Instituto Politécnico Nacional
Escuela Superior de Computo

Ejercicio 09

"Análisis de Algoritmos Recursivos"

Mora Ayala José Antonio

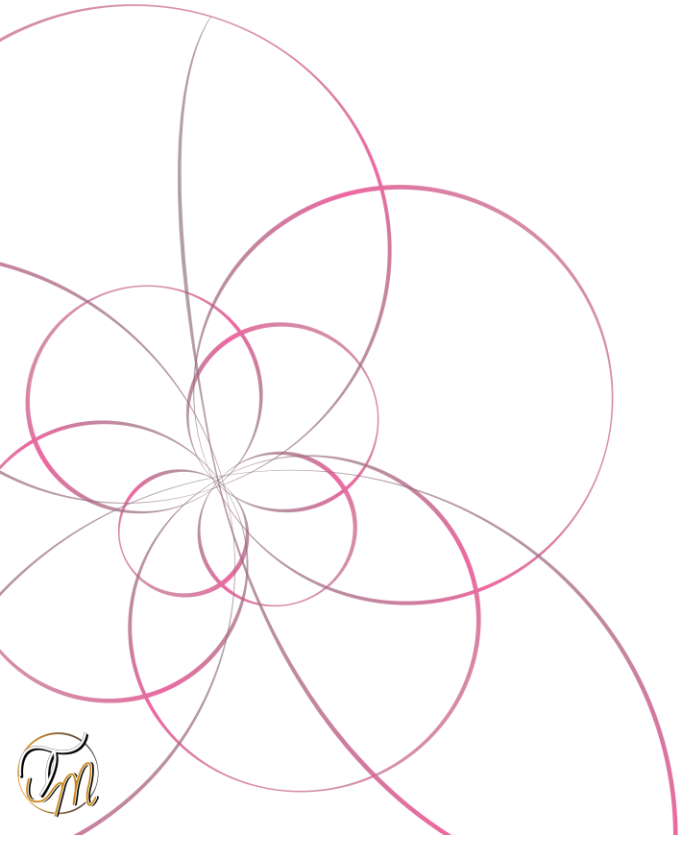
Análisis de Algoritmos

INSTRUCCIONES

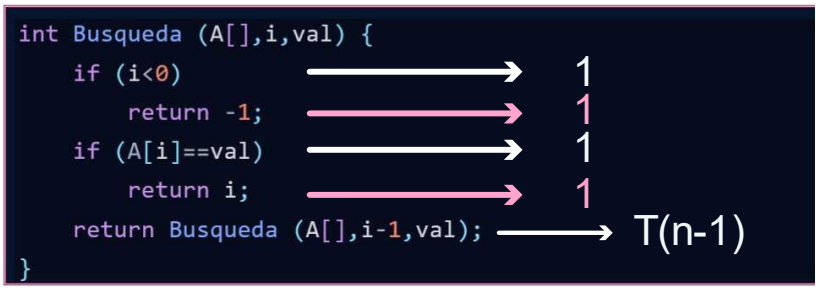
Para los siguientes 6 algoritmos determine la cota de complejidad O (cota superior ajustada) bajo el principio del peor caso, de cada algoritmo. Indique a detalle el procedimiento de obtención de la cota.

*Incluir portada con los de datos del alumno, datos del trabajo y fotografía del alumno

*Recordar manejar encabezados y pies de página.



EJERCICIO 1



$$T(n) = \begin{cases} 2 \; cuando \; n = 0 \\ T(n-1) + 4 \end{cases}$$

Para la solución de este ejercicio debemos comenzar analizando el algoritmo en cuestión, para así poder determinar las operaciones primitivas que son realizadas para un caso base (caso en el cual la recursión terminara), en caso de no existir este caso base la recursión nunca terminaría provocando una situación indeseada, un bucle infinito. Consideraremos las siguientes operaciones:

- aritméticas
- Comparaciones
- Asignaciones
- Return

Teniendo en mente esto, podemos observar como para nuestro caso base de n=0 tendremos solamente dos operaciones primitivas aplicadas.

