
TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN – A DISTANCIA

Trabajos Integradores – Programación I

Datos Generales

Título del trabajo: Algoritmos de Búsqueda y Ordenamiento en Python:
Implementación y Análisis de Eficiencia

Alumnos: Diego Raúl Montes y Ramiro Morales

Materia: Programación I

Profesor/a: Julieta Trape

Fecha de Entrega: 09/06/2025

Índice

Introducción

Marco Teórico

Caso Práctico

Metodología Utilizada

Resultados Obtenidos

Conclusiones

Bibliografía

Anexos

1. Introducción

Los algoritmos de búsqueda y ordenamiento son fundamentales en programación porque permiten organizar y recuperar datos eficientemente. Este trabajo se centra en:

Preguntas para los estudiantes:

¿Por qué se eligió este tema?

Elegimos este tema, ya que estos algoritmos son fundamentales y porque representa una de las bases fundamentales en la programación. algoritmos de entender pero con un enorme valor práctico, y permiten aplicar conceptos clave como estructuras de datos, ciclos, condicionales, y manipulación de listas. Este trabajo se centra en:

Relevancia:

Son la base para optimizar aplicaciones reales (bases de datos, motores de búsqueda).

Objetivos:

1. Implementar y comparar 4 algoritmos de ordenamiento y 2 de búsqueda en Python.
 2. Medir su eficiencia en listas de diferentes tamaños.
-

2. Marco Teórico

Bubble Sort: Compara elementos adyacentes o el que tiene al lado e intercambia al orden correcto. ideal para listas pequeñas

Insertion Sort: Construye una lista ordenada insertando uno por uno los elementos en su posición correspondiente, Ideal para listas pequeñas y/o casi ordenadas.

Selection Sort: Encuentra el elemento más pequeño y lo coloca en la primera posición del índice, Así sucesivamente con cada elemento en la posición que sigue

QuickSort = Selecciona un elemento base aleatorio (pivot), Divide la lista en 2. Una con elementos menores, y otra con elementos mayores al pivot. Se aplica de manera recursiva a cada sublista hasta ordenar todos los elementos. Ideal para listas grandes.

Búsqueda Binaria: Algoritmo eficiente que requiere una lista previamente ordenada. Divide repetidamente la lista en dos mitades, descartando la mitad donde el elemento buscado no puede estar.

Búsqueda Lineal: Algoritmo simple que recorre cada elemento de la lista en orden hasta encontrar el objetivo.

Tabla Comparativa de Algoritmos

Tipo	Algoritmo	Complejidad (Big-O)	Caso Ideal
Ordenamiento	Bubble Sort	$O(n^2)$	Listas pequeñas (<100 elementos).
Ordenamiento	Insertion Sort	$O(n^2)$	Listas casi ordenadas.
Ordenamiento	QuickSort	$O(n \log n)$	Listas grandes.
Búsqueda	Búsqueda Binaria	$O(\log n)$	Listas ordenadas.
Búsqueda	Búsqueda Lineal	$O(n)$	Listas desordenadas.

- Fuente: W3Schools Python Sorting Algorithms: [w3schools.com](https://www.w3schools.com/python/python_sorting.asp)
- Fuente: Búsqueda y Ordenamiento en Programación.pdf

3. Caso Práctico

Desarrollamos un script de comparación de algoritmos basado en una misma lista. Pudiendo así medir en tiempo de inicio y fin de ejecución que algoritmo es más eficiente según el caso de una lista de 1000 elementos, con un menú interactivo para facilitar su ejecución.

