Practica 2

Sebastian Britto

26 de septiembre de 2022

1. Ejercicio 1

Probar utilizando las definiciones que $f \in O(h)$ sabiendo que:

a) f, h: $\mathbb{N}$ son funciones tales que $f(n) = n^2 - 4n - 2$ y $h(n) = n^2$

- $f(n) = n^2 - 4n - 2$
- $h(n) = n^2$
- $f \in O(h) \Leftrightarrow \exists n_0 \in \mathbb{N}, k \in \mathbb{R} \wedge k > 0$ tal que $n \geq n_0 \Rightarrow f(n) \leq k \cdot h(n)$ con $n \in \mathbb{N}$
- Busco n, n_0 y k:
  $f(n) \leq k \cdot h(n) \Leftrightarrow n^2 - 4n - 2 \leq k \cdot n^2$
  $\Leftrightarrow -4n - 2 \leq k \cdot n^2 - n^2$
  $\Leftrightarrow -2(2 + 1) \leq n^2(k + 1)$
  $\Leftrightarrow 2(2n + 1) \geq n^2(1 - k)$ Elijo k = 1
  $\Leftrightarrow 2(2n + 1) \geq 0 \ \forall n \in \mathbb{N}$

b) f, g, h: $\mathbb{N}$ son funciones tales que $g(n) = n^k, h(n) = n^{k+1}$ y $f \in O(g)$

- $g(n) = n^k$
- $h(n) = n^{k+1}$
- $f(n) \in O(g)$
- Si $f \in O(h) \wedge g \in O(h) \Rightarrow f \in O(h)$
- $g \in O(h) \Leftrightarrow \exists n_0 \in \mathbb{N}, c \in \mathbb{R} \wedge c > 0$ tal que $n \geq n_0 \Rightarrow g(n) \leq c \cdot h(n)$ con $n \in \mathbb{N}$
- Busco n, n_0 y c:
  $g(n) \leq c \cdot h(n) \Leftrightarrow n^k \leq c \cdot n^{k+1}$
  $\Leftrightarrow 1 \leq c \cdot n$ Lo cual vale, $\forall c \in \mathbb{R}_{>0} \wedge \forall n \in \mathbb{N}$

c) f, g, h: $\mathbb{N}$ son funciones tales que $g(n) = log\ n, h(n) = n$ y $f \in O(g)$

- $g(n) = log(n)$
- $h(n) = n$
- $f(n) \in O(g)$
- Si $f \in O(h) \wedge g \in O(h) \Rightarrow f \in O(h)$
- Si $\lim_{x\to\infty} \frac{g(n)}{h(n)} = 0 \Rightarrow g \in O(h)$
- Compruebo por limite:
  $\lim_{x\to\infty} \frac{g(n)}{h(n)} = \lim_{x\to\infty} \frac{log(n)}{n} =_{L'H} \lim_{x\to\infty} \frac{1}{n} = 0$
  $\Rightarrow g \in O(h) \Rightarrow f \in O(g) \wedge g \in O(h) \Rightarrow f \in O(h)$

2. Ejercicio 2

Utilizando la nueva notacion, determinar la verdad o falsedad de cada una de las siguientes afirmaciones. JUSTIFICAR.

a) $2^n = O(1) \equiv$ Falso, no puede estar acotado por una constante

- $2^n \in O(1) \Leftrightarrow \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \leq n_0 \Rightarrow 2^n \leq c \cdot 1$
  $\iff 2^n \leq c \iff n \cdot \log_2 2 \leq \log_2 c \iff n \leq \log_2 c$
  Si elegimos un valor para c fijo, es imposible encontrar un n_0 tal que n $\geq n_0$, ya que siempre vamos a encontrar un $n > \log_2 c$

b) $n = O(n!)$

- $n \in O(n!) \Leftrightarrow \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \leq n_0 \Rightarrow n \leq c \cdot n!$
  $\iff n \leq c \cdot n! \iff n \leq c \cdot n \cdot (n-1)! \iff 1 \leq c \cdot (n-1)!$ Elijo c = 1
  $\iff 1 \leq (n-1)!$ Lo cual vale $\forall n \in \mathbb{N}$