Cuando n no sea 0 observamos que pasara por las 4 operaciones siguientes, las cuales son:

- 1 comparación
- 1 Comparación
- Return
- Finalmente la aritmética dentro de la recursividad

Teniendo como resultado 4 operaciones primitivas

$$T(n) = T(n-1) + 4$$

Ahora comenzaremos con la solución de nuestra ecuación, comenzando con agrupar los términos en sus respectivos lados mediante los despejes

$$T(n) - T(n-1) = 4$$

Una vez realizado esto podemos darnos cuenta que estamos frente a un caso de ecuación no homogénea por lo cual realizaremos los siguientes cambios

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

Una vez obtenida la ecuación con los exponentes correspondientes, podemos observar que se nos presenta una forma muy sencilla de poder realizar la obtención de los valores de las raíces, pues solo debemos igual a 0 ambos elementos del producto y realizar el despeje, teniendo como resultado 2 raíces con el mismo valor o bien 1 raíz de multiplicidad 2

$$r_1 = 1 \qquad r_2 = 1$$

Ya que obtuvimos raíces del mismo valor nuestra ecuación característica quedara de la siguiente forma:

$$T(n) = C_1(1)^n + C_2 n(1)^n$$

Ahora procedemos a buscar el valor de nuestras constantes, por lo cual nos auxiliaremos del caso base para poder sacar otra ecuación que nos auxilie para cumplir el objetivo principal, así que buscaremos una ecuación cuando el valor de n=1

$$\begin{aligned} \mathrm{T}(1) &= \mathrm{T}(1-1)+4 \\ \mathrm{T}(1) &= \mathrm{T}(0)+4 \\ \mathrm{T}(1) &= (2)+4 \\ \mathrm{T}(1) &= 6 \end{aligned}$$

Una vez realizado esto podemos observar que al momento de realizar la sustitución con n=0 obtenemos de forma inmediata el valor de C_1 el cual es 2, por lo cual solo basta con sustituir en nuestra segunda ecuación para obtener el valor de la constante que nos hace falta

$$\begin{aligned} T(0) &= C_1 + C_2(0) \quad \rightarrow \; C_1 = 2 \\ T(1) &= C_1 + C_2(1) \quad \rightarrow \; C_1 + C_2 = 6 \end{aligned}$$

$$\begin{aligned} C_1 &= 2 \\ 2 + C_2 &= 6 \\ C_2 &= 4 \end{aligned}$$

Ya con el valor de las contantes podemos determinar de forma completa la ecuación característica y por lo tanto una cota adecuada a esta.

$$C_1 = 2 \qquad C_2 = 4$$
$$\boldsymbol{T(n) = 2 + 4n \rightarrow \; \epsilon\, O(n)}$$

EJERCICIO 2

```
int coef (int n , int k ){
    if(k==0 || k==n)
        return 1;
    else if (k>0&&k<n)
        return coef (int n-1 , int k-1 )+coef (int n-1 , int k );
}
```

El funcionamiento de este algoritmo nos proporciona el coeficiente que tendrá cada elemento del algoritmo de pascal, el cual para ser implementado y visto de una forma lógica al momento de ser ejecutado, este debe verse envuelto en 2 ciclos anidados, donde se vaya pasando el valor de n y el valor de k, pues si la función solo es llamada tendremos que pasar nosotros mismos los argumentos, en caso de colocar un nivel 3 con k 2, tendremos que en la posición 2 de lo que podemos ver como arreglo tendremos un coeficiente de 3

1 - 3 - 3- 1 ==========➔ C(3,2)

[0] [1] [2] [3]

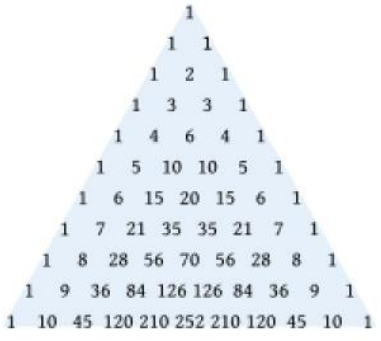
Lo anterior muestra un pequeño ejemplo del caso mencionado n=3 y k=2, es por eso que se hace mención de que para la obtención de los coeficientes sería necesario realizar este dentro de bucle anidado.