Implementacion de modulos/dependencias específicas para:

- **random**: para generar una lista aleatoria con una cantidad de números definida
- **colorama**: para diferenciar las salidas de cada función correspondiente a los algoritmos
- **time**: para poder medir el tiempo de ejecución
- **os**: para poder limpiar la terminal cada ejecución de una opcion del menu

Script en Python:

```
import random
import colorama
import time
import os

def limpiar_pantalla():
    os.system('cls' if os.name == 'nt' else 'clear')

# Random.sample(range(inicio, fin), cantidad de elementos)
listaDeNumerosIntermedia = random.sample(range(1, 51), 50)
listaDeNumerosGigante = random.sample(range(1, 1001), 1000)

# ===== ALGORITMOS DE ORDENAMIENTO =====

# Bubble Sort: compara elementos adyacentes y los intercambia si están desordenados
def BubbleSort(lista):
    num = len(lista)
    for i in range(num):
        for j in range(0, num - i - 1):
            if lista[j] > lista[j + 1]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
    return lista

def InsertionSort(listaGrande): # Ordena insertando cada elemento en su poscicion correcta dentro de una nueva lista ordenada.
```

```

for i in range(1, len(listaGrande)):
    key = listaGrande[i]
    j = i - 1
    while j >= 0 and key < listaGrande[j]:
        listaGrande[j + 1] = listaGrande[j]
        j -= 1
    listaGrande[j + 1] = key
return listaGrande

```

def QuickSort(listaGigante): # Ordena dividiendo la lista en sublistas menores y mayores al elemento pivote o elemento central, y luego ordenando esas sublistas recursivamente.

```

if len(listaGigante) <= 1:
    return listaGigante
else:
    elemento_pivot = listaGigante[len(listaGigante) // 2]
    izquierda = [x for x in listaGigante if x < elemento_pivot]
    medio = [x for x in listaGigante if x == elemento_pivot]
    derecha = [x for x in listaGigante if x > elemento_pivot]
    return QuickSort(izquierda) + medio + QuickSort(derecha)

```

def SelectionSort(lista): # Ordena seleccionando el elemento mas pequeño de la lista y colocandolo al principio, Es recursivo hasta completar la lista.

```

for i in range(len(lista)):
    min_idx = i
    for j in range(i + 1, len(lista)):
        if lista[j] < lista[min_idx]:
            min_idx = j
    lista[i], lista[min_idx] = lista[min_idx], lista[i]
return lista

```

sorted_list = sorted(listaDeNumerosGigante) # Ordena la lista gigante de numeros

===== ALGORITMOS DE BUSQUEDA
=====

Búsqueda Binaria: requiere lista ordenada, divide el rango de búsqueda a la mitad

```

def BusquedaBinaria(lista, objetivo):
    izquierda, derecha = 0, len(lista) - 1
    while izquierda <= derecha:
        medio = (izquierda + derecha) // 2
        if lista[medio] == objetivo:
            return medio
        elif lista[medio] < objetivo:
            izquierda = medio + 1
        else:

```

```

        derecha = medio - 1
    return -1 # Si no lo encuentra

def BusquedaLineal(lista, objetivo): # Bucle for que recorre la cada elemento
de la lista y compara si es igual al objetivo.

    for i in range(len(lista)):
        if lista[i] == objetivo:
            return i
    return -1 # Si no lo encuentra

def menu():
    while True:
        limpiar_pantalla()

        print("1. Ordenar lista de numeros con Bubble Sort")
        print("2. Ordenar lista grande de numeros con Insertion Sort")
        print("3. Ordenar lista gigante de numeros con Quick Sort")
        print("4. Ordenar lista de numeros enana con Selection Sort")
        print("5. Buscar un numero en la lista ordenada con Bubble Sort y
Busqueda Binaria")
        print("6. Buscar un numero en la lista ordenada con Selection Sort y
Busqueda Lineal")
        print("7. Ordenar con funcion integrada de Python (sorted)")
        print("8. Salir")
        opcion = input("Seleccione una opcion: ")
        if opcion == "8":
            print(colorama.Fore.CYAN + "Saliendo del programa...")
            break

        limpiar_pantalla() # Limpiar después de seleccionar cada opción

        inicio = time.time()

        if opcion == "1":
            print(colorama.Fore.RED + "Lista ordenada con Bubble Sort:",
BubbleSort(listaDeNumerosGigante))
            nombreAlgoritmo = "Bubble Sort"

        elif opcion == "2":
            print(colorama.Fore.YELLOW + f"Lista grande ordenada con Insertion
Sort:", InsertionSort(listaDeNumerosGigante))
            nombreAlgoritmo = "Insertion Sort"

        elif opcion == "3":
            print(colorama.Fore.YELLOW + f"Lista gigante ordenada con Quick
Sort: {QuickSort(listaDeNumerosGigante)}")
            nombreAlgoritmo = "Quick Sort"

        elif opcion == "4":

