Análisis de Algoritmos y Estructuras de Datos

Tema 2: Análisis de la complejidad de los algoritmos

Versión 2.0

1. Considere dos algoritmos que resuelven un mismo problema en un número de operaciones elementales distinto, pero definido en ambos casos por funciones del mismo orden. ¿Qué podemos decir de los tiempos de ejecución de los programas correspondientes a dichos algoritmos?

2. En un artículo de cierta revista, un «experto» realizaba el siguiente comentario:

> «El algoritmo de ordenación por *fusión* es $\Theta(n \log_2 n)$, así, en las mismas condiciones, un programa que ordena 1000 elementos por este método es aproximadamente 100 veces más rápido a otro que lo hace por el de *intercambio directo*, que es $\Theta(n^2)$.»

Conteste razonadamente a las siguientes cuestiones.

a) ¿A qué se refiere el experto con «en las mismas condiciones»?

b) ¿Qué razonamiento ha podido seguir el experto para llegar a dicha conclusión?

c) ¿Está de acuerdo con él?

d) ¿Importa realmente la base del logaritmo en la expresión $\Theta(n \log_2 n)$?

3. Dado un vector $v \in \mathbb{R}^n$ con $n \geq 2$, el siguiente algoritmo determina el *par más cercano*, es decir, un par de elementos (v_i, v_j) tal que $1 \leq i < j \leq n$ y $|v_i - v_j|$ es mínima.

```
pmc(v, n) → (x, y)
m ← ∞
desde i ← 1 hasta n − 1
        desde j ← i + 1 hasta n
                d ← |v[i] − v[j]|
                si d < m
                  m ← d
                  (x, y) ← (v[i], v[j])
```

Analícelo respecto a la operación crítica que considere oportuna. Calcule el número de dichas operaciones y su orden exacto. ¿Qué hay que suponer acerca de la representación interna de los datos de entrada para que las operaciones que se realizan sobre ellos sean elementales en la práctica?

4. El siguiente algoritmo trabaja con una matriz booleana de dimensión $n \times n$.

```
crt(a, n) → a
desde k ← 1 hasta n
        a[k, k] ← ⊤
        desde i ← 1 hasta n − 1
                desde j ← i + 1 hasta n
                        a[i, j] ← a[i, j] ∨ (a[i, k] ∧ a[k, j])
```

Analícelo respecto a la operación crítica que considere oportuna. Calcule el número de dichas operaciones y su orden exacto.

5. Este algoritmo obtiene de forma recursiva el mínimo m de un vector $v \in V^n$, cuyos n elementos están ordenados por la relación de orden $\langle V, < \rangle$:

$$
\begin{array}{l}
\mathit{mínimo}(v, n) \to m \\
\mathsf{si}\ n = 1 \\
\quad m \leftarrow v[n] \\
\mathsf{si\ no} \\
\quad m \leftarrow \mathit{mínimo}(v, n-1) \\
\quad \mathsf{si}\ v[n] < m \\
\quad\quad m \leftarrow v[n]
\end{array}
$$

Indique la operación crítica escogida y utilícela para expresar mediante ecuaciones de recurrencia el tiempo requerido por el algoritmo en el peor y el mejor caso. Obtenga el orden exacto de ambos tiempos. ¿Existe alguna diferencia? ¿Coincide el tiempo con la versión iterativa del algoritmo?

6. Sea v un vector de dimensión $n \in \mathbb{N}^*$ tal que $v[1] \leq \cdots \leq v[n-1]$, el siguiente algoritmo termina de ordenar el vector insertando en orden el elemento $v[n]$ en la posición adecuada.

$$
\begin{array}{l}
\mathit{inserción} : v \times n \to v \\
\mathsf{si}\ n > 1 \\
\quad \mathsf{si}\ v[n] < v[n-1] \\
\quad\quad v[n] \leftrightarrow v[n-1] \\
\quad\quad \mathit{inserción}(v, n-1)
\end{array}
$$

Analice, mediante ecuaciones de recurrencia, el número exacto de comparaciones entre elementos del vector que realiza el algoritmo en el mejor caso, el peor caso y el caso promedio. Para el análisis en el caso promedio, suponga que el elemento $v[n]$ tiene la misma probabilidad de acabar en cualquiera de las posiciones.
Pista. En el caso promedio le resultará útil considerar las relaciones existentes entre las probabilidades de que $v[n] < v[n-1]$, $v[n-1] \leq v[n]$ y $v[n] = \max_{1 \leq i \leq n} v[i]$.

