UNIVERSIDAD NACIONAL GUILLERMO BROWN

Tecnicatura Universitaria en

PROGRAMACIÓN

Algoritmos de Búsqueda

Algoritmos de búsqueda

Búsqueda Lineal

```python
def busqueda_lineal(lista, elemento):
    for i in range(len(lista)):
        if lista[i] == elemento:
            return i
    return -1
```

La búsqueda lineal es el algoritmo más simple y básico.

Recorre la lista de elementos uno por uno, comparando cada elemento con el valor buscado.

Si encuentra el elemento, devuelve su posición. Si llega al final sin encontrarlo, devuelve -1.

Tiene una complejidad temporal O(n) en el peor caso.

Búsqueda Lineal

```python
def busqueda_lineal(lista, elemento):
    for i in range(len(lista)):
        if lista[i] == elemento:
            return i
    return -1
```

La búsqueda lineal es el algoritmo más simple y básico.

Recorre la lista de elementos uno por uno, comparando cada elemento con el valor buscado.

Si encuentra el elemento, devuelve su posición. Si llega al final sin encontrarlo, devuelve -1.

Tiene una complejidad temporal O(n) en el peor caso.

Búsqueda Binaria

```python
def busqueda_binaria(lista, elemento):
    inicio = 0
    fin = len(lista) - 1
    while inicio <= fin:
        medio = (inicio + fin) // 2
        if lista[medio] == elemento:
            return medio
        elif lista[medio] < elemento:
            inicio = medio + 1
        else:
            fin = medio - 1
    return -1
```

La búsqueda binaria es más eficiente que la lineal.

Requiere que la lista esté ordenada previamente.

Compara el elemento buscado con el elemento del medio de la lista.

Si es menor, busca en la mitad inferior. Si es mayor, busca en la mitad superior.

Divide repetidamente la lista hasta encontrar el elemento o hasta que no queden más elementos.

Tiene una complejidad temporal O(log n) en el peor caso.

Búsqueda Binaria

```python
def busqueda_binaria(lista, elemento):
    inicio = 0
    fin = len(lista) - 1
    while inicio <= fin:
        medio = (inicio + fin) // 2
        if lista[medio] == elemento:
            return medio
        elif lista[medio] < elemento:
            inicio = medio + 1
        else:
            fin = medio - 1
    return -1
```

La búsqueda binaria es más eficiente que la lineal.

Requiere que la lista esté ordenada previamente.

Compara el elemento buscado con el elemento del medio de la lista.

Si es menor, busca en la mitad inferior. Si es mayor, busca en la mitad superior.

Divide repetidamente la lista hasta encontrar el elemento o hasta que no queden más elementos.

Tiene una complejidad temporal O(log n) en el peor caso.

Búsqueda por Saltos (Jump Search)

```python
import math


def busqueda_por_saltos(lista, elemento):
    n = len(lista)
    paso = int(math.floor(math.sqrt(n)))
    bloque_anterior = 0
    while lista[min(paso, n) - 1] < elemento:
        bloque_anterior = paso
        paso += int(math.floor(math.sqrt(n)))
        if bloque_anterior >= n:
            return -1
    while lista[bloque_anterior] < elemento:
        bloque_anterior += 1
        if bloque_anterior == min(paso, n):
            return -1
    if lista[bloque_anterior] == elemento:
        return bloque_anterior
    return -1

```

.Combina características de la búsqueda lineal y binaria.

Divide la lista en bloques de tamaño √n.

Busca en qué bloque puede estar el elemento y luego hace una búsqueda lineal dentro de ese bloque.

Tiene una complejidad temporal O(√n) en el peor caso

¿PREGUNTAS?