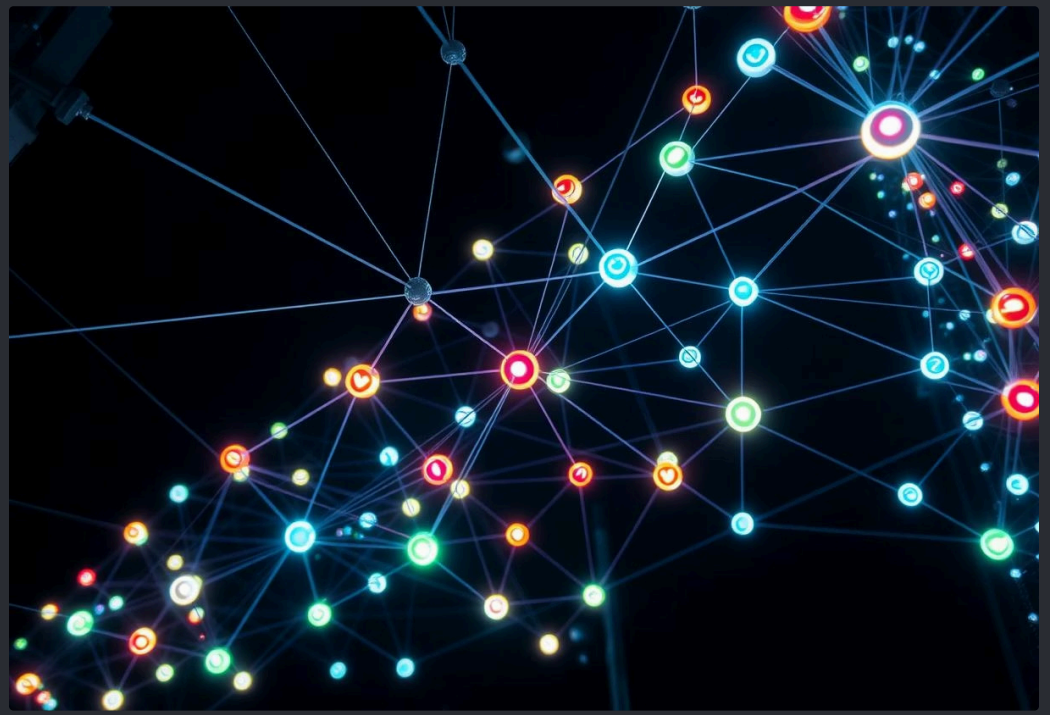
Integrador Programación I

Integrante: Prados Gonzalo

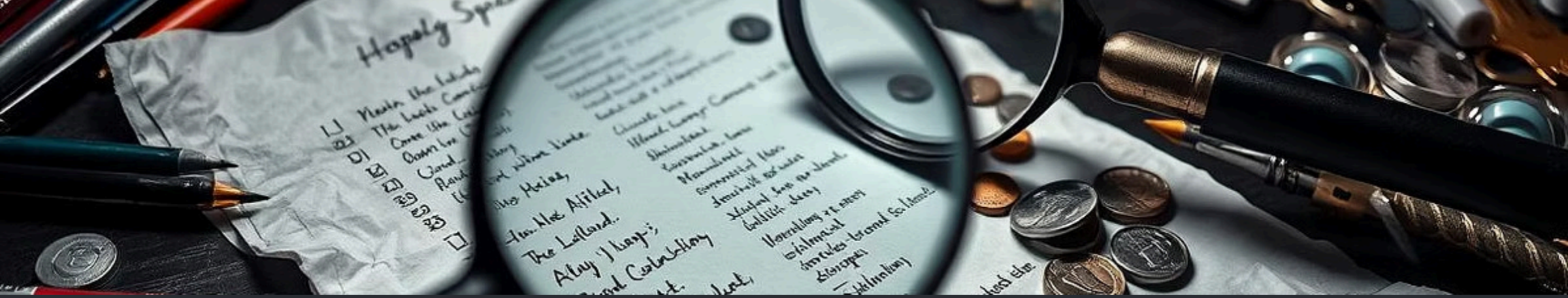
Introducción a los Algoritmos de Búsqueda y Ordenamiento



Este trabajo es un primer acercamiento a los algoritmos de búsqueda y ordenamiento. Exploraremos su funcionamiento, efectividad y casos de uso.

Una búsqueda efectiva optimiza el tiempo de cómputo y mejora los tiempos de respuesta.

