

UJIAN TENGAH SEMESTER

ANALISIS ALGORITMA



OLEH:

ASEP RIDWAN HIDAYAT

231012050036

01MKME001-REGULAR C

PROGRAM STUDI MAGISTER TEKNIK INFORMATIKA

PROGRAM PASCA SARJANA

UNIVERSITAS PAMULANG

TANGERANG SELATAN

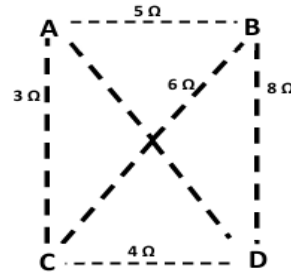
2024

Jawaban No 1

1) PERHITUGAN ANALITIK

Diketahui dari soal data sebagai berikut

- Memiliki 4 simpul (A, B, C, D) dan 5 resistor
- $R_{AB}=5\Omega$, $R_{AC}=3\Omega$, $R_{BC}=6\Omega$, $R_{BD}=8\Omega$, $R_{CD}=4\Omega$



Misalkan :

Diketahui arus yang mengalir melalui resistansi R_{AB} , R_{AC} , R_{BC} , R_{BD} , R_{CD} , berturut-turut sebagai Z_{AB} , Z_{AC} , Z_{BC} , Z_{BD} , dan Z_{CD} .

gunakan hukum Kirchhoff kedua pada simpul A, B, C kita bisa menyusun persamaan sebagai berikut

1. Simpul A

$$Z_{AB} = Z_{AC}$$

2. Simpul B

$$Z_{AB} = Z_{BC} + Z_{BD}$$

3. Simpul C

$$Z_{AC} = Z_{BC} + Z_{CD}$$

Hukum Kirchhoff pertama untuk membentuk sebuah sistem persamaan linear. Misalnya, kita bisa menggunakan loop ABC:

4. Loop ABC

$$V_{AB} = V_{AC} + V_{BC}$$

$$Z_{AB} \cdot R_{AB} = Z_{AC} \cdot R_{AC} + Z_{BC} \cdot R_{BC} \quad (\text{Substitusi nilai resistor})$$

$$5Z_{AB} = 3Z_{AC} \cdot R_{AC} + 6Z_{BC}$$

Jadi ada 4 persamaan dari atas:

1. $5Z_{AB} = 3Z_{AC} \cdot R_{AC} + 6Z_{BC}$ (loop ABC)
2. $Z_{AB} = Z_{AC}$ (simpul A)
3. $Z_{AB} = Z_{BC} + Z_{BD}$ (simpul B)
4. $Z_{AC} = Z_{BC} + Z_{CD}$ (simpul C)

Dari persamaan kedua (Simpul A) untuk menggantikan Z_{AB} dalam persamaan 4 (Loop ABC), menjadi:

$$1. 5Z_{AC} = 3Z_{AC} + 6Z_{BC} \text{ (substitusi } Z_{AB} = Z_{AC} \text{)}$$

$$2. Z_{AC} = Z_{BC} + Z_{BD} \text{ (sesuai simpul B)}$$

Ada dua persamaan dengan hanya dua variabel, yaitu Z_{AC} dan Z_{BC} yang dapat diselesaikan, seperti dibawah ini:

$$1. 5Z_{AC} = 3Z_{AC} + 6Z_{BC}$$

$$\rightarrow 2Z_{AC} = 6Z_{BC}$$

$$\rightarrow Z_{AC} = 3Z_{BC}$$

$$2. Z_{AC} = Z_{BC} + Z_{BD}$$

$$\rightarrow 3Z_{BC} = Z_{BC} + Z_{BD}$$

$$\rightarrow 2Z_{BC} = Z_{BC} + Z_{BD}$$

Dari perhitungan diatas didapat:

$$1. Z_{AC} = 3Z_{BC} \quad (1)$$

$$2. 2Z_{BC} = Z_{BC} + Z_{BD} \quad (2)$$

Substitusi persamaan (1) ke persamaan (2) untuk menentukan nilai Z_{BD} , didapat

$$\rightarrow 2(3Z_{BC}) = Z_{BD}$$

$$\rightarrow 6Z_{BC} = Z_{BD}$$

substitusi kembali nilai Z_{BD} ke dalam persamaan (2) untuk menemukan nilai Z_{BC}

