INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 2. Algoritmos Divide y Vencerás

ALUMNOS: Ramírez Jiménez Itzel Guadalupe

Colín Ramiro Joel

GRUPO: 3CM3

PROFESORA: Sánchez García Luz María

MATERIA: Análisis y Diseño de Algoritmos

I. Planteamiento del Problema

Se solicita elaborar, analizar y codificar 3 algoritmos recursivos con la estrategia de Divide y Vencerás. Los algoritmos con los que se trabajara son los siguientes:

- **Búsqueda Binaria.-** Es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado. Compara el valor con el elemento en el medio del array, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre.
- **Multiplicación de enteros largos. -** es un procedimiento para multiplicar números grandes eficientemente, que fue descubierto por Anatolii Alexeevitch Karatsuba en 1960 y publicado en 1962.
- **Máximo y mínimo. –** Este algoritmo se encarga de dado un arreglo de n números con n < 1, regresar, número mayor y menor del arreglo.

Estas pruebas experimentales, se realizaron en el IDE Dev C++

II. Actividades

Búsqueda Binaria

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

int busqueda_binaria(int arreglo[], int busqueda, int izquierda, int derecha);
int main(){
	int lim;
	int busqueda;
	int aux;
	int temp;
	int a,b;
	printf("Digite el numero de elementos en el arreglo : ");
	scanf("%i",&lim);
	int numeros[lim];
	 int g;
	for(int i = 0; i<lim; i++){
		g=i;
		aux = 1 + rand()%lim;
		for(int d = 0; d <= g; d++){
		if(aux == numeros[d]){
			g=g-g;
			i=i-1;
		}
```

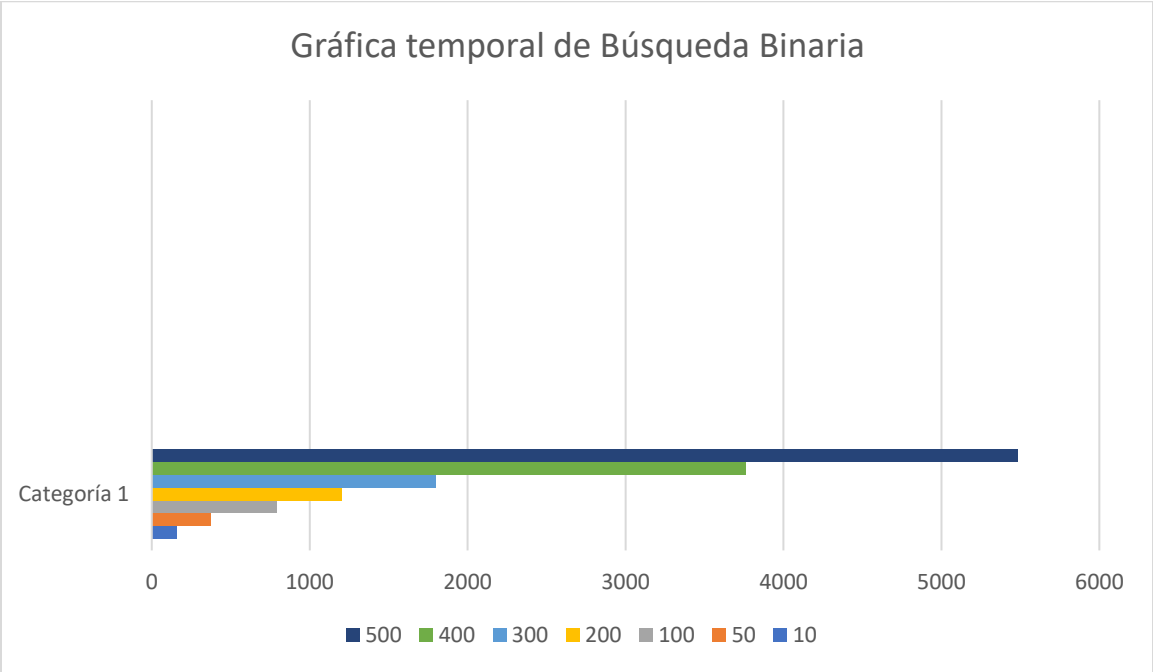
```
                while((g==i) && (aux != numeros[d]) && (d==i)){
                numeros[i] = aux;
                }
        }
  }
        printf("El arreglo es: [ ");
        for(int i = 0; i<lim; i++){
                printf(" %i ",numeros[i]);
        }
        printf(" ]");
        for(a = 0; a <lim; a++){
                for(b = 0; b <lim; b++){
                        if(numeros[b]> numeros[b+1]){
                                temp = numeros[b];
                                numeros[b] = numeros [b+1];
                                numeros[b+1] = temp;
                        }
                }
        }
        printf("\nEl arreglo ordenado es:  [ ");    //imprimimos el arreglo
        for(int o = 0; o<lim; o++){
                printf(" %i ",numeros[o]);
        }
        printf(" ]");
        int longitudDelArreglo = sizeof(numeros) / sizeof(numeros[0]);
        printf("\nDigite el numero a buscar: \n");
        scanf("%i", &busqueda);
        numeros[0] = busqueda;
        int resultado = busqueda_binaria(numeros, busqueda, 0, longitudDelArreglo);
        printf("\n\nEl numero %i , esta en la posicion:  %d\n", busqueda, resultado+1);
}
int busqueda_binaria(int A[],int X, int i, int j){
        int medio;
        if (i>j) return 0;
        medio = (i+j) / 2;
        if (A[medio] < X) {
        return busqueda_binaria(A,X,medio+1,j);
}
 else {
        if (A[medio] > X) {
                return busqueda_binaria(A,X,i,medio-1);
        }
        else {
                return medio;
        }
    }
}
```

- **Análisis temporal**

$$T(n) = T(n-1) + 2n$$
$$O(\log n)$$

k	Instrucciones
10	T(9)+20
50	T(49)+100
100	T(99)+200
200	T(199)+400
300	T(299)+600
400	T(399)+800
500	T(499)+1000

Gráfica



- Análisis espacial

k	Celdas

Gráfica espacial
Entorno

Multiplicación de enteros largos (Karatsuba)