Nos centraremos en la búsqueda lineal y binaria, que establecen las bases conceptuales.



Búsqueda Lineal: Conceptos Básicos

Recorrido Secuencial

Implica revisar cada elemento uno por uno hasta encontrar el objetivo. Es sencillo de implementar.

Ventajas

Su principal ventaja es su sencillez y su aplicación a cualquier tipo de lista.

Desventajas

Ineficiente en listas grandes y no es óptima para listas ordenadas.

Búsqueda Binaria: Un Enfoque Eficiente

Funcionamiento

Divide repetidamente la lista ordenada a la mitad para encontrar el elemento.

Eficiencia

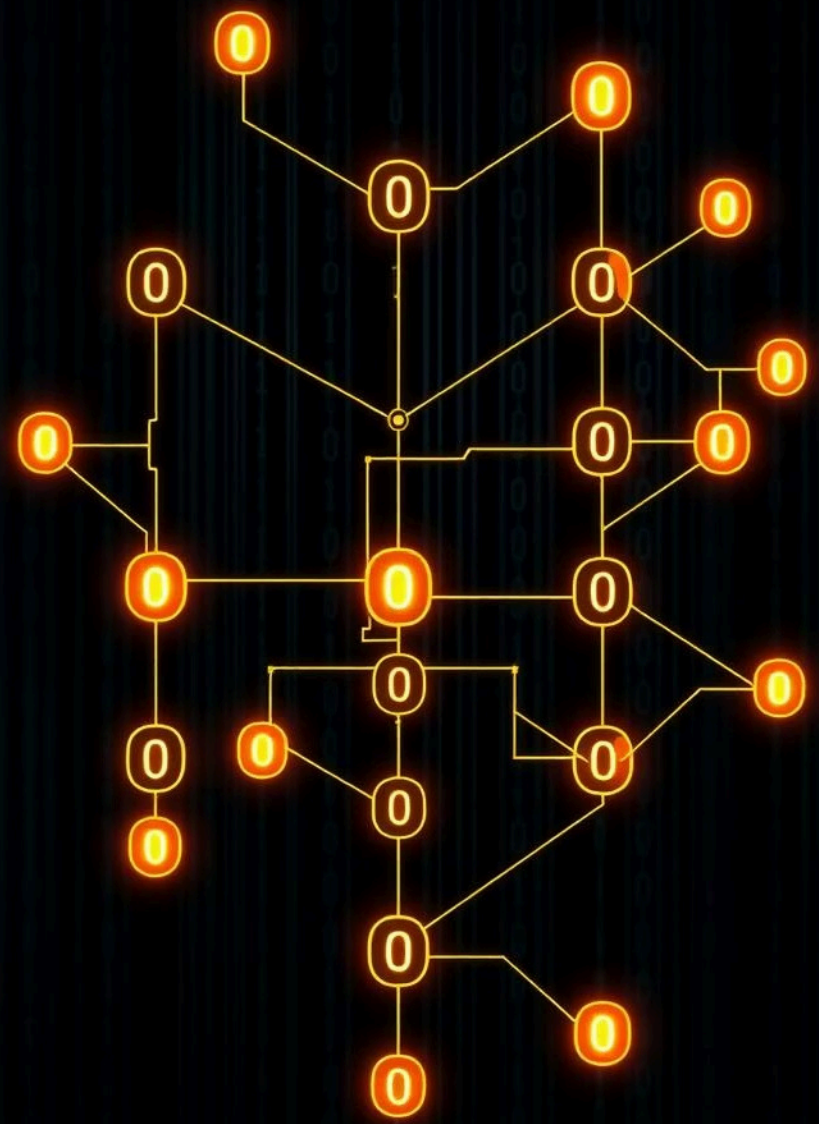
Su tiempo de ejecución es O(log n), muy rápido para listas grandes.

Menos Comparaciones

Realiza significativamente menos comparaciones que la búsqueda lineal.

Requisitos

Requiere una lista previamente ordenada, lo que puede implicar un costo.



Caso Práctico: Escenario 1

Comparamos ambos algoritmos en una lista ordenada de cien mil elementos buscando el numero 4900, midiendo el tiempo de ejecución.

