INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

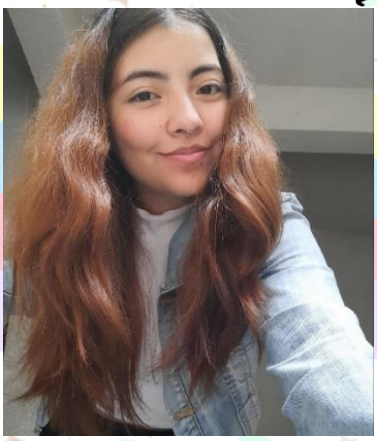
ANÁLISIS DE ALGORITMOS

EJERCICIOS 01:

"CALCULAR EL NÚMERO DE IMPRESIONES"

ALUMNA: MÉNDEZ CASTAÑEDA AURORA

GRUPO: 3CM13



PROFESOR: FRANCO MARTÍNEZ EDGARDO ADRIÁN

Determinar para los siguientes códigos el modelo matemático que determine el número de impresiones en términos de "n" que cada uno realiza de la palabra "Algoritmos" y comprobar empíricamente el resultado.

- Contraste sus funciones con la prueba empírica para los 20 valores de n = {−1,0,1,2,3,5,15,20,100,409,500,593,1000,1471,1500,2801,3000,5000,10000,20000}

Código 01.

```
1   for(i=10;i<n*5;i*=2)
2       printf("Algoritmos");
```

Partimos del 10, para el siguiente ciclo será 20, para el siguiente 40, para el siguiente 80 y así sucesivamente, hasta 5n. De modo que nos podemos dar cuenta que se trata de una potencia y tenemos que:

$$2^x = 5n, \text{ donde x es el número de saltos}$$

Despejando x,

$$\log_2 5n = x$$

Por lo tanto, podemos decir que $\mathrm{f(n)} = \log_2 5n.$ Sin embargo, como se dijo al principio, partimos desde 10, entonces de 0 a 10 tenemos más potencias, por lo que tendríamos que restar un $\log_2 5$ a nuestra función f(n).

Por lo tanto,

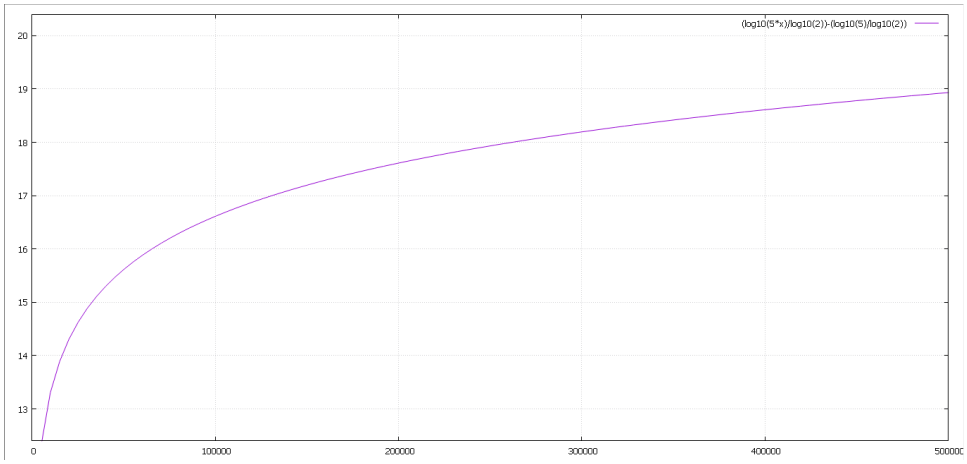
$$\mathbf{f(n) = \log_2 5n - \log_2 5}$$

Tomando piso de la función

Comparamos los resultados teórico y empíricos de algunos valores:

n	Valor teórico	Valor empírico
-1	Indefinido	0
0	Indefinido	0
1	0	0
2	1	0
3	1	1
5	2	2
15	3	3
20	4	4
100	6	6
409	8	8
500	8	8
593	9	9
1000	9	9
1471	10	10
1500	10	10
2801	11	11
3000	11	11
5000	12	12
10000	13	13
20000	14	14

Gráfica de la función $\mathbf{f(n) = \log_2 5n - \log_2 5}$, $\mathbf{0 < n < 500000}$



Código en C

```c
#include <stdio.h>
#include <stdlib.h>

int main (int narg, char** varg){
    int i, j=0, n,cont=0;

    if(narg!=2){
        printf("\nIntroduce una n");
        exit(1);
    }

    n = atoi(varg[1]);

    for(i=10; i<n*5; i*=2){
        printf("\n %d Algoritmos", ++j);
    }
}
```

