



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: **Manuel Enrique Castañeda Castañeda**

Asignatura:

Estructura de Datos y Algoritmos I

Grupo:

12

No de Práctica(s):

11 .

Integrante(s):

- **Lemus Ambrosio Aline Andrea**
- **Reyes Fuentes José Manuel**
- **Sánchez Alvirde Andrés Iván**

*No. de Equipo de cómputo
empleado:*

-

No. de Lista o Brigada:

12

Semestre:

2021-2

Fecha de entrega:

6 de agosto de 2021

Observaciones:

Calificación:

Estructura de Datos y Algoritmos I

Práctica 11

INTRODUCCIÓN

En esta práctica podremos ver algunos conceptos o estrategias para la elaboración de algoritmos, conceptos los cuales son: fuerza bruta, algoritmo ávido, bottom-up, top-down, divide y vencerás.

Fuerza bruta:

No requiere ninguna fase de preproceso previo, ni un espacio extra constante además del espacio asignado al patrón y al texto.

Para la búsqueda:

- Consiste en la comparación de todas las posiciones del texto entre 0 y el $n-m$, si una ocurrencia del patrón corresponde o no.
- Si encuentra una no ocurrencia, o una ocurrencia total del patrón, salta un carácter hacia la derecha.

Características:

- Es el algoritmo más simple posible.
- Consiste en probar todas las posibles posiciones del patrón en el texto.
- Requiere espacio constante.
- Realiza siempre saltos de un carácter.
- Compara de izquierda a derecha.
- Realiza la búsqueda del patrón en un tiempo $O(mn)$.
- Realiza $2n$ comparaciones previstas de los caracteres del texto.

Algoritmo ávido:

Los algoritmos voraces, ávidos o de avance rápido (en inglés greedy) se utilizan normalmente en problemas de optimización.

–El problema se interpreta como: “tomar algunos elementos de entre un conjunto de candidatos”.

– El orden el que se cogen puede ser importante o no.

Un algoritmo voraz funciona por pasos:

– Inicialmente partimos de una solución vacía.

– En cada paso se escoge el siguiente elemento para añadir a la solución, entre los candidatos.

– Una vez tomada esta decisión no se podrá deshacer.

– El algoritmo acabará cuando el conjunto de elementos seleccionados constituya una solución.

top-down:

El enfoque top-down enfatiza la planificación y conocimiento completo del sistema. Se entiende que la codificación no puede comenzar hasta que no se haya alcanzado un nivel de detalle suficiente, al menos en alguna parte del sistema. Esto retrasa las pruebas de las unidades funcionales del sistema hasta que gran parte

del diseño se ha completado

Bottom-up:

Bottom-up hace énfasis en la programación y pruebas tempranas, que pueden comenzar tan pronto se ha especificado el primer módulo. Este enfoque tiene el riesgo de programar cosas sin saber cómo se van a conectar al resto del sistema, y esta conexión puede no ser tan fácil como se creyó al comienzo. El reuso del código es uno de los mayores beneficios del enfoque bottom-up.