```

```

        print(colorama.Fore.MAGENTA + f"Lista de numeros ordenada con
Selection Sort: {SelectionSort(listaDeNumerosGigante)}")
        nombreAlgoritmo = "Selection Sort"

    elif opcion == "5":
        numeroObjetivo = int(input("Ingrese un numero entre 1 y 50: "))
        posicion = BusquedaBinaria(BubbleSort(listaDeNumerosIntermedia),
numeroObjetivo)
        if posicion != -1:
            print(f"Posicion del numero {numeroObjetivo} en la lista
ordenada con Bubble Sort Y Busqueda Binaria: {posicion}")
            nombreAlgoritmo = "Bubble Sort y Busqueda Binaria"
        else:
            print(f"El numero {numeroObjetivo} no se encuentra en la
lista.")

    elif opcion == "6":
        numeroObjetivoLineal = int(input("Ingrese un numero entre 1 y 50:
"))
        posicion = BusquedaLineal(SelectionSort(listaDeNumerosIntermedia),
numeroObjetivoLineal)
        if posicion != -1:
            print(f"Posicion del numero {numeroObjetivoLineal} en la lista
ordenada con Selection Sort Y Busqueda Lineal: {posicion}")
            nombreAlgoritmo = "Selection Sort y Busqueda Lineal"
        else:
            print(f"El numero {numeroObjetivoLineal} no se encuentra en la
lista.")

    elif opcion == "7":
        print(colorama.Fore.GREEN + f"Lista gigante ordenada con la
funcion integrada de Python (sorted): {sorted_list}")
        nombreAlgoritmo = "Funcion integrada de Python (sorted)"

    fin = time.time()
    duracion = fin - inicio

    print(colorama.Fore.BLUE + "Algoritmo usado: " + nombreAlgoritmo)
    print(colorama.Fore.BLUE + f"Tiempo de ejecucion: {duracion:.4f}
segundos")

    input("\nPresione Enter para volver al menú...")

menu()

```

4. Metodología Utilizada

1. Investigación:
 - Búsqueda y Ordenamiento en Programación.pdf
 - W3Schools Python Sorting Algorithms: [w3schools.com](https://www.w3schools.com/python/python_sorting.asp)
 2. Desarrollo:
 - Herramientas: VS Code, Git/GitHub.
 - División de tareas:
 - Ramiro: Implementó Bubble Sort, Insertion Sort, Quicksort, Búsqueda Binaria, Búsqueda Lineal.
 - Diego: Diseñó el menú interactivo, mediciones de tiempo y validó resultados.
 3. Pruebas:
 - 10 ejecuciones por algoritmo para obtener promedios confiables.
-

5. Resultados Obtenidos

Se midió el tiempo de ejecución de cada algoritmo utilizando listas aleatorias. Los resultados fueron los siguientes:

Algoritmo	Tiempo de Ejecución
Bubble Sort	0.0193 segundos
Insertion Sort	0.0000 segundos
Quick Sort	0.0010 segundos
Selection Sort	0.0150 segundos
Bubble Sort + Búsqueda Binaria	1.1858 segundos
Selection Sort + Búsqueda Lineal	2.9987 segundos
Función integrada de Python (sorted)	0.0010 segundos

Errores corregidos:

- Uso incorrecto de listas mutables compartidas.
- Ajustes de indentación y limpieza de pantalla según el sistema operativo.
- Validación de entradas numéricas en las búsquedas.

