



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

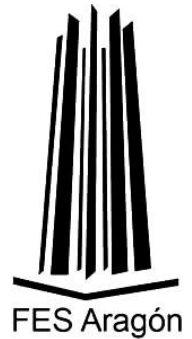
INGENIERÍA EN COMPUTACIÓN

DISEÑO Y ANÁLISIS DE ALGORITMOS

GRUPO 1558 2026 – 1

PROFESOR JESÚS HERNÁNDEZ CABRERA

ALUMNO PEDRO CÉSAR JUÁREZ NÚÑEZ



Tarea 2. Búsqueda Lineal

Primer código: **hw_02.py**

Para este código realicé dos funciones. **buscar_num()** que imprime si fue encontrado, o no, el número buscado; y la función **busq_lineal()** que devuelve dos valores, el número de veces que se hizo la comprobación (con el contador) y un booleano llamado encontrado. Realicé dos funciones porque quería poner en práctica las funciones. Considerar que es menos eficiente.

```
tarea_2 > hw_02.py > ...
1  list_01 = [4, 1, 4, 7, 9, 10]      # T(n) = n   S(n) = n
2
3  valor_A_Buscar = 5                # T(n) = 1   S(n) = 1
4
5  def busq_lineal (arreglo, valor):  # S(n) = 1
6      contador = 0                  # T(n) = 1   S(n) = 1
7      for i in arreglo:             # T(n) = n   S(n) = 1
8          contador += 1              # T(n) = 1
9          if(i == valor):            # T(n) = 1
10             return contador, True  # T(n) = 1
11             return contador, False # T(n) = 1
12
13 def buscar_num (arreglo, valor):    # S(n) = 1
14     contador, encontrado = busq_lineal(arreglo, valor) # T(n) = 1   S(n) = 2
15     if encontrado:                 # T(n) = 1
16         print(f"Número {valor} encontrado.\nNúmero de comparaciones: {contador}") # T(n) = 1
17     else:
18         print(f"Número {valor} no encontrado.\nNúmero de comparaciones: {contador}") # T(n) = 1
19
20 buscar_num(list_01, valor_A_Buscar) # T(n) = 1   S(n) = 1
21
22 # T(n) = n + 1 + 1 + n + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = [2n + 11]
23 # S(n) = n + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 1 = [n + 8]
```

ECUACIONES DE COMPLEJIDAD:

$$\# T(n) = n + 1 + 1 + n + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = [2n + 11]$$

$$\# S(n) = n + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 1 = [n + 8]$$

EJEMPLOS DE EJECUCIÓN:

EJEMPLO 1: valor encontrado

```
tarea_2 > hw_02.py > ...
1 list_01 = [4, 1, 3, 7, 9, 100, 29, 10, 50] # T(n) = n S(n) = n
2
3 valor_A_Buscar = 100 # T(n) = 1 S(n) = 1
4
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS C:\Users\kit21\Documents\5to semestre\algoritmos> & C:/Users/kit21/AppData/Local/Programs/Python/Python39-64/Python.exe C:\Users\kit21\Documents\5to semestre\algoritmos\tarea_2\hw_02.py"
Número 100 encontrado.
Número de comparaciones: 6
PS C:\Users\kit21\Documents\5to semestre\algoritmos>
```

EJEMPLO 2: valor no encontrado

```
tarea_2 > hw_02.py > ...
1 list_01 = [4, 1, 3, 7, 9, 100, 29, 10, 50] # T(n) = n S(n) = n
2
3 valor_A_Buscar = 400000 # T(n) = 1 S(n) = 1
4
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS C:\Users\kit21\Documents\5to semestre\algoritmos> & C:/Users/kit21/AppData/Local/Programs/Python/Python38-32/Python.exe C:\Users\kit21\Documents\5to semestre\algoritmos\tarea_2\hw_02.py
Número 400000 no encontrado.
Número de comparaciones: 9
PS C:\Users\kit21\Documents\5to semestre\algoritmos>
```