```
#include <stdio.h>
#include <math.h>



int Power(int x, int y);
int Digitos (int n, int &dig);
int ultimos(int digitos, int &numero);
int primeros(int digitos, int &numero);
int karatsuba (int &u, int &v);
int main () {
   int numero=0;
   int num1;
   int num2;
   printf("Digite el primer numero: ");
   scanf("%d", num1);
   printf("Digite el segundo numero: ");
   scanf("%d", num2);
   printf("\n\nEl resultado del producto es: ",karatsuba(num1, num2));
          return 0;
  }
  int Power(int x, int y) {
          if (y == 0)
          return (1);
  }
   else if (y == 1)
      return(x);
   else
      return(x * Power(x, y-1));
}
int Digitos (int n, int &dig) {
```

```
   if (n < 10) return (dig+1);
   else {
      dig++;
      return(Digitos(n/10, dig));
   }
}
 int ultimos(int digitos, int &numero) {
   return numero % Power(10, digitos);
}
 int primeros(int digitos, int &numero) {
   return numero/Power(10, digitos);
}
 int karatsuba (int &no1, int &no2) {
          int dig1=0, dig2=0;
          int w, x, y, z, p, q, wMasx, zMasy, r, noDigitos;
          noDigitos = max(Digitos(no1, dig1), Digitos(no2, dig2));
          if (noDigitos <= 1) {
                  return no1*no2;
          }
          noDigitos = (noDigitos / 2) + (noDigitos % 2);