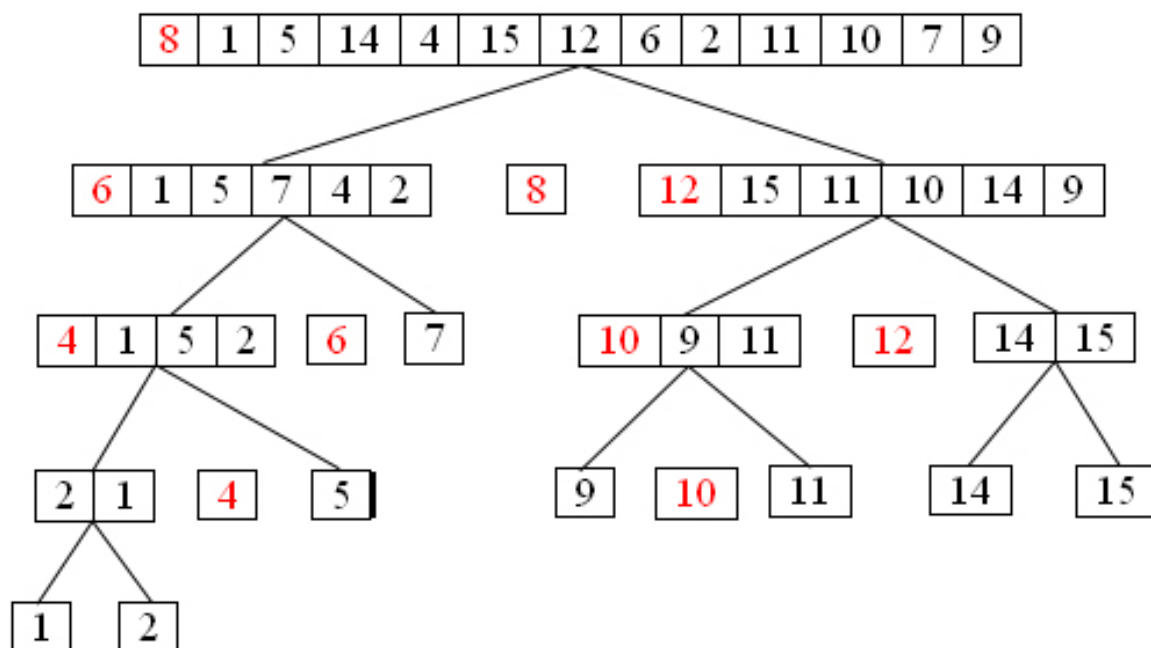
Divide y vencerás:

Es una estrategia que consiste en:

- Dividir el problema en subproblemas hasta que son suficientemente simples que se pueden resolver directamente.
- Después las soluciones son combinadas para generar la solución general del problema.

Quick sort:

Quicksort es un algoritmo de ordenación considerado entre los más rápidos y eficientes. El algoritmo usa la técnica divide y vencerás que básicamente se basa en dividir un problema en subproblemas y luego juntar las respuestas de estos subproblemas para obtener la solución al problema central.

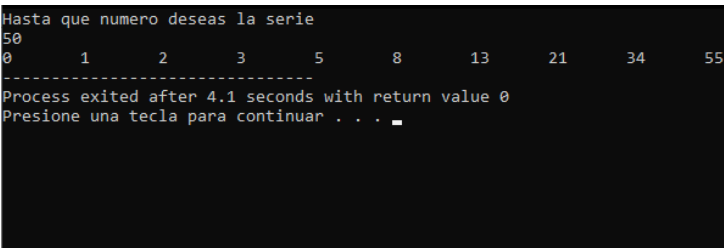


Juntando los elementos, el arreglo quedaría ordenado

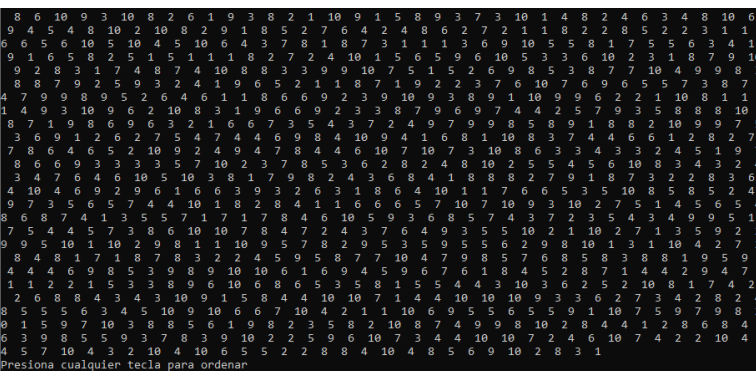
[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15]