Búsqueda Lineal

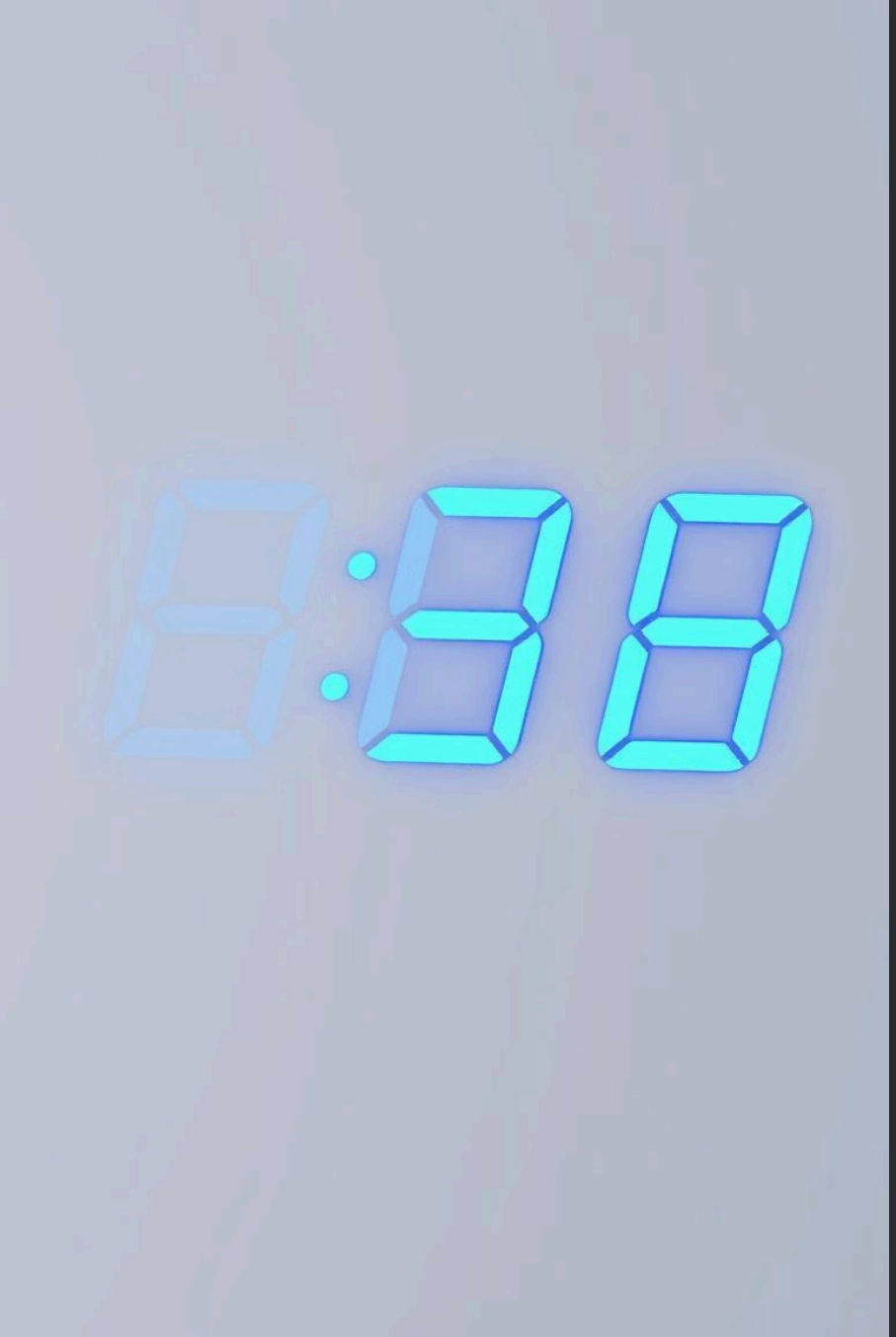
```
def busquedaLineal(lista, obj): for i in range(len(lista)): if lista[i] == obj: return i return -1
```

Tiempo: 0.61962 ms

Búsqueda Binaria

```
def busquedaBinaria(lista, objetivo, inicio, fin): if inicio > fin: return -1 centro = (inicio + fin) // 2 if lista[centro] == objetivo: return centro elif lista[centro] < objetivo: return busquedaBinaria(lista, objetivo, centro + 1, fin) else: return busquedaBinaria(lista, objetivo, inicio, centro - 1)
```

Tiempo: 0.02377 ms



Caso Práctico: Escenario 2

0.00678 ms

Búsqueda Lineal

0.02514 ms

Búsqueda Binaria

En este escenario buscamos el numero 13 que está casi al comienzo de la lista, la búsqueda lineal fue más rápida. Esto se debe a que el elemento deseado se encuentra al principio de la lista, reduciendo las iteraciones.

