INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ANÁLISIS DE ALGORITMOS

PROFESORA: LUZ MARÍA SÁNCHEZ GARCÍA

ALUMNO: VÁZQUEZ MORENO MARCOS OSWALDO

2016601777

PRÁCTICA 1 ANÁLISIS DE COMPLEJIDAD TEMPORAL Y TEMPORAL

3CM2

16 DE FEBRERO DE 2019

Introducción

A continuación, se presenta un reporte de la práctica número 1 en el cual se hablará de complejidad espacial y temporal, pero qué son ambos tipos de complejidad, bueno, hablaremos un poco de ellos.

- **Temporal:** Se denomina complejidad temporal a la función T(n) que mide el número de instrucciones realizadas por el algoritmo para procesar los n elementos de entrada. Cada instrucción tiene asociado un costo temporal.

Existen distintos tipos de funciones, por ejemplo:

Las funciones de n pueden ser de diferente tipo:
Funciones constantes: $f(n) = 5$, o bien $g(n) = 10$.
Funciones logarítmicas: $f(n) = \log(n)$, o bien $g(n) = n\log(n)$
Funciones polinomiales: $f(n) = 2n^2$, o bien $g(n) = 8n^2 + 5n$
Funciones exponenciales: $f(n) = 2^n$, o bien $g(n) = 2^{5n}$.
O mezclas de las anteriores, o cualquier función de n en un caso general.

- **Espacial:** Se denomina complejidad espacial al numero de declaraciones en espacio de memoria que se utilizan en un algoritmo. (Martínez)

Planteamiento del problema

El objetivo es calcular de cada uno de los algoritmos

- La función de complejidad temporal: *f(n)*
- La función de complejidad espacial: *f(n)*

El programa deberá mostrar el tiempo estimado de ejecución de cada algoritmo con diferentes entradas *(n)*.

Documentar en una tabla los resultados obtenidos de cada algoritmo.

Graficar el comportamiento temporal de cada uno.

Diseño de la solución

El diseño de la solución se muestra de la siguiente manera.

En la imagen 1.1 se aprecia el diagrama de flujo del algoritmo número 1.

