

Universidad de Guadalajara  
Centro Universitario de Ciencias Exactas e Ingenierías

# Búsqueda con GUI

César Alejandro Prieto Franco – 223991675  
Análisis de Algoritmos  
Sección D06

# Desarrollo

El desarrollo de la actividad se divide en dos áreas: los algoritmos implementados y la interfaz gráfica.

## Algoritmos

Se utilizó la librería de numpy para el manejo de arreglos y listas. Esta librería es muy eficiente al trabajar con grandes volúmenes de datos.

Para generar los arreglos, se utilizó la función *randint*, que genera un arreglo de elementos entre un límite inferior y superior.

```
def generate(list_size: int):  
    arr = numpy.random.randint(1, list_size + 1, list_size)  
    return arrD
```

La búsqueda lineal verifica cada elemento en orden para encontrar uno en particular dentro de un arreglo de valores. Si lo encuentra, regresa el índice del arreglo donde se encontró. En caso contrario, regresa un valor de -1.

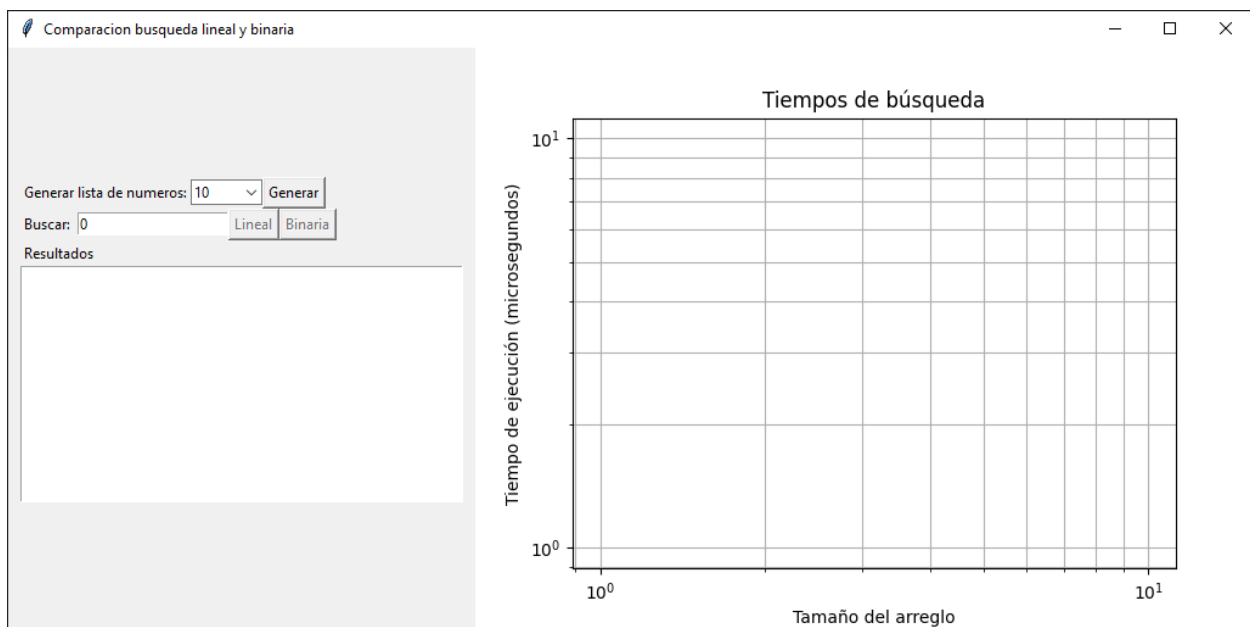
```
def linear_search(target: int, arr):  
    for i, num in enumerate(arr):  
        if num == target:  
            return i  
    return -1
```