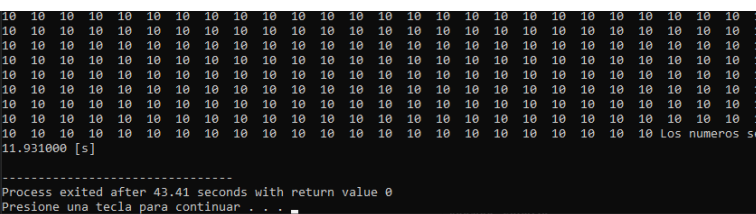
DESARROLLO

1. Implementar la serie de Fibonacci hasta n dado por el usuario, usando top down

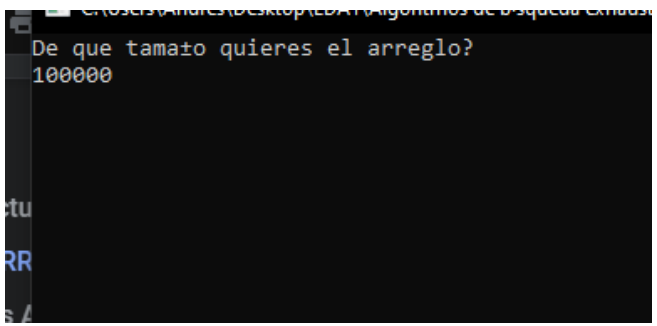
CÓDIGO FUENTE	EJECUCIÓN
<pre>#include<stdio.h> #include<stdlib.h> int fibonacci(int n); int pedirNumero(); int main(){ fibonacci(pedirNumero()); return 0; } int fibonacci(int n){ int n1=0,n2=1,n3; do{ n3=n1+n2; printf("%d\t",n3); n1=n2; n2=n3; }while(n2<n); } int pedirNumero(){ int x; printf("Hasta que numero deseas la serie\n"); scanf("%d",&x); printf("0\t"); return x; }</pre>	 <p>Hasta que numero deseas la serie 50 0 1 2 3 5 8 13 21 34 55 ----- Process exited after 4.1 seconds with return value 0 Presione una tecla para continuar . . .</p>

