PYTHON

SOLUSI STUDI KASUS

lembar solusi ini sebagai salah satu tugas dalam
mata kuliah Dasar Pemrograman

dosen pengampu
Indira Syawanodya, M.Kom.
Yulia Retnowati, S.Pd., M.T.

disusun oleh
Andika Eka Kurnia 2306033/1A RPL

PROGRAM STUDI REKAYASA PERANGKAT LUNAK
KAMPUS UPI DI CIBIRU
UNIVERSITAS PENDIDIKAN INDONESIA
2023

File dnsort.py

```
def bubble_sort(data):
    for i in range(len(data) - 1, 0, -1):
        for j in range(i):
            if data[j] > data[j + 1]:
                data[j], data[j + 1] = data[j + 1], data[j]
def selection_sort(data):
    for i in range(len(data) - 1, 0, -1):
        pos_max = 0
        for j in range(1, i + 1):
            if data[j] > data[pos_max]:
                pos_max = j
        data[i], data[pos_max] = data[pos_max], data[i]
def heap_sort(arr):
    def max_heap(arr, n, i):
        index_parent = i
        index_kiri = 2 * i + 1
        index_kanan = 2 * i + 2

        if index_kiri < n and arr[index_parent] <
arr[index_kiri]:
            index_parent = index_kiri

        if index_kanan < n and arr[index_parent] <
arr[index_kanan]:
            index_parent = index_kanan

        if index_parent != i:
            arr[i], arr[index_parent] = arr[index_parent],
arr[i]
            max_heap(arr, n, index_parent)

    n = len(arr)
```

```
    for i in range(n // 2 - 1, -1, -1):
        max_heap(arr, n, i)

    for i in range(n - 1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        max_heap(arr, i, 0)
```

File main.py

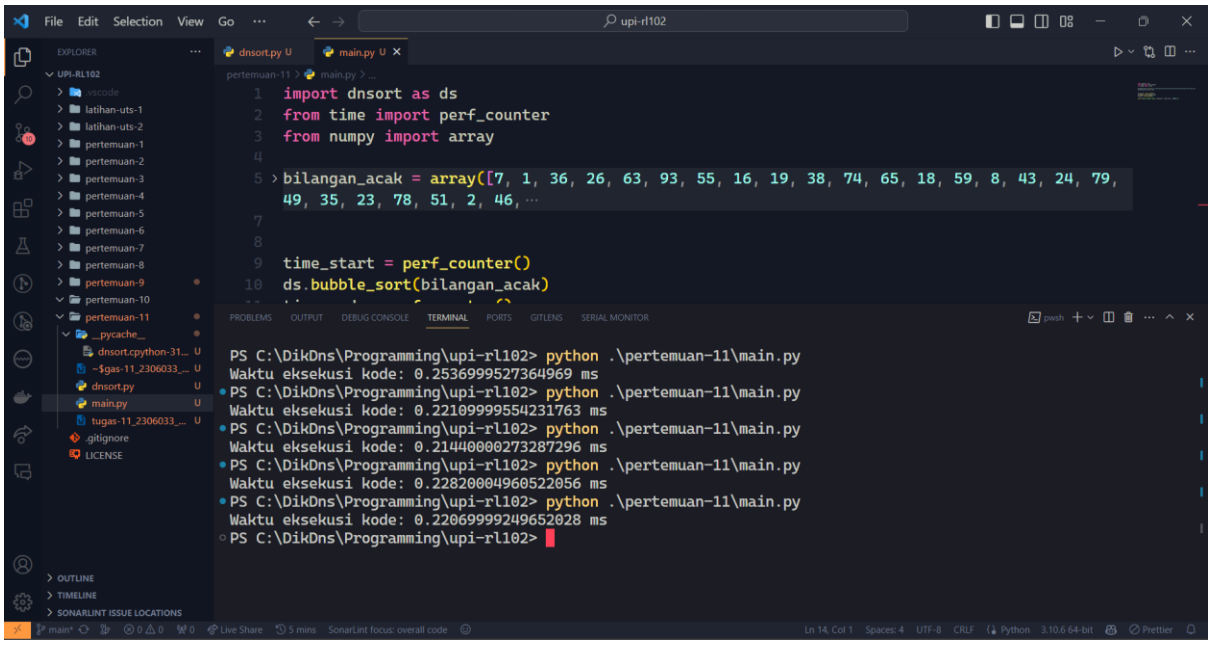
```
import dnsort as ds
from time import perf_counter
from numpy import array

bilangan_acak = array([7, 1, 36, 26, 63, 93, 55, 16, 19, 38, 74,
65, 18, 59, 8, 43, 24, 79, 49, 35, 23, 78, 51, 2, 46, 28, 60,
76, 10, 85, 66, 29, 82, 58, 69, 75, 48, 100, 5, 32, 40, 33, 34,
90, 81, 42, 57, 44, 41, 77])

time_start = perf_counter()
ds.bubble_sort(bilangan_acak)
time_end = perf_counter()

print(f"Waktu eksekusi kode: {(time_end - time_start) * 1000}
ms")
```

Terminal main.py



Tabel Data Percobaan

Jenis Sort	Percobaan (ms)					Rata-rata
	Ke-1	Ke-2	Ke-3	Ke-4	Ke-5	
Bubble	0.25810	0.21100	0.21180	0.21840	0.22210	0,22
Selection	0.13160	0.12810	0.13800	0.13940	0.13560	0,13
Heap	0.12450	0.13040	0.11560	0.11450	0.11450	0,12

Berdasarkan analisis data percobaan yang terdapat dalam tabel, dapat diambil kesimpulan bahwa Heap Sort menonjol sebagai jenis algoritma pengurutan yang paling efisien dan cepat di antara yang lain.