    w = primeros(noDigitos, no1);
    x = ultimos(noDigitos, no1);
    y = primeros(noDigitos, no2);
    z = ultimos(noDigitos, no2);
    p=karatsuba(w, y);
    q=karatsuba(x, z);
    wMasx = w + x;
    zMasy = z + y;
    r= karatsuba(wMasx, zMasy);
   return Power(10, 2*noDigitos)*p+Power(10, noDigitos)*(r-p-q)+q;
}
```

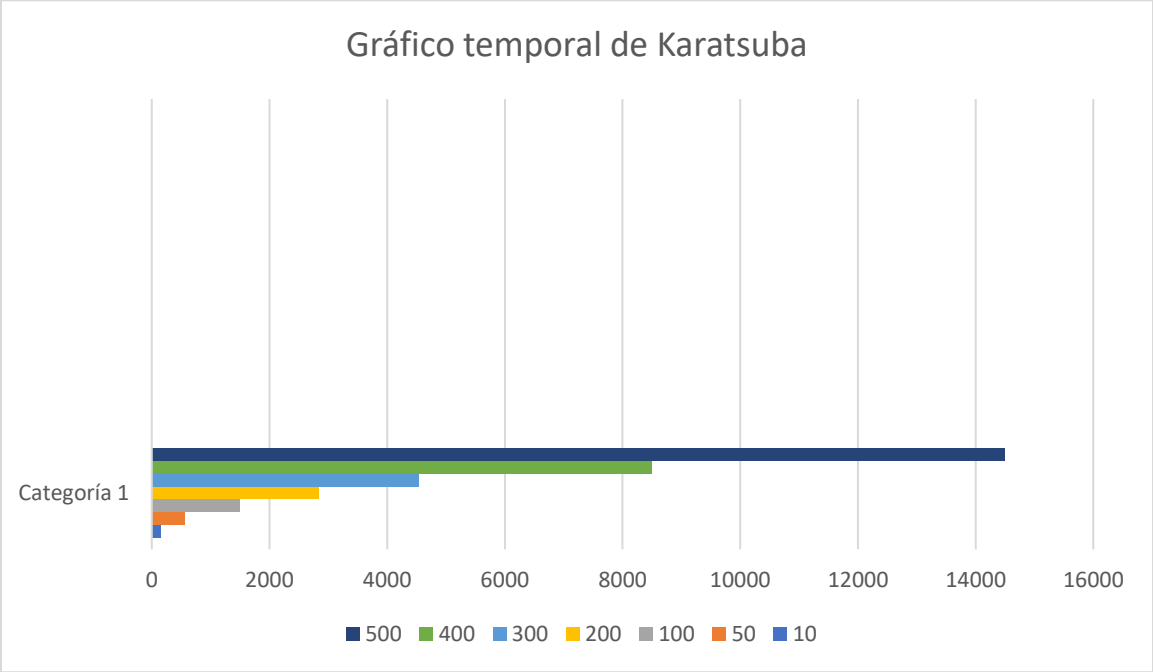
- **Análisis temporal**

$$\boldsymbol{T(n) = T(n^2 + 5n + 4) + 5n}$$
$$\boldsymbol{O(n^2)}$$

k	Instrucciones
10	$\boldsymbol{T(154) + 50}$
50	$\boldsymbol{T(2754) + 250}$
100	$\boldsymbol{T(10504) + 250}$