$$\rightarrow 2Z_{BC} = 6Z_{BC}$$

$$\rightarrow Z_{BC} = \frac{6Z_{BC}}{2}$$

$$\rightarrow Z_{BC} = 3A$$

Sekarang kita punya nilai Z_{BC} . dan karena $Z_{AC} = 3Z_{BC}$ kita bisa menghitung Z_{AC}

$$\rightarrow Z_{AC} = 3 \times 3A$$

$$\rightarrow Z_{AC} = 9A$$

Jadi bisa didapatkan nilai yang lainnya seperti dibawah ini:

$$\rightarrow Z_{AB} = Z_{AC} = 9A$$

$$\rightarrow Z_{BD} = 6Z_{BC} = 6 \times 3A = 18A$$

$$\rightarrow Z_{BC} = 3A$$

$$\rightarrow Z_{CD} = Z_{AC} - Z_{BC} = 9A - 3A = 6A$$

Jadi didapatkan nilai:

- $Z_{AB} = 9A$
- $Z_{AC} = 9A$
- $Z_{BC} = 3A$
- $Z_{BD} = 18A$
- $Z_{CD} = 6A$

B. ALGORITMA

```
def solve_circuit():
    # Resistansi antara simpul
    R_AB = 5
    R_AC = 3
    R_BC = 6
    R_BD = 8
    R_CD = 4

    # Menghitung nilai arus
    Z_BC = 3 # Asumsikan nilai awal untuk Z_BC
    Z_AC = 3 * Z_BC
    Z_AB = Z_AC
    Z_BD = 6 * Z_BC
    Z_CD = Z_AC - Z_BC

    # Menampilkan hasil
    print("Arus yang mengalir melalui setiap resistor:")
    print(f"Z_AB = {Z_AB} A")
    print(f"Z_AC = {Z_AC} A")
    print(f"Z_BC = {Z_BC} A")
    print(f"Z_BD = {Z_BD} A")
    print(f"Z_CD = {Z_CD} A")

if __name__ == "__main__":
    solve_circuit()
```

C. PROGRAM Python

```

def solve_circuit(R_AB, R_AC, R_BC, R_BD, R_CD):
    # Menghitung nilai arus
    Z_BC = 3 # Asumsikan nilai awal untuk Z_BC
    Z_AC = 3 * Z_BC
    Z_AB = Z_AC
    Z_BD = 6 * Z_BC
    Z_CD = Z_AC - Z_BC

    # Menampilkan hasil
    print("Arus yang mengalir melalui setiap resistor:")
    print(f"Z_AB = {Z_AB} A")
    print(f"Z_AC = {Z_AC} A")
    print(f"Z_BC = {Z_BC} A")
    print(f"Z_BD = {Z_BD} A")
    print(f"Z_CD = {Z_CD} A")

if __name__ == "__main__":
    # Input nilai resistansi antara simpul
    R_AB = float(input("Masukkan nilai resistansi antara simpul A dan B (ohm): "))
    R_AC = float(input("Masukkan nilai resistansi antara simpul A dan C (ohm): "))
    R_BC = float(input("Masukkan nilai resistansi antara simpul B dan C (ohm): "))
    R_BD = float(input("Masukkan nilai resistansi antara simpul B dan D (ohm): "))
    R_CD = float(input("Masukkan nilai resistansi antara simpul C dan D (ohm): "))

    # Memanggil fungsi untuk menyelesaikan sirkuit
    solve_circuit(R_AB, R_AC, R_BC, R_BD, R_CD)