c) $n! = O(n^n)$

- $n! \in O(n^n) \Leftrightarrow \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \leq n_0 \Rightarrow n! \leq c \cdot n^n$
- Lo pruebo por induccion con c = 1:
  $p(n) : n! \leq n^n$
  - Caso base:
    $p(1) : 1! \leq 1^1$ Se cumple
  - Paso Inductivo: Sea h $\in \mathbb{N}$
    - H.I.: $h! \leq h^h$
    - Q.V.Q.: $(h+1)! \leq (h+1)^{h+1}$
      Luego $(h+1)! \leq (h+1)^{h+1}$
      $\iff h! \cdot (h+1) \leq (h+1)^h \cdot (h+1) \iff h! \leq (h+1)^h \iff_{HI} h^h \leq (h+1)^h \iff h \leq h+1 \iff 0 \leq 1$
      Probe que p(1) es verdadero y que p(h) $\Rightarrow$ p(h+1) $\forall h \in \mathbb{N}$. Por induccion, puedo decir que p(n) es verdadero $\forall n \in \mathbb{N}$

d) $2^n = O(n!)$

- $2^n \in O(n!) \Leftrightarrow \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \leq n_0 \Rightarrow 2^n \leq c \cdot n!$
- Lo pruebo por induccion, tomo c = 1:
  $p(n) : 2^n \leq n!$
  - Caso base:
    $p(1) : 2 \leq 1$ No cumple.
    $p(2) : 4 \leq 2$ No cumple.
    $p(3) : 8 \leq 6$ No cumple.
    $p(4) : 16 \leq 24$ Cumple.
    $p(5) : 32 \leq 120$ Cumple
  - Paso Inductivo: Sea h $\in \mathbb{N}_{\geq 4}$. Supongo que p(h) es verdadero.
    - H.I.: $2^h \leq h!$
    - Q.V.Q.: $2^{h+1} \leq (h+1)!$
      Luego $2^{h+1} \leq (h+1)! \iff 2^h \cdot 2 \leq (h+1)! \iff_{HI} h! \cdot 2 \leq (h+1)! \iff 2 \cdot h! \leq (h+1) \cdot h! \iff 2 \leq h+1$
      Lo cual vale $\forall h \geq 4$. Asi, p(4) es verdadero y $\forall h \in \mathbb{N}_{\geq 4}$ p(h) es verdadero $\Longrightarrow$ p(h+1) es verdadero. Por el principio de induccion (corrida), p(n) es verdadero $\forall n \in \mathbb{N}_{\geq 4}$

e) $\forall i, j \in \mathbb{N}, i \cdot n = O(j \cdot n)$

- $i \cdot n \in O(j \cdot n) \iff \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \geq n_0 \Longrightarrow i \cdot n \leq c \cdot j \cdot n$
  $i \cdot n \leq c \cdot j \cdot n \iff i \leq j \cdot c \iff \frac{i}{j} \leq c$
  Si tomo $c \geq \frac{i}{j}$ esto vale $\forall i, j \in \mathbb{N}$

f) $\forall K \in \mathbb{N}, 2^k = O(1)$

- $2^k \in O(1) \iff \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \geq n_0 \longrightarrow 2^k \leq c \cdot 1$
  Pero 2^k es una constante, asi que $2^k \in O(1)$, luego $c = 2^k$

g) $log(n) = O(n)$

- $\log n \in O(n) \iff \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \geq n_0 \implies \log n \leq c \cdot n$
- Si $\lim_{x\to\infty} \frac{\log n}{n} = 0 \implies \log n \in O(n)$
  $\lim_{x\to\infty} \frac{\log n}{n} =_{L'H} \lim_{x\to\infty} \frac{1}{n} = 0 \implies \log n \in O(n)$

h) $n! = O(2^n)$

- $n! \in O(2^n) \iff \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \geq n_0 \implies n! \leq c \cdot 2^n$
  En el punto d) ya probe que $2^n \leq n!\ \forall n \in \mathbb{N}_{\geq 4}$.
  Entonces, $n! \leq 2^n \iff n \in (1, 2, 3)$

i) $2^n n^2 = O(3^n)$

- $2^n n^2 \in O(3^n) \iff \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \geq n_0 \implies 2^n n^2 \leq c \cdot 3^n$