Este algoritmo como tal no le podemos determinar una función de la forma en que lo hemos estado haciendo, pues en esta ocasión dependemos de 2 variables que afectan directamente el funcionamiento de la recursividad, por lo cual una depende de la otra.

Aunque es posible ver este algoritmo como un árbol, el cual tendrá una altura de n, así mismo se puede observar que el comportamiento que tiene k va creciendo conforme lo hace n

Una formula que describe el funcionamiento de este algoritmo es la siguiente:

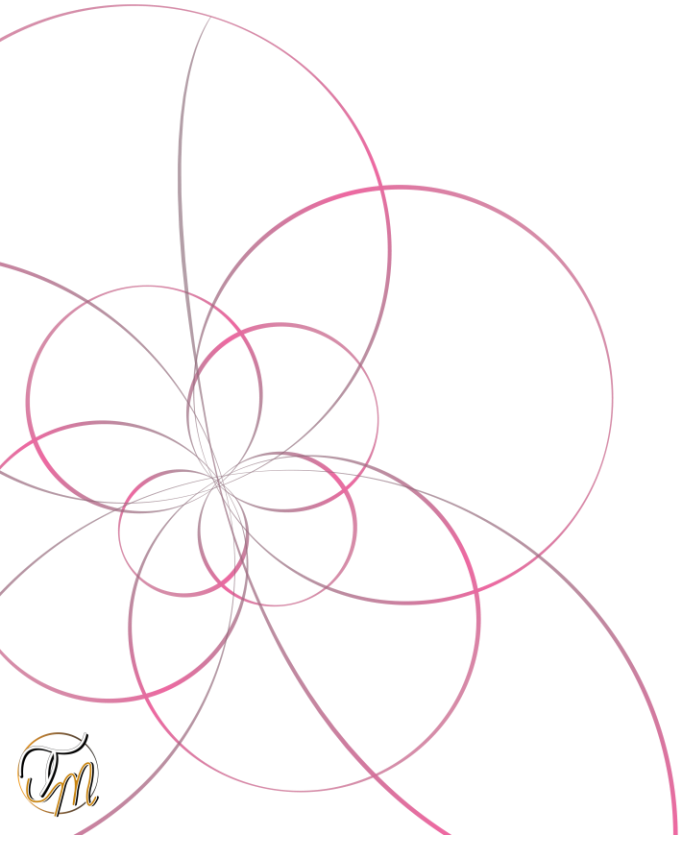
$$\frac{n!}{(n-k)! * k!}$$



Como se ha hecho mención a lo largo de la explicación, el triángulo de pascal es una excelente forma de poder establecer una cota aproximada a nuestro algoritmo, Las combinaciones posibles con nuestro ejemplo de n = 3 y k =2 es igual a 3 como podemos apreciar en el triángulo de pascal

Una vez dicho todo esto, podemos determinar una cota de n^2debido a el caso de que tenemos 2 ciclos anidados para la proporción de los coeficientes en la forma que se observa gráficamente en el triángulo, aunque cabe destacar que se realiza la mención de que este ciclo dependía de 2 variables distintas, una cumple la condición de ser menor que la otra, por lo cual podemos llegar a una cota aproximada de este valor:

$$O, k \leq n$$



EJERCICIO 3

```
Palindromo (cadena){
    if (longitud(cadena)==1)          ⟶  1
    {
        return TRUE                   ⟶  1
    }
    if (primer_caracter(cadena)!=ultimo_caracter(cadena))
    {
        return FALSE
    }
    cadena=remover_primer_ultimo_caracter(cadena);   ⟶  1
    Palindromo(cadena);                              ⟶  T(n-2)
}
```

Para el análisis de este algoritmo como en los ejercicios anteriores será establecer un caso base en cual la recursividad deje de darse para poder evitar la generación poco deseada de un bucle infinito, de igual manera cabe destacar que el algoritmo solo es valido cuando el numero n, es distinto de un numero par, pues todos los palíndromos que hemos buscado resultaron coincidir en que tienen un numero impar de letras.