Las instrucciones de compilación y ejecución son las siguientes:

→ gcc Codigo01.c -o Codigo01.exe
→ Codigo01.exe (valor de n)

Funcionamiento:

```
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo01.exe 1000

 1 Algoritmos
 2 Algoritmos
 3 Algoritmos
 4 Algoritmos
 5 Algoritmos
 6 Algoritmos
 7 Algoritmos
 8 Algoritmos
 9 Algoritmos
```

Código 02.

```
1   for(j=n;j>1;j/=2)
2   {
3       if(j<(n/2))
4       {
5           for(i=0;i<n;i+=2)
6           {
7               printf("Algoritmos");
8           }
9       }
10  }
```

Como podemos observar, el primer for del código anterior empieza en n y en cada salto se va dividiendo entre dos hasta llegar a uno, pero sin tocarlo. Entonces este avanza de la forma: $\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots$ y así podemos deducir que $\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots = \frac{n}{2^x} = 1$, siendo x el número de veces que entra al for.

Despejando x, tenemos qué,

$\frac{\mathrm{n}}{2^{\mathrm{x}}} = 1 \;;\; \mathrm{n} = 2^{\mathrm{x}} \;;\; \log_2 n = x$

Entrando al primer for nos encontramos con la condición de que nuestra variable debe ser menor que $\frac{n}{2}$ y al realizar pruebas de escritorio noté que en ese if entra hasta la 3era vez que entra en el for, es decir que, si este solo entra dos veces en el for, no logrará entrar al if y, por ende, no imprimirá nada. Así que por esta premisa restamos menos 2 a lo que llevamos de la función.

$$f(n) = \log_2 n - 2$$

Pero ahí no termina, ya que aún queda un for dentro del if, este podemos deducir a primera vista que imprimirá $\frac{\mathrm{n}}{2}$ veces.
Por lo tanto,

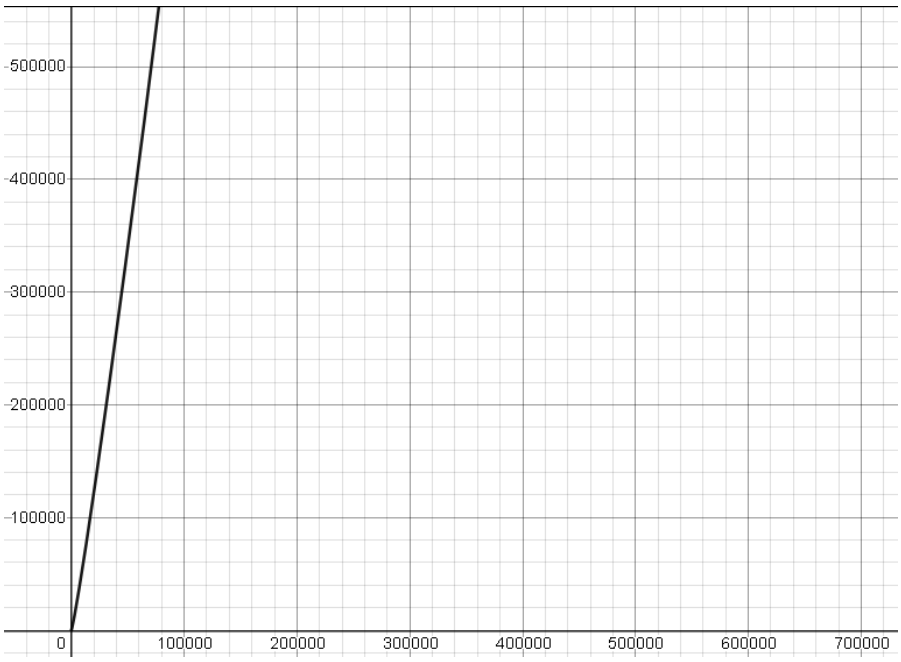
$$\boldsymbol{f(n) = (\log_2 n - 2)\left(\frac{n}{2}\right)}$$

Siendo el logaritmo base dos, piso.