Observaciones:

Quick Sort y sorted() fueron los más rápidos en ordenar la lista gigante.

Bubble Sort y Selection Sort son notablemente más lentos con listas grandes.

La búsqueda binaria (con lista ordenada) fue más rápida que la búsqueda lineal, como se esperaba.

Insertion Sort fue extremadamente rápido con la lista usada, probablemente debido al orden previo de los elementos.

6. Bibliografía

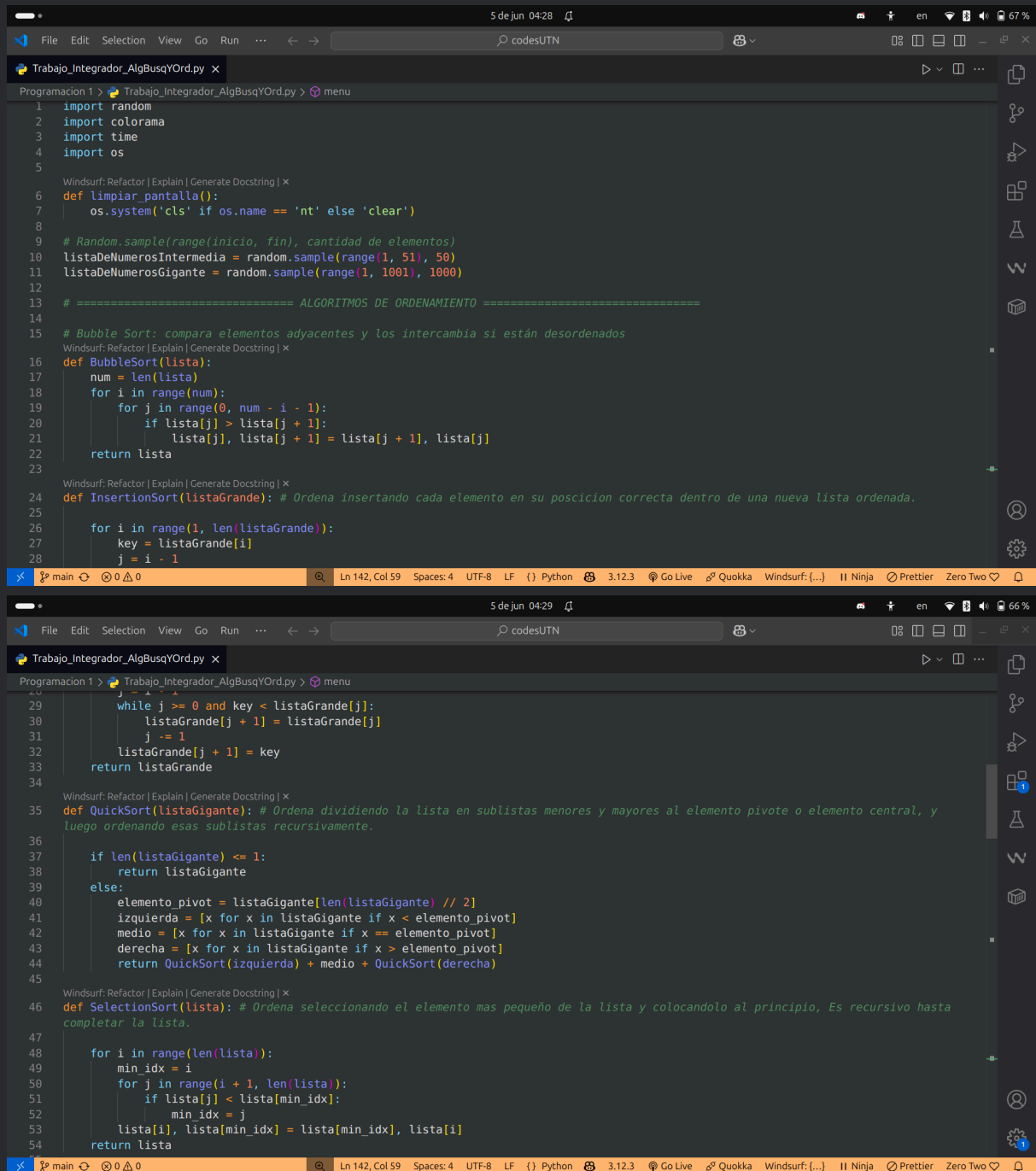
- Búsqueda y Ordenamiento en Programación.pdf
- W3Schools Python Sorting Algorithms: [w3schools.com](https://www.w3schools.com/python/python_sorting.asp)
- Documentación oficial de Python: <https://docs.python.org/3/>