La búsqueda binaria trabaja sobre un arreglo ordenado. En cada paso iterativo o recursivo, verifica el valor central de una secuencia del arreglo, y decide si debe buscar en la mitad inferior o superior del rango, o retornar el índice en el que se encuentra si ya lo encontró.

```
def binary_search(target: int, arr):
    L = 0
    R = arr.size - 1
    while L <= R:
        m = L + (R - L) // 2
        if arr[m] < target:
            L = m + 1
        elif arr[m] > target:
            R = m - 1
        else:
            return m
    return -1
```

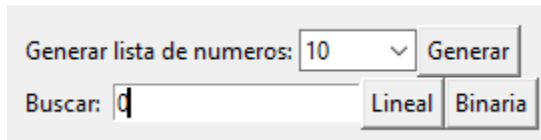
## Interfaz de usuario

En la mitad izquierda se ubican los controles de la aplicación. La primera línea se utiliza para cargar los datos en un arreglo, para luego poder buscarlos. Los tamaños disponibles van desde 10 hasta 100,000 elementos, variando cada opción en un orden de magnitud.



Los botones que realizan las búsquedas inicialmente se encuentran deshabilitados, porque no existen datos que puedan buscarse. Al generar el primer conjunto de datos estos se habilitan. La entrada de la línea de “Buscar” solo permite el ingreso de valores numéricos, ya que las listas están compuestas de enteros positivos.

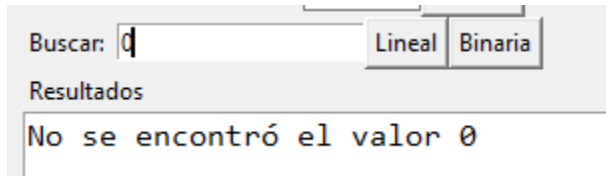
Botones de búsqueda habilitados.



Generar lista de numeros: 10

Buscar: 0

Al buscar un dato que no existe dentro del conjunto de datos, el cuadro de texto inferior muestra que no se encontró el elemento.

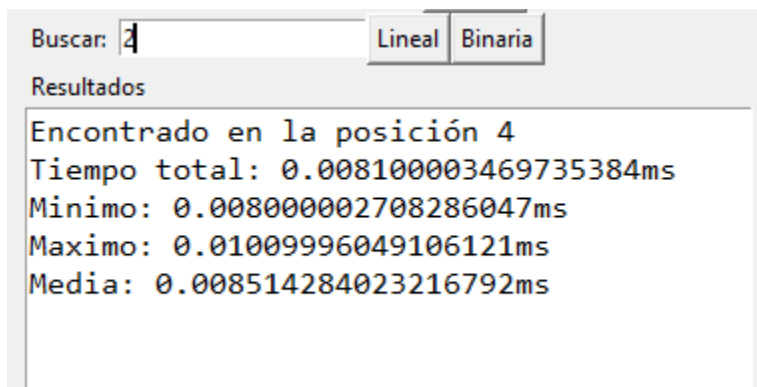


Buscar: 0

Resultados

No se encontró el valor 0

En caso contrario, se muestra el índice en el que se encontró el valor, junto con el tiempo que tomó en realizarse.