Con una complejidad temporal de $=10n^2-12+5.$

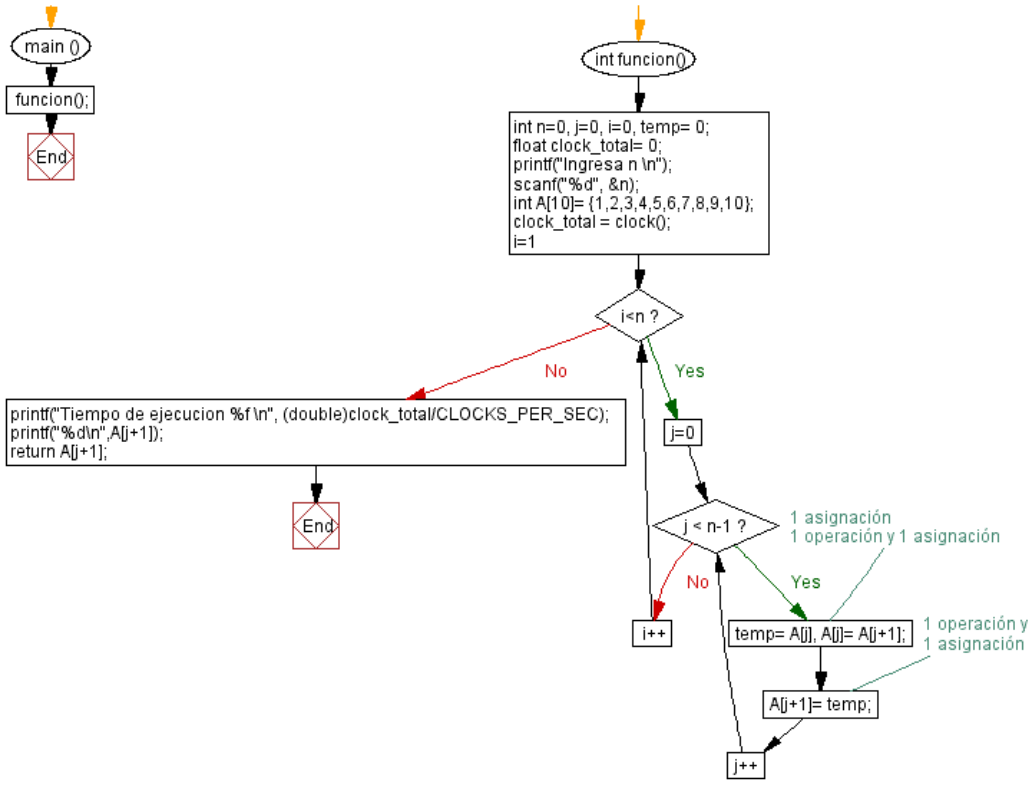


Imagen 1.1 Algoritmo 1.

```c
/*
Marcos Oswaldo Vazquez Moreno 2016601777
Practica 1 analisis de algoritmos
analisis de complejidad espacial y temporal
Profesora Luz Maria
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int funcion();

main () {

funcion();


}


int funcion(){
    int n=0, j=0, i=0, temp= 0;
```

```
23.     float  clock_total=  0;
24.     printf("Ingresa n \n");
25.     scanf("%d", &n);
26.
27.     int A[10]= {1,2,3,4,5,6,7,8,9,10};
28.     clock_total = clock();
29.     for(i=1; i<n; i++)
30.         for(j=0;j < n-1; j++)
31.         {
32.             temp= A[j],   // 1 asignación
33.             A[j]= A[j+1]; // 1 operación y 1 asignación
34.             A[j+1]= temp; // 1 operación y 1 asignación
35.         }
36.     printf("Tiempo de ejecucion %f \n", (double)clock_total/CLOCKS_PER_SEC);
37.     printf("%d\n",A[j+1]);
38.     return A[j+1];
39.
40. }
```

En la imagen 1.2 se aprecia el diagrama de flujo del algoritmo número 2.