7. Anexos

Enlace al repositorio: [Repositorio al GitHub](#)

Video Presentación: [Video YouTube](#)

Capturas de pantalla:



The image displays two screenshots of a code editor window, likely Visual Studio Code, showing Python code for sorting algorithms. The editor has a dark theme and a sidebar on the right with various icons.

Top Screenshot: The code is in a file named `Trabajo_Integrador_AlgBusqYOrd.py`. It includes imports for `random`, `colorama`, `time`, and `os`. A function `limpiar_pantalla()` is defined to clear the terminal. The code then generates two lists: `listaDeNumerosIntermedia` and `listaDeNumerosGigante`. It defines two sorting functions: `BubbleSort` and `InsertionSort`. The `BubbleSort` function is commented out, and the `InsertionSort` function is active. The status bar at the bottom shows the cursor is at line 142, column 59.

```
1 import random
2 import colorama
3 import time
4 import os
5
6 def limpiar_pantalla():
7     os.system('cls' if os.name == 'nt' else 'clear')
8
9 # Random.sample(range(inicio, fin), cantidad de elementos)
10 listaDeNumerosIntermedia = random.sample(range(1, 51), 50)
11 listaDeNumerosGigante = random.sample(range(1, 1001), 1000)
12
13 # ===== ALGORITMOS DE ORDENAMIENTO =====
14
15 # Bubble Sort: compara elementos adyacentes y los intercambia si están desordenados
16 def BubbleSort(lista):
17     num = len(lista)
18     for i in range(num):
19         for j in range(0, num - i - 1):
20             if lista[j] > lista[j + 1]:
21                 lista[j], lista[j + 1] = lista[j + 1], lista[j]
22     return lista
23
24 def InsertionSort(listaGrande): # Ordena insertando cada elemento en su poscion correcta dentro de una nueva lista ordenada.
25
26     for i in range(1, len(listaGrande)):
27         key = listaGrande[i]
28         j = i - 1
```

