Insertion Sort:

Implemente um algoritmo eficiente para realizar a ordenação:

O Insertion Sort é um algoritmo de ordenação eficiente para conjuntos de dados pequenos ou quase ordenados. Ele constrói uma lista ordenada uma posição por vez, movendo cada elemento para sua posição correta na sublista ordenada. O Insertion Sort é estável e tem complexidade quadrática, mas é mais eficiente que o Bubble Sort e o Selection Sort para conjuntos de dados pequenos.

```python
def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

arr = [64, 34, 25, 12, 122, 532, 22, 11, 90]
print("Antes da ordenação:", arr)
print("Após Insertion Sort:", insertion_sort(arr.copy()))
```

Execução:

```
.exe c:/Users/migue/teste/insertion.py
Antes da ordenação: [64, 34, 25, 12, 122, 532, 22, 11, 90]
Após Insertion Sort: [11, 12, 22, 25, 34, 64, 90, 122, 532]
PS C:\Users\migue\teste>
```

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

Insertion Sort constrói a lista ordenada um elemento de cada vez, inserindo cada novo elemento na posição correta na sub-lista ordenada.

Complexidade Temporal:

- Melhor caso: O(n) quando a lista já está ordenada.
- Caso médio: O(n^2).
- Pior caso: O(n^2) quando a lista está ordenada na ordem inversa.