Con una complejidad temporal de *=8n+12.*

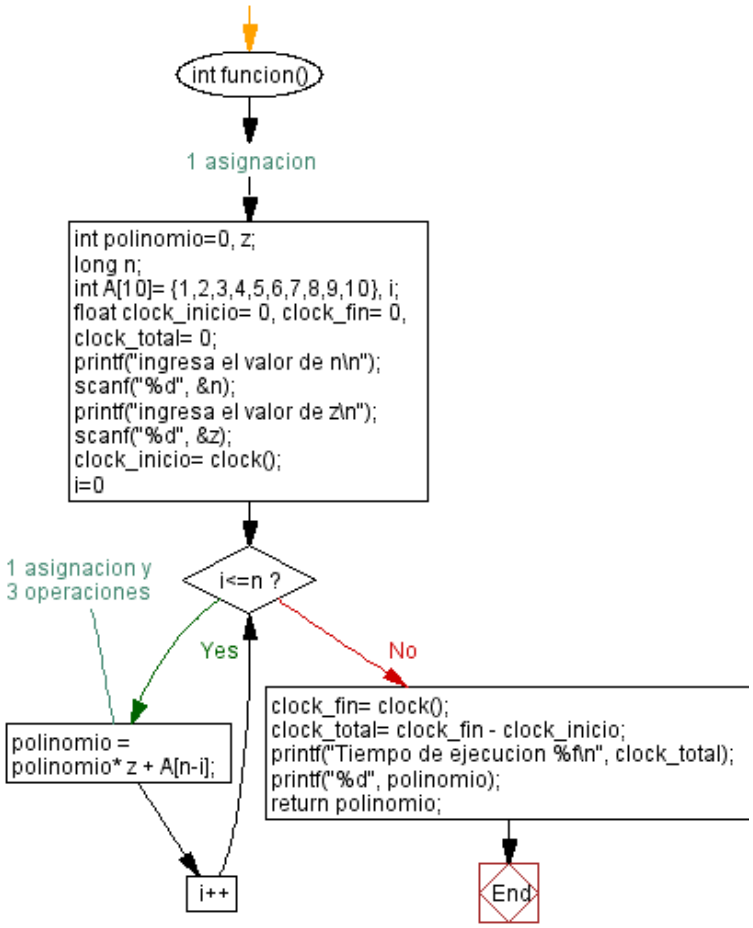


Imagen 1.2 Algoritmo 2.

```c
/*
Marcos Oswaldo Vazquez Moreno 2016601777
Practica 1 analisis de algoritmos
analisis de complejidad espacial y temporal
Profesora Luz Maria
 */

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

int funcion();

main () {

funcion();

}

int funcion(){
    int polinomio=0, z; // 1 asignacion
    long n;
    int A[10]=  {1,2,3,4,5,6,7,8,9,10}, i;
    float clock_inicio= 0,  clock_fin= 0,   clock_total=  0;
    printf("ingresa el valor de n\n");
    scanf("%d", &n);
    printf("ingresa el valor de z\n");
    scanf("%d", &z);

    clock_inicio= clock();
    for (i=0; i<=n; i++)
    {
        polinomio = polinomio* z + A[n-i]; //1 asignacion y 3 operaciones

    }clock_fin= clock();
    clock_total= clock_fin - clock_inicio;
    printf("Tiempo de ejecucion %f\n", clock_total);
    printf("%d", polinomio);
    return polinomio;
}
```

En la imagen 1.4 se aprecia el diagrama de flujo del algoritmo número 4.

