Estructura de Datos y Algoritmos 1

Teórico #5:

Análisis de Algoritmos (Ejercicios)

Ejercicio 1

Considere la siguiente función en C++, definido sobre un arreglo de enteros de tamaño n

```
bool F(int* A, int n){
  bool res = true;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      if (i < j)
        res = res && (A[j] < A[i]);
  return res;
}
```

a) ¿Qué calcula/retorna (conceptualmente) la función F, dado un arreglo de enteros de tamaño n?
b) Calcule el orden (O) de tiempo de ejecución para el peor caso de la función F
c) El problema que resuelve F, ¿podría resolverse en un menor orden de tiempo de ejecución en el peor caso?

Ejercicio 2

Considere la siguiente función en C++, definido sobre un arreglo de enteros de tamaño n

```
bool F(int* A, int n){
  bool res = true;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      if (i != j)
        res = res && (A[j] != A[i]);
  return res;
}
```

a) ¿Qué calcula/retorna (conceptualmente) la función F, dado un arreglo de enteros de tamaño n?

b) Calcule el orden (O) de tiempo de ejecución para el peor caso de la función F

c) Si se sabe que el arreglo A sólo puede contener valores enteros en el rango [0 : n-1], el problema que resuelve F ¿podría resolverse en un menor orden de tiempo de ejecución en el peor caso?

Ejercicio 3

Considere la siguiente función en C++, definido sobre un arreglo de enteros de tamaño n

```
bool F(int* A, int* B, int n){
  bool res = true;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      if (i+j == n-1)
        res = res && (A[i] == B[j]);
  return res;
}
```

a) ¿Qué calcula/retorna (conceptualmente) la función F, dado un arreglo de enteros de tamaño n?

b) Calcule el orden (O) de tiempo de ejecución para el peor caso de la función F

c) El problema que resuelve F, ¿podría resolverse en un menor orden de tiempo de ejecución en el peor caso?

Ejercicio 4

Dadas las siguientes expresiones que indican el tiempo de ejecución de ciertos algoritmos, exprese la complejidad temporal de tales algoritmos en función de la cota superior “O grande”. Fundamente en cada caso

$$a)\ T(n) = \log(n^n)$$

$$b)\ T(n) = n^3 + 3n^2 + 3n + 1$$

$$c)\ T(n) = \begin{cases} 1 & \text{Si } n = 1 \\ T(n-1) + 1 & \text{Si } n > 1 \end{cases}$$

$$d)\ T(n) = \begin{cases} 1 & \text{Si } n = 1 \\ T(n-1) + n & \text{Si } n > 1 \end{cases}$$

Ejercicio 5

Determine su complejidad temporal

```
int Potencia1(int x, int n)
{
  if (n == 0)
       return 1;
  return x*Potencia(x, n-1);
}
```

```
int Potencia2(int x, int n)
{
  if (n == 0) return 1;
  if (n == 1) return x;
  int temp = Potencia2(x, n/2);
  if (n % 2 == 0)
       return temp*temp;
  return x*temp*temp;
}
```