Nama : Arindita Prihastama
NIM : L200180058
Kelas : B

TUGAS MODUL 6

PENGURUTAN LANJUTAN

6.4 Soal-soal untuk Mahasiswa

1. Mengubah kode mergeSort dan quicksort agar bisa mengurutkan list yang berisi object-object MhsTIF.
   Dari class MhsTIF

```
class MhsTIF():
    def __init__(self, nim):
        self.nim = nim

    def __str__(self):
        return str(self.nim)

c0 = MhsTIF(10)
c1 = MhsTIF(51)
c2 = MhsTIF(2)
c3 = MhsTIF(18)
c4 = MhsTIF(4)
c5 = MhsTIF(31)
c6 = MhsTIF(13)
c7 = MhsTIF(5)
c8 = MhsTIF(23)
c9 = MhsTIF(64)
c10 = MhsTIF(29)

c0.next = c1
c1.next = c2
c2.next = c3
c3.next = c4
c4.next = c5
c5.next = c6
c6.next = c7
c7.next = c8
c8.next = c9
c9.next = c10
```

a. MergeSort

```
def mergeSort(A):
    #print("Membelah      ",A)
    if len(A) > 1:
        mid = len(A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]

        mergeSort(separuhkiri)
        mergeSort(separuhkanan)

        i = 0;j=0;k=0
        while i < len(separuhkiri) and j < len(separuhkanan):
            if separuhkiri[i] < separuhkanan[j]:
                A[k] = separuhkiri[i]
                i = i + 1
            else:
                A[k] = separuhkanan[j]
                j = j + 1
            k=k+1

        while i < len(separuhkiri):
            A[k] = separuhkiri[i]
            i = i + 1
            k=k+1

        while j < len(separuhkanan):
            A[k] = separuhkanan[j]
            j = j + 1
            k=k+1
    #print("Menggabungkan",A)

def convert(arr, obj):
    hasil=[]
    for x in range (len(arr)):
        for i in range (len(arr)):
            if arr[x] == obj[i].nim:
                hasil.append(obj[i])
    return hasil

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
A = []
for x in Daftar:
    A.append(x.nim)

print("MERGE SORT")
mergeSort(A)
for x in convert(A, Daftar):
    print (x.nim)
```

Saat dijalankan di python shell, hasilnya :

```
MERGE SORT
2
4
5
10
13
18
23
29
31
51
64
```

b. Quick Sort

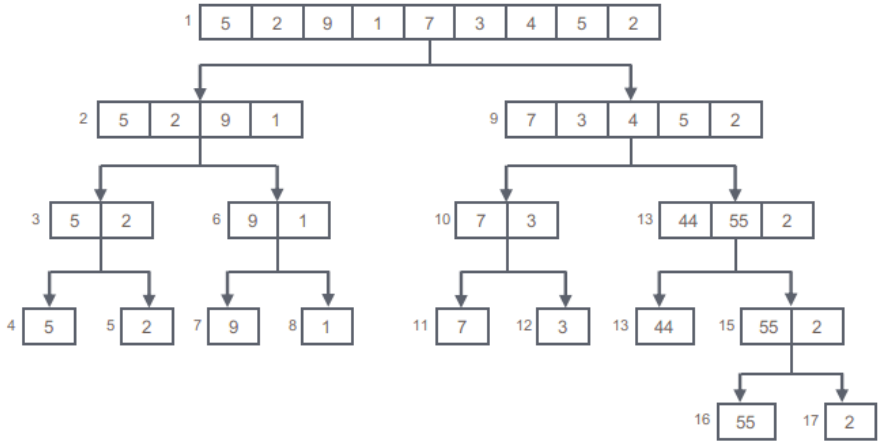
```
def partisi(A, awal, akhir):
    nilaipivot = A[awal]

    penandakiri = awal + 1
    penandakanan = akhir

    selesai = False
    while not selesai:

        while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
            penandakiri = penandakiri + 1

        while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
            penandakanan = penandakanan - 1

        if penandakanan < penandakiri:
            selesai = True
        else:
            temp = A[penandakiri]
            A[penandakiri] = A[penandakanan]
            A[penandakanan] = temp

    temp = A[awal]
    A[awal] = A[penandakanan]
    A[penandakanan] = temp

    return penandakanan

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)
        quickSortBantu(A, awal, titikBelah-1)
        quickSortBantu(A, titikBelah+1, akhir)

def quickSort(A):
    quickSortBantu (A, 0, len(A)-1)
```