Bottom Screenshot: This screenshot shows the continuation of the code. It includes the `while` loop for the `InsertionSort` function, the `QuickSort` function, and the `SelectionSort` function. The `QuickSort` function is commented out, and the `SelectionSort` function is active. The status bar at the bottom shows the cursor is at line 142, column 59.

```
29     while j >= 0 and key < listaGrande[j]:
30         listaGrande[j + 1] = listaGrande[j]
31         j -= 1
32     listaGrande[j + 1] = key
33     return listaGrande
34
35 def QuickSort(listaGigante): # Ordena dividiendo la lista en sublistas menores y mayores al elemento pivote o elemento central, y
36     # luego ordenando esas sublistas recursivamente.
37     if len(listaGigante) <= 1:
38         return listaGigante
39     else:
40         elemento_pivot = listaGigante[len(listaGigante) // 2]
41         izquierda = [x for x in listaGigante if x < elemento_pivot]
42         medio = [x for x in listaGigante if x == elemento_pivot]
43         derecha = [x for x in listaGigante if x > elemento_pivot]
44         return QuickSort(izquierda) + medio + QuickSort(derecha)
45
46 def SelectionSort(lista): # Ordena seleccionando el elemento mas pequeño de la lista y colocandolo al principio, Es recursivo hasta
47     # completar la lista.
48     for i in range(len(lista)):
49         min_idx = i
50         for j in range(i + 1, len(lista)):
51             if lista[j] < lista[min_idx]:
52                 min_idx = j
53     lista[i], lista[min_idx] = lista[min_idx], lista[i]
54     return lista
```

```
5 de jun 04:30
File Edit Selection View Go Run ... codesUTN
Trabajo_Integrador_AlgBusqYOrd.py x
Programacion 1 > Trabajo_Integrador_AlgBusqYOrd.py > menu

58 sorted_list = sorted(listaDeNumerosGigante) # Ordena la lista gigante de numeros
59
60 # ===== ALGORITMOS DE BUSQUEDA =====
61
62 # Búsqueda Binaria: requiere lista ordenada, divide el rango de búsqueda a la mitad
Windsurf: Refactor | Explain | Generate Docstring | x
63 def BusquedaBinaria(lista, objetivo):
64     izquierda, derecha = 0, len(lista) - 1
65     while izquierda <= derecha:
66         medio = (izquierda + derecha) // 2
67         if lista[medio] == objetivo:
68             return medio
69         elif lista[medio] < objetivo:
70             izquierda = medio + 1
71         else:
72             derecha = medio - 1
73     return -1 # Si no lo encuentra
74
75
Windsurf: Refactor | Explain | Generate Docstring | x
76 def BusquedaLineal(lista, objetivo): # Bucle for que recorre la cada elemento de la lista y compara si es igual al objetivo.
77
78     for i in range(len(lista)):
79         if lista[i] == objetivo:
80             return i
81     return -1 # Si no lo encuentra
82
83
Windsurf: Refactor | Explain | Generate Docstring | x
84 def menu():
85     while True:
86         limpiar_pantalla()
87
88         print("1. Ordenar lista de numeros con Bubble Sort")
89         print("2. Ordenar lista grande de numeros con Insertion Sort")
90         print("3. Ordenar lista gigante de numeros con Quick Sort")
91         print("4. Ordenar lista de numeros enana con Selection Sort")
92         print("5. Buscar un numero en la lista ordenada con Bubble Sort y Busqueda Binaria")
93         print("6. Buscar un numero en la lista ordenada con Selection Sort y Busqueda Lineal")
94         print("7. Ordenar con funcion integrada de Python (sorted)")
95         print("8. Salir")
96         opcion = input("Seleccione una opcion: ")
97         if opcion == "8":
98             print(colorama.Fore.CYAN + "Saliendo del programa...")
99             break
100
101         limpiar_pantalla() # Limpiar después de seleccionar cada opción
102
103         inicio = time.time()
104
105         if opcion == "1":
106             print(colorama.Fore.RED + "Lista ordenada con Bubble Sort:", BubbleSort(listaDeNumerosGigante))
107             nombreAlgoritmo = "Bubble Sort"
108
