

Nama: Nurul Rahmadani H Suryanto

Nim: F55121065

Kelas: B

A. Program Bubble Sort

```
1 def bubble_sort(arr):
2     n = len(arr)
3
4     for i in range(n - 1):
5         for j in range(n - i - 1):
6             if arr[j] > arr[j + 1]:
7                 arr[j], arr[j + 1] = arr[j + 1], arr[j]
8
9     return arr
10
11
12 # Contoh penggunaan
13 array = [64, 34, 25, 12, 22, 11, 90]
14 sorted_array = bubble_sort(array)
15 print("Array setelah diurutkan menggunakan Bubble Sort:")
16 print(sorted_array)
```

Run bubblesort x

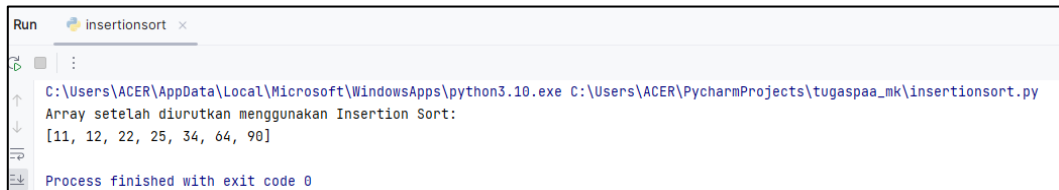
C:\Users\ACER\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\ACER\PycharmProjects\tugaspa_mk\bubblesort.py

Array setelah diurutkan menggunakan Bubble Sort:

[11, 12, 22, 25, 34, 64, 90]

B. Program Insertion Sort

```
def insertion_sort(arr):  
    n = len(arr)  
  
    for i in range(1, n):  
        key = arr[i]  
        j = i - 1  
  
        while j >= 0 and arr[j] > key:  
            arr[j + 1] = arr[j]  
            j -= 1  
  
        arr[j + 1] = key  
  
    return arr  
  
# Contoh penggunaan  
array = [64, 34, 25, 12, 22, 11, 90]  
sorted_array = insertion_sort(array)  
print("Array setelah diurutkan menggunakan Insertion Sort:")  
print(sorted_array)
```



```
Run insertionsort x  
C:\Users\ACER\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\ACER\PycharmProjects\tugaspaa_mk\insertionsort.py  
Array setelah diurutkan menggunakan Insertion Sort:  
[11, 12, 22, 25, 34, 64, 90]  
Process finished with exit code 0
```

C. Antara Bubble Sort dan Insertion Sort memiliki kompleksitas waktu yang sama pada kasus rata-rata yaitu $O(n^2)$. Namun, secara umum Insertion Sort cenderung sedikit lebih cepat daripada Bubble Sort.

Perbedaan kinerja antara kedua algoritma tergantung pada sifat data yang diurutkan. Jika array hampir terurut atau memiliki sejumlah kecil elemen yang tidak terurut, Insertion Sort akan lebih cepat karena memanfaatkan sifat data hampir terurut tersebut. Sedangkan Bubble Sort akan tetap memerlukan iterasi sebanyak yang diperlukan untuk menukar elemen di tempat yang benar.

Meskipun begitu, baik Bubble Sort maupun Insertion Sort tidak efisien untuk jumlah data yang sangat besar. Untuk kasus-kasus di mana efisiensi waktu sangat penting, algoritma pengurutan seperti Merge Sort, Quick Sort, atau Heap Sort biasanya lebih disarankan karena memiliki kompleksitas waktu yang lebih baik seperti $O(n \log n)$.