- Me fijo por limites:
  $\lim_{n\to\infty} \frac{2^n n^2}{3^n} =_{Cauchy} \lim_{n\to\infty} \sqrt[n]{\frac{2^n n^2}{3^n}} = \lim_{n\to\infty} \frac{\sqrt[n]{2^n n^2}}{\sqrt[n]{3^n}} = \lim_{n\to\infty} \frac{2 \cdot \sqrt[n]{n^2}}{3} = \lim_{n\to\infty} \frac{2 \cdot \sqrt[n]{n} \cdot \sqrt[n]{n}}{3} = \frac{2}{3} = L$
  $L < 1 \implies \lim_{n\to\infty} \frac{2^n n^2}{3^n} = 0$

j) $\forall f : \mathbb{N} \longrightarrow \mathbb{N}, f = O(f)$

- $f \in O(f) \iff \exists n_0 \in \mathbb{N}, c \in \mathbb{R}_{>0}$ tal que $n \geq n_0 \implies f \leq c \cdot f$
- Si tomo c = 1
  $f \leq f$ Lo cual se cumple siempre

3. Ejercicio 3

a) ¿Que significa, intuitivamente, $O(f) \subseteq O(g)$?¿Que se puede concluir acerca del crecimiento de f y g cuando, simultaneamente, tenemos $O(f) \subseteq O(g)$ y $O(g) \subseteq O(f)$?
$O(f) \subseteq O(g)$, significa que la cota superior de f esta contenida en la cota superior de g.
Si tengo $O(f) \subseteq O(g) \land O(g) \subseteq O(f) \implies O(f) \subseteq O(g)$, es decir ambas crecen simultaneamente

b) ¿Como ordena por inclusion las siguientes familias de funciones?
Veo claramente que $O(1) \in (O(x+1), O(x^2), O(\sqrt{x}))$, pero no veo claro $O(\frac{1}{x})$
$O(1) \subseteq O(\frac{1}{x}) \iff 1 \in O(\frac{1}{x}) \iff \exists x_0 \in \mathbb{N}, c \in \mathbb{R}, c > 0$ tal que $x \geq x_0 \implies 1 \leq c \cdot \frac{1}{x}$
$x \leq c$, pero esto no se cumple $\forall x \in \mathbb{N} \implies O(1) \notin O(\frac{1}{x})$ pero $O(\frac{1}{x}) \in O(1)$
Ahora me fijo si $O(\sqrt{x}) \subseteq O(x+1)$, a primera vista parece que si, pero me fijo de todas formas:
$O(\sqrt{x}) \subseteq O(x+1) \iff \sqrt{x} \in O(x+1)$
$\iff \exists x_0 \in \mathbb{N}, c \in \mathbb{R}, c > 0$ tal que $\forall x \in \mathbb{N}, x \geq x_0 \implies \sqrt{x} \leq c \cdot (x+1) \iff (\sqrt{x})^2 \leq c^2 \cdot (x+1)^2 \iff |x| \leq (x+1)^2$, pero ya se que $x \in \mathbb{N}$, $\iff x \leq (x+1)^2$ Vale, $\forall x \in \mathbb{N}$
Por lo tanto, $O(\frac{1}{x}) \subseteq O(1) \subseteq O(\sqrt{x}) \subseteq O(x+1) \subseteq O(x^2)$

4. Ejercicio 4

a) Sumatoria, que calcula la sumatoria de un arreglo de enteros

```
//Si n es la longitud de A
function Sumatoria(arreglo A){
   int i, total; //O(1)
   total := 0;   //O(1)
   for i := 0 ...Long(A) - 1 do //O(n)
      total := total + A[i];   //O(1)
   end for
end function
}
//O(n)
```

Si llamo a la funcion sumatoria f(n), $\Rightarrow$ f(n) $\in$ O(n)

b) SumatoriaLenta, que calcula la sumatoria de n, definida como la suma de todos los enteros entre 1 y n, de forma poco eficiente:

```
function SumatoriaLenta(natural N){
    int i, total;                //O(1)
    total := 0;                  //O(1)
    for i := 1 ...n do           //O(n)
        for j: 1...i do          //O(i)
            total := total + 1;  //O(1)
        end for
    end for
end function
}
//O(n^2)
```

$\sum_{i=1}^{n}(\sum_{j=1}^{i} 1) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, llamo a SumatoriaLenta como f(n) = $\frac{n(n+1)}{2}$; ¿f(n) $\in$ O(n^2)?

