

**Primer Examen**  
Fundamentos de Análisis y Diseño de Algoritmos  
Programa de Ingeniería de Sistemas  
Universidad del Valle



Profesor Carlos Alberto Ramírez Restrepo

Abril 22 de 2017

**DECLARACIÓN DE HONOR**

Yo, \_\_\_\_\_, declaro que todas mis respuestas en este parcial fueron desarrolladas por mi y no dí ni recibí copia en ningún punto. Además soy consciente de las implicaciones disciplinarias y éticas de incurrir en copia y entiendo que este tipo de acciones no contribuyen a mi formación como profesional y como persona.

**Firma:** \_\_\_\_\_

**Código:** \_\_\_\_\_

**Fecha:** \_\_\_\_\_

Este es el primer examen parcial del curso *Fundamentos de Análisis y Diseño de Algoritmos*, 2017-1. El examen tiene 8 preguntas: otorga un total de 100 puntos y 10 puntos de bono. El examen es *individual* y NO es permitido el uso de libros o apuntes, ni de equipos electrónicos; no puede hablar o compartir sus soluciones con sus compañeros. Tenga en cuenta los puntos de cada pregunta y planifique

adecuadamente su tiempo. El parcial tiene una duración de 3 horas.

1. [12 pts.] ¿Cuál es la complejidad del siguiente algoritmo? (justifique su respuesta):

```
def conundrum(n):  
    r = 0  
    i = 1  
    while i <= n:  
        j = i + 1  
        while j <= n:  
            k = i + j - 1  
            while k <= n:  
                r = r + 1  
                k = k + 1  
            j = j + 1  
        i = i + 1  
    return r
```

2. [14 pts.] Considere el siguiente algoritmo:

```

Algoritmo(m, n)
{
    c=0;
    while(m >= n){
        m = m - n;
        c++;
    }
    return c;
}

```

- (a) Que cálcula este algoritmo?
  - (b) Cúal es la precondition y poscondición del algoritmo?
  - (c) Proponer una invariante para los ciclos del algoritmo.
  - (d) Demostrar que el invariante se cumple y argumentar la corrección del algoritmo.
3. [15 pts.] Se tienen tres algoritmos que resuelven un mismo problema:
- El algoritmo *A* divide el problema en 5 subproblemas de un cuarto del tamaño inicial, los resuelve recursivamente y combina las soluciones en tiempo lineal.
  - El algoritmo *B* resuelve el problema de tamaño  $n$  planteando dos problemas de tamaño  $n - 1$  y combinando las soluciones en tiempo constante. tiempo cuadrático.
  - El algoritmo *C* divide el problema en 7 subproblemas de la mitad del tamaño inicial, los resuelve recursivamente y combina las soluciones en tiempo cuadrático.

Cuál es la complejidad temporal de cada uno de los tres algoritmos? Cuál es el más eficiente y por qué?

Para calcular la complejidad de cada algoritmo plantee una ecuación de recurrencia y resuelvala de ser posible mediante el método maestro, si no resuelvala con cualquiera de los otros métodos vistos en clase.

4. [15 pts.] Defina un algoritmo con complejidad lineal que reciba un número  $N$  y construya un arreglo con todos los factoriales entre 1 y  $N$  de forma descendente.
5. [10 pts.] Defina un algoritmo que utilice recursividad y permita calcular el valor de la serie de fibonacci para un número  $n$  con complejidad  $\Theta(n)$ . Su algoritmo no puede utilizar ningún tipo de ciclo.

## 6. [18 pts.] Problema: Puntos máximos en el plano

Dado un par de puntos  $(x, y)$ ,  $(x', y')$  en el plano, se dice que  $(x, y)$  domina a  $(x', y')$  si  $x > x'$  y  $(y > y')$ . También se dice que  $(x', y')$  es dominado por  $(x, y)$ .

Dados  $n$  puntos en el plano cartesiano, se dice que uno de ellos es máximo si no está dominado por ninguno de los otros puntos.

Considere el problema de los *puntos máximos en un plano* **PMP**:

**Entrada:**  $n$  puntos en el plano  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , tales que  $x_i < x_{i+1}$  ó  $(x_i = x_{i+1} \wedge y_i < y_{i+1})$ . Es decir, los  $n$  puntos vienen ordenados de izquierda a derecha y de abajo a arriba.

**Salida:**  $I \subseteq \{1, 2, \dots, n\}$  tal que  $\forall_{i \in I} (x_i, y_i)$  es un punto máximo y tal que  $\forall_{i \notin I} (x_i, y_i)$  es un punto dominado.

- (a) Suponga que la entrada es  $n = 8$  y el conjunto de puntos es  $\{(1, 2), (2, 3), (3, 4), (4, 5), (5, 4), (6, 3), (7, 3), (8, 1)\}$ . Cuál es la salida que arrojaría con esta entrada un algoritmo que resuelva este problema?
  - (b) Defina un algoritmo para solucionar este problema que use la técnica *divide y vencerás*. Cuál es la complejidad de su algoritmo?
7. [18 pts.] Considere el siguiente problema. Dados  $m$  libros enumerados  $1, 2, \dots, m$ , donde  $p_1, p_2, \dots, p_m$  corresponde al número de páginas en cada libro. Se requiere hacer una copia de cada libro. Su tarea es asignar estos libros entre  $k$  escribas tales que  $k \leq m$ . Cada libro puede ser asignado solamente a un escriba y cada escriba debe tener una secuencia continua de libros. Esto significa que existe una sucesión creciente de números  $0 = b_0 < b_1 < b_2 \dots < b_{k-1} \leq b_k = m$  de tal manera que al  $i$ -ésimo escriba le son asignados los libros con números entre  $b_{i-1}$  y  $b_i$ . El tiempo necesario para copiar todos los depende del escriba al que se le haya asignado más trabajo. De esta manera, se requiere minimizar el máximo número de páginas asignado a un solo escriba.

Por ejemplo, si se tiene que  $m = 9$ ,  $k = 3$  y

$p_1, p_2, \dots, p_9 = 100, 200, 300, 400, 500, 600, 700, 800, 900$

el resultado debería ser 1700, que corresponde a asignar  $\{100, 200, 300, 400, 500\}$  al escriba 1,  $\{600, 700\}$  al escriba 2 y  $\{800, 900\}$  al escriba 3.

- (a) Construya un algoritmo que utilice la estrategia *divide y vencerás* (búsqueda binaria) que resuelva el problema.
  - (b)Cuál es la complejidad de su algoritmo. Explique claramente su respuesta.
8. **[8 pts.]** Resolver la siguiente recurrencia mediante el método de iteración o el método de árbol de recursión:

$$\mathbf{T(n)} = 3\mathbf{T(n - 1)} + 1$$