2. Hacer un arreglo de tamaño grande (más de 100 mil) y ordenar con el método de la burbuja

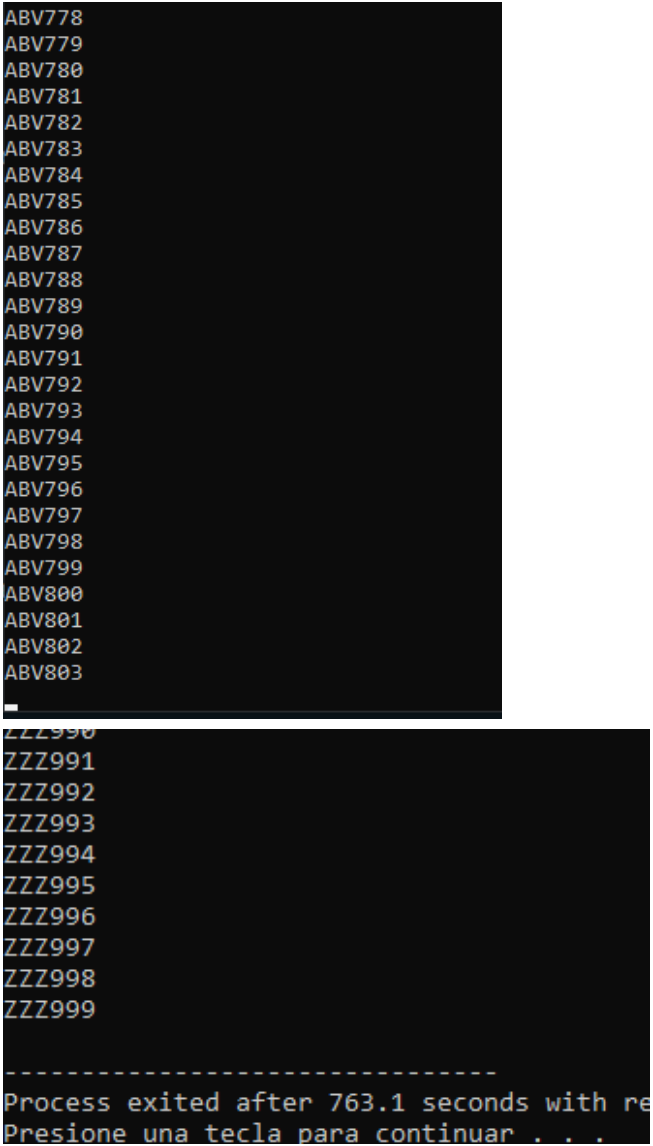
CÓDIGO FUENTE	EJECUCIÓN
<pre>#include <stdio.h> #include <stdlib.h> #include <time.h> int main(){ int n=100000; float t0,t1; char x; int arre[n]; srand(time(NULL)); for(int i=0;i<n;i++){ arre[i]=rand()%10+1; printf(" %d ",arre[i]); }</pre>	 <p>8 6 10 9 3 10 8 2 6 1 9 3 8 2 1 10 9 1 5 8 9 3 7 3 10 1 4 8 2 4 6 3 4 8 10 6 9 4 5 4 8 10 2 10 8 2 9 1 8 5 2 7 6 4 2 4 8 6 2 7 2 1 1 8 2 2 8 5 2 2 3 1 1 6 6 5 6 10 5 10 4 5 10 6 4 3 7 8 1 8 7 3 1 1 3 6 9 10 5 5 8 1 7 5 5 6 3 4 1 9 1 6 5 8 2 5 1 5 1 1 1 8 2 7 2 4 10 1 5 6 5 9 6 10 5 3 3 6 10 2 3 1 8 7 9 10 9 2 8 3 1 7 4 8 7 4 10 8 8 3 3 9 9 10 7 5 1 5 2 6 9 8 5 3 8 7 7 10 4 9 9 8 3 8 8 7 9 2 5 9 3 2 4 1 9 6 5 2 1 1 8 7 1 9 2 2 3 7 6 10 7 6 9 6 5 5 7 3 8 7 4 7 9 9 8 9 5 2 6 4 6 1 1 8 6 6 9 2 3 9 10 9 3 8 9 1 10 9 9 6 2 2 1 10 8 1 1 1 4 9 3 10 9 6 2 10 8 3 1 9 6 6 9 2 3 3 8 7 9 9 6 9 7 4 4 2 5 7 9 3 5 8 8 10 8 7 1 9 8 6 9 6 3 2 1 6 6 7 3 5 4 3 7 2 4 9 7 9 9 8 5 8 9 1 8 8 2 10 9 9 7 5 3 6 9 1 2 6 2 7 5 4 7 4 4 6 9 8 4 10 9 4 1 6 8 1 10 8 3 7 4 4 6 6 1 2 8 2 7 7 8 6 4 6 5 2 10 9 2 4 9 4 7 8 4 4 6 10 7 10 7 3 10 8 6 3 