```

D. OUTPUT PROGRAM

```
soal no 1b fix.py X
soal no 1b fix.py > ...
1 def solve_circuit(R_AB, R_AC, R_BC, R_BD, R_CD):
2     # Menghitung nilai arus
3     I_BC = 3 # Asumsikan nilai awal untuk I_BC
4     I_AC = 3 * I_BC
5     I_AB = I_AC
6     I_BD = 6 * I_BC
7     I_CD = I_AC - I_BC
8
9     # Menampilkan hasil
10    print("Arus yang mengalir melalui setiap resistor:
11    print(f"I_AB = {I_AB} A")
12    print(f"I_AC = {I_AC} A")
13    print(f"I_BC = {I_BC} A")
14    print(f"I_BD = {I_BD} A")
15    print(f"I_CD = {I_CD} A")
16
17 if __name__ == "__main__":
18     # Input nilai resistansi antara simpul
19     R_AB = float(input("Masukkan nilai resistansi a
20     R_AC = float(input("Masukkan nilai resistansi a
21     R_BC = float(input("Masukkan nilai resistansi a
22     R_BD = float(input("Masukkan nilai resistansi a
23     R_CD = float(input("Masukkan nilai resistansi a
24
25     # Memanggil fungsi untuk menyelesaikan sirkuit
26     solve_circuit(R_AB, R_AC, R_BC, R_BD, R_CD)

PS D:\ABI\_MATA KULIAH PASCA\clone 2\SAKOLA-PASCA\SEMESTER 1\ANALISA ALGORITMA\_TUGAS\UTS> & C:/Users/reads/AppData/Local/Programs/Python/Python312/python.exe "d:/ABI/_MATA KULIAH PASCA/clone 2/SAKOLA-PASCA/SEMESTER 1/ANALISA ALGORITMA/_TUGAS/UTS/soal no 1b fix.py"
Masukkan nilai resistansi antara simpul A dan B (ohm): 5
Masukkan nilai resistansi antara simpul A dan C (ohm): 3
Masukkan nilai resistansi antara simpul B dan C (ohm): 6
Masukkan nilai resistansi antara simpul B dan D (ohm): 8
Masukkan nilai resistansi antara simpul C dan D (ohm): 4
Arus yang mengalir melalui setiap resistor:
I_AB = 9 A
I_AC = 9 A
I_BC = 3 A
I_BD = 18 A
I_CD = 6 A
PS D:\ABI\_MATA KULIAH PASCA\clone 2\SAKOLA-PASCA\SEMESTER 1\ANALISA ALGORITMA\_TUGAS\UTS>
```

Jawaban No 2

1. Buktikan 3^n dan 5^n selalu habis dibagi oleh 8

A. PERHITUNGAN SECARA ANALITIK

Langkah induksi pertama :

1. Untuk $n = 1$

3^n dan 5^n selalu habis dibagi oleh 8

$3^1 = 3$ (**tidak habis dibagi 8**)

$5^1 = 5$ (**tidak habis dibagi 8**)

Ini artinya pernyataan diatas tidak benar untuk $n=1$

2. Untuk $n = z$ (kita asumsikan pernyataan ini benar untuk $n = z$)

Asumsikan k bilangan bulangan positif dan 3^k dan 5^k habis dibagi 8, dengan kata lain $3^z = 8x$ dan $5^z = 8y$, dengan x dan y bilangan bulat positif

3. Buktikan untuk $n = z + 1$

Jika asumsi $n = z$ benar untuk pernyataan diatas (3^n dan 5^n selalu habis dibagi oleh 8), maka pernyataan tersebut juga benar untuk $n = z + 1$.

Mari kita coba:

$$\begin{aligned} 3^{z+1} &= 3^z \times 3^1 = 3^z \times 3 \quad (\text{sifat perpangkatan } a^{m+n} = a^m \times a^n) \\ 5^{z+1} &= 5^z \times 5^1 = 5^z \times 5 \end{aligned}$$

Dari asumsi kita sebelumnya, diketahui bahwa 3^z dan 5^z habis dibagi 8 untuk $n = z$

Maka, jika ditulis ulang persamaan nya sebagai berikut:

$$\checkmark \quad 3^{z+1} = 8x \times 3$$

$$\checkmark \quad 3^{z+1} = 24x$$

$$\checkmark \quad 5^{z+1} = 8y \times 5$$

$$\checkmark \quad 5^{z+1} = 40y$$

Karena 24 dan 40 adalah kelipatan dari 8, maka 3^{z+1} dan 5^{z+1} juga habis dibagi 8

Dengan demikian, dengan metode induksi matematika, kita telah membuktikan bahwa untuk setiap bilangan bulat positif n , 3^n dan 5^n selalu habis dibagi oleh 8 dan $n \neq 1$

