Investigación aplicada en Python: Eficiencia y aplicación de la búsqueda binaria

Alumno

Maximo Perrotta

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Marco Teórico – Búsqueda Binaria en Python

1. Introducción

En programación, una de las tareas más comunes es buscar un dato dentro de una colección de elementos. La eficiencia de la búsqueda depende tanto del tipo de estructura de datos como del algoritmo utilizado. En este trabajo se investigan y comparan dos métodos clásicos de búsqueda en listas: **búsqueda lineal** y **búsqueda binaria**.

---

2. Búsqueda lineal

La **búsqueda lineal** es el método más sencillo: recorre la lista **uno por uno** hasta encontrar el valor deseado o hasta llegar al final. No requiere que la lista esté ordenada, pero es **ineficiente para listas largas**, ya que su tiempo de ejecución crece proporcionalmente a la cantidad de elementos (complejidad O(n)).

Ejemplo:

Buscar el número 8 en la lista [1, 3, 5, 8, 10] requerirá 4 pasos (comparaciones).

---

3. Búsqueda binaria

La **búsqueda binaria** es mucho más eficiente, pero **requiere que la lista esté ordenada**. Se basa en dividir la lista en mitades, comparando el valor buscado con el elemento central:

- Si el número buscado es menor, se sigue buscando en la mitad izquierda.
- Si es mayor, en la mitad derecha.
- Si es igual, se ha encontrado el valor.

Este proceso **reduce a la mitad** el espacio de búsqueda en cada paso, con una complejidad de **O(log n)**.

Ejemplo:
Buscar el número 8 en [1, 3, 5, 8, 10]:

1. Compara con 5 → es mayor
2. Busca en [8, 10], compara con 8 → lo encuentra

Solo 2 pasos, contra 4 de la búsqueda lineal.

---

4. Recursividad en búsqueda binaria

Una forma elegante de implementar búsqueda binaria es mediante **recursividad**. La función se llama a sí misma reduciendo el rango de búsqueda hasta encontrar el valor o agotarse las opciones.

Esto permite un código más compacto y claro, aunque requiere entender el flujo de llamadas internas.

---

5. **Código Fuente:**

```
import random

import time

# Función de búsqueda lineal

def busqueda_lineal(lista, objetivo):

    pasos = 0

    for i in range(len(lista)):

        pasos += 1

        if lista[i] == objetivo:

            return i, pasos

    return -1, pasos


# Función de búsqueda binaria iterativa

def busqueda_binaria_iterativa(lista, objetivo):

    inicio = 0

    fin = len(lista) - 1

    pasos = 0

    while inicio <= fin:

        pasos += 1

        medio = (inicio + fin) // 2

        if lista[medio] == objetivo:

            return medio, pasos

        elif lista[medio] < objetivo:

            inicio = medio + 1
```

```
 else:
            fin = medio - 1


    return -1, pasos


# Función de búsqueda binaria recursiva
def busqueda_binaria_recursiva(lista, objetivo, inicio=0, fin=None, pasos=0):
    if fin is None:
        fin = len(lista) - 1

    if inicio > fin:
        return -1, pasos

    pasos += 1
    medio = (inicio + fin) // 2

    if lista[medio] == objetivo:
        return medio, pasos
    elif lista[medio] < objetivo:
        return busqueda_binaria_recursiva(lista, objetivo, medio + 1, fin, pasos)
    else:
        return busqueda_binaria_recursiva(lista, objetivo, inicio, medio - 1, pasos)

# Programa principal
```

```
# Generamos una lista ordenada de 100 elementos

lista = sorted(random.sample(range(1, 200), 100))

print("Lista generada (ordenada):")

print(lista)


# Pedimos al usuario un número para buscar

objetivo = int(input("\nIngrese el número que desea buscar: "))


# Búsqueda lineal

print("\n--- Búsqueda Lineal ---")

indice, pasos = busqueda_lineal(lista, objetivo)

print(f"Resultado: {'Encontrado en índice ' + str(indice) if indice != -1 else 'No encontrado'}")

print(f"Pasos realizados: {pasos}")


# Búsqueda binaria iterativa

print("\n--- Búsqueda Binaria Iterativa ---")

indice, pasos = busqueda_binaria_iterativa(lista, objetivo)

print(f"Resultado: {'Encontrado en índice ' + str(indice) if indice != -1 else 'No encontrado'}")

print(f"Pasos realizados: {pasos}")


# Búsqueda binaria recursiva

print("\n--- Búsqueda Binaria Recursiva ---")
```

```
indice, pasos = busqueda_binaria_recursiva(lista, objetivo)


print(f"Resultado: {'Encontrado en índice ' + str(indice) if indice != -1 else 'No encontrado'}")

print(f"Pasos realizados: {pasos}")
```

---

6. **Resultados del código:**

```
Maxgard@DESKTOP-7IFHRL1 MINGW64 /d/FacuUTN/PrimerCuatrimestre/Programacion/IntegradorFinal

$ C:/Users/Maxgard/AppData/Local/Microsoft/WindowsApps/python3.13.exe d:/FacuUTN/PrimerCuatrimestre/Programacion/IntegradorFinal/busquedas.py

Lista generada (ordenada):

[2, 3, 4, 5, 7, 9, 12, 15, 16, 17, 20, 21, 23, 24, 25, 26, 28, 29, 30, 32, 37, 38, 39, 41, 42, 43, 46, 48, 50, 53, 54, 55, 56, 58, 61, 63, 64, 65, 66, 67, 70, 71, 73, 74, 75, 78, 79, 86, 87, 92, 93, 96, 97, 99, 102, 105, 107, 109, 110, 114, 117, 118, 120, 121, 125, 126, 127, 128, 129, 131, 132, 134, 135, 137, 141, 145, 147, 150, 151, 153, 155, 157, 158, 160, 162, 164, 165, 166, 168, 169, 171, 174, 177, 180, 182, 184, 186, 190, 191, 199]


Ingrese el número que desea buscar: 199

--- Búsqueda Lineal ---

Resultado: Encontrado en índice 99

Pasos realizados: 100

--- Búsqueda Binaria Iterativa ---

Resultado: Encontrado en índice 99

Pasos realizados: 7

--- Búsqueda Binaria Recursiva ---

Resultado: Encontrado en índice 99

Pasos realizados: 7
```

7. Comparación entre métodos

Criterio	Búsqueda lineal	Búsqueda binaria
Lista ordenada	No necesaria	Necesaria
Eficiencia promedio	Baja (O(n))	Alta (O(log n))
Implementación	Muy sencilla	Requiere más lógica
Uso de recursividad	No	Opcional

---

8. Conclusiones teóricas

La búsqueda binaria es claramente más eficiente que la búsqueda lineal en listas grandes, siempre que estén ordenadas. Su implementación puede hacerse de forma iterativa o recursiva, y representa una excelente oportunidad para aplicar conceptos de **funciones**, **listas**, **condicionales**, **ciclos** y **recursividad**.

---

9. Fuentes consultadas

- Python official docs: https://docs.python.org/3/
- GeeksforGeeks – Binary Search: https://www.geeksforgeeks.org/binary-search/
- W3Schools – Python Search Algorithms: https://www.w3schools.com/python/python_lists.asp
- Cormen, Thomas et al. "Introduction to Algorithms", MIT Press, 2009.

---

10. Repositorio Github:

https://github.com/MaximoPerrotta/busqueda-binaria-python

---

11. Link Video:

https://youtu.be/j1W3e-x39hE