Comparamos los resultados teórico y empíricos de algunos valores:

n	Valor teórico	Valor empírico
-1	Indefinido	0
0	Indefinido	0
1	-1	0
2	-1	0
3	-1	0
5	0	0
15	7	8
20	20	20
100	200	200
409	1227	1230
500	1500	1500
593	2075	2079
1000	3500	3500
1471	5884	5888
1500	6000	6000
2801	12604	12609
3000	13500	13500
5000	25000	25000
10000	55000	55000
20000	120000	120000

Gráfica de la función $\boldsymbol{f(n) = (\log_2 n - 2)\left(\frac{n}{2}\right),\ \ 0 < n < 500000}$



Código en C.

```c
#include <stdio.h>
#include <stdlib.h>

int main (int narg, char** varg){
    int i, j, n,cont=0;

    if(narg!=2){
        printf("\nIntroduce una n");
        exit(1);
    }

    n = atoi(varg[1]);

    for(j=n; j>1; j/=2){
        if(j<(n/2)){
            for(i=0;i<n;i+=2){
                printf("\n %d Algoritmos", ++cont);
            }
        }
    }
}
```

Las instrucciones de compilación y ejecución son las siguientes:

- → gcc Codigo02.c -o Codigo02.exe
- → Codigo02.exe (valor de n)

Funcionamiento:

```
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo02.exe 8

 1 Algoritmos
 2 Algoritmos
 3 Algoritmos
 4 Algoritmos
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo02.exe 15

 1 Algoritmos
 2 Algoritmos
 3 Algoritmos
 4 Algoritmos
 5 Algoritmos
 6 Algoritmos
 7 Algoritmos
 8 Algoritmos
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo02.exe 18

 1 Algoritmos
 2 Algoritmos
 3 Algoritmos
 4 Algoritmos
 5 Algoritmos
 6 Algoritmos
 7 Algoritmos
 8 Algoritmos
 9 Algoritmos
 10 Algoritmos
 11 Algoritmos
 12 Algoritmos
 13 Algoritmos
 14 Algoritmos
 15 Algoritmos
 16 Algoritmos
 17 Algoritmos
 18 Algoritmos
```

Código 03.

```
1   for(i=0;i<n*5;i+=2)
2   {
3      for(j=0;j<2*n;j++)
4      {
5         for(k=j;k<n;k++)
6         {
7            printf("Algoritmos",cont);
8         }
9      }
10  }
```

Para analizar este algoritmo, lo primero que hice fue obtener las veces que entra en cada ciclo, entonces tenemos que $\frac{5n}{2}$ corresponde al primer ciclo, $2n$ al segundo ciclo y $n-j$ al tercero. Siendo esta última una suma que va aumentando j y nos quedaría $\sum_{j=0}^{n}(n-j)$.Ha simple vista podríamos decir que nuestra función sería la multiplicación de las expresiones obtenidas, sin embargo, al realizar pruebas de escritorio, podemos notar que en el segundo ciclo si entra 2n veces, pero solo entra n veces al tercer ciclo, por lo que descartaríamos esa expresión y nos quedaría:

$$\left(\sum_{j=0}^{n}(n-j)\right)\left(\frac{5n}{2}\right)$$

Obteniendo la convergencia de la suma, tenemos que $\sum_{j=0}^{n}(n-j) = n + \frac{n^2-n}{2}$.

$$f(n) = \left(n + \frac{n^2-n}{2}\right)\left(\frac{5n}{2}\right) = \left(\frac{2n+n^2-n}{2}\right)\left(\frac{5n}{2}\right) = \frac{5n^2+5n^3}{4} = \frac{5n^2(1+n)}{4} = \frac{5n}{2}\frac{n(n+1)}{2}$$

Por lo tanto,

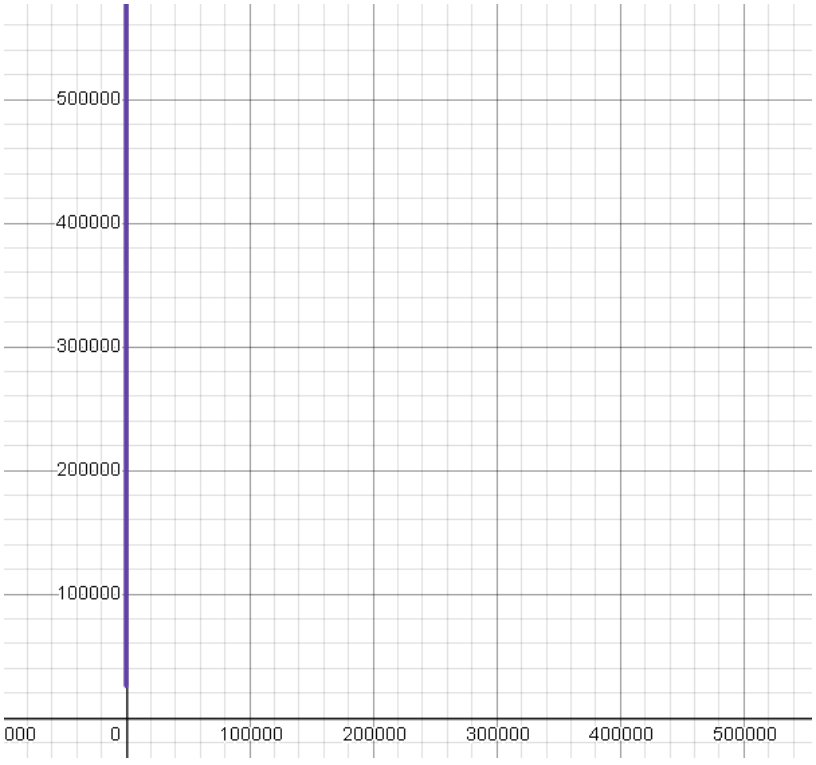
$$\boldsymbol{f(n) = \frac{5n}{2}\frac{n(n+1)}{2}}$$