B. ALGORITMA

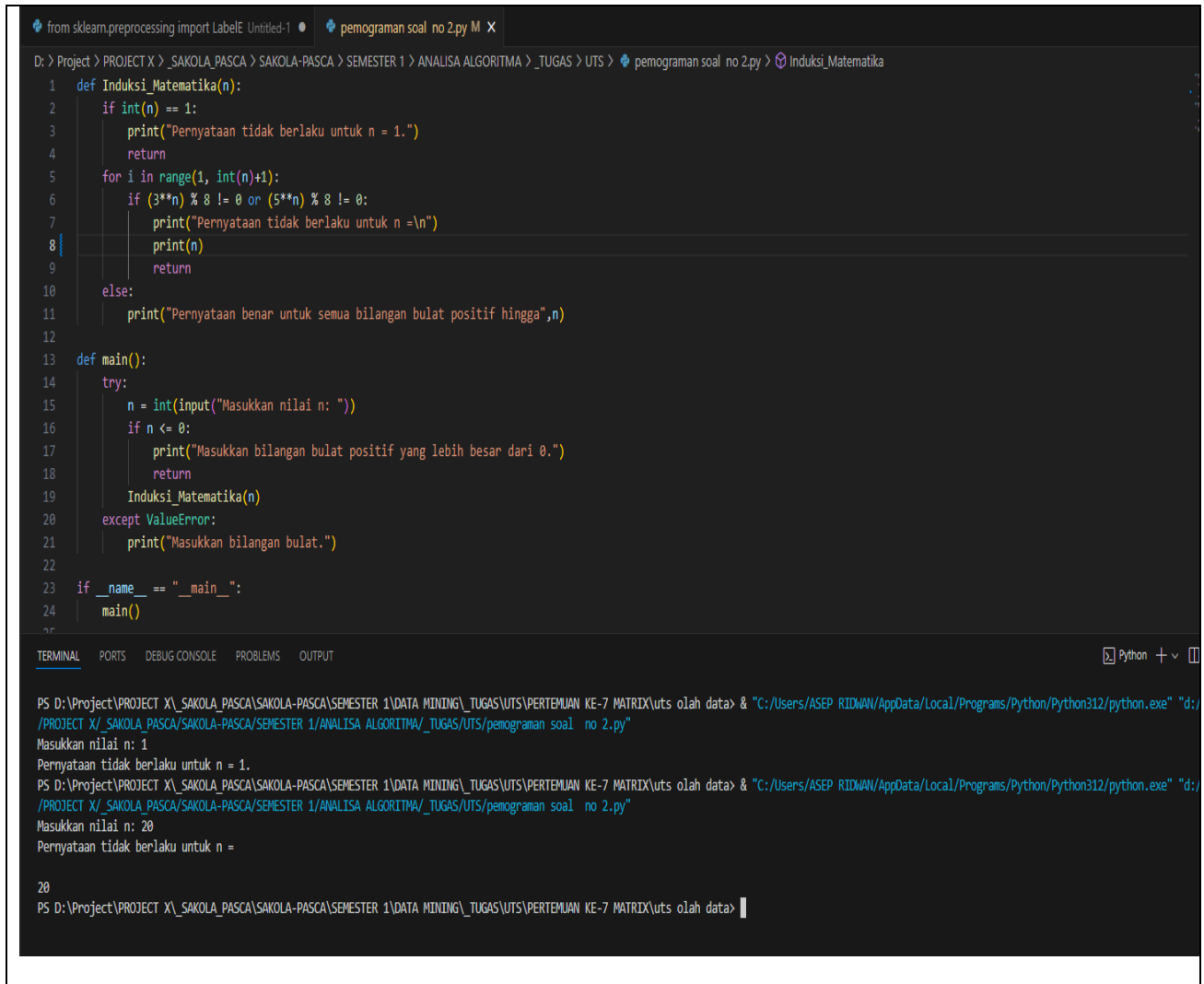
1. Fungsi Induksi_Matematika(n): //buat fungsi untuk input n
2. //jika $n = 1$ tidak terpenuhi karena $n \neq 1$
3. Jika $n = 1$:
4. Tampilkan "Pernyataan tidak berlaku untuk $n = 1$."
5. Berhenti.
6. Jika $n > 1$:
7. Untuk i dari 1 hingga n :
8. Jika $(3^n) / 8 \neq 0$ atau $(5^n) / 8 \neq 0$:
9. Tampilkan "Pernyataan tidak berlaku untuk $n =$ ", i .
10. Berhenti.
- 11.
12. Tampilkan "Pernyataan benar untuk semua bilangan bulat positif hingga n ."

