TAREA 2: Analisis y Diseño de Algoritmos

Andrey Arguedas Espinoza - 2020426569

September 11, 2022

1. Se tiene la siguiente función:

$$t(n) = \begin{cases} 2 & n = 2 \\ 1 & n = 3 \\ 2 \cdot t\left(\left\lceil \frac{n+1}{2} \right\rceil\right) + t\left(\left\lceil \frac{n}{2} \right\rceil\right) & n > 3 \end{cases}$$

Mostrar que t(n) **no es** asintóticamente no decreciente.

Primero debemos solucionar la recurrencia, para esto vamos a utilizar el cambio de variable sobre **n**. Si **n** es potencia de 2:

$$n => 2^i \tag{1}$$

Ahora trabajaremos con **ti**

$$\begin{aligned} ti &= T(2^i) \\ &= 2T(2^{i-1} + \tfrac{1}{2}) + T(2^{i-1}) \\ &= 2t_{i-1} + 1 + t_{i-1} \\ &= 3t_{i-1} + 1 \end{aligned} \tag{2}$$

Ahora continuamos con la ecuación caracteristica:

$$t_i - 3t_{i-1} = 1 \tag{3}$$

- Con la ecuación caracteristica obtendremos las raices del lado izquierdo tenemos:

$$t_i - 3t_{i-1} => (x - 3) \tag{4}$$

- Con la ecuación caracteristica obtendremos las raices del lado derecho tenemos:

$$\begin{aligned} p(i) &= 1 \\ b &= 1 \\ d &= 0 \end{aligned} \tag{5}$$

$$(x - 1) \tag{6}$$

- Con las nuevas raices generamos la solución general:

$$t_i = c_1 * 3^i + c_2 * 1^i \tag{7}$$

- Ahora debemos deshacer el cambio de variable:

$$T(n) = \text{ logn when n =}2^i \tag{8}$$

$$\begin{aligned} T(n) &= c_1 * 3^{logn} + c_2 * 1^{logn} \\ &= c_1 * n^{log3} + c_2 \end{aligned} \tag{9}$$

- Ahora armamos el sistema de ecauciones de 2x2:

$$\begin{aligned} c_1 + c_2 &= 2|\text{ n = 1} \\ 3c_1 + c_2 &= 1|\text{ n = 2} \end{aligned} \tag{10}$$

- Como resultado obtenemos que $\boxed{c_1 = -1/2}$ y $\boxed{c_2 = 5/2}$

$$t_n = -1/2 * n^{log3} + 5/2 * n \tag{11}$$

- Finalmente tenemos que $\boxed{5_n}$ es la parte que domina por lo tanto

$$T(n)\epsilon\theta(\text{ n IF n is power of 2}) \tag{12}$$

Ahora debemos demostrar de que no es asintoticamente no decreciente

2. Para cada función en el centro de la figura siguiente escoja la mejor función Ω de la izquierda (la más ajustada: la más "grande" que cumple con la definición), y la mejor función O grande de la derecha (la más ajustada: la más "pequeña" que cumple con la definición). **Justificar brevemente**.

$\Omega(1/n)$		$O(1/n)$
$\Omega(1)$		$O(1)$
$\Omega(\log\log n)$		$O(\log\log n)$
$\Omega(\log n)$		$O(\log n)$
$\Omega(\log^2 n)$		$O(\log^2 n)$
$\Omega(\sqrt[3]{n})$		$O(\sqrt[3]{n})$
$\Omega(n/\log n)$		$O(n/\log n)$
$\Omega(n)$	$1/\log n$	$O(n)$
$\Omega(n^{1.00001})$	$7n^5 - 3n + 2$	$O(n^{1.00001})$
$\Omega(n^2/\log^2 n)$	$(n^2 + n)/(\log^2 n + \log n)$	$O(n^2/\log^2 n)$
$\Omega(n^2/\log n)$	$2^{\log^2 n}$	$O(n^2/\log n)$
$\Omega(n^2)$	3^n	$O(n^2)$
$\Omega(n^{3/2})$		$O(n^{3/2})$
$\Omega(2^n)$		$O(2^n)$
$\Omega(5^n)$		$O(5^n)$
$\Omega(n^n)$		$O(n^n)$
$\Omega(n^{n^2})$		$O(n^{n^2})$

