PRACTICUM REPORT
ALGORITHM AND DATA STRUCTURES
MODUL 5 : SORTING

Disusun Oleh :

ONIC AGUSTINO

L200234275

X

INFORMATICS ENGINEERING

FACULTY OF COMMUNICATION AND INFORMATICS

UNIVERSITAS MUHAMMADIYAH SURAKARTA

YEARS 2024/2025

1.11 Questions

1. Create a program to sort an array of students based on NIM, whose elements are made from the MhsTIF class, which you created earlier.

- **Program Code**

Module_5 > Task > task.py

```python
# 1. Buatlah suatu program untuk mengurutkan array mahasiswa berdasarkan NIM, yang elemennya terbuat dari class MhsTIF, yang telah kamu buat sebelumnya.
class MhsTIF:
    def __init__(self, nim, nama, prodi):
        self.nim = nim
        self.nama = nama
        self.prodi = prodi

    def __str__(self):
        return f"NIM: {self.nim}, Nama: {self.nama}, Prodi: {self.prodi}"

    def __repr__(self):
        return f"MhsTIF({self.nim}, '{self.nama}', '{self.prodi}')"

def urutkan_mahasiswa(mahasiswa_list):
    return sorted(mahasiswa_list, key=lambda m: m.nim)

mahasiswa_list = [
    MhsTIF("L200234230", "Andi", "TI"),
    MhsTIF("L200234291", "Budi", "TI"),
    MhsTIF("L200234275", "Tino", "TI"),
    MhsTIF("L200234276", "Budi", "TI"),
    MhsTIF("L200234277", "Siti", "TI")
]
print("Sebelum diurutkan:", mahasiswa_list)
mahasiswa_list = urutkan_mahasiswa(mahasiswa_list)
print("Setelah diurutkan:", mahasiswa_list)
```

Code 1.1 the code.

This program creates a class called MhsTIF to represent student data, including attributes like student ID (NIM), name, and study program. Then, a function urutkan_mahasiswa() is defined, using the sorted() method to sort the student list based on their NIM. Initially, a list of students is created using multiple MhsTIF objects, each containing their respective NIM, name, and study program. Before sorting, the student list is displayed using the print() function. Once sorting is completed, the updated and sorted student list is shown again.

- **Practicum Result Screenshot**

```
PS D:\Semester 4\PrakAl_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAl_and_StrDat/Module_5/Task/task.py"
Sebelum diurutkan: [MhsTIF(L200234230, 'Andi', 'TI'), MhsTIF(L200234291, 'Budi', 'TI'), MhsTIF(L200234275, 'Tino', 'TI'), MhsTIF(L200234276, 'Budi', 'TI'), MhsTIF(L200234277, 'Siti'
, 'TI')]
Setelah diurutkan: [MhsTIF(L200234230, 'Andi', 'TI'), MhsTIF(L200234275, 'Tino', 'TI'), MhsTIF(L200234276, 'Budi', 'TI'), MhsTIF(L200234277, 'Siti', 'TI'), MhsTIF(L200234291, 'Budi'
, 'TI')]
```

Gambar 1.2 the results

2. For example, there are two sorted arrays A and B. Create a program to efficiently combine the two arrays into a sorted array C.

- **Program Code**

```python
# Misal terdapat dua buah array yang sudah urut A dan B. Buatlah suatu program untukmenggabungkan, secara efisien, kedua array itu menjadi suatu array C yang urut.
def gabungkan_array_urut(A, B):
    C = []
    i = j = 0
    while i < len(A) and j < len(B):
        if A[i] < B[j]:
            C.append(A[i])
            i += 1
        else:
            C.append(B[j])
            j += 1
    C.extend(A[i:])
    C.extend(B[j:])
    return C

# Contoh penggunaan
A = [1, 3, 5, 7]
B = [2, 4, 6, 8]
print("Array A:", A)
print("Array B:", B)
C = gabungkan_array_urut(A, B)
print("Array C (gabungan):", C)
```

2.1 the code.

This program efficiently merges two sorted arrays, A and B, into a single sorted array, C. It initializes an empty list C and uses two pointers, i and j, to traverse A and B while comparing elements one by one. The smaller element is appended to C, and the corresponding pointer is incremented until one of the lists is fully traversed. The remaining elements of either A or B are then added to C to complete the merging process. Finally, the merged and sorted array C is displayed.

- **Practicum Result Screenshot**

```
Array A: [1, 3, 5, 7]
Array B: [2, 4, 6, 8]
Array C (gabungan): [1, 2, 3, 4, 5, 6, 7, 8]
```

2.2 the result

3. You might have guessed that bubble sort is slower than selection sort and also insertion sort. But which is faster, selection sort or insertion sort?7
   To start investigating, you can compare the time it takes to sort a large array, say 6000 elements long.

- **Program Code**

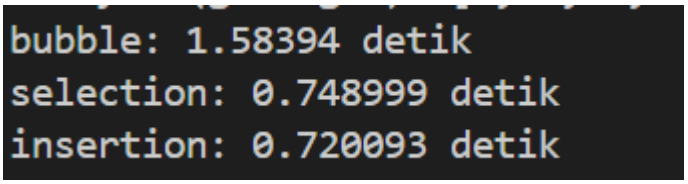
```
53  # 3. Membandingkan waktu bubble sort, selection sort, dan insertion sort
54  def bubbleSort(arr):
55      n = len(arr)
56      for i in range(n):
57          for j in range(0, n-i-1):
58              if arr[j] > arr[j+1]:
59                  arr[j], arr[j+1] = arr[j+1], arr[j]
60
61  def selectionSort(arr):
62      n = len(arr)
63      for i in range(n):
64          min_idx = i
65          for j in range(i+1, n):
66              if arr[j] < arr[min_idx]:
67                  min_idx = j
68          arr[i], arr[min_idx] = arr[min_idx], arr[i]
69
70  def insertionSort(arr):
71      for i in range(1, len(arr)):
72          key = arr[i]
73          j = i - 1
74          while j >= 0 and key < arr[j]:
75              arr[j + 1] = arr[j]
76              j -= 1
77          arr[j + 1] = key
```

```
80  from time import time as detak
81  from random import shuffle as kocok
82  # Membandingkan waktu eksekusi
83  k = list(range(1, 6001))
84  kocok(k)
85  u_bub = k[:]
86  u_sel = k[:]
87  u_ins = k[:]
88
89  aw = detak(); bubbleSort(u_bub); ak = detak(); print('bubble: %g detik' % (ak-aw))
90  aw = detak(); selectionSort(u_sel); ak = detak(); print('selection: %g detik' % (ak-aw))
91  aw = detak(); insertionSort(u_ins); ak = detak(); print('insertion: %g detik' % (ak-aw))
92
```

3.1 the code

This program compares the execution time of three sorting algorithms: ***Bubble Sort, Selection Sort, and Insertion Sort****. Each sorting function processes an input array by applying its respective method to reorder the elements in ascending order. The program generates a shuffled list containing numbers from 1 to 6000, then applies all three sorting algorithms on separate copies of this list. Using Python's time() function, it measures how long each algorithm takes to sort the list. Finally, the execution times are printed for comparison, helping to analyze which sorting algorithm performs faster.*

- **Practicum Results Program**



3.1 the results.