C. PEMOGRAMAN Python

```
def Induksi_Matematika(n):
    if n == 1:
        print("Pernyataan tidak berlaku untuk n = 1.")
        return
    for i in range(1, n+1):
        if (3**n) % 8 != 0 or (5**n) % 8 != 0:
            print("Pernyataan tidak berlaku untuk n =",i)
            return
    else:
        print("Pernyataan benar untuk semua bilangan bulat positif hingga",n)
def main():
    try:
        n = int(input("Masukkan nilai n: "))
        if n <= 0:
            print("Masukkan bilangan bulat positif yang lebih besar dari 0.")
            return
        Induksi_Matematika(n)
    except ValueError:
        print("Masukkan bilangan bulat.")

if __name__ == "__main__":
    main()
```


D. OUTPUT



```
from sklearn.preprocessing import LabelEncoder
pemograman soal no 2.py X

D: > Project > PROJECT X > _SAKOLA_PASCA > SAKOLA-PASCA > SEMESTER 1 > ANALISA ALGORITMA > _TUGAS > UTS > pemograman soal no 2.py > Induksi_Matematika

1 def Induksi_Matematika(n):
2     if int(n) == 1:
3         print("Pernyataan tidak berlaku untuk n = 1.")
4         return
5     for i in range(1, int(n)+1):
6         if (3**i) % 8 != 0 or (5**i) % 8 != 0:
7             print("Pernyataan tidak berlaku untuk n =\n")
8     print(n)
9     return
10 else:
11     print("Pernyataan benar untuk semua bilangan bulat positif hingga",n)
12
13 def main():
14     try:
15         n = int(input("Masukkan nilai n: "))
16         if n <= 0:
17             print("Masukkan bilangan bulat positif yang lebih besar dari 0.")
18             return
19         Induksi_Matematika(n)
20     except ValueError:
21         print("Masukkan bilangan bulat.")
22
23 if __name__ == "__main__":
24     main()
25
```

TERMINAL PORTS DEBUG CONSOLE PROBLEMS OUTPUT Python + v

```
PS D:\Project\PROJECT X\_SAKOLA_PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\_TUGAS\UTS\PERTEMUAN KE-7 MATRIX\uts olah data> & "C:/Users/ASEP RIDWAN/AppData/Local/Programs/Python/Python312/python.exe" "d:/PROJECT X/_SAKOLA_PASCA/SAKOLA-PASCA/SEMESTER 1/ANALISA ALGORITMA/_TUGAS/UTS/pemograman soal no 2.py"
Masukkan nilai n: 1
Pernyataan tidak berlaku untuk n = 1.
PS D:\Project\PROJECT X\_SAKOLA_PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\_TUGAS\UTS\PERTEMUAN KE-7 MATRIX\uts olah data> & "C:/Users/ASEP RIDWAN/AppData/Local/Programs/Python/Python312/python.exe" "d:/PROJECT X/_SAKOLA_PASCA/SAKOLA-PASCA/SEMESTER 1/ANALISA ALGORITMA/_TUGAS/UTS/pemograman soal no 2.py"
Masukkan nilai n: 20
Pernyataan tidak berlaku untuk n =
20
PS D:\Project\PROJECT X\_SAKOLA_PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\_TUGAS\UTS\PERTEMUAN KE-7 MATRIX\uts olah data>
```

Jawaban No. 3

A. PERHITUNGAN ANALITIK

Untuk memodifikasi algoritma Heap Sort agar dapat mengurutkan array dengan efisien dalam kondisi terbaik dan terburuk, diantaranya dengan langkah berikut:

1. **Kondisi terbaik** untuk algoritma Heap Sort terjadi ketika array masukan sudah dalam bentuk heap maksimum atau heap minimum (bergantung pada jenis Heap yang digunakan). Dalam hal ini, algoritma hanya membutuhkan waktu $O(n)$ untuk membangun heap, karena tidak ada perubahan yang diperlukan untuk memastikan properti heap terpenuhi.
2. **Kondisi terburuk** untuk algoritma Heap Sort terjadi ketika elemen-elemen dalam larik berada dalam urutan terbalik dari urutan yang diinginkan. Dalam kondisi ini, setiap kali sebuah elemen dimasukkan ke dalam heap saat membangunnya, elemen tersebut harus bergeser ke atas melewati semua elemen yang ada dalam heap untuk mencapai posisinya yang seharusnya. Hal ini menyebabkan kompleksitas waktu yang paling buruk dalam setiap iterasi membangun heap.

