

# **Algoritmos de Búsqueda y Ordenamiento**

**Alumnos:**

Santiago Nicolas Nievas – santiagon98@hotmail.com

Gonzalo Ojeada – gonzalowojeada@outlook.com

**Materia:** Programación I

**Profesora:** Julieta Trapé

**Tutor:** Marcos Vega

**Fecha de Entrega:** 8 de Junio de 2025

**Índice**

- 1. Introducción**
- 2. Marco Teórico**
- 3. Caso Práctico**
- 4. Metodología Utilizada**
- 5. Resultados Obtenidos**
- 6. Conclusiones**
- 7. Bibliografía**
- 8. Anexos**

## 1. Introducción

En este trabajo integrador se aborda el estudio y aplicación de los algoritmos de búsqueda y ordenamiento, temáticas fundamentales en el desarrollo de software y resolución de problemas computacionales. A lo largo del proyecto, se busca comprender su funcionamiento, analizar su eficiencia y poner en práctica su implementación en Python. Este trabajo no solo refleja los conocimientos adquiridos durante la cursada de Programación I, sino también el desarrollo de habilidades para investigar, planificar, programar y documentar soluciones algorítmicas.

## 2. Marco Teórico

Los algoritmos de búsqueda permiten localizar un elemento dentro de una estructura de datos. En este trabajo nos centramos en dos de ellos:

**Búsqueda Lineal:** recorre la estructura de forma secuencial. Su complejidad temporal es  $O(n)$ .

**Búsqueda Binaria:** requiere que los datos estén ordenados. Divide y conquista, comparando el elemento medio con el buscado. Su complejidad es  $O(\log n)$ .

Por otro lado, los algoritmos de ordenamiento permiten organizar elementos siguiendo un criterio, generalmente numérico o alfabético:

**Bubble Sort:** compara pares de elementos adyacentes y los intercambia si están desordenados. Es simple pero poco eficiente ( $O(n^2)$ ).

**Insertion Sort:** inserta elementos uno a uno en su posición correcta. Es más eficiente en listas pequeñas o casi ordenadas ( $O(n^2)$  en el peor caso,  $O(n)$  en el mejor).

Estos algoritmos son esenciales para optimizar búsquedas y otras operaciones sobre datos.

## 3. Caso Práctico

Se diseñó un programa en Python que permite al usuario:

1. Ingresar una lista de números.
2. Ordenar la lista con Bubble Sort o Insertion Sort (a elección del usuario).
3. Realizar una búsqueda de un número mediante Búsqueda Lineal o Binaria (según si la lista está ordenada).

La interacción es sencilla y por consola, pensando en la claridad y la comprensión del proceso por parte del usuario.

## 4. Metodología Utilizada

Se utilizó una metodología de desarrollo incremental y basada en prueba y error. El proceso incluyó:

**Investigación teórica:** recopilación de información sobre los algoritmos.

**Diseño del programa:** planteo de funciones independientes para cada algoritmo.

**Implementación:** codificación en Python, con énfasis en la legibilidad del código y modularidad.

**Pruebas:** ejecución de casos simples y casos límite.

**Documentación:** registro del proceso, decisiones tomadas y estructura del código.

El enfoque fue práctico y progresivo, priorizando la comprensión real de los algoritmos sobre la simple copia de código.

## 5. Resultados Obtenidos

El programa desarrollado cumple con las funciones esperadas. Entre los logros más destacados:

Comprobamos empíricamente la diferencia de eficiencia entre los algoritmos.

Se verificó la necesidad de ordenar previamente para que funcione la búsqueda binaria.

Desarrollamos un programa modular y reutilizable, fácilmente extensible con otros algoritmos.

Se consolidaron conceptos fundamentales como complejidad algorítmica y estructuras de control.

## 6. Conclusiones

Este trabajo permitió afianzar los conceptos aprendidos en clase y trasladarlos a una situación práctica. Se aprendió no solo a programar algoritmos, sino a analizarlos, compararlos y aplicarlos con criterio. También se valoró la importancia de una buena planificación y la claridad en el código y la documentación. Queda claro que la programación no es solo escribir código, sino construir soluciones pensadas, optimizadas y comunicables.