Con una complejidad temporal del *=8n-12 si n >= 3*

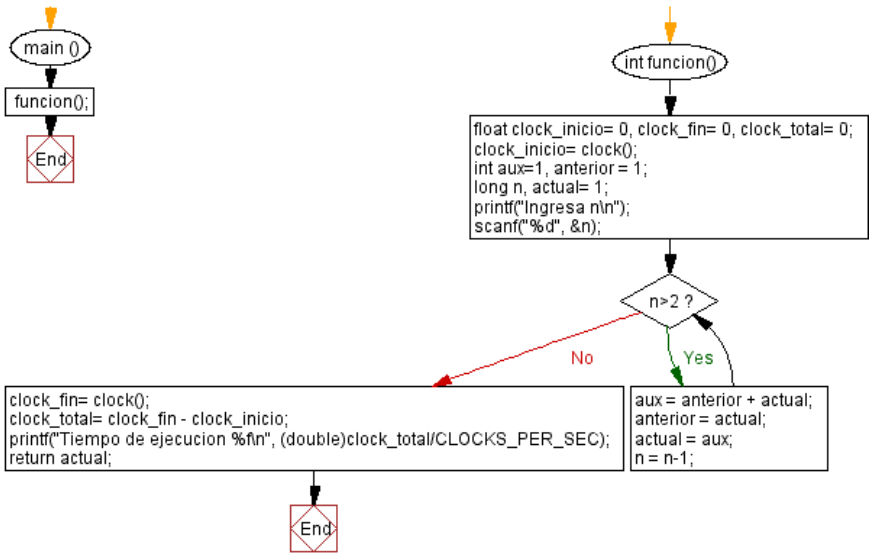


Imagen 1.4 Algoritmo 4.

```
1.  /*
2.  Marcos Oswaldo Vazquez Moreno 2016601777
3.  Practica 1 analisis de algoritmos
4.  analisis de complejidad espacial y temporal
5.  Profesora Luz Maria
6.   */
7.
8.  #include <stdio.h>
9.  #include <stdlib.h>
10. #include <time.h>
11.
12. int funcion();
13.
14. main () {
15.
16. funcion();
17.
18.
19. }
20.
21. int funcion(){
22.
23. float clock_inicio= 0,  clock_fin= 0,   clock_total=  0;
24.     clock_inicio= clock();
25.
26. int aux=1, anterior = 1;
27. long n, actual= 1;
28.
29. printf("Ingresa n\n");
30. scanf("%d", &n);
31. while (n>2)
32. {
33.     aux = anterior + actual;
34.     anterior = actual;
35.     actual = aux;
```

```
    n = n-1;
}clock_fin= clock();
    clock_total= clock_fin - clock_inicio;

    printf("Tiempo de ejecucion %f\n", (double)clock_total/CLOCKS_PER_SEC);
    return actual;


}
```

En la imagen 1.5 se aprecia el diagrama de flujo del algoritmo número 5.

Con una complejidad temporal de *=12n+8.*

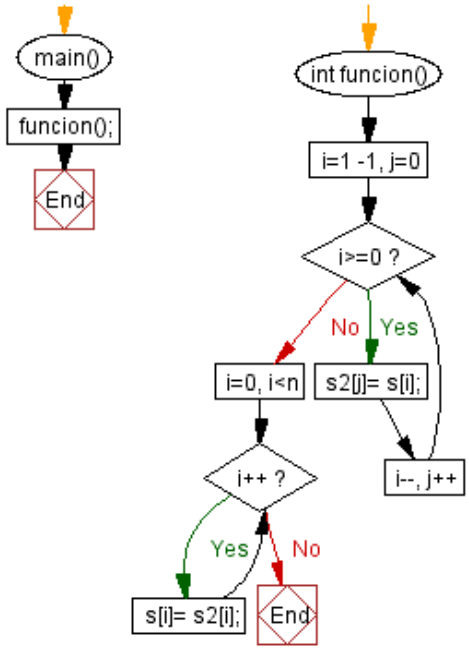


Imagen 1.5 Algoritmo 5.

```
/*
Marcos Oswaldo Vazquez Moreno 2016601777
Practica 1 analisis de algoritmos
analisis de complejidad espacial y temporal
Profesora Luz Maria
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int funcion();

main(){
    funcion();
```

```
16. }
17.
18. int funcion(){
19.
20. for (i=1 -1, j=0; i>=0; i--, j++)
21.     s2[j]= s[i];
22.
23. for (i=0, i<n; i++)
24.     s[i]= s2[i];
25.
26. }
```

Cada diagrama de flujo y diseño en ojo de ave fue llevado a cabo con ayuda del software de aplicación llamado Visustin v8.05.

Implementación de la solución

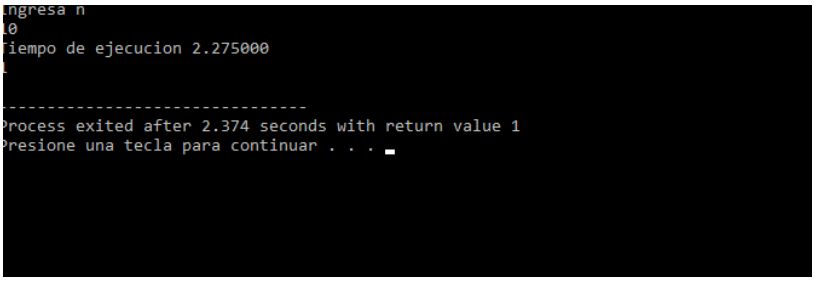
En la siguiente tabla algunos de los algoritmos se ejecutan demasiado rápido que la función que calcula el tiempo de ejecución del algoritmo es siempre 0.0000000 por lo que se ha decidido tomar el tiempo completo que toma ejecutar el programa completo y se denotará de la siguiente manera (**).