Redondeando $\frac{5n}{2}$ de la función $f(n)$.

Comparamos los resultados teórico y empíricos de algunos valores:

n	Valor teórico	Valor empírico
-1	0	0
0	0	0
1	3	3
2	15	15
3	48	48
5	195	195
15	4560	4560
20	10500	10500
100	1262500	1262500
409	85773435	85773435
500	156562500	156562500
593	261187443	261187443
1000	1251250000	1251250000
1471	3982008768	-31958528
1500	4221562500	-73404796
2801	27481179603	1711375827
3000	33761250000	-598488368
5000	156281250000	1662427344
10000	1250125×10^{12}	289516864
20000	100005×10^{13}	1816134912

Gráfica de la función $\boldsymbol{f(n) = \frac{5n}{2}\frac{n(n+1)}{2}}$, $\boldsymbol{0 < n < 500000}$



Código C

```c
#include <stdio.h>
#include <stdlib.h>

int main (int narg, char** varg){
    int i, j, n,cont=0,k;

    if(narg!=2){
        printf("\nIntroduce una n");
        exit(1);
    }

    n = atoi(varg[1]);

    for(i=0; i<n*5; i+=2){
        for(j=0; j<2*n; j++){
            for(k=j; k<n; k++){
                printf("\n%d Argoritmos",++cont);
            }
        }
    }
}
```

Las instrucciones de compilación y ejecución son las siguientes:

→ gcc Codigo03.c -o Codigo03.exe

→ Codigo03.exe (valor de n)

Funcionamiento:

```
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo03.exe 2

1 Argoritmos
2 Argoritmos
3 Argoritmos
4 Argoritmos
5 Argoritmos
6 Argoritmos
7 Argoritmos
8 Argoritmos
9 Argoritmos
10 Argoritmos
11 Argoritmos
12 Argoritmos
13 Argoritmos
14 Argoritmos
15 Argoritmos
```

Código 04.

```
1   i=n;
2   while(i>=0)
3       {
4     for (j=n;i<j;i-=2,j/=2)
5     {
6       printf("Algoritmos");
7     }
8   }
```

Para este programa, al realizar pruebas de escritorio, pude notar que nunca se cumple la condición del for, de modo que las variables i y j no cambian, por lo que se queda dentro del while infinitamente porque i siempre será mayor que 0, sino no entraría al while en primera.

Por lo tanto,

$$\boldsymbol{f(n) = 0}$$

Código 05.

```
1   for (i=1;i<4*n;i*=2)
2     for(j=i;j<5*n;j+=3)
3       printf("Algoritmos");
```

Del primer ciclo podemos observar que se trata de una potencia de dos, así que tenemos que:

$2^x = 4n$, donde x es el número de saltos

Despejando x, $\log_2 4n = x$

Dentro de este ciclo tenemos otro ciclo, este comienza en i, por lo que hay que considerar como avanza i en el ciclo anterior y esta avanza de modo que abarca los números pares y se trata de una sumatoria:

$$\frac{5n}{3} + \left(\frac{5n}{3} - 2\right) + \left(\frac{5n}{3} - 4\right) + \left(\frac{5n}{3} - 6\right) \ldots \sum_{i=1}^{n} \left(\frac{5n}{3} - i\right)$$

Sin embargo, hay un error en el sumatorio, si lo representamos así $\sum_{i=1}^{n} \left(\frac{5n}{3} - i\right)$, estamos considerando números impares de igual forma, por lo que habría que encontrar una expresión que, para cada iteración, i nos de 0, 2, 4, 6, 8…