$\Leftrightarrow \exists n_0, k > 0$ tal que $n \geq n_0 \Rightarrow f(n) \leq k \cdot n^2$

Pruebo por induccion, con k = 1:

Defino p(n): $\frac{n(n+1)}{2} \leq n^2$

- Caso base:
  - p(1): $\frac{2}{2} \leq 2^2 \Leftrightarrow 1 \leq 4$ p(1) es verdadero
- Paso Inductivo: Sea h $\in \mathbb{N}$ p(h) $\Rightarrow$ p(h+1)
  - H.I.: $\frac{h(h+1)}{2} \leq h^2$
  - Q.V.Q.: $\frac{(h+1)(h+2)}{2} \leq (h+1)^2$
  - Luego, $\frac{(h+1)(h+2)}{2} \leq (h+1)^2$
    $\iff \frac{(h+1)(h+2)}{2} \leq h^2 + 2h + 1$
    $\iff_{HI} \frac{(h+1)(h+2)}{2} \leq \frac{h(h+1)}{2} + 2h + 1$
    $\iff \frac{h^2+3h+2}{2} \leq \frac{h(h+1)}{2} + 2h + 1$
    $\iff \frac{h^2+3h+2}{2} \leq \frac{h^2+h+4h+2}{2}$
    $\iff 3h \leq 4h \iff 3 \leq 4$ Lo cual vale $\forall$ h $\in \mathbb{N}$

Como p(1) es verdadero y p(h) $\Rightarrow$ p(h+1). Por induccion puedo decir que p(n) es verdadero $\forall$ n $\in \mathbb{N}$

Luego f(n) $\in$ O(n^2)

c) ProductoMat, que dadas dos matrices A (de p x q) y B (de q x r) devuelve su producto AB(de p x r)

```
//Sea p y q, filas y columnas de A.
//Sea q y r, filas y columnas de B.
function ProductoMat(matriz A, matriz B){
    int fil, col, val colAFilB;                    //O(1)
    matriz res(Filas(A), Columnas(B));             //O(1)
    for fil := 0...Filas(A) - 1 do                 //O(p)
        for col := 0...Columnas(B) - 1 do          //O(r)
            val := 0;                              //O(1)
            for colAFilB := 0...Columnas(A) - 1 do //O(q)
                val := val + (A[fil][colAFilB]*B[colAFilB][col]); //O(1)
            end for
            res[fil][col] := val;                  //O(1)
        end for
    end for
    return res;
end function
}
//O(p*r*q)
```

Si ambas matrices tuvieran la misma cantidad de filas y columnas, n, $\Rightarrow$ f(n) $\in$ O(n^3)

5. Ejercicio 5

Determinar el orden de complejidad temporal de mejor y peor caso de los siguientes algoritmos, asumiendo que todas las operaciones sobre arreglos y matrices toman tiempo O(1).
La complejidad se debe calcular en funcion de una medida de los parametros de entrada, por ejemplo, la cantidad de elementos en el caso de los arreglos y matrices y el valor en el caso de parametros naturales.

a) InsertionSort, que ordena un arreglo pasado como parametro

```
function InsertionSort(arreglo A){
    int i, j, valor;                    //O(1)
    for i := 0...Long(A) - 1 do         //O(n)
        valor := A[i]                   //O(1)
        j := i - 1;                     //O(1)
        while j >= 0 && A[j] > valor do //O(i)
            A[j+1] := A[j];             //O(1)
            j := j - 1;                 //O(1)
        end while
        A[j+1] := valor;                //O(1)
    end for
end function
}
```