109         elif opcion == "2":
110             print(colorama.Fore.YELLOW + f"Lista grande ordenada con Insertion Sort:", InsertionSort(listaDeNumerosGigante))
111             nombreAlgoritmo = "Insertion Sort"
112
113         elif opcion == "3":
114             print(colorama.Fore.BLUE + f"Lista gigante ordenada con Quick Sort:", QuickSort(listaDeNumerosGigante))
115             nombreAlgoritmo = "Quick Sort"
116
117         elif opcion == "4":
118             print(colorama.Fore.MAGENTA + f"Lista enana ordenada con Selection Sort:", SelectionSort(listaDeNumerosEnana))
119             nombreAlgoritmo = "Selection Sort"
120
121         elif opcion == "5":
122             print(colorama.Fore.CYAN + f"Busqueda Binaria de {objetivo} en la lista ordenada: {BusquedaBinaria(listaOrdenada, objetivo)}")
123
124         elif opcion == "6":
125             print(colorama.Fore.CYAN + f"Busqueda Lineal de {objetivo} en la lista ordenada: {BusquedaLineal(listaOrdenada, objetivo)}")
126
127         elif opcion == "7":
128             print(colorama.Fore.GREEN + f"Lista ordenada con sorted: {sorted(listaDeNumerosGigante)}")
129
130         elif opcion == "8":
131             print(colorama.Fore.CYAN + "Saliendo del programa...")
132             break
133
134         # Mostrar tiempo de ejecución
135         tiempo = time.time() - inicio
136         print(f"Tiempo de ejecución: {tiempo} segundos")
137
138         # Preguntar si se desea continuar
139         continuar = input("¿Desea continuar? (s/n): ")
140         if continuar.lower() != 's':
141             break
142
143     return nombreAlgoritmo
144
145 if __name__ == "__main__":
146     menu()
147
148 main 0 0 0 Ln 142, Col 59 Spaces: 4 UTF-8 LF Python 3.12.3 Go Live Quokka Windsurf: [...] It Ninja Prettier Zero Two
```

```
5 de jun 04:30
File Edit Selection View Go Run ... codesUTN
Trabajo_Integrador_AlgBusqYOrd.py x
Programacion 1 > Trabajo_Integrador_AlgBusqYOrd.py > menu

84 def menu():
85     while True:
86         limpiar_pantalla()
87
88         print("1. Ordenar lista de numeros con Bubble Sort")
89         print("2. Ordenar lista grande de numeros con Insertion Sort")
90         print("3. Ordenar lista gigante de numeros con Quick Sort")
91         print("4. Ordenar lista de numeros enana con Selection Sort")
92         print("5. Buscar un numero en la lista ordenada con Bubble Sort y Busqueda Binaria")
93         print("6. Buscar un numero en la lista ordenada con Selection Sort y Busqueda Lineal")
94         print("7. Ordenar con funcion integrada de Python (sorted)")
95         print("8. Salir")
96         opcion = input("Seleccione una opcion: ")
97         if opcion == "8":
98             print(colorama.Fore.CYAN + "Saliendo del programa...")
99             break
100
101         limpiar_pantalla() # Limpiar después de seleccionar cada opción
102
103         inicio = time.time()
104
105         if opcion == "1":
106             print(colorama.Fore.RED + "Lista ordenada con Bubble Sort:", BubbleSort(listaDeNumerosGigante))
107             nombreAlgoritmo = "Bubble Sort"
108
109         elif opcion == "2":
110             print(colorama.Fore.YELLOW + f"Lista grande ordenada con Insertion Sort:", InsertionSort(listaDeNumerosGigante))
111             nombreAlgoritmo = "Insertion Sort"
112
113         elif opcion == "3":
114             print(colorama.Fore.BLUE + f"Lista gigante ordenada con Quick Sort:", QuickSort(listaDeNumerosGigante))
115             nombreAlgoritmo = "Quick Sort"
116
117         elif opcion == "4":
118             print(colorama.Fore.MAGENTA + f"Lista enana ordenada con Selection Sort:", SelectionSort(listaDeNumerosEnana))
119             nombreAlgoritmo = "Selection Sort"
120
121         elif opcion == "5":
122             print(colorama.Fore.CYAN + f"Busqueda Binaria de {objetivo} en la lista ordenada: {BusquedaBinaria(listaOrdenada, objetivo)}")
123
124         elif opcion == "6":