3 4 3 2 4 5 1 9 7 8 6 6 9 3 3 3 3 5 7 10 2 3 7 8 5 3 6 2 8 2 4 8 10 2 5 4 5 6 10 8 3 4 3 2 4 3 4 7 6 4 6 10 5 10 3 8 1 7 9 8 2 4 3 6 8 4 1 8 8 8 2 7 9 1 8 7 3 2 2 8 3 6 4 10 4 6 9 2 9 6 1 6 6 3 9 3 2 6 3 1 8 6 4 10 1 1 7 6 6 5 3 5 10 8 5 8 5 2 4 9 7 3 5 6 5 7 4 4 10 1 8 2 8 4 1 1 6 6 6 5 7 10 7 10 9 3 10 2 7 5 1 4 5 6 5 4 8 6 8 7 4 1 3 5 5 7 1 7 1 7 8 4 6 10 5 9 3 6 8 5 7 4 3 7 2 3 5 4 3 4 9 9 5 1 7 5 4 4 5 7 3 8 6 10 7 8 4 7 2 4 3 7 6 4 9 3 5 5 10 2 1 10 2 7 1 3 5 9 2 2 9 9 5 10 1 10 2 9 8 1 1 10 9 5 7 8 2 9 5 3 5 9 5 5 6 2 9 8 10 1 3 1 10 4 2 7 2 8 4 8 1 7 1 8 7 8 3 2 2 4 5 9 5 8 7 7 10 4 7 9 8 5 7 6 8 5 8 3 8 8 1 9 5 9 4 4 4 6 9 8 5 3 9 8 9 10 10 6 1 6 9 4 5 9 6 7 6 1 8 4 5 2 8 7 1 4 4 2 9 4 7 1 1 2 2 1 5 3 3 8 9 6 10 6 8 6 5 3 5 8 1 5 5 4 4 3 10 3 6 2 5 2 10 8 1 7 4 2 2 6 8 8 4 3 4 3 10 9 1 5 8 4 4 10 10 7 1 4 4 10 10 10 9 3 3 6 2 7 3 4 2 8 2 4 8 5 5 5 6 3 4 5 10 9 10 6 6 7 10 4 2 1 1 10 6 9 5 5 6 5 5 9 1 10 7 5 9 7 9 8 4 0 1 5 9 7 10 3 8 8 5 6 1 9 8 2 3 5 8 2 10 8 7 4 9 9 8 10 2 8 4 4 1 2 8 6 8 4 6 3 9 8 5 5 9 3 7 8 3 9 10 2 2 5 9 6 10 7 3 4 4 10 10 7 2 4 6 10 7 4 2 2 10 4 4 5 7 10 4 3 2 10 4 10 6 5 5 2 2 8 8 4 10 4 8 5 6 9 10 2 8 3 1 Presione cualquier tecla para ordenar</p>