Además, trabajar en equipo permitió compartir ideas, dividir tareas de forma efectiva y ejercitar la colaboración, algo fundamental en entornos reales de desarrollo. Este proyecto también fortaleció habilidades como la resolución de problemas, la autonomía en el aprendizaje y el uso de herramientas como GitHub y Python para llevar adelante proyectos integradores. En definitiva, fue una experiencia completa que consolidó nuestros conocimientos técnicos.

## 7. Bibliografía

Estos fueron los materiales de lectura y videos proporcionados en el aula virtual que utilizamos para la realización de este trabajo.

Introduccion al Analisis de Algoritmos

[http://www.youtube.com/watch?v=KaPkdPMh6K8&embeds\\_referring\\_euri=https%3A%2F%2Ftup.sied.utn.edu.ar%2F&source\\_ve\\_path=MjM4NTE](http://www.youtube.com/watch?v=KaPkdPMh6K8&embeds_referring_euri=https%3A%2F%2Ftup.sied.utn.edu.ar%2F&source_ve_path=MjM4NTE)

<https://www.youtube.com/watch?v=KdJnKVXmI1A>

Analisis empirico sumN

<https://www.youtube.com/watch?v=ptsijUkV0il>

Analisis empirico de sumGaus

<https://www.youtube.com/watch?v=UKRa9HZCysY>

Introduccion al analisis teorico

<https://www.youtube.com/watch?v=rdRrKIKP9T8>

Estructuras de control en analisis teorico

<https://www.youtube.com/watch?v=AJOiiMo8IVE>

Analisis teorico de sumN y sumGaus

<https://www.youtube.com/watch?v=XoiAMy6XM9s>

Primer ejemplo de analisis teorico

<https://www.youtube.com/watch?v=o-Oe8qh9irM>

Segundo ejemplo analisis teorico

<https://www.youtube.com/watch?v=QzEN0OVALBs>

Análisis Teórico de Algoritmos

[https://tup.sied.utn.edu.ar/pluginfile.php/10251/mod\\_label/intro/Analisis-Teorico-de-Algoritmos.pdf](https://tup.sied.utn.edu.ar/pluginfile.php/10251/mod_label/intro/Analisis-Teorico-de-Algoritmos.pdf)

Análisis Teórico y Notación Big-O

[https://tup.sied.utn.edu.ar/pluginfile.php/10251/mod\\_label/intro/Analisis%20de%20algoritmo%20Teorico%20y%20Big%20O.pdf](https://tup.sied.utn.edu.ar/pluginfile.php/10251/mod_label/intro/Analisis%20de%20algoritmo%20Teorico%20y%20Big%20O.pdf)

Introduccion a Big O

<https://www.youtube.com/watch?v=j8PqpS21Gpo>

Tipos de ordenes de Big O

<https://www.youtube.com/watch?v=CooM8LMLp9s>

Ejemplo de como obtener la Big O

<https://www.youtube.com/watch?v=xFiFBwT7Jn0>

Notación Big-O Conceptos clave y ejemplos de código

[https://tup.sied.utn.edu.ar/pluginfile.php/10247/mod\\_label/intro/Notacion-Big-O.pdf](https://tup.sied.utn.edu.ar/pluginfile.php/10247/mod_label/intro/Notacion-Big-O.pdf)