Las operaciones que se van a contar son:

1. Valores retornados (return)
2. Operaciones aritméticas
3. Comparaciones (if)
4. Asignaciones

Dicho esto podemos proceder a la búsqueda de una cota de complejidad mediante las funciones vistas en clase, teniendo como base lo siguiente

$$T(n) = \begin{cases} 2 \; cuando \; n = 1 \\ T(n-2) + 3 \end{cases}$$

Nuestro caso base sucede cuando n=1, debido a que realizara una comparación y finalizara con la devolución de un "verdadero", teniendo simplemente dos operaciones básicas en cuestión.

Para el caso de fuera un valor distinto podremos llegar a la recursión, teniendo primero:

- La primera comparación
- Segunda comparación
- 1 asignación
- Recursividad T(n-2)

Aquí podrá surgir una interrogante para algunos, ¿De donde es que sale la resta de 2 elementos a esa n en la parte recursiva?, fácil observando el código podemos observar como hay una función que quitara la letra inicial y la ultima letra en cuestión de la cadena (función que es ejecutada al momento de hacer la asignación a la cadena), es por eso que se le deben restar 2 unidades a nuestra n en cuestión

Dicho esto, podemos comenzar a realizar las operatividades correspondientes:

$$T(n) = T(n-2) + 3$$
$$T(n) - T(n-2) = 3$$

Podemos observar que tenemos una ecuación no homogénea y que nuestra K=2, por lo cual este será y grado del polinomio que a continuación vamos a establecer, así como la asignación con respecto a la estructura que las ecuaciones no homogéneas poseen, la cual es:

$$T(n) - T(n-2) = b^n p(n)$$
$$b^n = 1^0 \quad d = 0$$

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

Realizado esto, podemos proceder a la búsqueda de las raíces correspondientes, factorizamos el primer elemento del producto:

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

Tenemos las siguientes raíces mediante la igualación de cada elemento del producto con 0 y su posterior despeje $r_1 = -1 \quad r_2 = 1 \quad r_3 = 1$

Tenemos dos raíces con el mismo valor, por lo que nuestra ecuación característica quedara de la siguiente manera:

$$T(n) = C_1(-1)^n + C_2(1)^n + C_3 n(1)^n$$
$$T(n) = C_1(-1)^n + C_2 + C_3 n$$

Para encontrar nuestras constantes será necesario encontrar otras dos ecuaciones que nos sirva como apoyo, como se menciono los valores de n deben ser números impares, por lo que lo haremos para n=3 y n=5, ya que por la estructura que tenemos de nuestro caso de recursión serán valores con los cuales podremos sacar otras ecuaciones de apoyo:

$$2 = -C_1 + C_2 + C_3$$
$$5 = -C_1 + C_2 + 3C_3$$
$$8 = -C_1 + C_2 + 5C_3$$

Mediante una herramienta de solución de ecuaciones tenemos como resultado lo siguientes valores

$$C_2 = C_1 + \frac{1}{2}. \qquad C_3 = \frac{3}{2}$$

Como podemos observar nuestro sistema nos proporciona la información de que nuestros valores depende de valores asignados a otras variables, sustituyendo en nuestra formula tendríamos lo siguiente:

$$T(n) = C_1(-1)^n + C_1 + \frac{1}{2} + \frac{3}{2}n$$

Teniendo en cuenta lo que ya se había comentado, "solo acepta algoritmos de n impar" ya que estamos teniendo como caso base cuando n=1, y en caso de tomar un numero par para n las restas no llegarían a nuestro caso base de 1, por lo cual sería infinito

$$T(n) = -C_1 + C_1 + \frac{1}{2} + \frac{3}{2}n$$
$$T(n) = \frac{1}{2} + \frac{3}{2}n$$