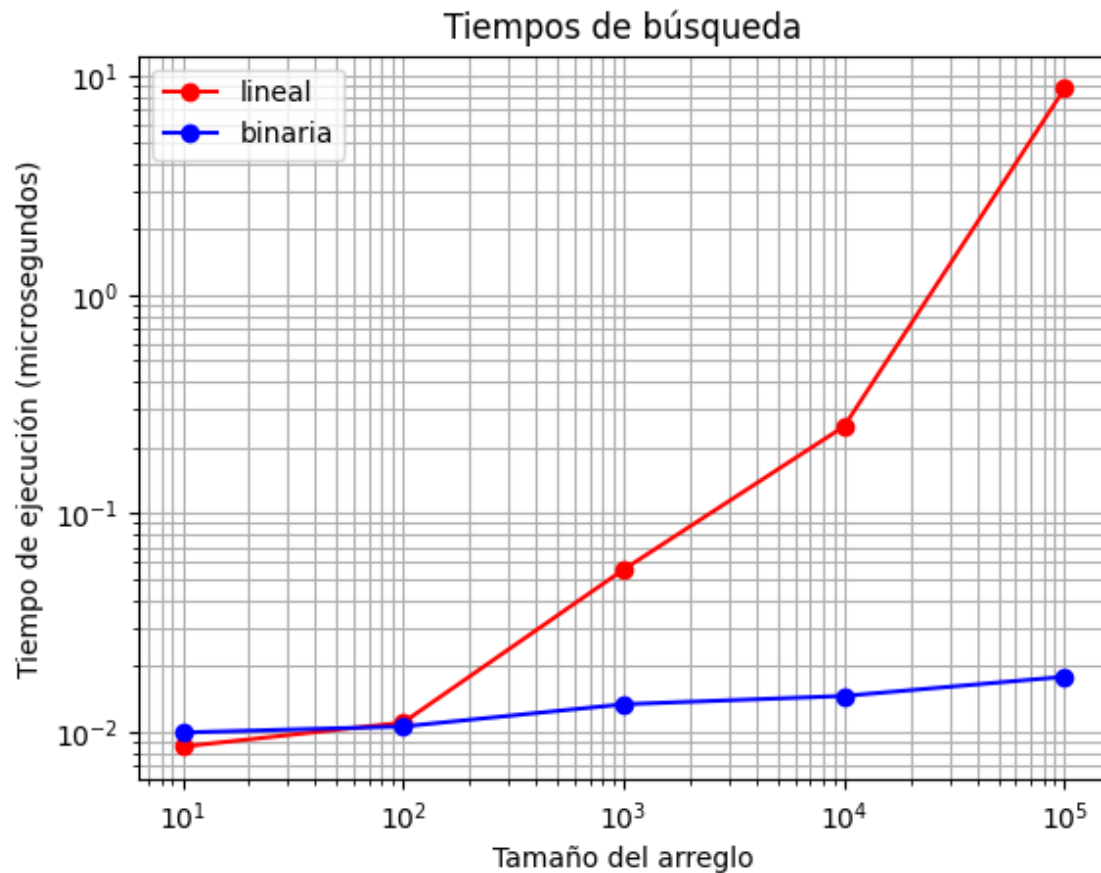


Buscar: 4

Resultados

Encontrado en la posición 4  
Tiempo total: 0.008100003469735384ms  
Minimo: 0.008000002708286047ms  
Maximo: 0.01009996049106121ms  
Media: 0.008514284023216792ms

Al realizar múltiples ejecuciones del programa, se puede observar que la gráfica refleja los tiempos en promedio que tomó cada operación para cada tamaño del arreglo de entrada.



## Conclusión

Con los resultados de esta actividad podemos observar la ventaja que tiene el algoritmo de búsqueda binaria sobre la búsqueda lineal, siempre y cuando opere sobre un conjunto de datos previamente ordenados. La búsqueda lineal presenta un mejor rendimiento en tamaños de entrada reducidos, debido a que la función lineal que sigue crece más lentamente que la función logarítmica que sigue la búsqueda binaria en valores pequeños o cercanos a cero.

## Referencias

tkinter — Python interface to Tcl/Tk. (s. f.). Python Documentation.

<https://docs.python.org/3.13/library/tkinter.html>

tkinter.ttk — Tk themed widgets. (s. f.). Python Documentation.

<https://docs.python.org/3.13/library/tkinter.ttk.html>

NumPy reference — NumPy v2.3 Manual. (s. f.).

<https://numpy.org/doc/stable/reference/index.html>

## Repositorio

[https://github.com/StarsAreDigital/Analisis\\_Algoritmos/](https://github.com/StarsAreDigital/Analisis_Algoritmos/)