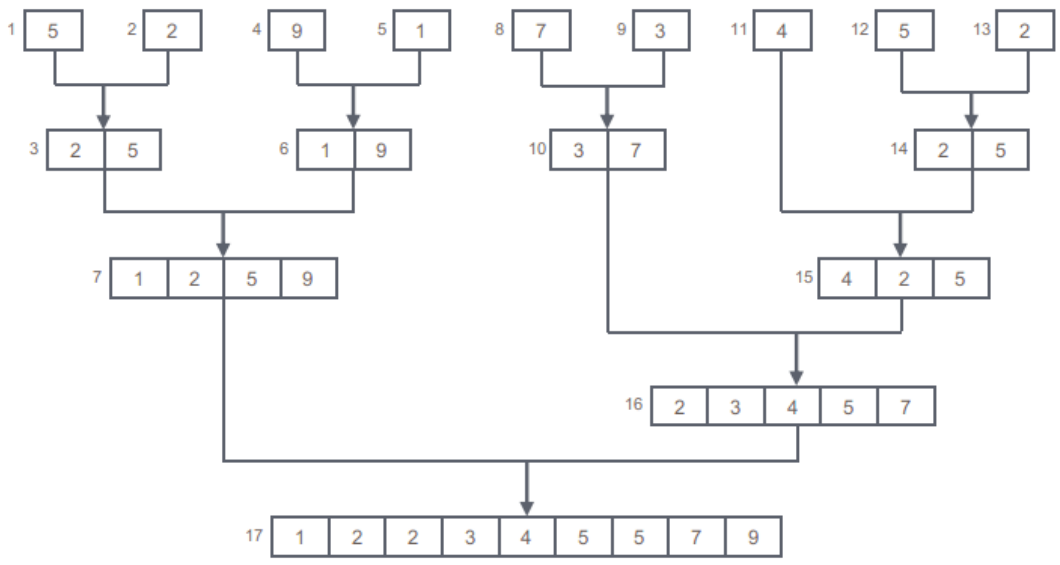
```
def convert(arr, obj):
    hasil=[]
    for x in range (len(arr)):
        for i in range (len(arr)):
            if arr[x] == obj[i].nim:
                hasil.append(obj[i])
    return hasil

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
A = []
for x in Daftar:
    A.append(x.nim)

print("QUICK SORT")
quickSort(A)
for x in convert(A, Daftar):
    print (x.nim)
```

Saat dijalankan di python shell, hasilnya :

```
QUICK SORT
2
4
5
10
13
18
23
29
31
51
64
```