Langkah-langkah optimisasi Heapify dengan Bottom-up Approach:

- a) Ubah proses membangun heap dari pendekatan top-down menjadi bottom-up.
- b) Lakukan heapify untuk setiap node dimulai dari node terakhir yang bukan daun hingga root

B. ALGORITMA

Algoritma Psudo code seperti ini untuk pengurutan heapify_bottom_up:

```
def heapify_bottom_up(arr, n, i):  
    largest = i  
    left = 2 * i + 1  
    right = 2 * i + 2  
  
    if left < n and arr[left] > arr[largest]:  
        largest = left  
  
    if right < n and arr[right] > arr[largest]:
```

```

        largest = right

    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify_bottom_up(arr, n, largest)

def heap_sort_bottom_up(arr):
    n = len(arr)

    # Membangun heap secara bottom-up
    for i in range(n // 2 - 1, -1, -1):
        heapify_bottom_up(arr, n, i)

    # Ekstraksi elemen satu per satu dari heap
    for i in range(n-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        heapify_bottom_up(arr, i, 0)

# Contoh penggunaan:
arr = [12, 11, 13, 5, 6, 7]
heap_sort_bottom_up(arr)
print("Array yang diurutkan adalah:", arr)

```

C. PROGRAM Python

```

def heapify_bottom_up(arr, n, i):
    largest = i
    left = 2 * i + 1
    right = 2 * i + 2

    if left < n and arr[left] > arr[largest]:
        largest = left

    if right < n and arr[right] > arr[largest]:
        largest = right

    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify_bottom_up(arr, n, largest)

def build_max_heap(arr):
    n = len(arr)
    # Starting from the last non-leaf node and heapify them in reverse order
    for i in range(n // 2 - 1, -1, -1):
        heapify_bottom_up(arr, n, i)

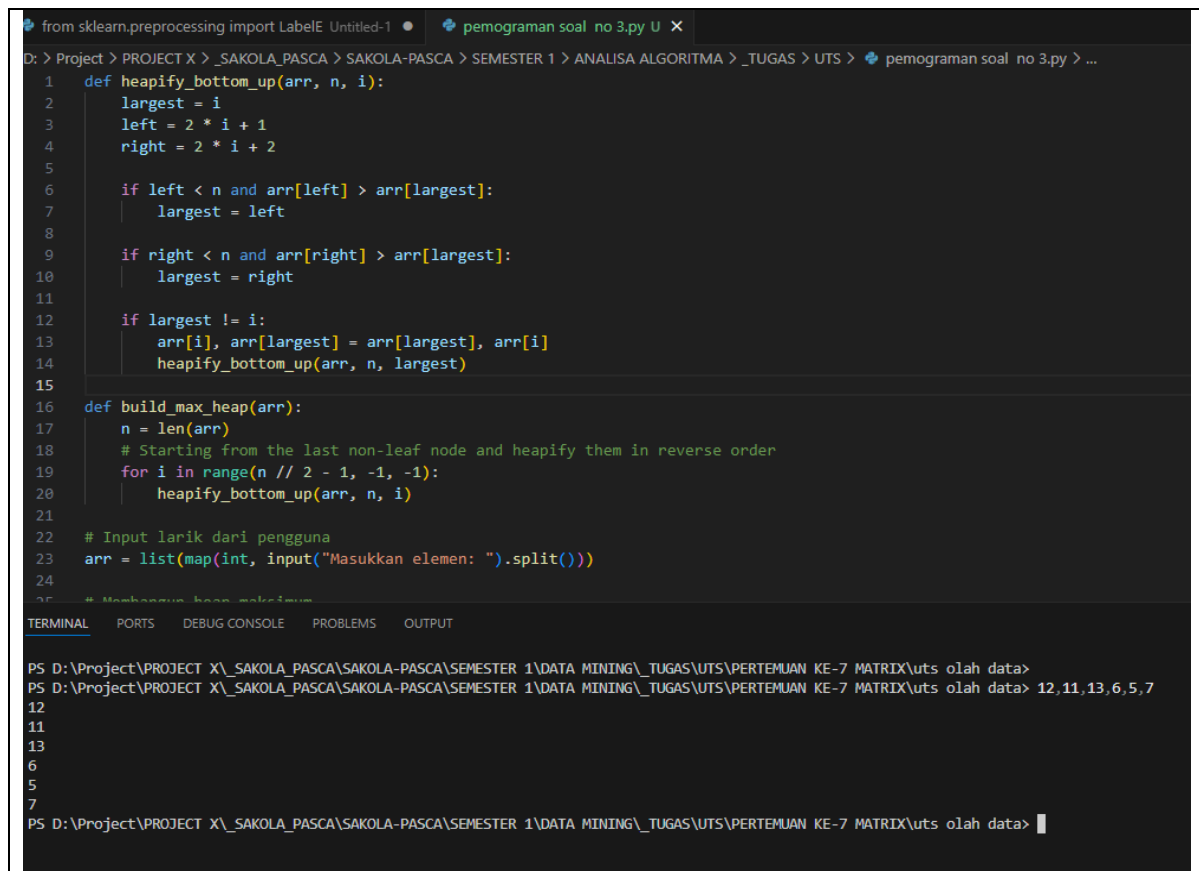
# Input larik dari pengguna
arr = list(map(int, input("Masukkan elemen: ").split()))

