Cálculo eficiencia teórica:

En cada línea de código se producen las siguientes operaciones elementales:
Línea 2: 1 o.e. inicialización booleana
Línea 3: 4 o.e. inicial. " i ", comparación "<", resta "n-1", incremento " i++"
Línea 4: 1 o.e. asignación booleana
Línea 5: 5 o.e. inicial. "j", comp. "<", doble resta "n-i-1", incremento "j++"

Línea 6: 4 o.e. accesos a v[j] y v[j+1], suma "j+1", comp. ">"
Línea 7: 1 o.e. asign. bool.
Línea 8: 2 o.e. inicial. " aux", acceso a v[j]
Línea 9: 4 o.e. acceso a v[j] y v[j+1], suma "j+1", asignación
Línea 10: 3 o.e. acceso a v[j+1], suma "j+1", asignación

*Las líneas 7-10 no sucederían en el mejor de los casos, cuando el vector está ordenado

Por tanto, en el mejor de los casos, donde el primer bucle sólo produce una iteración:

$$T(n) = 1 \ + \ 1 \ + \sum_{j=0}^{n-1} 4 = 1 + 1 + (\frac{4+4}{2}) \cdot (n-1) = 4n - 2$$