- Peor caso: $\sum_{i=0}^{n-1}(\sum_{j=0}^{i} 1) = \sum_{i=0}^{n-1} i = \frac{(n-1)n}{2} \Rightarrow \mathrm{O}(n^2)$
- Mejor caso: $\sum_{i=0}^{n-1} 1 = n, \Rightarrow \Omega(\mathrm{n})$

b) BusquedaBinaria, que determina si un elemento se encuentra en un arreglo, que debe estar ordenado

```
function BusquedaBinaria(arreglo A, elem valor){
    int izq := 0, der := Long(A) - 1; //O(1)
    while izq < der do                //O(log_2(n))
        int medio := (izq + der) / 2; //O(1)
        if valor < A[medio] then      //O(1)
            der := medio;             //O(1)
        else
            izq := medio;             //O(1)
        end if
    end while
    return A[izq] = valor;            //O(1)
end function
}
/*
Ejemplo: [2,5,9,11,12,13,14,15]
Iteraciones:
it 1: 8 -> medio = (8 + 0)/2
it 2: 4 -> medio = (4 + 0)/2
it 3: 2 -> medio = (2 + 0)/2
it k: n/(2^k)
Luego, n/(2^k) = 1 <-> n = 2^k <-> log_2(n) = k
*/
```

- Peor caso: $log_2(\mathrm{n}) \Rightarrow f(n) \in O(log_2(n))$
- Mejor caso: $log_2(\mathrm{n}) \Rightarrow f(n) \in \Omega(log_2(n))$

c) AlgoritmoQueHaceAlgo

```
//Sea tam(A) = n
function AlgoritmoQueHaceAlgo(arreglo A){
    i <- 1; j <- 1                  //O(1)
    suma <- 1; count <- 0           //O(1)
    while i <= tam(A) do            //O(n)
```

```
        if i != A[i] then            //O(1)
            count <- count + 1       //O(1)
        end if
        j <- 1                       //O(1)
        while j <= count do          //O(i)
            k <- 1                   //O(1)
            while k <= tam(A) do     //O(n/2)
                suma <- suma + A[k]  //O(1)
                k <- k*2             //O(1)
            end while
            j <- j + 1               //O(1)
        end while
        i <- i + 1                   //O(1)
    end while
    return suma
end function
}
```

- Peor Caso: $\sum_{i=1}^{n} \sum_{j=1}^{i} \frac{n}{2} = \sum_{i=1}^{n} i \cdot \frac{n}{2} = \frac{n \cdot (n+1)}{2} \cdot \frac{n}{2} = \frac{n^2 \cdot (n+1)}{4}$. Parece que $\frac{n^2 \cdot (n+1)}{4} \in O(n^3)$
  - $\frac{n^2 \cdot (n+1)}{4} \in O(n^3) \Leftrightarrow \exists n_0, k > 0$ tal que n $\geq n_0 \Rightarrow \frac{n^2 \cdot (n+1)}{4} \leq k \cdot n^3$
  - Elijo k = 1:
    $\frac{n^2 \cdot (n+1)}{4} \leq n^3 \Leftrightarrow n^3 + n^2 \leq 4n^3 \Leftrightarrow n^2 \leq 3n^3 \Leftrightarrow 1 \leq 3n \Leftrightarrow \frac{1}{3} \leq n$. Lo cual se cumple $\forall n \in \mathbb{N}$
  - $\Rightarrow f(n) \in O(n^3)$
- Mejor Caso: $\sum_{i=1}^{n} 1 = n \Rightarrow f(n) \in \Omega(n)$

6. Ejercicio 6

Pensar un algoritmo que resuelva cada uno de los siguientes problemas y, en cada caso, determinar su complejidad temporal. No es necesario escribir formalmente el algoritmo, basta con delinear los pasos importantes que permitan estimar su complejidad.

a) Calcular la media de un arreglo de enteros

```
function Media(arreglo A){ //O(n)
    suma := 0;                   //O(1)
    for i := 0...Long(A) - 1 do  //O(n)
        suma := suma + A[i];     //O(1)
    end for
    return suma / Long(A);       //O(1)
}
```

b) Calcular la mediana de un arreglo de una cantidad impar de enteros

```
//Primero ordeno el arreglo.

function ordenar(arreglo A){ //O(n^2)
    for i := 0...Long(A)-1 do
        posMinimo := i;
        for j := i...Long(A)-1 do
            if A[posMinimo] > A[j] then
                posMinimo := j;
            end if
        end for
        k := A[i];
        A[i] := A[posMinimo];
        A[posMinimo] := k;
    end for
}

function Mediana(arreglo A){ //O(n^2)
```

```
    ordenar(A);              //O(n^2)
    mediana := A[Long(A)/2]; //O(1)
    return mediana;
}
```

c) Determinar, dado un n natural, si n es (o no) primo.

```
function esPrimo(natural n){ //O(n/2)
    if n == 1 then        //O(1)
        return false;     //O(1)
    else //Aca n > 1
        cant := 0;               //O(1)
        for i := 2...n/2 do      //O(n/2)
            if (n % i == 0) then //O(1)
                cant := cant + 1; //O(1)
            end if
        end for
        return cant == 0;
    end if
}
```