200	$\boldsymbol{T(41004) + 250}$
300	$\boldsymbol{T(91504) + 250}$

400	$\boldsymbol{T(162004) + 250}$
500	$\boldsymbol{T(252504) + 250}$

Gráfica



- Análisis espacial

k	Celdas

Gráfica espacial
Entorno

Máximo y mínimo

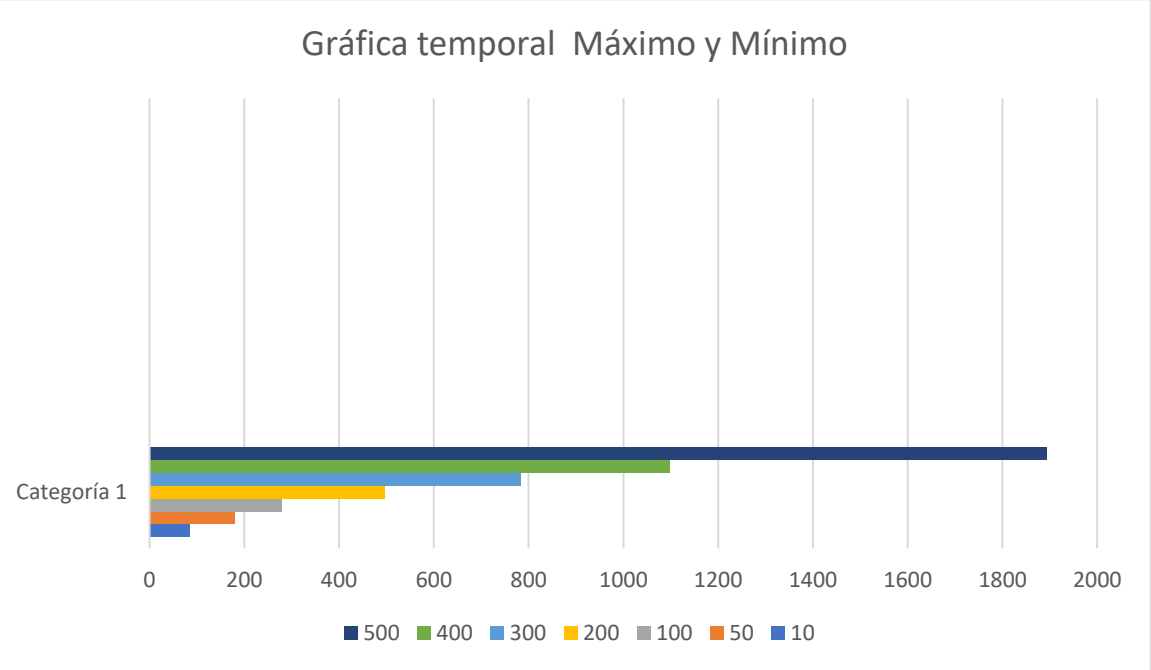
```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
int maximo (int [], int, int);
int minimo (int [], int, int);
int main() {
	int lim;
	printf("Digite el numero de elementos del arreglo : ");
	srand (time(NULL));
	scanf("%i",&lim);
	int num[lim];
	for(int i = 0; i<lim; i++){
	num[i] =  1 + rand()%lim;