Operatividad que puede ser realizada debido a que estamos quitando del camino aquellas potencias que sean pares, C_1 nunca llegara a ser positivo (razón por la cual quitamos esa potencia y simplemente lo dejamos como negativo), teniendo finalmente:

$$\boldsymbol{T(n) = \frac{1}{2} + \frac{3}{2}n \ \rightarrow \ \epsilon\, O(n)}$$

EJERCICIO 4

```
SubAlgoritmo Volados (n,cadena)
    Si n!=0                                          ⟶  1
        Volados(n-1,concatenar(cadena,'S'));         ⟶  T(n-1)
        Volados(n-1,concatenar(cadena,'A'));         ⟶  T(n-1)
    SiNo
        Mostrar cadena                               ⟶  1
    FinSi
FinSubAlgoritmo
```

$$T(n) = \begin{cases} 2 \; cuando \; n = 0 \\ 2T(n-1) + 1 \end{cases}$$

Tal como en el resto de ejercicios realizados hasta el momento comenzaremos con el análisis de nuestro algoritmo y tendremos que determinar un caso base adecuado para este, podemos observar que cuando n=0 simplemente realizara dos operaciones primitivas, las cuales son:

- 1 comparación
- Llamada a función mostrar cadena

E n caso contrario realizará la recursión dos veces más aparte la comparativa inicial, la cual en este caso si será cumplida, por lo cual tendremos la siguiente ecuación para este algoritmo:

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

Procedemos a colocar las T del lado izquierdo y dejas las constantes solas en el lado derecho.

Nos podemos dar cuenta de que se nos presenta una ecuación no homogénea nuevamente por lo cual realizaremos el siguiente cambio:

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

Dado que esta expresión no puede ser más factorizada simplemente igualamos a 0 cada uno de los elementos del producto para posteriormente despejas las x que corresponder a cada uno, teniendo como resultado las siguientes raíces:

$$r_1 = 2 \qquad r_2 = 1$$

Dado que nuestras raíces son de valores distintos, nuestra ecuación característica quedará de la siguiente manera:

$$T(n) = C_1(2)^n + C_2(1)^n$$

Apoyándonos de que T(0)=2 (caso base que hemos establecido), llegamos a esta ecuación

$$C_1 + C_2 = 2$$

Por lo cual procedemos a realizar la búsqueda de la ecuación resultante cuando n=1:

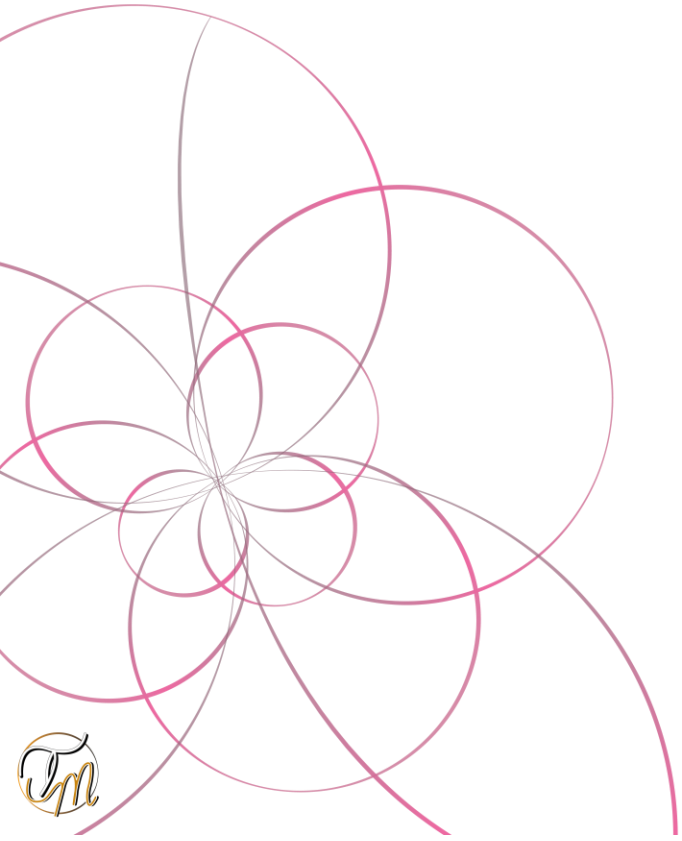
$$T(1) = 2T(1-1) + 1$$
$$T(1) = 2T(0) + 1$$
$$T(1) = 2(2) + 1$$
$$T(1) = 5$$