1- Comenzamos con el analisis de **1/logn** :

Podemos ver que al estar 1 siendo dividio por logn vamos a tener numeros menores a 1 como resultado por lo que la unica forma de acotar por abajo a **1/logn** de las opciones disponibles sería con:

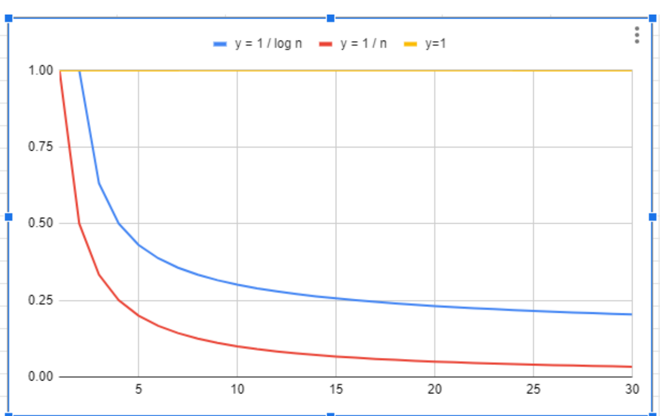
$$\Omega(1/n) \tag{13}$$

Esto debido a que no se podría encontrar un **c** dado que $fn \geq c * g(n)$, con ninguna de las otras opciones disponibles en la columna izquierda.

Mientras que la forma mas proxima de acotar por arriba sería con:

$$O(1) \tag{14}$$

Debido a que toda constante **c** dado que $fn \leq c * g(n)$, tendría a decrecer con numeros menores a 1



2 - Continuamos con el analisis de $7n^5 - 3n + 2$:

Para obtener el Big Oh podemos ver que en el peor de los casos exisitiría un que va a ser elevado a la 5, con esto ya podemos descartar muchas opciones de la derecha, y podemos empezar a probar con la mas grande que fuese valida, en este caso probaremos si cumple con:

$$O(2^n) \tag{15}$$

$$fn \leq c * g(n) \tag{16}$$

$$7n^5 - 3n + 2 \leq c * g(n) \tag{17}$$

$$7n^5 - 3n^5 + 2n^5 \leq c * 2^n \tag{18}$$

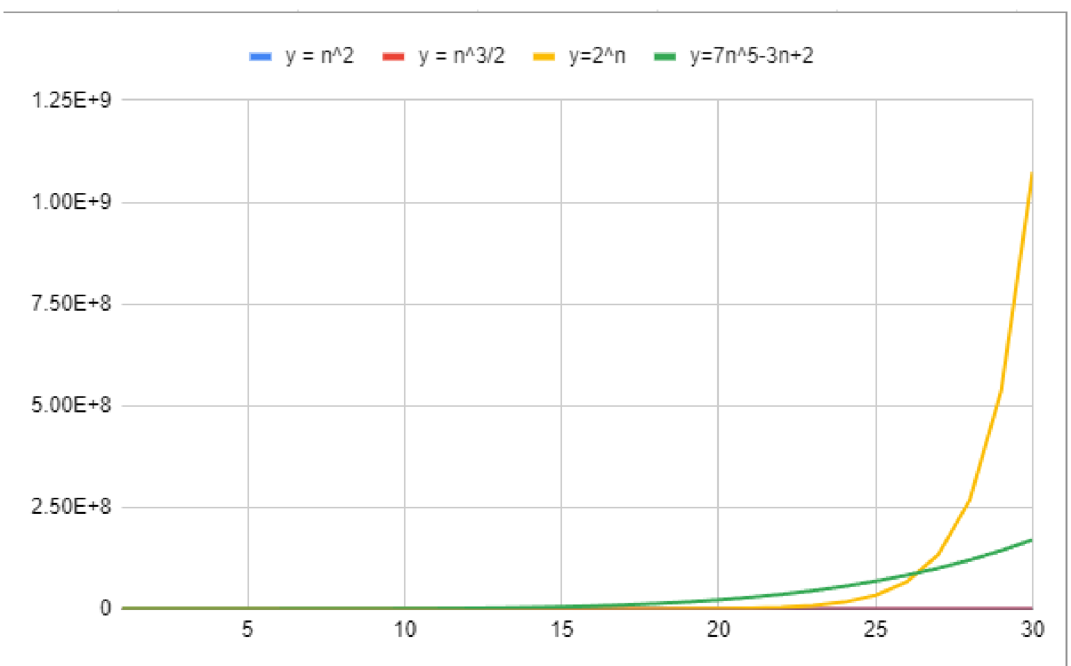
$$6n^5 \leq c * 2^n \tag{19}$$

Con $\boxed{n = 1}$ $\boxed{c = 4}$

$$6 \leq 8 \tag{20}$$

Para este caso la función más baja siguiente no sería posible encontrar un **c** para acotarla por debajo y que tenga la misma dirección por lo que tambien la funcion Omega es:

$$\Omega(2^n) \tag{21}$$



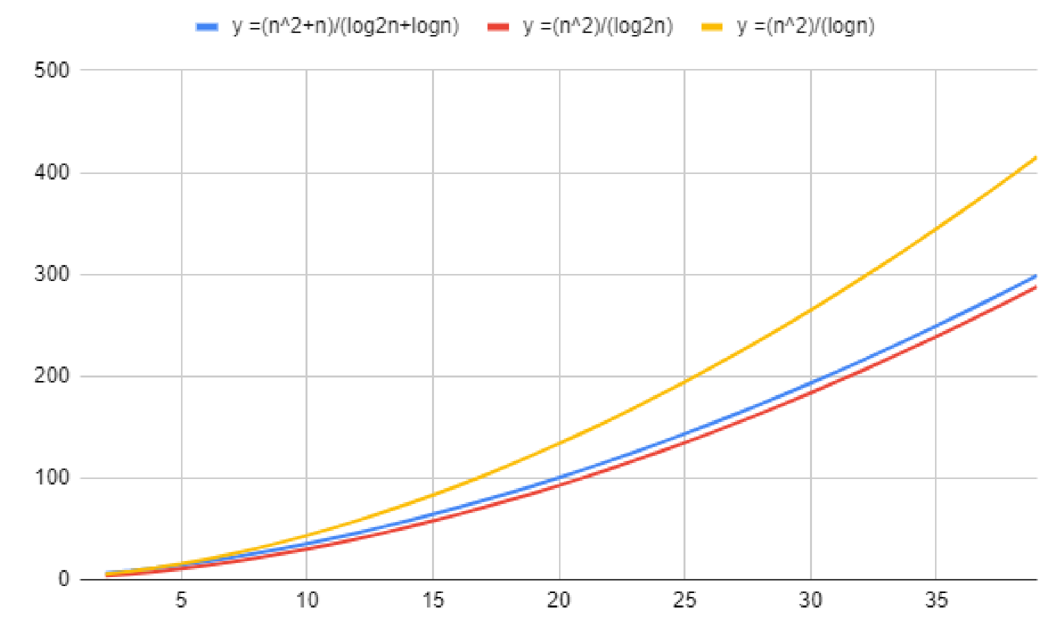
3- Continuamos con el analisis de $n^2 + n/log2n + logn$:

Por la definicion de la función podemos analizar que las opciones que la acotarían mejor son entre las siguientes:

$$n^2/log2n \tag{22}$$

$$n^2/logn \tag{23}$$

Si graficamos las funciones obtenemos que:



Con la visuazlización del grafico vemos que cualquier constante c nos serviría para acotar exitosamente, por lo que tenemos:

$$O(n^2/logn) \tag{24}$$

$$\Omega(n^2/log2n) \tag{25}$$

2.4- Continuamos con el analisis de 2^{log2n} :