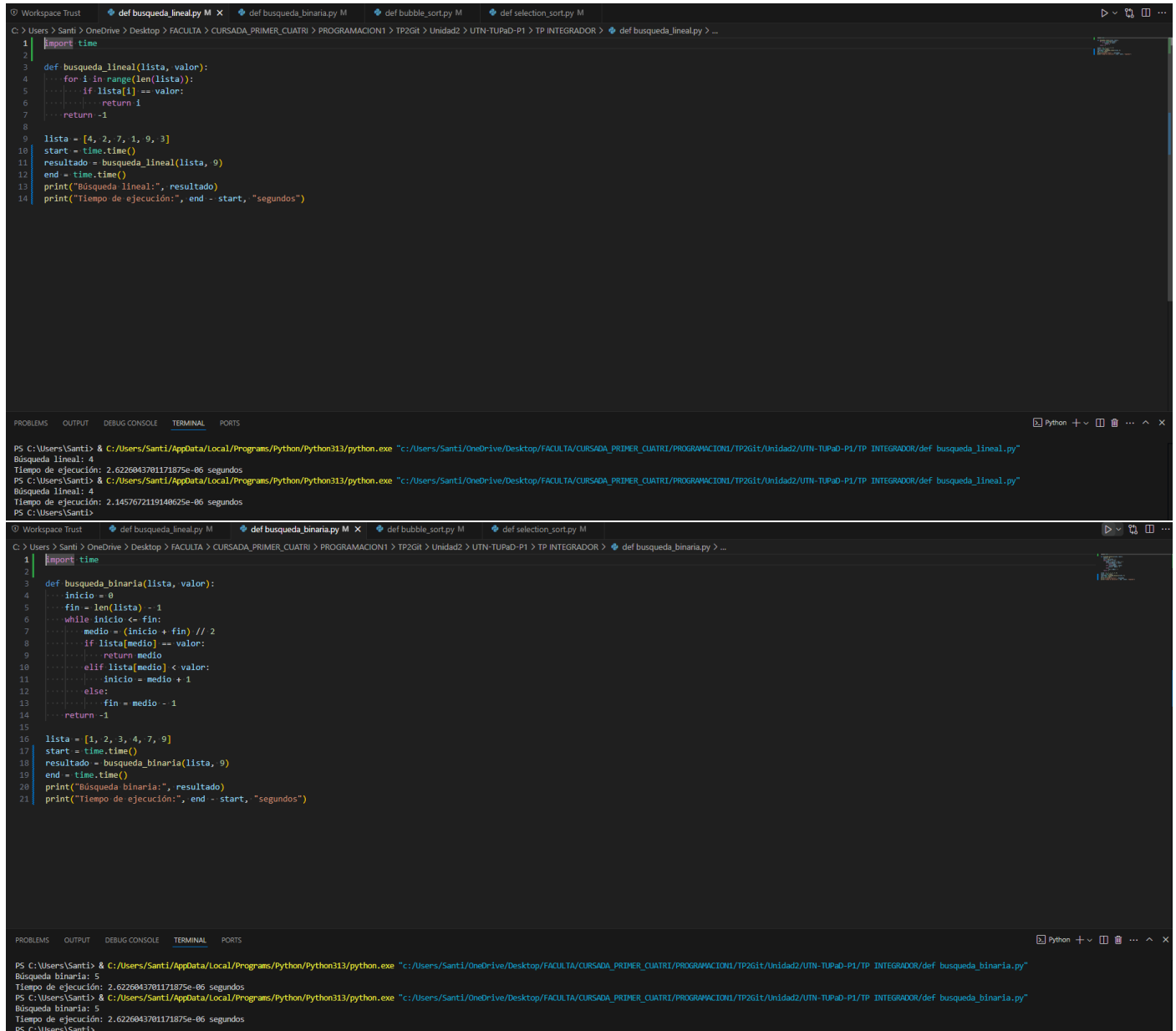
## 8. Anexos

Código fuente de los algoritmos implementados (ver repositorio GitHub).

Capturas de pantalla de pruebas realizadas.

Enlace al video explicativo: <https://www.youtube.com/watch?v=BjiFP9p3QJ4>

README con guía de ejecución.



```
1 import time
2
3 def busqueda_lineal(lista, valor):
4     for i in range(len(lista)):
5         if lista[i] == valor:
6             return i
7     return -1
8
9 lista = [4, 2, 7, 1, 9, 3]
10 start = time.time()
11 resultado = busqueda_lineal(lista, 9)
12 end = time.time()
13 print("Búsqueda lineal:", resultado)
14 print("Tiempo de ejecución:", end - start, "segundos")
```

```
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA_PRIMER_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPad-P1/TP INTEGRADOR/def busqueda_lineal.py"
Búsqueda lineal: 4
Tiempo de ejecución: 2.6226843701171875e-06 segundos
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA_PRIMER_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPad-P1/TP INTEGRADOR/def busqueda_lineal.py"
Búsqueda lineal: 4
Tiempo de ejecución: 2.1457672119148625e-06 segundos
PS C:\Users\Santi>
```

```
1 import time
2
3 def busqueda_binaria(lista, valor):
4     inicio = 0
5     fin = len(lista) - 1
6     while inicio <= fin:
7         medio = (inicio + fin) // 2
8         if lista[medio] == valor:
9             return medio
10        elif lista[medio] < valor:
11            inicio = medio + 1
12        else:
13            fin = medio - 1
14    return -1
15
16 lista = [1, 2, 3, 4, 7, 9]
17 start = time.time()
18 resultado = busqueda_binaria(lista, 9)
19 end = time.time()
20 print("Búsqueda binaria:", resultado)
21 print("Tiempo de ejecución:", end - start, "segundos")
```

```
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA_PRIMER_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPad-P1/TP INTEGRADOR/def busqueda_binaria.py"
Búsqueda binaria: 5
Tiempo de ejecución: 2.6226843701171875e-06 segundos
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA_PRIMER_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPad-P1/TP INTEGRADOR/def busqueda_binaria.py"
Búsqueda binaria: 5
Tiempo de ejecución: 2.6226843701171875e-06 segundos
PS C:\Users\Santi>
```

