PRÁCTICA 1.4 – Análisis de algoritmos - 2

Del Angel Flores Diego Emmanuel

Resolver lo siguiente:

a) Suponer que $T_1(n) = O(F(n))$ y $T_2(n) = O(F(n))$. Marcar qué de lo siguiente es verdadero:

   a. $T_1(n) + T_2(n) = O(F(n))$ (v)

   b. $T_1(n) * T_2(n) = O(F(n))$ (f)
   c. $T_1(n) = O(T_2(n))$ (v)
   d. Ninguno de los anteriores (f)

b) Suponga que el algoritmo 1 ejecuta $f(n) = n^2 + 4n$ pasos en el peor caso, y el algoritmo 2 ejecuta $g(n) = 29n + 3$ pasos en el peor caso, con entradas de tamaño n.

   ¿Cuál es la complejidad de cada algoritmo en notación O? R1:_____O(n2)_____
   R2:____O(n)_____

   ¿Cuál de los dos algoritmos es más rápido en ejecución? R: __algoritmo2___

   ¿Con entradas de qué tamaño es más rápido el algoritmo 1 que el algoritmo 2 en el peor caso (es decir, encuentre el valor de n_0)? R:____n26___

c) Considerar el siguiente método:

```
// Precondición:  m representa la matriz con N filas, N columnas,
//                en cada fila y en cada columna los elementos se incrementan
// Poscondición: regresa true si algún elemento en m almacena val, false en
                 caso contrario
public static boolean contains ( int [ ][ ] m, int val ) {
    for ( int r = 0; r < m.length; r++ )
      for ( int c = 0; c < m.length; c++ )
        if ( m[r] [c] == val )
          return true;
    return false;
}
```

   Ejemplo de matriz que satisface las precondiciones es:

```
int [ ][ ] m1={ {4, 6, 8}, {5, 9, 11}, {7, 11, 14} }
```

   a) ¿Cuál es el tiempo de corrida del método *contains* (en notación *O*)? R:_____(n2)_____
   b) Suponer que toma 4 segundos correr *contains* con una matriz de 100 x 100, ¿cuánto tiempo le tomará a *contains* correr con una matriz de 400 x 400? R:____1024seg_____