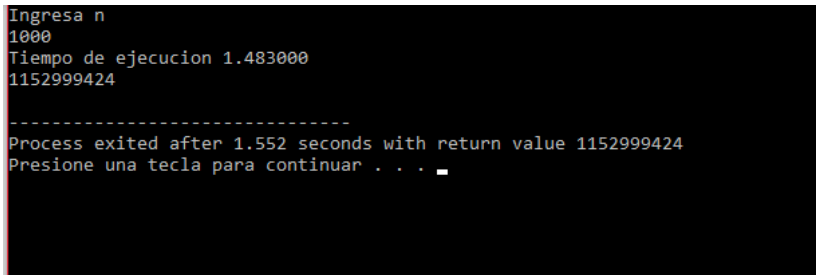
n	Alg 1	Alg 2	Alg 3	Alg 4	Alg 5
1	2.523 seconds**	1.533 seconds**	**--**	0.937 seconds	0.235 seconds
10	3.849 seconds**	2.007 seconds**	**--**	1.774 seconds	0.658 seconds
100	4.905 seconds**	2.875 seconds**	**--**	2.987 seconds	0.849 seconds
1000	6.334 seconds**	8.906 seconds**	**--**	3.807 seconds	2.245 seconds
10000	7.456 seconds**	**--**	**--**	5.0830 seconds	3.125 seconds
100000	9.118 Seconds**	**--**	**--**	7.718 seconds	3.952 seconds

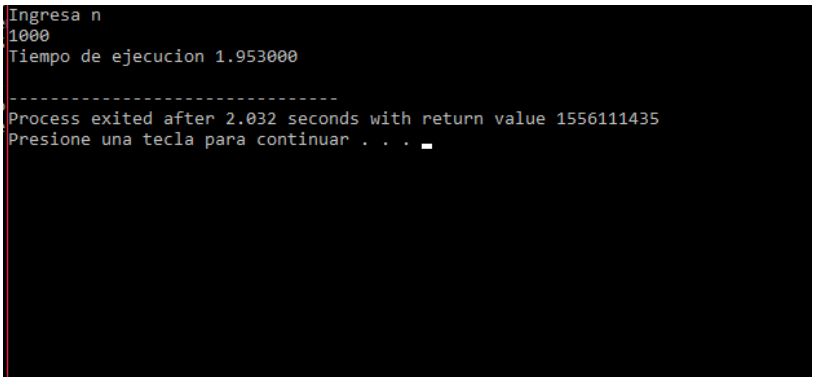
1000000	11.48 seconds**	**--**	**--**	8.346 seconds	4.498 seconds

En el algoritmo 3 surge una discrepancia ya que el algoritmo en el lenguaje de programación Python no funciona, no corre y se desborda al parecer demasiado rápido por lo que se decidió que no hay tiempos estimados y por supuesto no existe el algoritmo.

Funcionamiento







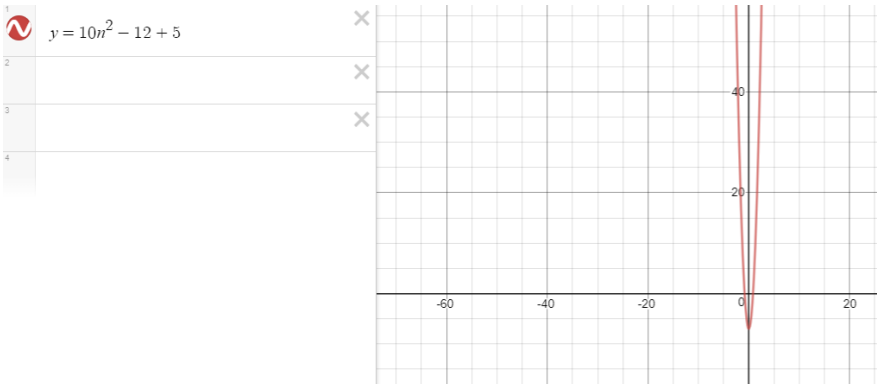
Plataforma experimental

La ejecución de los algoritmos anteriores se llevó a cabo en una computadora personal que se describe en la siguiente imagen.