Teniendo ya dos ecuaciones podemos proceder a la solución del sistema de ecuaciones 2x2, para el cual se ha hecho uso de un software matemático online

$$C_1 + C_2 = 2$$
$$2C_1 + C_2 = 5$$

Una vez obtenidos los valores de las constantes podemos realizar la sustitución pertinente en la ecuación característica ya con los valores de las constantes en su respectivo lugar, teniendo como resultado la siguiente ecuación y su respectiva cota

$$C_1 = 3 \qquad C_2 = -1$$
$$\boldsymbol{T(n) = 3(2)^n - 1 \rightarrow \epsilon\, O(C^n)}$$



EJERCICIO 5

```
DecABin(n){
    if(n>1)
    DecABiin(n/2);
    Mostrar(n%2);
}
```

⟶ 1
⟶ T(n/2)
⟶ 1

$$T(n) = \begin{cases} 2 \; cuando \; n = 0 \\ T\left(\frac{n}{2}\right) + 1 \end{cases}$$

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

Como podemos observar en esta ecuación se nos presenta una situación diferente, pues la forma en que cambia el valor de n ya no se ve afectada por una operatividad de adición o substracción con un entero, si no que esta siendo modificada mediante una división, situación que provoca un comportamiento logarítmico dentro de nuestras funciones, como hemos visto en el análisis de ejercicios anteriores.

Dada esta situación y mediante los recursos vistos en clase, podemos determinar la cota mediante el "Teorema maestro", el cual nos presenta una forma mas sencilla de poder hacer frente a estas situaciones.

El **teorema maestro** se ve compuesto de la siguiente forma:

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

Relacionándolo directamente con la estructura que nosotros tenemos en nuestra expresión, tenemos como resultado las siguientes asignaciones:

$$a = 1 \quad n = n \quad b = 2 \quad f(n) = 1$$

Por lo cual podemos proceder a realizar la sustitución con respecto al uso del segundo caso:

$$f(n) = \theta\left(n^{\log_b a}\right) \therefore \;\; T(n) = \theta\left(n^{\log_b a} \lg(n)\right)$$
$$f(n) = \theta\left(n^{\log_2 1}\right)$$
$$f(n) = \theta(1^0)$$
$$f(n) = \theta(1)$$

Dado que hemos conseguido la misma cota de complejidad que en efecto tiene nuestra f(n), la aplicación del segundo método funciona de forma correcta, por lo cual no es necesario realizar la está de algún valor ε, por lo cual nuestra cota exacta quedará de la siguiente manera:

$$T(n) = \theta\left(n^{\log_2 1} \lg(n)\right)$$
$$T(n) = \theta(1^0 \lg(n))$$
$$\boldsymbol{T(n) = \theta(\lg(n))}$$

EJERCICIO 6

```
int Producto(int a, int b){
    if(b==0)
        return 0;
    else
        return a+Prodcuto(a,b-1);
}
```