```

```
# Membangun heap maksimum
build_max_heap(arr)
print("Heap maksimum:", arr)
```

D. Output

Implementasi dengan menggunakan python :



```
from sklearn.preprocessing import LabelEncoder
pemograman soal no 3.py U X
D: > Project > PROJECT X > _SAKOLA_PASCA > SAKOLA-PASCA > SEMESTER 1 > ANALISA ALGORITMA > _TUGAS > UTS > pemograman soal no 3.py > ...
1 def heapify_bottom_up(arr, n, i):
2     largest = i
3     left = 2 * i + 1
4     right = 2 * i + 2
5
6     if left < n and arr[left] > arr[largest]:
7         largest = left
8
9     if right < n and arr[right] > arr[largest]:
10        largest = right
11
12    if largest != i:
13        arr[i], arr[largest] = arr[largest], arr[i]
14        heapify_bottom_up(arr, n, largest)
15
16 def build_max_heap(arr):
17     n = len(arr)
18     # Starting from the last non-leaf node and heapify them in reverse order
19     for i in range(n // 2 - 1, -1, -1):
20         heapify_bottom_up(arr, n, i)
21
22 # Input larik dari pengguna
23 arr = list(map(int, input("Masukkan elemen: ").split()))
24
25 # Membangun heap maksimum
26
TERMINAL    PORTS    DEBUG CONSOLE    PROBLEMS    OUTPUT
PS D:\Project\PROJECT X\_SAKOLA_PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\_TUGAS\UTS\PERTEMUAN KE-7 MATRIX\uts olah data>
PS D:\Project\PROJECT X\_SAKOLA_PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\_TUGAS\UTS\PERTEMUAN KE-7 MATRIX\uts olah data> 12,11,13,6,5,7
12
11
13
6
5
7
PS D:\Project\PROJECT X\_SAKOLA_PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\_TUGAS\UTS\PERTEMUAN KE-7 MATRIX\uts olah data> |
```

Jawaban No. 4

A. Perhitungan Analitik

Untuk menghitung jumlah operasi perbandingan yang dilakukan dalam algoritma Merge Sort untuk mengurutkan array [5, 1, 6, 2, 3, 4, 7] secara menurun dengan algoritma Merge Sort (perbandingan antara elemen-elemen dalam subarray)

1) Saat membagi array menjadi subarray, array akan dibagi menjadi subarray berikut:

- [5, 1, 6, 2] dan [3, 4, 7]
- [5, 1] dan [6, 2]
- [5] dan [1]
- [6] dan [2]
- [3, 4] dan [7]
- [3] dan [4]

2) Selanjutnya, subarray akan digabungkan secara berurutan untuk menghasilkan array terurut:

- [1, 5, 2, 6] dan [3, 4, 7]
- [1, 5] dan [2, 6]
- [1] dan [5]
- [2] dan [6]
- [3, 4] dan [7]
- [3] dan [4]

dalam penggabungan array jumlah elemen dalam subarray adalah $n/2$, di mana n adalah panjang array. karena jumlah langkah yang diperlukan adalah 2 kali subarray.

$$\text{maka: } \frac{n}{2} \times \frac{n}{2} = \frac{n^2}{4}$$

Didalam array ini [5, 1, 6, 2, 3, 4, 7] Panjang array adalah 7 berarti jika disubstitusikan seperti ini $\frac{7^2}{4} \approx 12$ operasi perbandingan.