Para este caso podemos primero aplicar una de las propiedades de los logaritmos y tendriamos que la función tambien la podemos representar como:

$$2^{log2n} => n \tag{26}$$

Seguidamente podemos demostrar que Big Oh y Omega tambien son **n**:

$$O(n) \tag{27}$$

$$fn \leq c * g(n) \tag{28}$$

$$n \leq c * g(n) \tag{29}$$

$$1 \leq c * 1 \tag{30}$$

$$1 \leq 1 \tag{31}$$

Con $\boxed{n = 1}$ $\boxed{c \geq 1}$

$$O(n) \tag{32}$$

Para demostrar el Omega realizamos:

$$\Omega(n) \tag{33}$$

$$fn \geq c * g(n) \tag{34}$$

$$n \geq c * g(n) \tag{35}$$

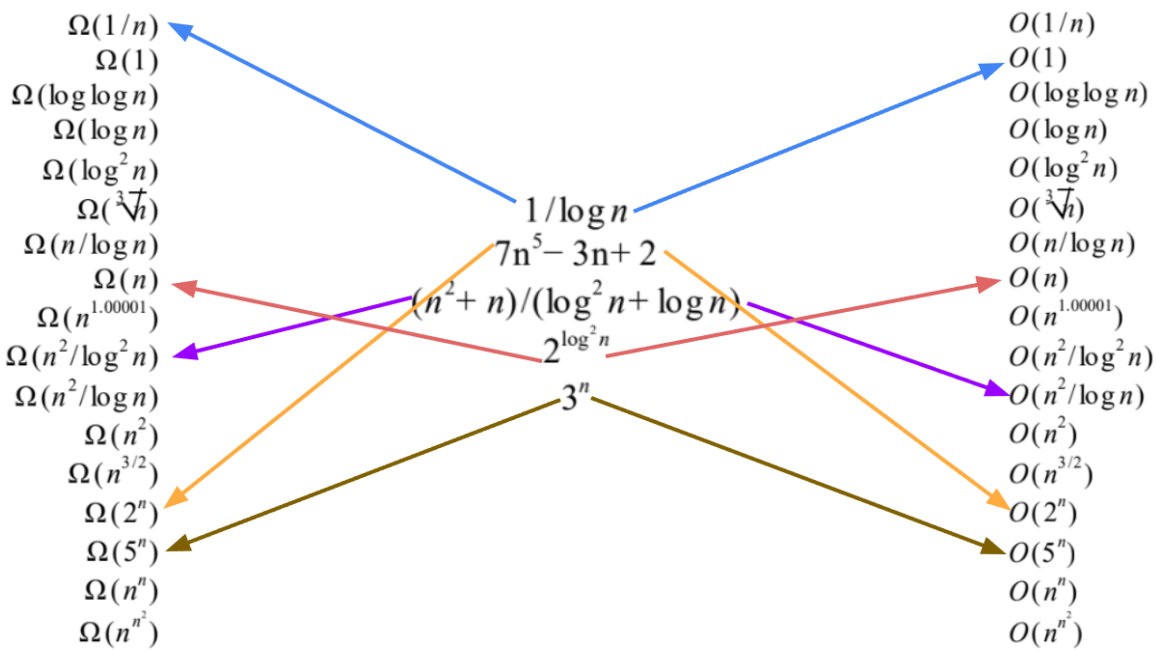
$$n/n \leq (c * n)/n \tag{36}$$

$$1 \leq c/n \tag{37}$$

Con $\boxed{n \geq 1}$ $\boxed{c = 1}$

$$\Omega(n) \tag{38}$$

Finalmente el asocie queda de la siguiente manera:



3. ¿Cuánto tiempo requiere el siguiente "algoritmo" como función de n?

```
s←0
for i←1 to n do
  for j←i to n do
    for k←1 to j do
      s←s+1
```

No interesa que cuente cada operación que ocurre, sino el número de veces que se ejecuta la instrucción más anidada.

