Selection Sort:

Implemente um algoritmo eficiente para realizar a ordenação:

O Selection Sort é um algoritmo de ordenação que divide a lista em duas partes: uma sublista ordenada e uma sublista não ordenada. Ele seleciona repetidamente o menor (ou maior, dependendo da ordem desejada) elemento da sublista não ordenada e o move para o final da sublista ordenada. Esse processo é repetido até que toda a lista esteja ordenada. Apesar de simples, o Selection Sort também possui complexidade quadrática, tornando-o ineficiente para grandes conjuntos de dados.

```python
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_index = i
        for j in range(i+1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]
    return arr

arr = [64, 34, 25, 12, 22, 11, 90, 41, 64, 102, 999]
print("Antes da ordenação:", arr)
print("Após Selection Sort:", selection_sort(arr.copy()))
```

Execução:

```
.exe c:/Users/migue/teste/selection.py
Antes da ordenação: [64, 34, 25, 12, 22, 11, 90, 41, 64, 102, 999]
Após Selection Sort: [11, 12, 22, 25, 34, 41, 64, 64, 90, 102, 999]
PS C:\Users\migue\teste>
```

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

Selection Sort divide a lista em uma sub-lista ordenada e uma sub-lista não ordenada. Em cada iteração, encontra o menor elemento da sub-lista não ordenada e troca-o com o primeiro elemento da sub-lista não ordenada. O Selection Sort sempre tem a mesma complexidade temporal independente da ordem dos elementos na lista.

Complexidade Temporal:

- Melhor caso: O(n^2).
- Caso médio: O(n^2).
- Pior caso: O(n^2).