}
	printf("El arreglo es: \n [ ");    //imprimimos el arreglo
	for(int i = 0; i<lim; i++){
	printf(" %i ",num[i]);
}
	printf(" ]");
	printf("\nEl numero maximo del arreglo es: ", maximo(num, lim-1, num[0]));
	printf("\nEl numero minimo del arreglo es: ", minimo(num, lim-1, num[0]));
}
int maximo(int num[], int lim, int max){
		if(lim==0){
		return max;
		}
		else {
			if(num[lim]>max){
			max=num[lim];
			}
			return maximo(num, lim-1, max);
		}
}
int minimo(int num[], int lim, int min){
		if(lim==0){
			return min;
		}
		else{
			if(num[lim]<min){
			min=num[lim];
		}
			return minimo(num, lim-1, min);
	}
}
```

- **Análisis temporal**

$$T(n) = 4T(\frac{n}{2}) + 4$$
$$0(n)$$

k	Instrucciones
10	$4T(5) + 4$
50	$4T(25) + 4$
100	$4T(50) + 4$
200	$4T(100) + 4$
300	$4T(150) + 4$
400	$4T(200) + 4$
500	$4T(250) + 4$

Gráfica temporal



- Análisis espacial

k	Celdas

10	
50	
100	
200	
300	
400	
500	

Gráfica espacial

a) Para cada algoritmo ¿Existe alguna solución que no sea recursiva? ¿Cuál es?

Entre los algoritmos que realizamos, concluimos que únicamente para el de máximo y mínimo existe una solución iterativa.

b) ¿Cuál de los 3 algoritmos es más fácil de implementar?

El más sencillo de implementar fue el de Máximo y mínimo

c) ¿Cuál de los 2 algoritmos es el más difícil de implementar?

El más complejo fue el de la multiplicación de enteros largos (Karatsuba)

d) ¿El comportamiento experimental de los algoritmos era el esperado? ¿Por qué?

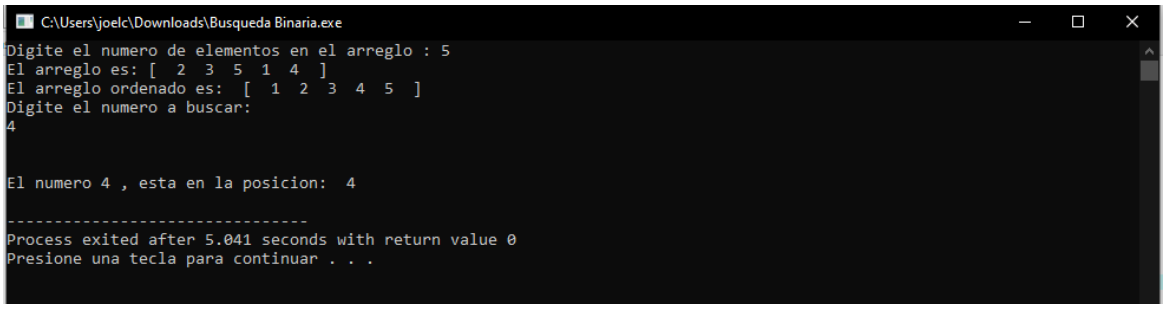
No realmente, se nos complicó un poco a la hora de implementarlos de forma recursiva, y sin elementos iterativos, finalmente, logramos que funcionaran.

e) ¿Qué recomendaciones darían a nuevos equipos para realizar esta práctica?
Comprender a fondo el tema de algoritmos divide y venceras y primordialmente el comprender la recursividad.

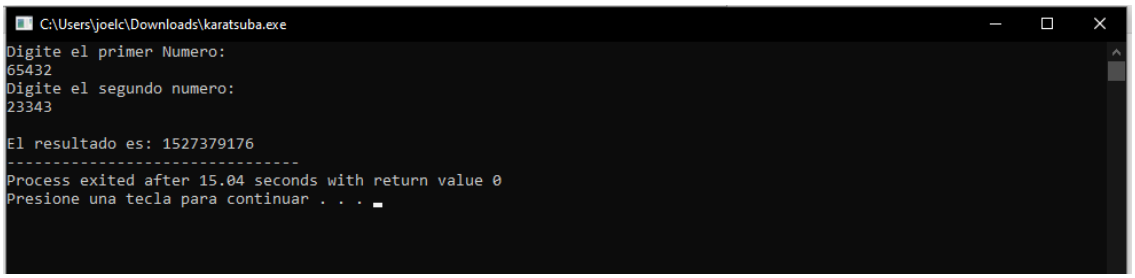
III. Pruebas

A continuación, se presentarán los pantallazos de los resultados de los códigos implementados.

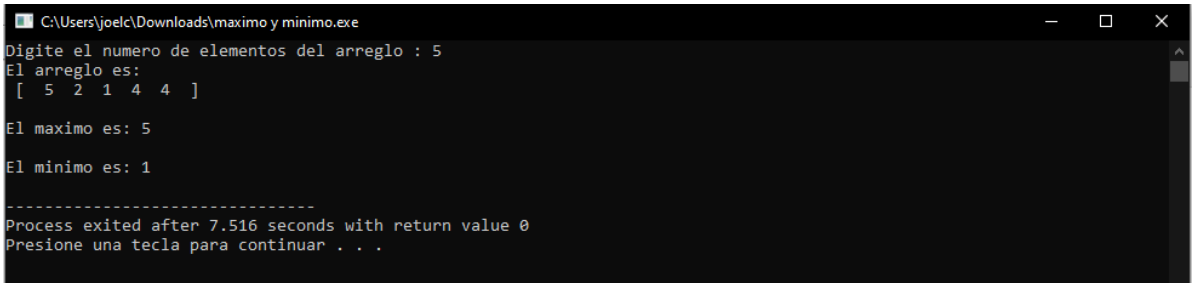
- Búsqueda binaria



- Multiplicación de enteros largos (Karatsuba)



- Máximo y mínimo de un arreglo



IV. Bibliografía

1. https://www.3ciencias.com/wp-content/uploads/2018/03/Art_2.pdf
2. https://webdiis.unizar.es/asignaturas/AB/?p=1479
3. https://programacion1z.wordpress.com/2020/01/25/mayor-y-menor-elemento-de-un-vector-recursividad/
4. http://www.forosdelweb.com/f96/minimo-vector-recursividad-986050/
5. https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/implementing-binary-search-of-an-array
6. https://www.fing.edu.uy/tecnoinf/mvd/cursos/eda/material/teo/EDA-teorico6.pdf