Alberto Rodríguez Santana

ED - Relación de Problemas - EFICIENCIA

1.1. Probar que las siguientes sentencias son verdad

- 17 es $O(1)$

  ↳ $O(a)$ donde a es constante $= 17$

  $$\lim_{n\to\infty} \frac{17}{1} = 17 \neq 0 \rightarrow O(17) \subseteq O(1) \text{, luego } 17 \in O(1)$$

- $\frac{n(n-1)}{2}$ es $O(n^2)$ y $\Omega(n^2)$ (es decir $\Theta(n^2)$)

  $$\lim_{n\to\infty} \frac{\frac{n(n-1)}{2}}{n^2} \in \Theta(n^2) \Rightarrow \lim_{n\to\infty} \frac{\frac{n(n-1)}{2}}{n^2} = \lim_{n\to\infty} \frac{\cancel{n}(n-1)}{2n^{\cancel{2}}} = \lim_{n\to\infty} \frac{n-1}{2n} \neq 0$$

  $$\Theta\left(\frac{n(n-1)}{2}\right) = \Theta(n^2)$$

- $\max(\overset{f(n)}{n^3}, \overset{g(n)}{10n^2})$ es $O(n^3)$

  - Regla de la suma → si $n^3 \in O(n^3)$ y $10n^2 \in O(n^2)$ entonces

  $$n^3 + 10n^2 \in O(\max(n^3, 10n^2))$$

  $$\lim_{n\to\infty} \frac{n^3+10n^2}{1 n^3} = 1 \neq 0 \rightarrow O(n^3+10n^2) = O(n^3)$$

- $\log_2 n$ es $\Theta(\log_3 n)$

  $$\lim_{n\to\infty} \frac{\log_2 n}{\log_3 n} = \lim_{n\to\infty} \frac{\frac{\log n}{\log 2}}{\frac{\log n}{\log 3}} = \lim_{n\to\infty} \frac{\log n \cdot \log 3}{\log n \cdot \log 2} \neq 0 \Rightarrow \Theta(\log_2 n) = \Theta(\log_3 n)$$

1.2. Encontrar el entero k mas pequeño tal que $f(n)$ es $O(n^k)$ en los siguientes casos:

1. $f(n) = 13n^2 + 4n + 73 \in O(n^2) \Rightarrow k = 2$
2. $f(n) = 1/n+1 \rightarrow \lim_{n\to\infty} \frac{\frac{1}{n+1}}{n^k} = \lim_{n\to\infty} \frac{1}{n^k(n+1)} = 0 \rightarrow$ para cualquier valor de k

   se cumple $O\left(\frac{1}{n+1}\right) = O(n^k)$, valor mas pequeño de $\boxed{k=0}$

3. $f(n) = 1/n-1 \rightarrow$ igual que el anterior, $k=0$

4. $f(n) = (n-1)^3 \Rightarrow n^3 - 3n^2 + 3n - 1 \in O(n^3) \Rightarrow k=3$

5. $f(n) = (n^3+2n-1)/n+1 \rightarrow \lim_{n\to\infty} \frac{\left(\frac{n^3+2n-1}{n+1}\right)}{n^k} = \lim_{n\to\infty} \frac{n^3+2n-1}{n^k(n+1)} \neq 0 \Rightarrow \left(\frac{n^3+2n-1}{n+1}\right) = (n^k)$

$O(f(n)) = O(n^k) \rightarrow k=3$

6. $f(n) = \sqrt{n^2-1} \rightarrow \lim_{n\to\infty} \frac{\sqrt{n^2-1}}{n^k} = \lim_{n\to\infty} \frac{(n^2-1)^{1/2}}{n^k} \neq 0 \Rightarrow k \geq 2 \rightarrow \boxed{k=2}$

Problema 1.3. Ordenar de menor a mayor eficiencia:

$n, \sqrt{n}, n^3+1, n^2, n\log_2(n^2), n\log_2\log_2(n^2), 3^{\log_2(n)}, 3^n, 2^n, 2^n+3^{n-1}, 2000, n+100$

Ordenados $\rightarrow 2000 \subseteq \sqrt{n} \subseteq n \subseteq n+100 \subseteq n\log_2\log_2(n^2) \subseteq n\log_2(n^2) \subseteq 3^{\log_2(n)} \subseteq n^2 \subseteq$
$n^3+1 \subseteq 2^n \subseteq 3^n \subseteq 2^n+3^{n-1} \subseteq n2^n$