125             print(colorama.Fore.CYAN + f"Busqueda Lineal de {objetivo} en la lista ordenada: {BusquedaLineal(listaOrdenada, objetivo)}")
126
127         elif opcion == "7":
128             print(colorama.Fore.GREEN + f"Lista ordenada con sorted: {sorted(listaDeNumerosGigante)}")
129
130         elif opcion == "8":
131             print(colorama.Fore.CYAN + "Saliendo del programa...")
132             break
133
134         # Mostrar tiempo de ejecución
135         tiempo = time.time() - inicio
136         print(f"Tiempo de ejecución: {tiempo} segundos")
137
138         # Preguntar si se desea continuar
139         continuar = input("¿Desea continuar? (s/n): ")
140         if continuar.lower() != 's':
141             break
142
143     return nombreAlgoritmo
144
145 if __name__ == "__main__":
146     menu()
147
148 main 0 0 0 Ln 142, Col 59 Spaces: 4 UTF-8 LF Python 3.12.3 Go Live Quokka Windsurf: [...] It Ninja Prettier Zero Two
```

```
5 de jun 04:30
File Edit Selection View Go Run ... codesUTN
Trabajo_Integrador_AlgBusqYOrd.py x
Programacion 1 > Trabajo_Integrador_AlgBusqYOrd.py > menu
112
113 elif opcion == "3":
114     print(colorama.Fore.YELLOW + f"Lista gigante ordenada con Quick Sort: {QuickSort(listaDeNumerosGigante)}")
115     nombreAlgoritmo = "Quick Sort"
116
117 elif opcion == "4":
118     print(colorama.Fore.MAGENTA + f"Lista de numeros ordenada con Selection Sort: {SelectionSort(listaDeNumerosGigante)}")
119     nombreAlgoritmo = "Selection Sort"
120
121 elif opcion == "5":
122     numeroObjetivo = int(input("Ingrese un numero entre 1 y 50: "))
123     posicion = BusquedaBinaria(BubbleSort(listaDeNumerosIntermedia), numeroObjetivo)
124     if posicion != -1:
125         print(f"Posicion del numero {numeroObjetivo} en la lista ordenada con Bubble Sort Y Busqueda Binaria: {posicion}")
126         nombreAlgoritmo = "Bubble Sort y Busqueda Binaria"
127     else:
128         print(f"El numero {numeroObjetivo} no se encuentra en la lista.")
129
130
131 elif opcion == "6":
132     numeroObjetivoLineal = int(input("Ingrese un numero entre 1 y 50: "))
133     posicion = BusquedaLineal(SelectionSort(listaDeNumerosIntermedia), numeroObjetivoLineal)
134     if posicion != -1:
135         print(f"Posicion del numero {numeroObjetivoLineal} en la lista ordenada con Selection Sort Y Busqueda Lineal: {posicion}")
136         nombreAlgoritmo = "Selection Sort y Busqueda Lineal"
137     else:
138         print(f"El numero {numeroObjetivoLineal} no se encuentra en la lista.")
139
140 elif opcion == "7":
141     print(colorama.Fore.GREEN + f"Lista gigante ordenada con la funcion integrada de Python (sorted): {sorted(listaDeNumerosGigante)}")
```

```
5 de jun 04:30
File Edit Selection View Go Run ... codesUTN
Trabajo_Integrador_AlgBusqYOrd.py x
Programacion 1 > Trabajo_Integrador_AlgBusqYOrd.py > menu
129
130
131 elif opcion == "6":
132     numeroObjetivoLineal = int(input("Ingrese un numero entre 1 y 50: "))
133     posicion = BusquedaLineal(SelectionSort(listaDeNumerosIntermedia), numeroObjetivoLineal)
134     if posicion != -1:
135         print(f"Posicion del numero {numeroObjetivoLineal} en la lista ordenada con Selection Sort Y Busqueda Lineal: {posicion}")
136         nombreAlgoritmo = "Selection Sort y Busqueda Lineal"
137     else:
138         print(f"El numero {numeroObjetivoLineal} no se encuentra en la lista.")
139
140 elif opcion == "7":
141     print(colorama.Fore.GREEN + f"Lista gigante ordenada con la funcion integrada de Python (sorted): {sorted(listaDeNumerosGigante)}")
142     nombreAlgoritmo = "Funcion integrada de Python (sorted)"
143
144     fin = time.time()
145     duracion = fin - inicio
146
147     print(colorama.Fore.BLUE + "Algoritmo usado: " + nombreAlgoritmo)
148     print(colorama.Fore.BLUE + f"Tiempo de ejecucion: {duracion:.4f} segundos")
149
150     input("\nPresione Enter para volver al menú...")
151
152     menu()
```