CS023 – Algoritmia y Complejidad

Instrucciones: Resuelva los siguientes ejercicios de forma clara y ordenada, dejando constancia de todo su procedimiento.

Ejercicio 1. Compruebe que $f(n) = n^2 \log n + n$ es $O(n^2 \log n), \Omega(n^2 \log n)$ & $\Theta(n^2 \log n)$.

Ejercicio 2. Compruebe que $f(n) = \log n!$ es $O(n \log n)$ & $\Omega(1)$.

Ejercicio 3. Compare las funciones $f(n) = n^{\log n}$ & $g(n) = 2^{\sqrt{n}}$. *R*: $n^{\log n} > 2^{\sqrt{n}}$.

Ejercicio 4. Compare las funciones $f(n) = 2^{\log n}$ & $g(n) = n^{\sqrt{n}}$. *R*: $2^{\log n} < n^{\sqrt{n}}$.

Ejercicio 5. Compare las funciones $f(n) = 2n$ & $g(n) = 3n$. *R*: $2n < 3n$.

Ejercicio 6. Dé una estimación en notación *big-oh* para el número de operaciones (comparación o multiplicación) usadas en el siguiente segmento de un algoritmo:

```
a=[a1,a2,…,an]      #array de números
m=0
for (i=1;i<=n;i++){
     for (j=i+1;j<n;j++){
          m=max(a[i]*a[j], m)
     }
}
```

R: $O(n^2)$

Ejercicio 7. El algoritmo convencional para evaluar un polinomio

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \text{ en } x = c$$

puede expresarse mediante el siguiente pseudocódigo:

```
a=[a0,a1,…,an]      #coeficientes del polinomio
power=1
y=a[0]
for(i=1;i<=n;i++){
     power=power*c
     y=y+a[i]*power
}
```

a. Evalúe $3x^2 + x + 1$ en $x = 2$ usando el algoritmo descrito.
b. Dé una estimación en notación *big-oh* para el número de operaciones (suma o multiplicación) usadas en el algoritmo. *R*: multiplicaciones $2n \rightarrow O(n)$; sumas $n \rightarrow O(n)$

Ejercicio 8. Hay un algoritmo más eficiente (en términos del número de sumas y multiplicaciones usadas) para evaluar un polinomio y se llama **método de Horner**. Este puede expresarse mediante el siguiente pseudocódigo:

```
a=[a0,a1,…,an]       #coeficientes del polinomio
y=a[n]
for(i=1;i<=n;i++){
     y=y*c+a[n-i]
}
```

a. Evalúe $3x^2 + x + 1$ en $x = 2$ usando el método de Horner.
b. Dé una estimación en notación *big-oh* para el número de operaciones (suma o multiplicación) usadas en el método de Horner. *R*: multiplicaciones $n \to O(n)$; sumas $n \to O(n)$; el número de multiplicaciones se reduce a la mitad.

Ejercicio 9. Indique cuál es el efecto en el tiempo de ejecución de un algoritmo cuando el número de datos se duplica, si se sabe que dicho algoritmo es:

a. $O(\log \log n)$
Para comparar, evaluamos la resta $\log \log 2n - \log \log n = \log \frac{\log 2n}{\log n} = \log \frac{\log 2 + \log n}{\log n}$.

Para valores muy grandes de n, el cociente $\frac{\log 2 + \log n}{\log n} \approx \frac{\log n}{\log n} = 1$.

Entonces, $\log \frac{\log 2 + \log n}{\log n} \approx \log 1 = 0$. Esto quiere decir que el tiempo de ejecución es casi igual.

b. $O(\log n)$
Para comparar, evaluamos la resta $\log 2n - \log n = \log \frac{2n}{n} = \log 2$.

Entonces, se requieren $\log 2$ unidades de tiempo adicional para la ejecución.

c. $O(n \log n)$
Para comparar, evaluamos la división $\frac{2n \log 2n}{n \log n} = 2 \cdot \frac{\log 2 + \log n}{\log n}$.

Para valores muy grandes de n, el cociente $\frac{\log 2 + \log n}{\log n} \approx \frac{\log n}{\log n} = 1$.

Entonces, se requieren aproximadamente 2 unidades de tiempo adicional para la ejecución.

d. $O(n^2)$
Para comparar, evaluamos la resta $(2n)^2 - n^2 = 4n^2 - n^2 = 3n^2$.
Entonces, se requieren $3n^2$ unidades de tiempo adicional para la ejecución.

e. $O(2^n)$

Para comparar, evaluamos la división $\frac{2^{2n}}{2^n} = 2^{2n-n} = 2^n$.

Entonces, se requieren 2^n unidades de tiempo adicional para la ejecución.

Ejercicio 10.

a. Describa un algoritmo que encuentre el entero más pequeño en una secuencia finita de números naturales.

```
a=[a1,a2,…,an]      #array de datos
min=a1
for(i=2;i<=n;i++){
     if min > a[i]{
          min=a[i]
     }
}
return min
```

b. Dé una estimación en notación *big-oh* para el número de comparaciones usadas en el algoritmo. *R*: Se requieren $2n - 1$ comparaciones.