Segundo código: hw_02_01.py

Para este código sólo realicé una función. Es más óptimo y eficiente.

```
tarea_2 > hw_02_01.py > ...
1 # TAREA 1: Búsqueda lineal
2 list_01 = [3,8,4,7,9,5,2] # T(n) = n S(n) = n
3
4 valor_A_Buscar = 100 # T(n) = 1 S(n) = 1
5
6 def busq_lineal (arreglo, valor): # S(n) = 1
7
8     contador = 0 # T(n) = 1 S(n) = 1
9     for i in arreglo: # T(n) = n S(n) = 1
10         contador += 1 # T(n) = 1
11         if(i == valor): # T(n) = 1
12             return contador, True # T(n) = 1
13     return contador, False # T(n) = 1
14
15 contador, encontrado = busq_lineal(list_01, valor_A_Buscar) # T(n) = 1 S(n) = 2
16 if encontrado: # T(n) = 1
17     print(f"Número {valor_A_Buscar} encontrado.\nNúmero de comparaciones: {contador}") # T(n) = 1
18 else:
19     print(f"Número {valor_A_Buscar} no encontrado.\nNúmero de comparaciones: {contador}") # T(n) = 1
20
21 # T(n) = n + 1 + 1 + n + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = [2n + 10]
22 # S(n) = n + 1 + 1 + 1 + 1 + 1 + 2 = [n + 6]
```

ECUACIONES DE COMPLEJIDAD:

$$\# T(n) = n + 1 + 1 + n + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = [2n + 10]$$

$$\# S(n) = n + 1 + 1 + 1 + 1 + 1 + 2 = [n + 6]$$

EJEMPLOS DE EJECUCIÓN:


EJEMPLO 1: valor encontrado

```
tarea_2 > hw_02_01.py > ...
1 # TAREA 1: Búsqueda lineal
2 list_01 = [4, 1, 4, 7, 9, 10] # T(n) = n S(n) = n
3
4 valor_A_Buscar = 9 # T(n) = 1 S(n) = 1
5
6 def busq_lineal (arreglo, valor): # S(n) = 1
7
8     contador = 0 # T(n) = 1 S(n) = 1
9     for i in arreglo: # T(n) = n S(n) = 1
10         contador += 1 # T(n) = 1
11         if(i == valor): # T(n) = 1
12             return contador, True # T(n) = 1
13     return contador, False # T(n) = 1
14
15 contador, encontrado = busq_lineal(list_01, valor_A_Buscar) # T(n) = 1 S(n) = 2
16 if encontrado: # T(n) = 1
17     print(f"Número {valor_A_Buscar} encontrado.\nNúmero de comparaciones: {contador}") # T(n) = 1
18 else:
19     print(f"Número {valor_A_Buscar} no encontrado.\nNúmero de comparaciones: {contador}") # T(n) = 1
20
21 # T(n) = n + 1 + 1 + n + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = [2n + 10]
22 # S(n) = n + 1 + 1 + 1 + 1 + 1 + 2 = [n + 6]
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS C:\Users\kit21\Documents\5to semestre\algoritmos> & C:/Users/kit21/
uments/5to semestre/algoritmos/tarea_2/hw_02_01.py"
Número 9 encontrado.
Número de comparaciones: 5
PS C:\Users\kit21\Documents\5to semestre\algoritmos>
```

EJEMPLO 2: valor no encontrado

tarea_2 >  hw_02_01.py > ...

2 list_01 = [3,8,4,7,9,5,2] # $T(n) = n$ $S(n) = n$

3

4 valor_A_Buscar = 100 # $T(n) = 1$ $S(n) = 1$

5

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

PUERTOS

PS C:\Users\kit21\Documents\5to semestre\algoritmos> & C:/Users/kit21/Documents/5to semestre/algoritmos/tarea_2/hw_02_01.py"

Número 100 no encontrado.

Número de comparaciones: 7

PS C:\Users\kit21\Documents\5to semestre\algoritmos>