⟶ 1
⟶ 1
⟶ T(n-1)+2

$$T(n) = \begin{cases} 2 \; cuando \; n = 0 \\ T(n-1) + 4 \end{cases}$$

Como en el resto de ejercicios debemos comenzar con el análisis de este algoritmo y debemos establecer un caso base, en el cual la recursión finalizara, para no provocar resultados poco deseados, dadas las operatividades primitivas que se han establecido para contar, podemos notar que cuando sucede que b=0 (o en caso de la aplicación de nuestra función n=0), solo se realizaran dos operatividades, pues la condición inicial será cumplida y entrara, posteriormente realizara un return con valor de 0. Dado esto solo tuvimos 2 operaciones, para el resto, podremos llegar a la aplicación de la recursión la cual esta establecida por la resta en una unidad a el valor sobre el cual aplica la recursión y 4 operaciones más:

- 1 Comparación
- Return
- Suma de (a+Funcion)
- Decremento a b
- Aplicación de recursividad T(n-1)

Dada la explicación anterior, nuestra función queda de esta forma

$$T(n) - T(n-1) = 4$$

Ahora procedemos a realizar el procedimiento que llevamos haciendo hasta el momento con el resto de ejercicios, acomodando los términos en su lugar correspondiente para poder determinar si estamos frente a una ecuación homogénea o no homogénea

$$T(n) = T(n-1) + 4$$

Como se observó, la ecuación fue no homogénea, por lo cual será necesario realizar el cambio pertinente para poder llegar a esta ecuación:

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

Una vez realizado este procedimiento procedemos a la obtención de raíces, las cuales como podemos observar son del mismo valor, por lo cual podemos decir que tenemos una raíz de multiplicidad 2

$$r_1 = 1 \qquad r_2 = 1$$

Dado que la multiplicidad de nuestras raíces fue de valor 2, la ecuación característica para este ejercicio quedará de la siguiente manera:

$$T(n) = C_1(1)^n + C_2 n(1)^n$$

$$C_1 + C_2 n = 2$$

Partiendo del caso base que nosotros establecimos podemos realizar la obtención de otro caso, cuando n=1 tendremos lo siguiente:

$$Sabiendo\ que\ T(0) = 2$$

$$\mathrm{T}(1) = \mathrm{T}(1-1) + 4$$

Como podemos observar tenemos T(0), entonces simplemente sustituimos por el valor que establecimos

$$\mathrm{T}(1) = \mathrm{T}(0) + 4$$
$$\mathrm{T}(1) = 2 + 4$$
$$\mathrm{T}(1) = 6$$

Procedemos a establecer el sistema de ecuaciones correspondiente

$$C_1 + C_2(0) = 2$$
$$C_1 = 2$$

Sustituyendo el valor

$$C_1 + C_2 = 6$$
$$2 + \mathrm{C}_2 = 6$$
$$C_2 = 4$$
$$\boldsymbol{T(n) = 2 + 4n \ \rightarrow \ \epsilon\, O\,(n)}$$

PROCEDIMIENTOS A MANO

Dec a Bin

$$T(n)=\begin{cases} 2 & \text{Si } n=0 \\ T(n)=1+T(n/2) \end{cases}$$

$T(n)=T(\frac{n}{2})+1$ Tenemos que recurrir al Teorema maestro dado que tenemos presente la siguiente estructura

$$T(n)=aT(n/b)+f(n)$$

Tenemos que
$a=1 \quad n=n \quad b=2 \quad f(n)=1$

Si $f(n)=\Theta(n^{\log_b a})$ $\therefore$ $T(n)=\Theta(n^{\log_b a}\lg(n))$

$f(n)=\Theta(1^{\log_2 1})$
$=\Theta(1^0)$
$=\Theta(1)$ $\longrightarrow$ $T(n)=\Theta[n^{\log_2 1}\lg(n)]$
$=\Theta[1\lg(n)]$
$\Theta[\lg(n)]$