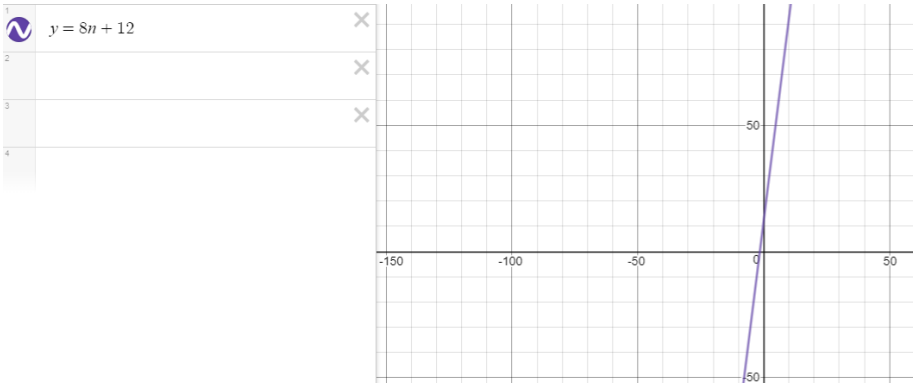
El compilador utilizado fue gcc integrado dentro del IDE DevC en un sistema operativo de 64 bits Windows 10.