Para realizar este analisis utilizamos la tecnica del barometro y escogeremos el tercer ”for” como nuestro barometro:

```
s←0
for i←1 to n do
  for j←i to n do
    for k←1 to j do
      s←s+1
```

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=i}^{n}\mathbf{1} &= \sum_{i=1}^{n} n-i+1 \\
&= \sum_{i=1}^{n} n-i+1 \\
&= (n-i)*(n-i+1)/2+1 \\
&= n^2-2ni+i^2+n-i/2+1 \\
&= 1/2*n^2-2ni+i+n+1
\end{aligned} \tag{39}
$$

Finalmente el tiempo que requiere el algoritmo por la regla del máxinmo sería:

$$\theta(n^2) \tag{40}$$

4. Resolver exactamente la siguiente recurrencia:

$$t(n) = \begin{cases} 2^n + 10 & n = 0,1,2 \\ t_{n-1} + 4t_{n-2} - 4t_{n-3} & n \geq 3 \end{cases}$$

Primero generamos la ecuación característica:

$$t_n - t_{n-1} - 4t_{n-2} + 4t_{n-3} = 0 \tag{41}$$

Segundo generamos el polinomio característico:

$$x^3 - x^2 - 4x + 4 = 0 \tag{42}$$

Tercero obtenemos las nuevas raices con las soluciones de la ecuación cúbica: $\boxed{r_1 = -2}$ $\boxed{r_2 = 2}$ $\boxed{r_3 = 1}$

$$c_1 * -2^n + c_2 * 2^n + c_3 * 1^n \tag{43}$$

- Ahora armamos el sistema de ecauciones de 3x3:

$$\begin{aligned} c_1 + c_2 + c_3 &= 11|\text{ n} = 0 \\ -2c_1 + 2c_2 + c_3 &= 12|\text{ n} = 1 \\ -4c_1 + 4c_2 + c_3 &= 18|\text{ n} = 2 \end{aligned} \tag{44}$$

Las soluciones del sistema son: $\boxed{c_1 = 1}$ $\boxed{c_2 = 4}$ $\boxed{c_3 = 6}$

$$\begin{aligned} tn &= -2^n + 4 * 2_n + 6 * 1^n \\ tn &= -2^n + 2^2 * 2_n + 6 \\ tn &= -2^n + 2^{n+2} + 6 \end{aligned} \tag{45}$$

- Finalmente tenemos que $\boxed{2^{n+2}}$ es la parte que domina por lo tanto

$$T(n)\epsilon\theta(2^{n+2}) \tag{46}$$

5. Resolver exactamente la siguiente recurrencia:

$$t(n) = \begin{cases} n+1 & n = 0,1 \\ 3 \cdot t_{n-1} - 2 \cdot t_{n-2} + 3 \cdot 2^{n-2} & n \geq 3 \end{cases}$$

Primero generamos la ecuación característica:

$$t_n - 3t_{n-1} + 2t_{n-2} = 3 * 2^{n-2} \tag{47}$$

Del lado izquierdo de la ecuación obtenemos las siguientes raices:

$$x^2 - 3x + 2 = 0 => (x-2)(x-1) \tag{48}$$

Del lado derecho de la ecuación tenemos que:

$$3 * 2^{n-2} \tag{49}$$

$\boxed{b = 2}$ $\boxed{d = 1}$

Por lo que tenemos las siguientes raices finales: $\boxed{(x-2)^3}$ $\boxed{(x-1)}$

Con esto creamos la nueva ecuacion:

$$t_n = c_1 1^n + c_2 2^n + c_3 n 2^n + c_4 n^2 2^n \tag{50}$$

- Ahora armamos el sistema de ecauciones de 4x4:

$$\begin{aligned} c_1 + c_2 + 0 + 0 &= 1| \text{ n = 0} \\ c_1 + 2c_2 + 2c_3 + 2c_4 &= 2| \text{ n = 1} \\ c_1 + 4c_2 + 8c_3 + 16c_4 &= 10| \text{ n = 3} \\ c_1 + 8c_2 + 24c_3 + 17c_4 &= 38| \text{ n = 4} \end{aligned} \tag{51}$$

Las soluciones del sistema son: $\boxed{c_1 = 6}$ $\boxed{c_2 = -5}$ $\boxed{c_3 = 3}$ $\boxed{c_4 = 0}$

$$tn = -5 * 2^n + 3 * n2_n + 0 * n^2 2_n \tag{52}$$

- Finalmente tenemos que $\boxed{n2^n}$ es la parte que domina por lo tanto

$$T(n) \epsilon \theta(n2^n) \tag{53}$$