B. Algoritma

Pseudo dengan python

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

    i = j = k = 0
```

```

# Menggabungkan dua bagian menjadi arr dalam urutan menurun
while i < len(left_half) and j < len(right_half):
    if left_half[i] >= right_half[j]:
        arr[k] = left_half[i]
        i += 1
    else:
        arr[k] = right_half[j]
        j += 1
    k += 1

# Menyalin elemen yang tersisa dari left_half, jika ada
while i < len(left_half):
    arr[k] = left_half[i]
    i += 1
    k += 1

# Menyalin elemen yang tersisa dari right_half, jika ada
while j < len(right_half):
    arr[k] = right_half[j]
    j += 1
    k += 1

# Masukan Array:
arr = [5, 1, 6, 2, 3, 4, 7]
merge_sort(arr)
print("Array setelah diurutkan secara menurun:", arr)

```

C. Pemograman

```

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0

        # Menggabungkan dua bagian menjadi arr dalam urutan menurun
        while i < len(left_half) and j < len(right_half):
            if left_half[i] >= right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:

```

```

        arr[k] = right_half[j]
        j += 1
        k += 1

    # Menyalin elemen i yang tersisa dari right_half, jika ada
    while i < len(left_half):
        arr[k] = left_half[i]
        i += 1
        k += 1

    # CMenyalin elemen j yang tersisa dari right_half, jika ada
    while j < len(right_half):
        arr[k] = right_half[j]
        j += 1
        k += 1

# array dari soal
arr = [5, 1, 6, 2, 3, 4, 7]
merge_sort(arr)
print("Array setelah diurutkan secara menurun (desc):", arr)

```

D. Output

Implementasi dengan menggunakan python :

The screenshot shows a Python IDE with a file named 'merge_sort.py'. The code implements a merge sort algorithm to sort an array in descending order. The array is [5, 1, 6, 2, 3, 4, 7]. The output in the terminal is 'Array setelah diurutkan secara menurun (desc): [7, 6, 5, 4, 3, 2, 1]'. The terminal also shows the file path and the command used to run the script.

```

1 def merge_sort(arr):
2     if len(arr) > 1:
3         mid = len(arr) // 2
4         left_half = arr[:mid]
5         right_half = arr[mid:]
6
7         merge_sort(left_half)
8         merge_sort(right_half)
9
10        i = j = k = 0
11
12        # Menggabungkan dua bagian menjadi arr dalam urutan menurun
13        while i < len(left_half) and j < len(right_half):
14            if left_half[i] >= right_half[j]:
15                arr[k] = left_half[i]
16                i += 1
17            else:
18                arr[k] = right_half[j]
19                j += 1
20            k += 1
21
22        # Menyalin elemen i yang tersisa dari right_half, jika ada
23        while i < len(left_half):
24            arr[k] = left_half[i]
25            i += 1
26            k += 1
27
28        # Menyalin elemen j yang tersisa dari right_half, jika ada
29        while j < len(right_half):
30            arr[k] = right_half[j]
31            j += 1
32            k += 1
33
34        return arr
35
36 # array dari soal
37 arr = [5, 1, 6, 2, 3, 4, 7]
38 merge_sort(arr)
39 print("Array setelah diurutkan secara menurun (desc):", arr)

```

Terminal Output:

```

PS D:\Project\PROJECT X\SAKOLA PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\TUGAS\UTS\PERTEMUAN KE-7\MATRIK\uts lah data> & "C:/Users/ASEP RIDWAN/AppData/Local/Programs/Python/Python312/python.exe" "d:/Project/PROJECT X/SAKOLA PASCA/SAKOLA-PASCA/SEMESTER 1/ANALISA ALGORITMA/TUGAS/UTS/pemograman soal no 4.py"
Array setelah diurutkan secara menurun (desc): [7, 6, 5, 4, 3, 2, 1]
PS D:\Project\PROJECT X\SAKOLA PASCA\SAKOLA-PASCA\SEMESTER 1\DATA MINING\TUGAS\UTS\PERTEMUAN KE-7\MATRIK\uts lah data>

```