<pre> } printf("\n"); printf("Presiona cualquier tecla para ordenar\n"); x=getchar(); t0=clock(); for(int i;i<n-1;i++){ for(int j=i+1;j<n;j++){ if(arre[i]>arre[j]){ int temp=arre[i]; arre[i]=arre[j]; arre[j]=temp; } } } t1=clock(); printf("Presione cualquier tecla para mostrar los datos ordenados\n"); x=getchar(); for(int i=0;i<n;i++){ printf(" %d ",arre[i]); } printf("Los numeros se ordenaron en: %f [s]\n",(t1-t0)/CLOCKS_PER_SEC); } </pre>	
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------

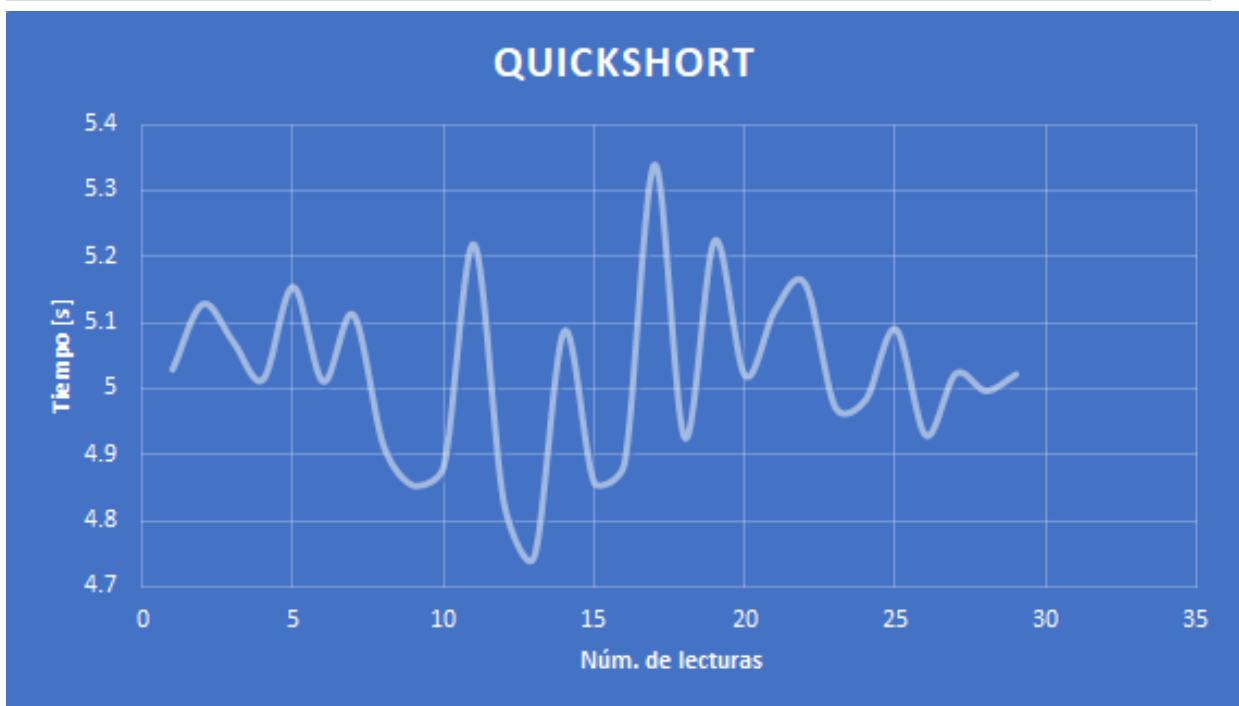
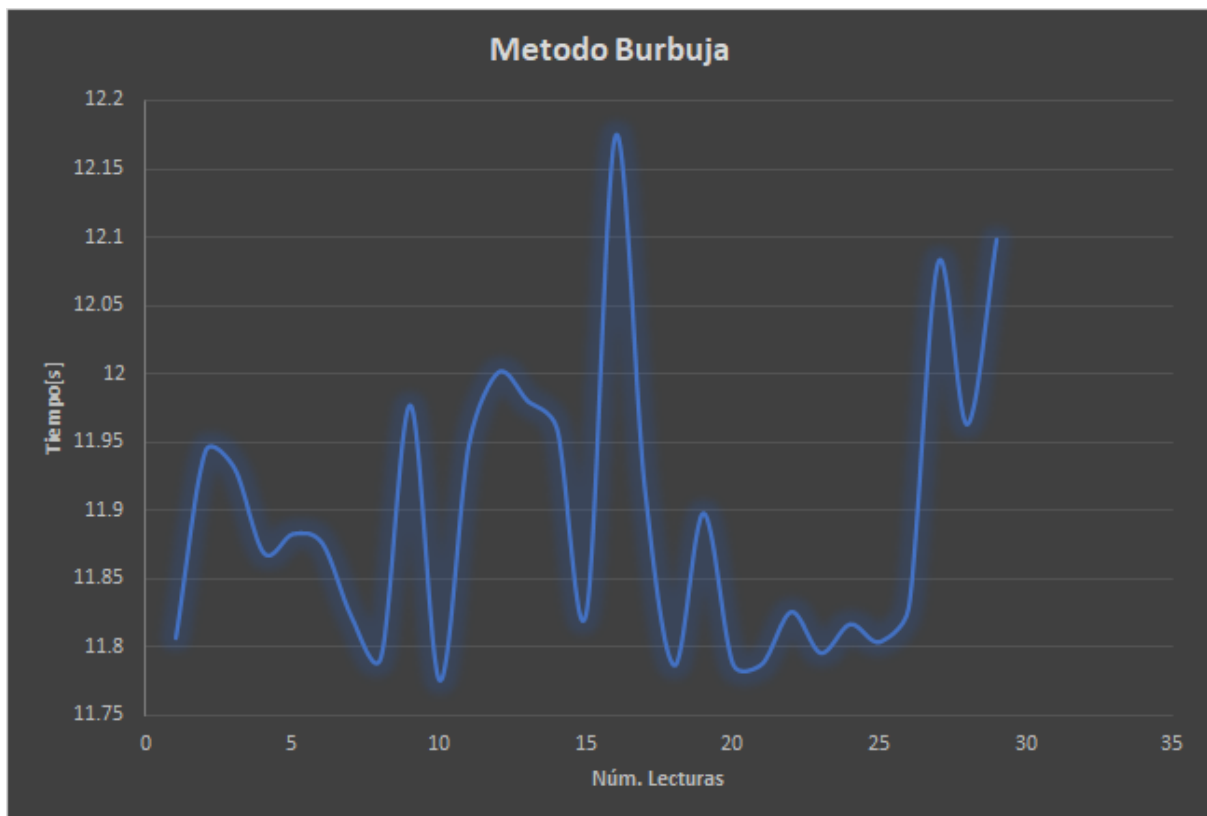
3. Ordenar un arreglo de tamaño grande (más de 100 mil) con el método de Quicksort

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include <stdio.h> #include <stdlib.h> #include <time.h> void qs(int arr[],int i,int j); int main(){ int n,i,j; float t1,t0; char x; printf("De que tamaño quieres el arreglo?\n"); scanf("%i",&n); int *arr; arr=(int *)malloc(n*sizeof(int)); srand(time(NULL)); for (i=0;i<n;i++){ arr[i]=rand()%10+1; printf(" %i",arr[i]); } qs(arr,0,i); printf("\n"); for (i=0;i<n;i++){ printf(" %i",arr[i]); } } </pre>	

