Nama : I Putu Ananda Mahayana

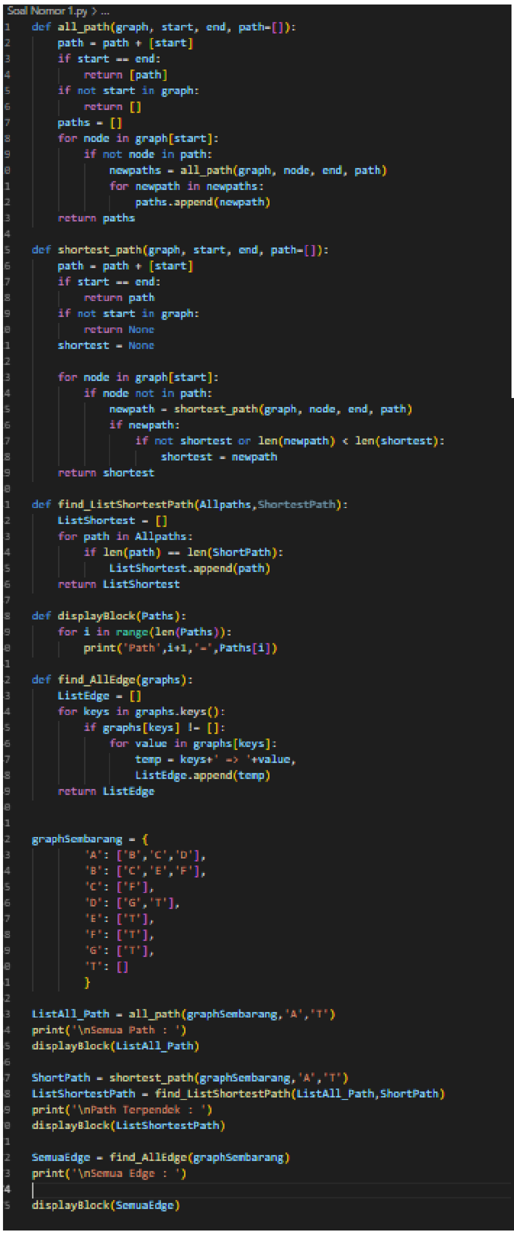
Nim : 2201010346

Kelas : M

Prodi : TI-MDI

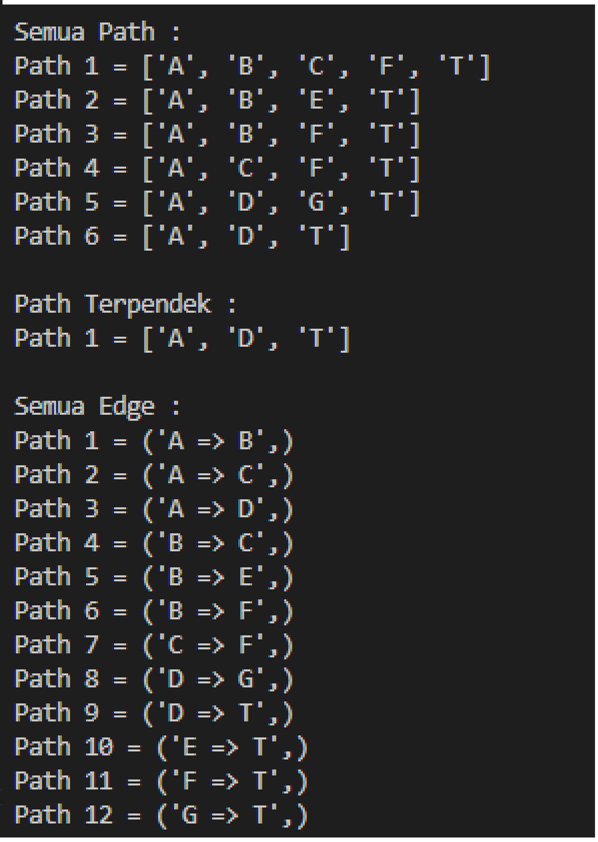
UAS STRUKTUR DATA

1. CODE GRAPH



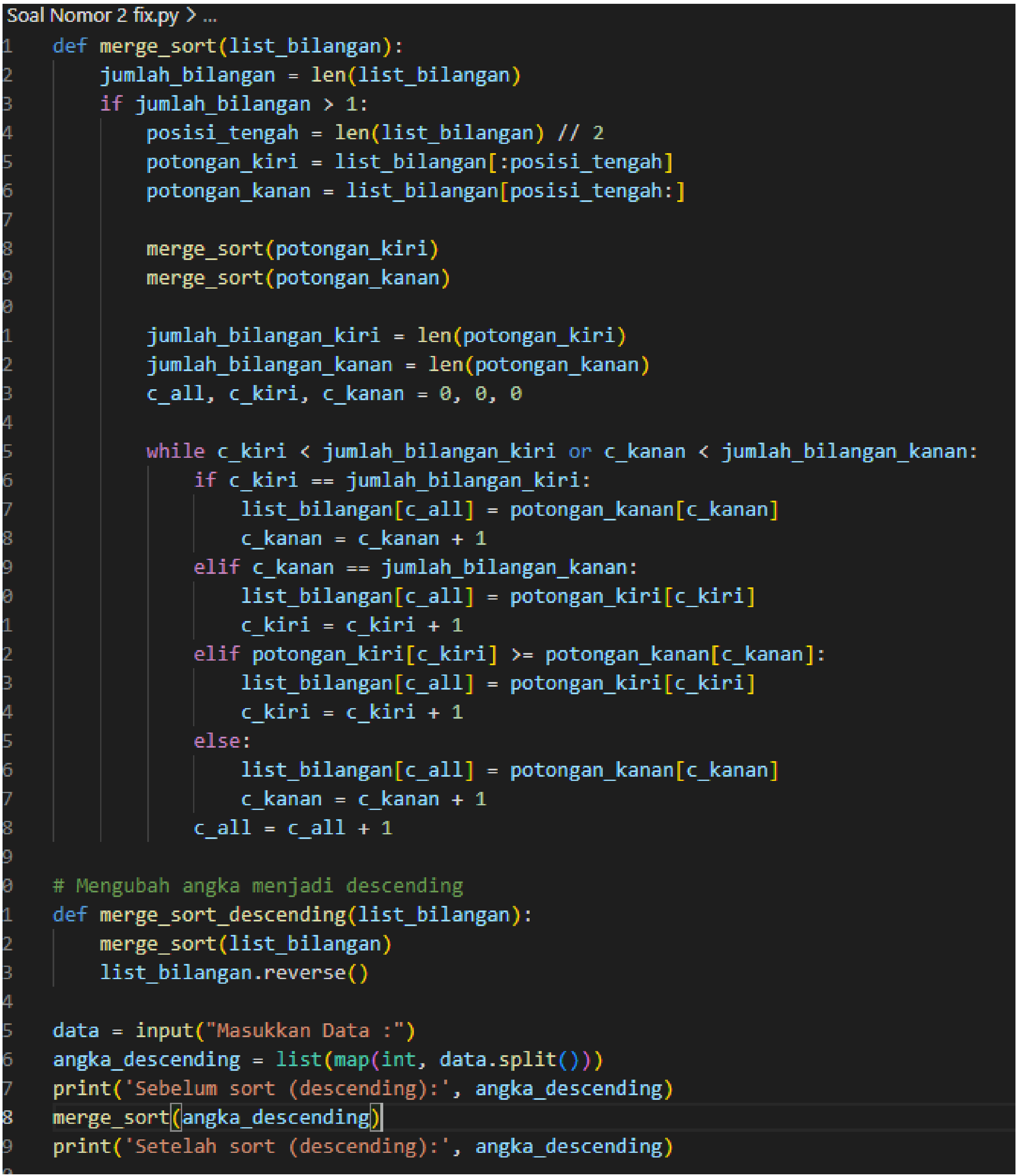
Soal Nomor 1.py > ...

```python
def all_path(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return [path]
    if not start in graph:
        return []
    paths = []
    for node in graph[start]:
        if not node in path:
            newpaths = all_path(graph, node, end, path)
            for newpath in newpaths:
                paths.append(newpath)
    return paths

def shortest_path(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return path
    if not start in graph:
        return None
    shortest = None

    for node in graph[start]:
        if node not in path:
            newpath = shortest_path(graph, node, end, path)
            if newpath:
                if not shortest or len(newpath) < len(shortest):
                    shortest = newpath
    return shortest

def find_ListShortestPath(Allpaths,ShortestPath):
    ListShortest = []
    for path in Allpaths:
        if len(path) == len(ShortPath):
            ListShortest.append(path)
    return ListShortest

def displayBlock(Paths):
    for i in range(len(Paths)):
        print('Path',i+1,'=',Paths[i])

def find_AllEdge(graphs):
    ListEdge = []
    for keys in graphs.keys():
        if graphs[keys] != []:
            for value in graphs[keys]:
                temp = keys+' => '+value,
                ListEdge.append(temp)
    return ListEdge


graphSembarang = {
            'A': ['B','C','D'],
            'B': ['C','E','F'],
            'C': ['F'],
            'D': ['G','T'],
            'E': ['T'],
            'F': ['T'],
            'G': ['T'],
            'T': []
            }

ListAll_Path = all_path(graphSembarang,'A','T')
print('\nSemua Path : ')
displayBlock(ListAll_Path)

ShortPath = shortest_path(graphSembarang,'A','T')
ListShortestPath = find_ListShortestPath(ListAll_Path,ShortPath)
print('\nPath Terpendek : ')
displayBlock(ListShortestPath)

SemuaEdge = find_AllEdge(graphSembarang)
print('\nSemua Edge : ')

displayBlock(SemuaEdge)
```



