UFM UNIVERSIDAD FRANCISCO MARROQUÍN | FACULTAD de CIENCIAS ECONÓMICAS

CS023 Algoritmia y Complejidad

Examen Parcial 1

Nombre: _____

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

1. (a) (5 puntos) Explique brevemente, ¿qué significa que una función sea $\mathcal{O}(1)$?

   (b) (5 puntos) Suponga que se tienen dos algoritmos distintos que resuelven el mismo problema. Para resolver un problema de tamaño n, el primer algoritmo realiza $T_1(n) = \frac{n^2}{8} \log n$ operaciones y el segundo algoritmo realiza $T_2(n) = n^{3/2}$ operaciones. A medida que n crece, ¿cuál de los dos algoritmos usa menos operaciones?

   (c) (5 puntos) ¿Cuál es el efecto en el tiempo de ejecución de un algoritmo $\mathcal{O}(n^3)$ si el número de datos se duplica?

2. (20 puntos) Determine la complejidad del algoritmo mostrado.

```
void function(int n)
{
    int count = 0;
    for (int i=n/2; i<=n; i++)
        for (int j=1; j<=n; j = 2 * j)
            for (int k=1; k<=n; k = k * 2)
                count++;
}
```

3. Resuelva las EDF:

   (a) (15 puntos)

   $$a_n = a_{n-1} + 2a_{n-2},\ n \geq 2,\ a_0 = 1,\ a_1 = 8$$

   (b) (15 puntos)

   $$a(n) - a(n-1) = 2n,\ n \geq 1,\ a(0) = -2$$

4. La ordenación rápida (*quick sort*) es un algoritmo eficiente para ordenar una lista de números $a_1, a_2, \ldots, a_n$. Este algoritmo comienza tomando el primer elemento (llamado **pivote**) y formando dos sublistas, de forma que la primera contiene aquellos elementos que son menores que a_1, en el orden que aparecen, y la segunda contiene los términos mayores que a_1, también en el orden en el que aparecen. Entonces a_1 se pone al final de la primera lista. Este procedimiento se repite recursivamente para cada sublista, hasta que todas las sublistas contienen un solo término. La lista ordenada de n términos se obtiene combinando las sublistas de un término en el orden en el que aparecen.

   (a) (5 puntos) Ordene la lista $3, 5, 7, 8, 1, 9, 2, 4, 6$ usando *quick sort*.

   (b) (5 puntos) Describa el algoritmo *quick sort* usando pseudocódigo.

   (c) (5 puntos) Sea $a_1, a_2, \ldots, a_n$ una lista de n números distintos. ¿Cuántas comparaciones son necesarias para formar dos sublistas a partir de la original, la primera con todos los elementos menores que a_1 y la segunda con aquellos elementos mayores que a_1?

   (d) (10 puntos) ¿Cuál es el mínimo/máximo número de comparaciones necesarias para ordenar una lista de 4 términos utilizando el algoritmo *quick sort*?

   (e) (10 puntos) Determine la complejidad en el peor caso del algoritmo *quick sort* en términos del número de comparaciones realizadas (asuma una lista de n datos).

Modalidad de entrega:

- Subir scan en PDF a MiU; use alguna aplicación para escanear su trabajo, no se aceptarán fotografías ni se aceptará la entrega de trabajos por correo electrónico. **La hora límite de entrega es 10:00**.
- Asegúrese que su trabajo es legible y que el archivo puede abrirse correctamente antes de subirlo a MiU, ya que de no ser posible verlo, el trabajo se calificará con una nota de cero puntos.
- El **Ejercicio 4** debe entregarse resuelto a más tardar el **viernes 25 de septiembre a las 18:00 horas**.