Taller Preparcial #1

Descripción general

Este taller Preparcial tiene como objetivo acercar al estudiante a los contenidos y competencias de la asignatura. Es un material de apoyo para la preparación de los parciales.

1. Sean n un entero positivo, seleccione una notación Θ para cada una de las siguientes funciones, demuestre que su selección es correcta sin utilizar la aplicación de límite ni las fórmulas para calcular directamente sumatorias.

   a) $\sum_{i=1}^{n} i$
   b) $9n^3 + 12n^2 + 1$
   c) $n^3 + 3n^2 + 2n - 21$
   d) $3n^2 + 2n \lg n$
   e) $2 \lg n + 4n + 3n \lg n$
   f) $(6n + 4)(1 + \lg n)$
   g) $(6n + 1)^2$
   h) $\sum_{i=1}^{n} i^2$
   i) $\dfrac{(n + 1)(n + 3)}{(n + 2)}$
   j) $\dfrac{(n^2 + \lg n)(n + 1)}{n + n^2}$
   k) $\dfrac{\left(\sqrt{10000^{\log n} + \lg^4 n}\right)(n^2 + 4n)}{n + 3}$
   l) $n \lg(n^4) + n^2 \lg(n^2) + n^4 \lg n$
   m) $\sum_{i=1}^{n} 2^i$

2. Sean n un entero positivo, demuestre que:

   a) $n^5 + n^4 + n^3 + n^2 + 2n - 2000 = \Theta(n^5)$
   b) $n^3 - 25n^2 - 50n - 100 = \Theta(n^3)$
   c) $n^4 + n^3 + 6n^2 - 11n - 10 = \Theta(n^4)$
   d) $\lg n! = \Theta(n \lg n)$
   e) $\log_a n = \Theta(\lg n)$, con $a > 1$
   f) $\log(n^2 + 1) = \Theta(\lg n)$
   g) $\sqrt{2n + 1} = \Theta(\sqrt{n})$
   h) $\sqrt[3]{n} = \mathcal{O}(\sqrt{n})$
   i) $\sqrt[3]{n} = \Omega(\sqrt[4]{n})$
   j) $\log(\log n) = \mathcal{O}(\log n)$
   k) $2^n = \mathcal{O}(n!)$
   l) $2^n = \mathcal{O}(n^n)$
   m) $n! = \mathcal{O}(n^n)$
   n) $n^{n+1} = \mathcal{O}\big(2^{n^2}\big)$

3. Si $f(n) = \Theta\big(n^{3/2}\big)$ y $g(n) = \Theta\big(n^{5/2}\big)$, entonces ¿cuál es la notación de $\Theta\big(f(n) + g(n)\big)$?.

4. Demuestre que $2^n = \mathcal{O}(3^n)$, pero que 3^n no es $\mathcal{O}(2^n)$, es decir, que $2^n = o(3^n)$.

5. Demuestre o refute que se cumple que $1 + \operatorname{sen}(n) = \Theta(1 + \cos(n))$.

6. Sean f y g un par de funciones asintóticamente no negativas tales que $f(n) = \mathcal{O}\big(g(n)\big)$. Demuestre que $f(n) + g(n) = \mathcal{O}\big(g(n)\big)$.

7. Sean f, g y h funciones asintóticamente no negativas tales que $f(n) = \mathcal{O}\big(h(n)\big)$ y $g(n) = \mathcal{O}\big(h(n)\big)$. Demuestre que $f(n) + g(n) = \mathcal{O}\big(h(n)\big)$.

8. Sean f, g, F y G funciones asintóticamente no negativas tales que $f(n) = \mathcal{O}\big(F(n)\big)$ y $g(n) = \mathcal{O}\big(G(n)\big)$. Demuestre que $f(n) + g(n) = \mathcal{O}\big(\text{máx}\,\{F(n), G(n)\}\big)$.

9. Sean f, g, F y G funciones asintóticamente no negativas tales que $f(n) = \mathcal{O}\big(F(n)\big)$ y $g(n) = \mathcal{O}\big(G(n)\big)$. Demuestre que $f(n)g(n) = \mathcal{O}\big(F(n)(G(n)\big)$.

10. Calcule la complejidad asintótica en términos de la notación Θ, del número de veces que se ejecuta la instrucción $x = x + 1$, para los siguientes algoritmos en donde cada uno depende de n:

a)
```
x = 0
for i = 1 to n do
    for j = 1 to n do
        x = x + 1
```

b)
```
x = 0
for i = 1 to n do
    for j = 1 to 2i do
        x = x + 1
```

c)
```
x = 0
for i = 1 to n do
    for j = 1 to ⌈i/2⌉ do
        x = x + 1
```

d)
```
x = 0
for i = 1 to n do
    for j = n downto i do
        x = x + 1
```

e)
```
x = 0
for i = n downto 1 do
    for j = i downto 1 do
        x = x + 1
```

f)
```
x = 0
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            x = x + 1
```

g)
```
x = 0
for i = 1 to n do
    for j = 1 to i do
        for k = 1 to j do
            x = x + 1
```

h)
```
i = n
x = 1
while i ≥ 1 do
    x = x + 1
    i = i/2
```

i)
```
x = 0
i = 0
while i < n do
    x = x + 1
    i = x * x
```

j)
```
x = 0
i = n
while i ≥ 1 do
    for j = 1 to n do
        x = x + 1
    i = i/2
```

k)
```
x = 0
i = n
while i ≥ 1 do
    for j = 1 to i do
        x = x + 1
    i = i/2
```

l)
```
x = 1
for i = 1 to n do
    j = 1
    while j ≤ i do
        x = x + 1
        j = j + 1/2
```

m)
```
x = 1
m = 1
for i = 1 to n do
    for j = 1 to m do
        x = x + 1
    m = 2 * m
```

11. Calcule la complejidad asintótica en términos de la notación Θ de las siguientes funciones, las cuales dependen de n:

a)

```
FUN(n)
for i = 1 to 2n do
    j = 1
    while j ≤ i do
        j = j + 1
```

b)

```
FUN(n)
for i = 1 to n do
    j = 1
    while j ≤ 2n do
        j = j + 1
```

c)

```
FUN(n)
x = 0
i = n
while i > 1 do
    for j = 1 to n do
        for k = 1 to n do
            x = x + 1
    i = ⌈i/2⌉
i = 1
while i ≤ 2n do
    for j = 1 to n do
        x = x + 1
    i = i + 1
return(x)
```

12. Calcule la complejidad asintótica en términos de las notaciones $\mathcal{O}$ y Ω de las siguientes funciones, las cuales dependen de n:

a)

```
x = 1
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            if input() == "stop" then
                break
            x = x + 1
```

b)

```
x = 1
for i = 1 to n do
    for j = 1 to n do
        if input() == "stop" then
            break
        for k = 1 to n do
            x = x + 1
```