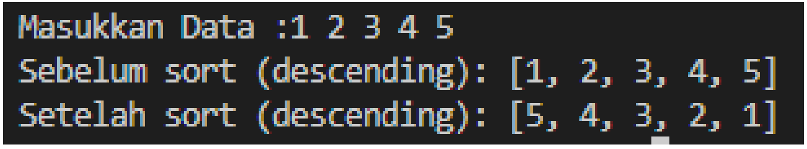
```
Semua Path :
Path 1 = ['A', 'B', 'C', 'F', 'T']
Path 2 = ['A', 'B', 'E', 'T']
Path 3 = ['A', 'B', 'F', 'T']
Path 4 = ['A', 'C', 'F', 'T']
Path 5 = ['A', 'D', 'G', 'T']
Path 6 = ['A', 'D', 'T']

Path Terpendek :
Path 1 = ['A', 'D', 'T']

Semua Edge :
Path 1 = ('A => B',)
Path 2 = ('A => C',)
Path 3 = ('A => D',)
Path 4 = ('B => C',)
Path 5 = ('B => E',)
Path 6 = ('B => F',)
Path 7 = ('C => F',)
Path 8 = ('D => G',)
Path 9 = ('D => T',)
Path 10 = ('E => T',)
Path 11 = ('F => T',)
Path 12 = ('G => T',)
```

