Busca Binária:

Implemente um algoritmo eficiente para realizar a ordenação:

A busca binária é um algoritmo de busca eficiente para encontrar um elemento em uma lista ordenada. Ele divide repetidamente a lista ao meio e compara o elemento do meio com o elemento de destino. Se forem iguais, o elemento é encontrado. Se o elemento do meio for menor que o elemento de destino, a busca continua na metade superior da lista; caso contrário, continua na metade inferior. Esse processo é repetido até que o elemento seja encontrado ou a lista seja esgotada. A busca binária possui complexidade de tempo O(log n), tornando-a muito eficiente para grandes conjuntos de dados ordenados.

```python
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

arr = [11, 12, 21, 62, 77, 7, 22, 25, 34, 64, 90]
target = 7
print("Lista ordenada:", arr)
print("O índice de", target, "é:", binary_search(arr, target))
```

Execução:

```
.exe c:/Users/migue/teste/buscaBi.py
Lista ordenada: [11, 12, 21, 62, 77, 7, 22, 25, 34, 64, 90]
O índice de 7 é: 5
PS C:\Users\migue\teste>
```

Documente o algoritmo utilizado e forneça uma análise da sua complexidade temporal:

A busca binária é um algoritmo de busca eficiente para encontrar a posição de um elemento em uma lista ordenada. Ela funciona dividindo repetidamente a lista ao meio e comparando o elemento do meio com o valor buscado, reduzindo assim o espaço de busca pela metade a cada passo.

Complexidade Temporal

- Melhor caso: O(1) quando o elemento está no meio da lista na primeira tentativa.
- Caso médio: O(log n).
- Pior caso: O(log n).