Caso Práctico: Escenario 3

Búsqueda Lineal	14.45768
Búsqueda Binaria	0.06502

Buscando el número 100000 está al final de la lista. La búsqueda lineal tuvo que recorrer toda la lista.

La diferencia de rendimiento es drástica; la búsqueda binaria fue más de 200 veces más rápida.

Metodología Utilizada

Investigación

Fuentes como 4Geeks, freeCodeCamp y TutorialsPoint.

Pruebas Empíricas

Algoritmos de prueba para verificar tiempos de cómputo.

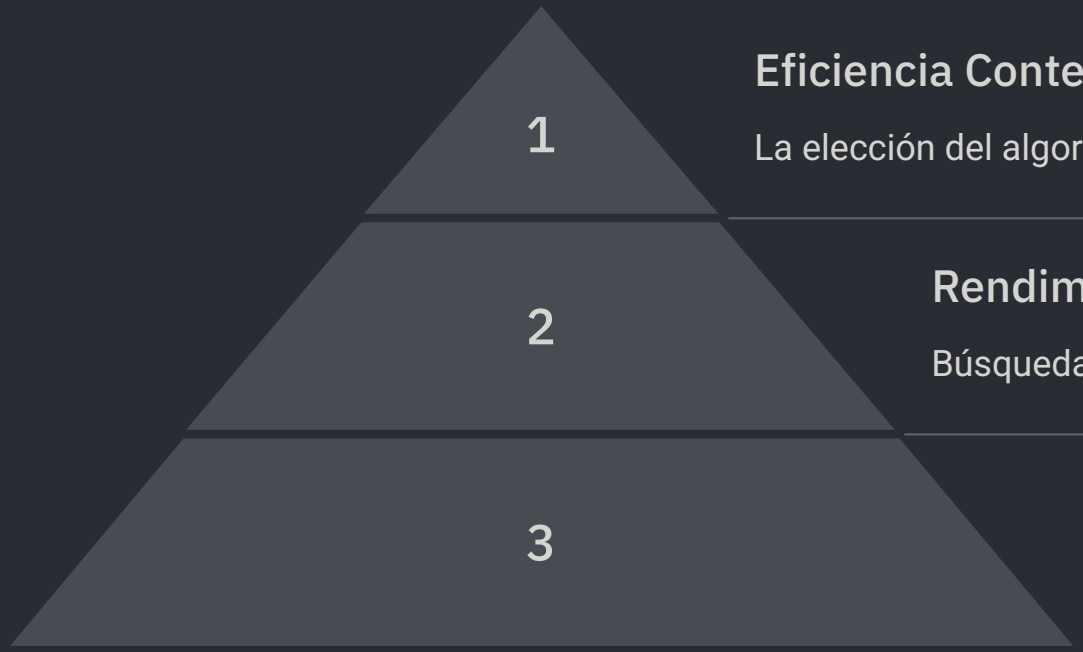
Herramientas

Visual Studio Code, GitHub Copilot, GitHub, Gamma.

Medición

Módulo 'time' para tiempos de ejecución y 'sort' para ordenar.

Análisis de Resultados



Eficiencia Contextual

La elección del algoritmo depende del escenario específico.

Rendimiento en Listas Grandes

Búsqueda Binaria es superior para grandes volúmenes.

Sencillez vs. Complejidad

Lineal es simple; Binaria, más compleja de implementar.

La búsqueda lineal es simple, pero menos eficiente en listas grandes.

La búsqueda binaria, aunque más compleja, es mucho más rápida en listas ordenadas y extensas.

Conclusión Final y Próximos Pasos

La búsqueda lineal es útil para listas pequeñas y desordenadas. Sin embargo, su rendimiento decrece con el tamaño de la lista.

La búsqueda binaria es más eficiente en listas ordenadas, reduciendo el tiempo de búsqueda con la estrategia "divide y conquistarás".

Ambos algoritmos son herramientas valiosas en el repertorio de todo desarrollador, dependiendo del contexto.

Identificar Contexto

Evaluar tamaño y orden de la lista.

Elegir Algoritmo

Seleccionar el más adecuado para el escenario.

Optimizar

Considerar costos de ordenamiento y complejidad.