2. CODE MERGE SORT



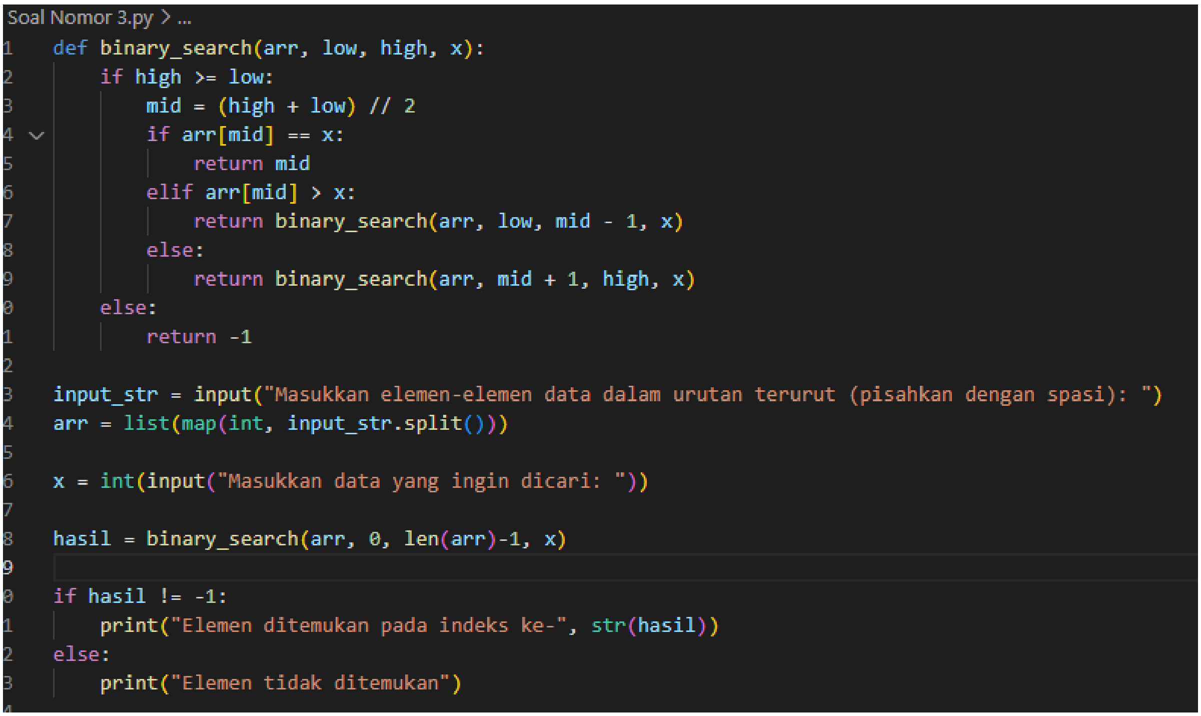
Soal Nomor 2 fix.py > ...

```python
def merge_sort(list_bilangan):
    jumlah_bilangan = len(list_bilangan)
    if jumlah_bilangan > 1:
        posisi_tengah = len(list_bilangan) // 2
        potongan_kiri = list_bilangan[:posisi_tengah]
        potongan_kanan = list_bilangan[posisi_tengah:]

        merge_sort(potongan_kiri)
        merge_sort(potongan_kanan)

        jumlah_bilangan_kiri = len(potongan_kiri)
        jumlah_bilangan_kanan = len(potongan_kanan)
        c_all, c_kiri, c_kanan = 0, 0, 0

        while c_kiri < jumlah_bilangan_kiri or c_kanan < jumlah_bilangan_kanan:
            if c_kiri == jumlah_bilangan_kiri:
                list_bilangan[c_all] = potongan_kanan[c_kanan]
                c_kanan = c_kanan + 1
            elif c_kanan == jumlah_bilangan_kanan:
                list_bilangan[c_all] = potongan_kiri[c_kiri]
                c_kiri = c_kiri + 1
            elif potongan_kiri[c_kiri] >= potongan_kanan[c_kanan]:
                list_bilangan[c_all] = potongan_kiri[c_kiri]
                c_kiri = c_kiri + 1
            else:
                list_bilangan[c_all] = potongan_kanan[c_kanan]
                c_kanan = c_kanan + 1
            c_all = c_all + 1

# Mengubah angka menjadi descending
def merge_sort_descending(list_bilangan):
    merge_sort(list_bilangan)
    list_bilangan.reverse()

data = input("Masukkan Data :")
angka_descending = list(map(int, data.split()))
print('Sebelum sort (descending):', angka_descending)
merge_sort(angka_descending)
print('Setelah sort (descending):', angka_descending)
```



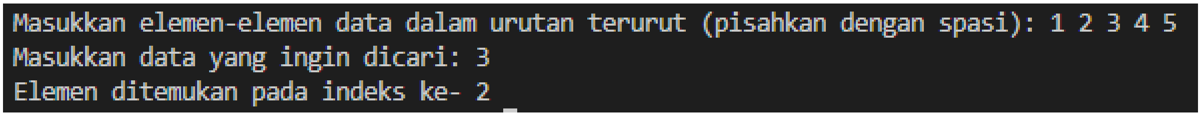
```
Masukkan Data :1 2 3 4 5
Sebelum sort (descending): [1, 2, 3, 4, 5]
Setelah sort (descending): [5, 4, 3, 2, 1]
```

3. CODE BINARY SEARCH (pendekatan interatif)



Soal Nomor 3.py > ...

```python
def binary_search(arr, low, high, x):
    if high >= low:
        mid = (high + low) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)
        else:
            return binary_search(arr, mid + 1, high, x)
    else:
        return -1

input_str = input("Masukkan elemen-elemen data dalam urutan terurut (pisahkan dengan spasi): ")
arr = list(map(int, input_str.split()))

x = int(input("Masukkan data yang ingin dicari: "))

hasil = binary_search(arr, 0, len(arr)-1, x)

if hasil != -1:
    print("Elemen ditemukan pada indeks ke-", str(hasil))
else:
    print("Elemen tidak ditemukan")
```



```
Masukkan elemen-elemen data dalam urutan terurut (pisahkan dengan spasi): 1 2 3 4 5
Masukkan data yang ingin dicari: 3
Elemen ditemukan pada indeks ke- 2
```