Estructura de Datos y Algoritmos

Practico Nº1. Complejidad de Algoritmos Iterativos

Ejercicio Nº1

Sea $f : \mathbb{N} \rightarrow \mathbb{R}^+$ una función de los naturales en los reales positivos. Definimos las siguientes familias de funciones:

$$O(f(n)) \doteq \{t : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \quad t(n) \leq cf(n)\},$$
$$\Omega(f(n)) \doteq \{t : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \quad t(n) \geq cf(n)\}.$$

Teniendo en cuenta las definiciones dadas, responder las siguientes preguntas y justificar su respuesta.

Queremos probar que: $\exists$ c>0, n_0 ,tal que $\forall n \geq n_0$, $t(n) \leq c \cdot f(n)$

a) t(n)=3n+2 y f(n)=n, ¿pertenece t(n) a O(f(n))?
b) t(n)=100n+6 y f(n)= n ¿pertenece t(n) a O(f(n))?
c) $t(n)=10n^2 +4n + 2$ y $f(n)=n^2$, ¿pertenece t(n) a $O(f(n^2))$?
d) $t(n)=10n^2 +4n + 2$ y f(n)=n, ¿pertenece t(n) a O(f(n))?

Ejercicio Nº2

Dados los siguientes algoritmos que resuelven el mismo problema: calcular el cuadrado de un número ingresado previamente por teclado.

1) *Producto*
```
def  producto():
 m=0
n = int(input("Ingrese un número:"))
m = n * n
print("Resultado:", m)
```

2) *Suma*
```
def Suma():
n = int(input("Ingrese un número: "))
m = 0
for i in range(n):
    m = m + n
print("Resultado:", m)
```

3) *Incremento*
```
def incremento():
n = int(input("Ingrese un número "))
m=0
 for i in range(n):
       for j in range(n):
           m = m + 1
  print("Resultado:",m)
```

1 -Calcular e indicar el Orden de Complejidad correspondiente a:
   a. T_1(n)(producto)
   b. T_2(n)(Suma)
   c. T_3(n)(Incremento)

2-Ejecute cada código dado agregando un contador en cada uno para contar el tiempo de ejecución

3-Determinar para cada uno de los ítems del punto anterior el valor de n_0 a partir del cual se cumple que: t(n) <=c.f(n)para algún valor constante de c ϵ R

4 -Representar gráficamente en un mismo eje cartesiano, la evolución del costo temporal en función del tamaño de n, de cada una de las funciones halladas en el punto 1.

5- Realizar un análisis del gráfico anterior y determinar qué algoritmo es asintóticamente más eficiente que los otros, para distintos tamaños de la entrada.