7. Ejercicio 7

Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decision.

a) $O(n^2) \cap \Omega(n) = \Theta(n^2)$

- Si esto es verdadero entonces se tiene que cumplir lo siguiente:
  $f \in O(n^2) \wedge f \in \Omega(n) \Rightarrow f \in \Theta(n^2)$, un f que cumple esto seria f(n) = n, ya que:
  $n \in O(n^2) \wedge n \in \Omega(n)$. Luego:
  a) $n \in O(n^2) \Leftrightarrow \exists n_0, k > 0$ tal que $\forall n \geq n_0 \Rightarrow n \leq k \cdot n^2$
    - Luego, con k = 1: $n \leq n^2 \Leftrightarrow 1 \leq n$ Lo cual se cumple $\forall n \in \mathbb{N}$
  b) $n \in \Omega(n) \Leftrightarrow \exists n_0, k > 0$ tal que $\forall n \geq n_0 \Rightarrow n \geq k \cdot n$
    - Elijo k = 1: $n \geq n \Leftrightarrow 1 \geq 1$ Lo cual se cumple $\forall n \in \mathbb{N}$
  c) Entonces como $n \in O(n^2) \wedge n \in \Omega(n) \Rightarrow n \in \Theta(n^2)$, pero por simetria: $n \in \Theta(n^2) \Leftrightarrow n^2 \in \Theta(n)$ Lo cual no se cumple, por lo que $O(n^2) \cap \Omega(n) = \Theta(n^2)$ es Falso.

b) $\Theta(n) \cup \Theta(n \cdot log\ n) = \Omega(n \cdot log\ n) \cap O(n)$

- Si $\Omega(n \cdot log\ n) \cap O(n) \Rightarrow$ existe una funcion f tal que $\Theta(f) = \Omega(n \cdot log\ n) \cap O(n)$.
  Es decir, $\exists n_0, k_1, k_2 > 0$ tal que $\forall n \geq n_0 \Rightarrow k_1 \cdot n \cdot log\ n \leq f \leq k_2 \cdot n$
  Pero $n \cdot log\ n$ crece mas rapido que n, por lo que no hay f que tenga como cota superior a n e inferior a $n \cdot log\ n$
  Por lo que $\Omega(n \cdot log\ n) \cap O(n) = \emptyset$. Por lo tanto es Falso.

c) Existe una $f : \mathbb{N} \longrightarrow \mathbb{N}$ tal que $\forall g : \mathbb{N} \longrightarrow \mathbb{N}$ se cumple que $g \in O(f)$

- Falso, no puede existir una cota superior maxima para todas las funciones, siempre podes encontrar una mayor.

d) Sean $f, g : \mathbb{N} \longrightarrow \mathbb{N}$, entonces se cumple que $O(f) \subseteq O(g)$ o $O(g) \subseteq O(f)$ (es decir, el orden sobre funciones dado por la inclusion de la O es total).
Te la debo.

8. Ejercicio 8

Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decision.

a) $n + m = O(nm)$
Falso. Por regla de la suma se que:
Si $f1 \in O(g) \wedge f2 \in O(h) \Longrightarrow f1 + f2 \in O(max(g, h))$
Pero $n \in O(n) \wedge m \in O(m) \Longrightarrow n + m \in O(max(n, m)) \neq O(nm)$

b) $n^2 + m^2 = O(nm)$
Para aplicar la regla de la suma

c) $n + m^5 = O(m^5)$

d) $n \log n + m \log m = O(n \log m + m \log n)$

e) $nm = O(n + m)$

f) $nm = O(n^2 + m^2)$

g) $m^5 = O(n + m^5)$

h) $n \log m + m \log n = O(n \log n + m \log m)$