Problema 1.4. Supongamos que $T_1(n) \in O(f(n))$ y $T_2(n) \in O(f(n))$. Razonar verdadero o falso:

a.- $T_1(n)+T_2(n) \in O(f(n))$

Es cierta ya que si $T_1 \in O(f(n))$ y $T_2 \in O(f(n))$, $\exists c_1$ y $\forall n \geq n_0$,
$T_1(n) \leq c_1 \cdot f(n)$ y $\exists c_2, \forall n \geq n_1$ $T_2(n) \leq c_2 \cdot f(n)$.
Sea $c = c_1+c_2$ y $n_2 = \max\{n_0, n_1\} \Rightarrow T_1(n)+T_2(n) \leq (c_1+c_2)\cdot f(n) \; \forall n \geq n_2$

b.- $T_1(n) \in O(f^2(n))$

Si $T_1(n) = n^3$ y $f(n) = n \Rightarrow f^2(n) = n^2 \Rightarrow T_1(n) \notin O(f^2(n))$ luego es falsa.

c.- $T_1(n)/T_2(n) \in O(1)$

Es falso, ya que si $T_1(n) = n^2$ y $T_2(n) = n$ y $f(n) = n^3$, entonces
$\frac{T_1(n)}{T_2(n)} = n \notin O(1)$

Problema 1.6. Demostrar que si $f(n) \in O(g(n))$ y $g(n) \in O(h(n))$ entonces $f(n) \in O(h(n))$

$$\left.\begin{array}{l} f(n) \in O(g(n)) \Rightarrow f(n) \leq c_1 \cdot g(n) \ \forall n \geq n_0 \\ g(n) \in O(h(n)) \rightarrow g(n) \leq c_2 \cdot h(n) \ \forall n \geq n_1 \end{array}\right\} f(n) \leq c_1 c_2 \ h(n) \ \forall n \geq \max\{n_0, n_1\}$$

Problema 1.7. Demostrar que $O(f(n)) \in O(g(n))$ si $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$

$$\left.\begin{array}{l} f \in O(g) \Rightarrow O(f) \leq O(g) \\ g \in O(f) \Rightarrow O(g) \leq O(f) \end{array}\right\} O(f(n)) \in O(g(n))$$

Problema 1.8 Demostrar la siguiente jerarquía de ordenes de complejidad:

$$O(1) \subseteq O(\log(n)) \subseteq O(n) \subseteq O(n^2) \subseteq O(2^n) \subset O(n!)$$

$$\lim_{n\to\infty} \frac{1}{\log(n)} = 0 \Rightarrow 1 \in O(\log(n)) \text{ y } \log n \notin O(1) \rightarrow O(1) \subset O(\log(n))$$

$$\lim_{n\to\infty} \frac{\log(n)}{n} \overset{\text{L'Hopital}}{=} \lim_{n\to\infty} \frac{1}{n} = 0 \Rightarrow \log n \in O(n) \text{ y } n \notin O(\log(n)) \Rightarrow O(\log(n)) \subseteq O(n).$$

$$\lim_{n\to\infty} \frac{\cancel{n}}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0 \rightarrow n \in O(n^2) \text{ y } n^2 \notin O(n) \Rightarrow O(n) \subseteq O(n^2)$$

$$\lim_{n\to\infty} \frac{n^2}{2^n} \overset{\text{L'hopital}}{=} \lim_{n\to\infty} \frac{2\cancel{n}}{\cancel{n}\, 2^{n-1}} = 0 \Rightarrow n^2 \in O(2^n) \text{ y } 2^n \notin O(n) \Rightarrow O(n^2) \subseteq O(2^n)$$

$$\lim_{n\to\infty} \frac{2^n}{n!} = 0 \rightarrow O(2^n) \subset O(n!)$$

Problema 1.9. Obtener usando la O-grande la eficiencia del sig. codigo:

for(i=0; i<n; i++) $\rightarrow O(n)$

for(j=0; j<n; j++){ $\rightarrow O(n)$] $O(n)$

C[i][j]=0; $\rightarrow O(1)$

for(k=0; k<n k++) $\rightarrow O(n)$

C[i][j]+=A[i][k]*B[k][j]; $\rightarrow O(1)$] $O(n)$

}

multiplicación

$O(n^2)$

$O(n^3)$

Problema 1.10. Obtener usando O grande la eficiencia de la función:

```
void ejemplo(int n){
    int i,j,k;
    for(i=1, i<n, i++)
        for(j=i+1; j<=n; j++)
            for(k=1; k<=j; k++)
                Global += k*i;  → O(1)
}
```