Workspace Trustdef busqueda\_lineal.py Mdef busqueda\_binaria.py Mdef bubble\_sort.py M Xdef selection\_sort.py M

C:\Users\Santi> OneDrive > Desktop > FACULTA > CURSADA\_PRIMER\_CUATRI > PROGRAMACION1 > TP2Git > Unidad2 > UTN-TUPaD-P1 > TP INTEGRADOR > def bubble\_sort.py > ...

```
1 import time
2
3 def bubble_sort(lista):
4     n = len(lista)
5     for i in range(n):
6         for j in range(0, n-1-i):
7             if lista[j] > lista[j+1]:
8                 lista[j], lista[j+1] = lista[j+1], lista[j]
9
10 lista = [64, 34, 25, 12, 22, 11, 90]
11 start = time.time()
12 bubble_sort(lista)
13 end = time.time()
14 print("Bubble Sort:", lista)
15 print("Tiempo de ejecución:", end - start, "segundos")
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

Python + Python ...

PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA\_PRIMER\_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPaD-P1/TP INTEGRADOR/def bubble\_sort.py"
Bubble Sort: [11, 12, 22, 25, 34, 64, 90]
Tiempo de ejecución: 5.245288748234375e-06 segundos
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA\_PRIMER\_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPaD-P1/TP INTEGRADOR/def bubble\_sort.py"
Bubble Sort: [11, 12, 22, 25, 34, 64, 90]
Tiempo de ejecución: 5.245288748234375e-06 segundos
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA\_PRIMER\_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPaD-P1/TP INTEGRADOR/def bubble\_sort.py"
Bubble Sort: [11, 12, 22, 25, 34, 64, 90]
Tiempo de ejecución: 5.228458984375e-06 segundos
PS C:\Users\Santi>

Workspace Trustdef busqueda\_lineal.py Mdef busqueda\_binaria.py Mdef bubble\_sort.py M Xdef selection\_sort.py M

C:\Users\Santi> OneDrive > Desktop > FACULTA > CURSADA\_PRIMER\_CUATRI > PROGRAMACION1 > TP2Git > Unidad2 > UTN-TUPaD-P1 > TP INTEGRADOR > def selection\_sort.py > ...

```
1 import time
2
3 def selection_sort(lista):
4     for i in range(len(lista)):
5         min_idx = i
6         for j in range(i+1, len(lista)):
7             if lista[j] < lista[min_idx]:
8                 min_idx = j
9         lista[i], lista[min_idx] = lista[min_idx], lista[i]
10
11 lista = [64, 25, 12, 22, 11]
12 start = time.time()
13 selection_sort(lista)
14 end = time.time()
15 print("Selection Sort:", lista)
16 print("Tiempo de ejecución:", end - start, "segundos")
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

Python + Python ...

PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA\_PRIMER\_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPaD-P1/TP INTEGRADOR/def selection\_sort.py"
Selection Sort: [11, 12, 22, 25, 64]
Tiempo de ejecución: 4.76837158283125e-06 segundos
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA\_PRIMER\_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPaD-P1/TP INTEGRADOR/def selection\_sort.py"
Selection Sort: [11, 12, 22, 25, 64]
Tiempo de ejecución: 4.5299538029296875e-06 segundos
PS C:\Users\Santi> & C:/Users/Santi/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Santi/OneDrive/Desktop/FACULTA/CURSADA\_PRIMER\_CUATRI/PROGRAMACION1/TP2Git/Unidad2/UTN-TUPaD-P1/TP INTEGRADOR/def selection\_sort.py"
Selection Sort: [11, 12, 22, 25, 64]
Tiempo de ejecución: 5.8067981611328125e-06 segundos
PS C:\Users\Santi>