UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

Guía de Ejercicios
Estructuras lineales

Algoritmos y **Estructura de datos**

Curso Ing. **Gustavo Schmidt**

Aquí tienes una **guía de ejercicios de complejidad algorítmica**, dividida en **tres niveles de dificultad**. Los ejercicios requieren que los estudiantes planteen la ecuación de recurrencia para un algoritmo dado y resuelvan la complejidad usando **expansión** o el **teorema maestro**.

Nivel 1: Básico

1. **Recursión simple**
   Un algoritmo divide un problema de tamaño n en dos subproblemas de tamaño $n/2$, con un costo constante en cada paso adicional. Plantea la ecuación de recurrencia y resuélvela.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$$

2. **Búsqueda binaria**
   Un algoritmo busca un valor en un array ordenado reduciendo el problema a la mitad en cada paso, con un costo constante para la comparación.

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

3. **Suma recursiva**
   Un algoritmo suma los elementos de una lista de n números dividiendo la lista en dos partes de igual tamaño. Encuentra la complejidad.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

Nivel 2: Intermedio

4. **Merge Sort**
   Un algoritmo de ordenamiento divide el array en dos partes, las ordena de manera recursiva y luego las fusiona en tiempo lineal.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

5. **Búsqueda en árbol binario**
   En un árbol binario balanceado, un algoritmo busca un elemento comparando el nodo actual con el valor buscado. El problema se reduce a la mitad en cada nivel.

$$T(n) = T\left(\frac{n}{2}\right) + O(\log n)$$

6. **Quick Sort (caso promedio)**
   Un algoritmo de ordenamiento selecciona un pivote y divide el array en dos partes, ordenando cada parte de manera recursiva. Suponiendo que los pivotes dividen el array en partes iguales en promedio, plantea y resuelve la recurrencia.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

Nivel 3: Avanzado

7. **Multiplicación de matrices de Strassen**
   Un algoritmo divide dos matrices $n \times n$ en matrices más pequeñas y realiza 7 multiplicaciones en lugar de 8, logrando mejorar el método convencional.

$$T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$$

8. **Torre de Hanoi**
   Un algoritmo resuelve el problema de las Torres de Hanoi moviendo discos entre 3 postes. Plantea la recurrencia y encuentra la complejidad.

$$T(n) = 2T(n-1) + O(1)$$

9. **Algoritmo de búsqueda exponencial**
   Un algoritmo busca un elemento en un array ordenado duplicando el tamaño del intervalo de búsqueda en cada paso. Plantea la recurrencia y resuélvela.

$$T(n) = T(n/2) + O(\log n)$$

10. **Karatsuba (Multiplicación rápida de enteros)**
    Un algoritmo de multiplicación de enteros de gran tamaño utiliza la técnica de dividir y vencerás para reducir el número de multiplicaciones necesarias.

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$