7. El algoritmo de búsqueda binaria o dicotómica devuelve la posición del valor x dentro del intervalo $[i, j]$ de un vector v cuyos elementos están en orden creciente. Si x no está en v, devuelve 0.

$$
\begin{array}{l}
\mathit{busbin}(x, v, i, j) \to p \\
p \leftarrow 0 \\
\mathsf{si}\ i \leq j \\
\quad p \leftarrow (i + j)/2 \\
\quad \mathsf{si}\ x < v[p] \\
\quad\quad p \leftarrow \mathit{busbin}(x, v, i, p-1) \\
\quad \mathsf{si\ no} \\
\quad\quad \mathsf{si}\ x > v[p] \\
\quad\quad\quad p \leftarrow \mathit{busbin}(x, v, p+1, j)
\end{array}
$$

Estudie su eficiencia en el caso peor, eligiendo la operación crítica que considere adecuada, y compare el resultado con la eficiencia de la búsqueda secuencial.

8. El algoritmo recursivo trivial para el cálculo del determinante de una matriz cuadrada de orden n procede de la siguiente forma:

- Si $n = 1$, el determinante se calcula sin ninguna operación escalar, ya que es el único elemento que posee la matriz.
- En caso contrario, se calculan los determinantes de los adjuntos de cada uno de los elementos de, por ejemplo, la primera fila (que son a su vez matrices cuadradas de orden $n-1$). El

determinante de la matriz original se obtiene a partir de éstos tras $2n-1$ operaciones escalares.

Dé un valor aproximado del número de operaciones escalares que realiza este algoritmo cuando n es lo suficientemente grande. Calcule su orden.
Pista. Suponga n lo suficientemente grande como para poder considerar $\sum_{i=0}^{n} \frac{1}{i!} \approx e$.

9. Calcular el orden de complejidad del siguiente algoritmo recursivo, que ordena un vector v de longitud n mediante el método de selección directa.

```
ordena-selec(v, n) → v
si n > 1
  p ← posición-máximo(v, n)
  v[p] ↔ v[n]
  ordena-selec(v, n − 1)
```

```
posición-máximo(v, n) → p
(p, m) ← (1, v[1])
desde i ← 2 hasta n
    si v[i] > m
      (p, m) ← (i, v[i])
```

10. El siguiente algoritmo permite calcular el número de *inversiones* de un vector.

```
inversiones(v, n) → r
si n = 0
  r ← 0
si no
  r ← inversiones(v, n − 1) +
      mayores(v[n], v, n − 1)
```

```
mayores(x, v, n) → r
si n = 0
  r ← 0
si no
  si x < v[n]
    r ← mayores(x, v, n − 1) + 1
  si no
    r ← mayores(x, v, n − 1)
```

Elija la operación crítica que considere oportuna y realice un análisis detallado del tiempo resultante mediante ecuaciones de recurrencia.

11. Un algoritmo recursivo para marcar una regla traza una marca en el centro de la regla, a continuación marcas más pequeñas en el centro de cada una de las dos mitades que genera la primera marca, y así sucesivamente. Supóngase que se dispone de otro algoritmo, $marcar(x, h)$, que hace una marca de altura h unidades en la posición x. La primera marca, trazada en el centro de la regla, tendrá n unidades de altura, la altura de las marcas en el centro de las dos mitades será de $n-1$ unidades, etc. El procedimiento de marcado termina cuando $n = 0$.

```
graduar(i, j, n)
si n > 0
  c ← (i + j)/2
  marcar(c, n)
  graduar(i, c, n − 1)
  graduar(c, j, n − 1)
```

Obtenga el tiempo de graduar la regla y su orden exacto, suponiendo:

a) Que el algoritmo $marcar(x, h)$ es de orden constante, $\Theta(1)$

b) Que el algoritmo $marcar(x, h)$ es de orden lineal, $\Theta(h)$