4. Obtener las posibles combinaciones de placas vehiculares en el formato (letraletraletra-numnum), usando fuerza bruta

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include<stdio.h> #include<stdlib.h> struct placa{ char l1,l2,l3; int p; }; int main(){ placa *mamalon; mamalon=(placa*)malloc(17576000*sizeof(placa)); int i,j,k,l,m=-1; for(i='A';i<='Z';i++){ for(j='A';j<='Z';j++){ for(k='A';k<='Z';k++){ for(l=0;l<=999;l++){ m++; mamalon[m].l1=i; mamalon[m].l2=j; mamalon[m].l3=k; mamalon[m].p=l; printf("%c%c%c%.3d\n",mamalon[m].l1,mamalon[m].l2,mamalon[m].l3,mamalon[m].p); } } } } }</pre>	 <p>The screenshot shows the output of the program, which generates all possible license plate combinations in the format (letraletraletra-numnum). The output starts with ABV778 and continues sequentially through ABV779, ABV780, ABV781, ABV782, ABV783, ABV784, ABV785, ABV786, ABV787, ABV788, ABV789, ABV790, ABV791, ABV792, ABV793, ABV794, ABV795, ABV796, ABV797, ABV798, ABV799, ABV800, ABV801, ABV802, ABV803, and continues through ZZZ990, ZZZ991, ZZZ992, ZZZ993, ZZZ994, ZZZ995, ZZZ996, ZZZ997, ZZZ998, and ZZZ999. The output is displayed in a terminal window with a black background and white text. At the bottom of the screenshot, a message indicates that the process exited after 763.1 seconds with the return code 0, and prompts the user to press a key to continue.</p>

5. Elaborar una grafica comparativa del comportamiento del método de burbuja y el Quicksort, por lo menos 30 lectura de tiempo



OBSERVACIONES (Individuales)

- **Lemus Ambrosio Aline Andrea**

En esta práctica pude observar que me confundía un poco en los for, ya que en ocasiones se me hacía muy repetitivo o intentaba hacerlo rápido y así me terminaba equivocando en algún dato, algo que me ayudo mucho para la resolución de los ejercicios fueron los videos de clase, ya que en la clase hicimos ejercicios similares, si tenía alguna duda más veía otros videos o le preguntaba a mis compañeros, ya que siempre me ayudan a resolver dudas.

- **Reyes Fuentes José Manuel**

Esta práctica fue un poco fácil de entender ya que durante clase trabajamos con ejercicios como estos, me apoye de los videos de clase y uno que otro externo, así pude terminar de resolver dudas y entender mejor el uso del método de burbuja y fuerza bruta, en general la práctica es sencilla, solo hay que poner atención en los detalles para no cometer algún error, ya que es fácil confundirse.

- **Sánchez Alvirde Andrés Iván**

La práctica de hoy me agrado porque pude guiarme mucho con los videos de la clase para realizar los ejercicios que nos dejaron, ya que en clase habíamos realizado ejercicios similares, si me llegué a confundir en algunos detalles como punto y coma o dentro de los for, pero al volverlo a revisar encontraba el error y si no lo hacía, pedía ayuda a mis compañeros, por lo que pude acabar bien mis códigos.

CONCLUSIONES (Individuales)

- **Lemus Ambrosio Aline Andrea**

Esta práctica me ayudó a comprender que debo de realizar mis ejercicios con calma para no tener tantos errores, también me permite comprender mejor el funcionamiento del método de burbuja, quicksort, top down, etc., ya que estos me permiten agilizar mis códigos y en el caso del método de burbuja y quicksort comprender que ambos se pueden comparar por el tiempo de lecturas.

- **Reyes Fuentes José Manuel**

Esta práctica nos ayuda mucho a comprender, reforzar y aprender los conceptos de fuerza bruta, burbuja, quicksort, etc., porque estos nos permiten en ocasiones agilizar nuestros códigos y hacer que sean más funcionales, además de que podemos comprender la diferencia que hay de tiempo en códigos hechos con el método de burbuja y con quicksort.

- **Sánchez Alvirde Andrés Iván**

Esta práctica me ayuda a reforzar los conocimientos que adquirí en clase, además de permitirme aprender de los errores que se pueden cometer al realizar los códigos, con la práctica pude comparar el comportamiento del método de burbuja y el quicksort, además de manejar los conceptos de fuerza bruta, quicksort, top down, etc.