Palindromo

$$T(n)=\begin{cases} 2 & n=1 \\ T(n-2)+3 \end{cases}$$

$T(n)=T(n-2)+3$
$T(n)-T(n-2)=3$ Ecuación no homogenea

$(x^2-1)(x-1)=0$
$(x+1)(x-1)(x-1)=0$
$r_1=-1$
$r_2=1$
$r_3=1$ } 2 raices iguales $\longrightarrow$ $T(n)=C_1(-1)^n+C_2(1)^n+C_3n(1)^n$
$=C_1(-1)^n+C_2+nC_3$

$T(1)=-C_1+C_2+C_3=2$ $\quad C_2=C_1+1/2$
$T(3)=-C_1+C_2+3C_3=5$
$T(5)=-C_1+C_2+5C_3=8$ $\quad C_3=0/2$

Ejercicio Producto

$$T(n) = \begin{cases} 2 & \text{si } n=0 \\ T(n-1)+4 \end{cases}$$

Consideramos los llamados a la función cuando ocurre la recursividad
Realiza 4 operaciones aparte

Solución

$T(n) = T(n-1)+4$
$T(n) - T(n-1) = 4$
$(x-1)(x-1) = 0$

$b^n p(n)$
$(b^n) = 1^0$ Ecuación no homogenea

$x=1$ multiplicidad 2

$T(n) = C_1(1)^n + C_2 n(1)^n$
$T(n) = C_1 + C_2 n$

Sabiendo que $T(0)=2$

$T(0) = C_1 + C_2(0)$
$2 = C_1$

Si $T(1) = T(1-1)+4$
$T(1) = T(0)+4$
$T(1) = 2+4$
$T(1) = 6$

$T(1) = C_1 + C_2(1)$
$6 = C_1 + C_2$ Sustituyendo $C_1=2$

$2 + C_2 = 6$
$C_2 = 4$ ∴ $T(n) = 2+4n \rightarrow O(n)$

Ejercicio Volados

$$T(n) = \begin{cases} 2 & \text{Si } n=0 \\ 2T(n-1)+1 \end{cases}$$

$T(n) = 2T(n-1)+1$
$T(n) - 2T(n-1) = 1$ Ecuación no homogenea

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

$r_1 = 2 \quad r_2 = 1$ Raices distintas

$T(n) = C_1(2)^n + C_2(1)^n$
Si $T(0) = 2$
$T(0) = C_1(2)^0 + C_2(1)^0$
$2 = C_1 + C_2$

Si $T(1) = 5$
$T(1) = C_1(2)^1 + C_2$
$5 = 2C_1 + C_2$

$T(n) = 3(2)^n - 1$ ∴

$T(1) = 2T(1-1)+1$
$T(1) = 2(2)+1$
$T(1) = 5$

$C_1 + C_2 = 2$
$2C_1 + C_2 = 5 \quad (-)$

$-C_1 = -3 \qquad C_2 = -1$
$C_1 = 3$

$18 = C_1 + C_2 + C_3$
$61 = -3C_1 - 2C_2 + 4C_3$
$259 = 9C_1 + 4C_2 + 16C_3$

Despejando C_3 de $T(0)$

$C_3 = 18 - C_1 - C_2$

Sustituyendo en ecuación 2

$-3C_1 - 2C_2 + 4(18 - C_1 - C_2) = 61$
$-3C_1 - 2C_2 + 72 - 4C_1 - 4C_2 = 61$
$-7C_1 - 6C_2 = 61 - 72$
$-7C_1 - 6C_2 = -11$

Sustituyendo en ecuación 3

$9C_1 + 4C_2 + 16(18 - C_1 - C_2) = 259$
$9C_1 + 4C_2 - 16C_1 - 16C_2 = 259 - 288$
$-7C_1 - 12C_2 = -29$

$-7C_1 - 6C_2 = -11 \quad (-2)$
$-7C_1 - 12C_2 = -29$

$7C_1 = -7$
$C_1 = -1$

$-7(-1) - 6C_2 = -11$
$-6C_2 = -11 - 7$
$C_2 = 3$

$C_3 = 18 - (-1) - 3$
$C_3 = 16$

$T(n) = 3(-2)^n - (-3)^n + 16(4)^n$