Lo que se me ocurrió fue la expresión $(2i - 2)$ para obtener esos números pares por medio de nuestra variable i entonces probando la expresión tenemos que:

Con i = 1, $(2i - 2) = 2(1) - 2 = 0$
Con i = 2, $(2i - 2) = 2(2) - 2 = 2$
Con i = 3, $(2i - 2) = 2(3) - 2 = 4$

Con i = 4, $(2i-2) = 2(4) - 2 = 6$
Con i = 5, $(2i-2) = 2(5) - 2 = 8$

De esta manera podemos notar que se obtuvo la expresión para los números pares, o eso aparenta. Si suponemos que la i de nuestra sumatoria es diferente a la i de nuestro ciclo, podría funcionarnos. Entonces la sumatoria quedaría de la siguiente forma:

$$\sum_{i=1}^{n} \left(\frac{5n}{3} - (2i-2)\right)$$

Obteniendo su convergencia,

$$\sum_{i=1}^{n} \left(\frac{5n}{3} - (2i-2)\right) = \frac{2}{3}n^2 + n$$

Ahora para obtener la función $f(n)$, multiplicaremos las expresiones obtenidas de cada ciclo.

Por lo tanto,

$$\boldsymbol{f(n) = (\log_2 4n)\left(\frac{2}{3}n^2 + n\right)}$$

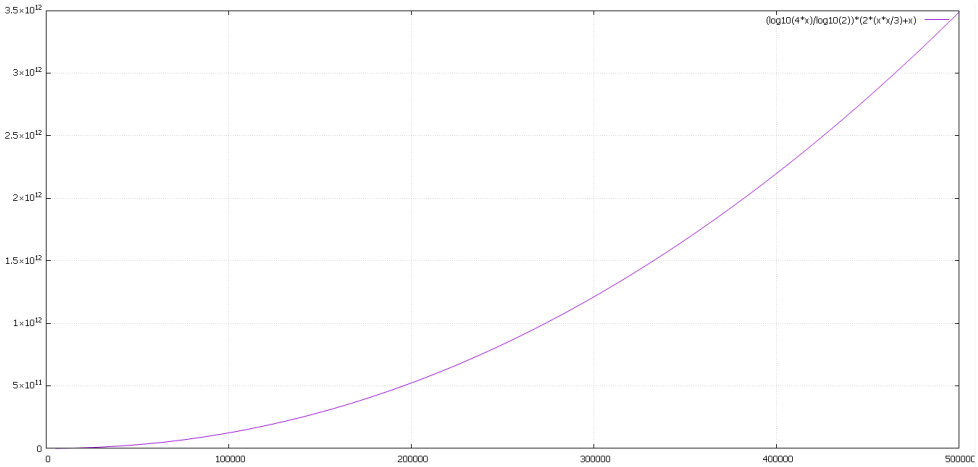
Tomaremos piso de las expresiones.

Comparamos los resultados teórico y empíricos de algunos valores:

n	Valor teórico	Valor empírico
-1	Indefinido	0
0	Indefinido	0
1	3	3
2	14	8
3	32	17
5	84	32
15	825	132
20	1716	192

No hace falta realizar más comparaciones para darnos cuenta de que la función f(n) no nos funciona ya que difiere bastante con el valor empírico, así que podemos concluir que solo se puede mediante una suma, que no es posible obtener con el sumatorio.

Gráfica de la función $\boldsymbol{f(n) = (\log_2 4n)\left(\frac{2}{3}n^2 + n\right),\ 0 < n < 500000}$



Código en C.

```c
#include <stdio.h>
#include <stdlib.h>

int main (int narg, char** varg){
    int i, j, n,cont=0,k;

    if(narg!=2){
        printf("\nIntroduce una n");
        exit(1);
    }

    n = atoi(varg[1]);

    for(i=1; i<4*n; i*=2){
        for(j=i; j<5*n; j+=3){
            printf("\n %d Algoritmos",++cont);
        }
    }
}
```

Las instrucciones de compilación y ejecución son las siguientes:

- → gcc Codigo03.c -o Codigo03.exe
- → Codigo03.exe (valor de n)

Funcionamiento:

```
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo05.exe 1

 1 Algoritmos
 2 Algoritmos
 3 Algoritmos
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas>Codigo05.exe 2

 1 Algoritmos
 2 Algoritmos
 3 Algoritmos
 4 Algoritmos
 5 Algoritmos
 6 Algoritmos
 7 Algoritmos
 8 Algoritmos
```