$\sum_{k=1}^{j} 1 = j-1+1 = j$

$\sum_{j=i+1}^{n} j = (i+1)+(i+2)+ \dots +n = \dots$

$\to \frac{(n+i+1)(n-1)}{2} \to \sum_{i=1}^{n-1} \frac{(n+i+1)(n-1)}{2} = \sum_{i=1}^{n-1} \frac{n^2-i^2+n-i}{2} =$

$= \frac{1}{2}\sum_{i=1}^{n-1} n^2 - \frac{1}{2}\sum_{i=1}^{n-1} i^2 + \frac{1}{2}\sum_{i=1}^{n-1} n - \frac{1}{2}\sum_{i=1}^{n-1} i = \dots\dots = \frac{n^3-n}{3} \in O(n^3)$

Problema 1.11. Obtener la eficiencia a traves de O grande:

```
for (i=0; i<n; i++)
    if(i%2){ //solo entra cuando es par (1/2)
        for(j=i; j<n; j++)
            x*= i; → O(1)
        for(j=1; j<i; j++)
            y*=j; → O(1)
    }
```

$\sum_{j=1}^{n-1} 1 = n-i$

$\sum_{j=1}^{i-1} 1 = i-1$

$T_1(n)+T_2(n) = n-i+i-1 = n-1$

$\sum_{i=0}^{n-1} n-1 = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} 1 = n^2-n$

$n^2-n \in O(n^2)$

Problema 1.12. Suponer que n es una potencia positiva de 2, n=2,4,8,16 ...
Dar la formula que expresa el valor de cont cuando termina.

```
void ejemplo(int n){
    int x,cont;   ↳ n=2,4,8,16...
    cont=0;
    x=2;
    while(x<n){
        x=2*x;
        cont++;
    };
    printf(...)
}
```

n=2	n=4	n=8	n=16
cont=0	cont=1	cont=2	cont=3

$\sum_{x=2}^{n-1} (x-2)$

Problema 1.15. Resolver la siguiente recurrencia en función de k y s.

$T(n) = k \times T(n-1) + s^2 \quad n>1 \quad k,s>1 \quad T(1)=1.$

$T(2) = k \cdot T(2) + s^2 = k + s^2$

$T(3) = k \cdot T(3) + s^2 = k(k+s^2) + s^2$

$T(4) = k \cdot T(4) + s^2 = k(k(k+s^2)+s^2)+s^2$

$T(5) = k \cdot T(5) + s^2 = k(k(k(k+s^2)+s^2+s^2)+s^2$

$\vdots$

$T(n) = k \cdot T(n) + s^2 = k \cdot T(n-1) + s^2$

Problema 1.18. Implementar una funcion recursiva que resuelva las torres de Hanoi y estudiar la eficiencia en función de la altura.

```
void Hanoi (int n , int T1 , int T2) {
    if(n==1)
        cout<<"Moviendo de" <<T1<<"a"<<T2;
    else{
        Hanoi(n-1,T1,6-T1-T2);
        Hanoi(1,T1,T2);
        Hanoi(n-1,6-T1-T2,T2);
    }
}
```

$$T(n) = \begin{cases} 1 & n=1 \\ 2T(n-1)+1 & n>1. \end{cases}$$

$T(n) - 2T(n-1) = 1.$

$(x-2)(x-1) = 0$

$T(n) = c_1 2^n + c_2 1^n \in O(2^n)$

Problema 1.17. Resolver la recurrencia:

$$T(n) = \begin{cases} T(n/2) + n & n \geq 2 \\ 1 & n = 1 \end{cases}$$

$$T(n) - T(n/2) = n$$

$$n = 2^m$$

$$T(2^m) - T(2^{m-1}) = 2^m$$

$$(x-1)(x-2) = 0$$

$$T(2^m) = c_1 1^m + c_2 2^m$$

$$T(n) = c_1 + c_2 n \quad \underline{\underline{\in O(n)}}$$