Graficas de funciones

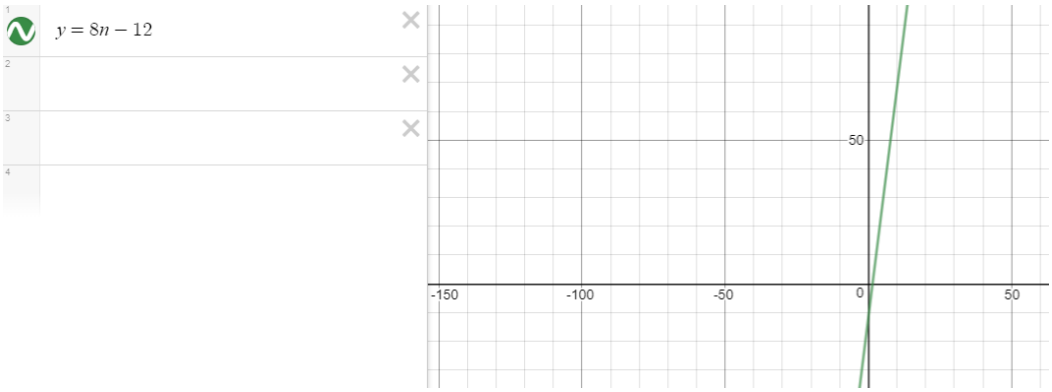
Algoritmo 1



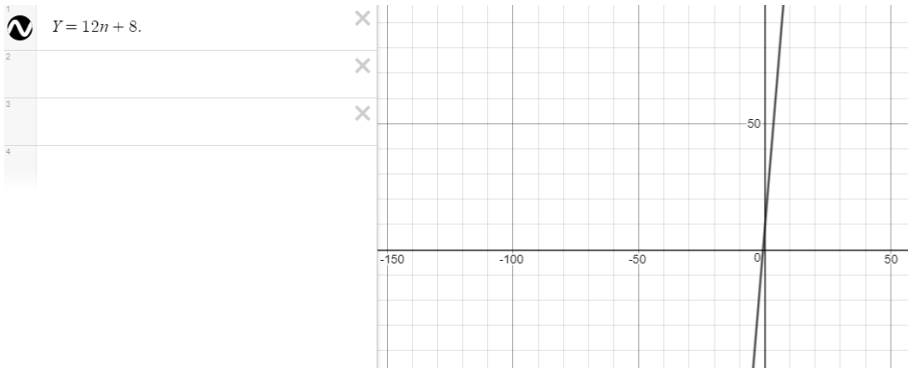
Algoritmo 2



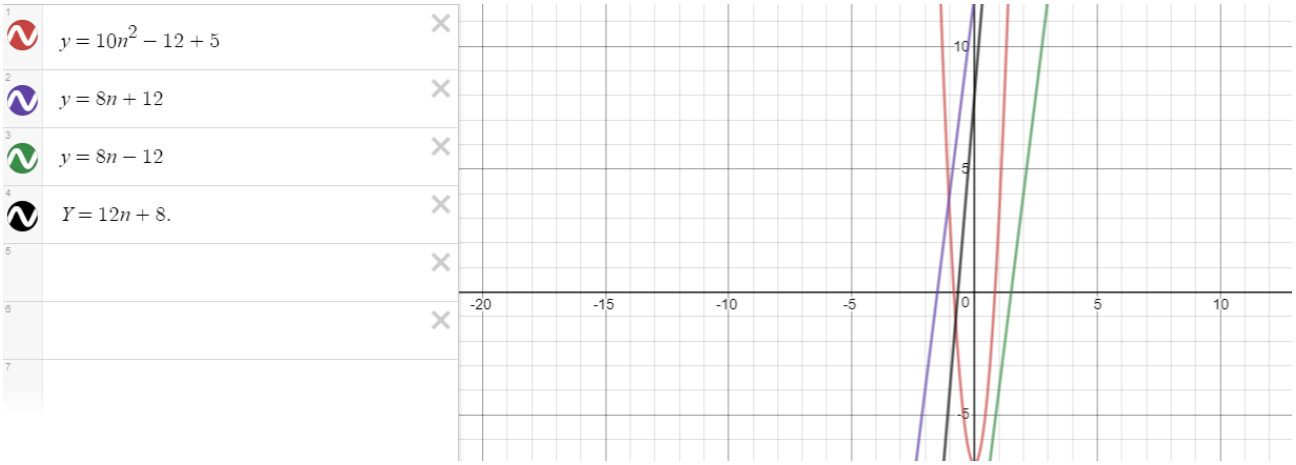
Algoritmo 4



Algoritmo 5



Haciendo la comparativa entre funciones, una es cuadrática y todas las demás lineales quedando de la siguiente manera.



Conclusiones

En esta práctica se codearon 4 algoritmos ya que se tuvo problemas con el tercer algoritmo que estaba en un lenguaje de programación Python y de esta manera no se pudieron obtener resultados, por otro lado con los cuatro algoritmos que si utilizamos nos dimos cuenta de lo curioso que puede ser cada algoritmo al programar, analizamos los tiempos de ejecución y los espacios utilizados, dicho lo anterior, se apreció que algoritmos con mayor numero de complejidad tardan más tiempo en ejecutarse y esto ya que eran funciones cuadráticas mismas que puedes apreciar en las gráficas.

Por otro lado, es importante mencionar que lo visto en clase lo llevamos a cabo en cada uno de los momentos programando y obteniendo las funciones de complejidad temporal, es de utilidad saber esto porque cuando se tienen algoritmos tan grandes que quiebran fácilmente debemos aumentar las propiedades del ordenador o en su defecto hacer más eficiente el algoritmo, implementarlo de una manera más eficaz.

Bibliografía

Bibliografía

Gimeno, J., & González, J. (s.f.). *ocw.udl.cat.* Recuperado el 2017, de http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/2-algoritmos.pdf

Martínez, E. A. (s.f.). *www.eafranco.com.* Recuperado el 2017, de http://www.eafranco.com/docencia/analisisdealgoritmos/files/06/Tema06.pdf

Visustin v8.05

Syntax Highlight Code In Word Documents *http://planetb.ca/syntax-highlight-word*

Desmos Graphic *https://www.desmos.com/calculator*