2. Tandai dan beri nomor urut eksekusi proses pada gambar halaman 59.





3. Menguji kecepatan mergeSort dan quicksort

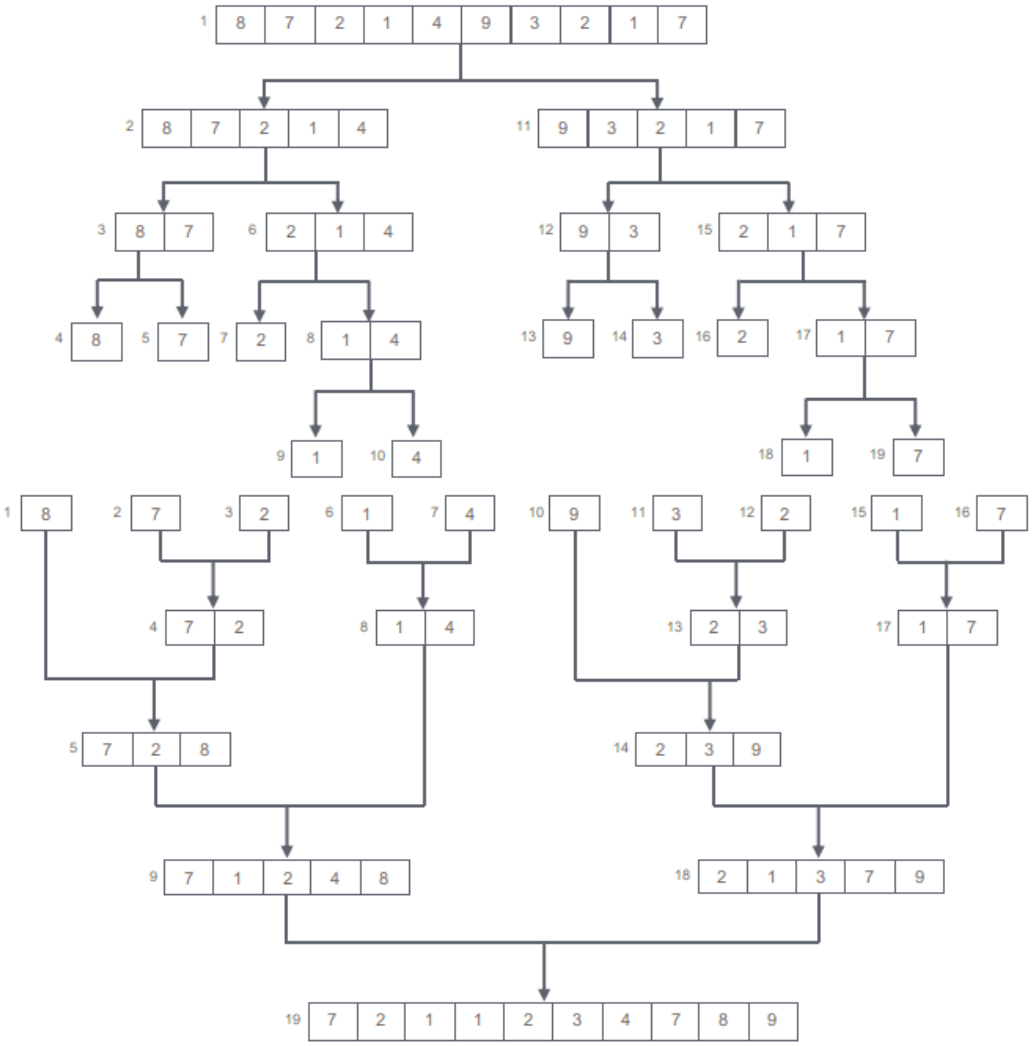
```
from time import time as detak
from random import shuffle as kocok
import time
```

```
aw=detak();bubbleSort(u_bub);ak=detak();print("bubble: %g detik" %(ak-aw));
aw=detak();selectionSort(u_sel);ak=detak();print("selection: %g detik" %(ak-aw))
aw=detak();insertionSort(u_ins);ak=detak();print("insertion: %g detik" %(ak-aw))
aw=detak();mergeSort(u_mrg);ak=detak();print("merge: %g detik" %(ak-aw));
aw=detak();quickSort(u_qck);ak=detak();print("quick: %g detik" %(ak-aw));
```

Hasilnya :

```
bubble: 5.75773 detik
selection: 2.43541 detik
insertion: 2.8584 detik
merge: 0.0389307 detik
quick: 0.0259638 detik
```

4. Diberikan list L = [80, 7, 24, 16, 43, 91, 35, 2, 19, 72], gambarkan trace pengurutan untuk algoritma



5. Meningkatkan efisiensi program mergeSort dengan tidak memakai operator slice, dan lalu mem-pass index awal dan index akhir bersama list-nya saat memanggil mergeSort secara rekursif.

   Dari class MhsTIF

```
class MhsTIF():
    def __init__(self, nama, nim, kota, us):
        self.nama = nama
        self.nim = nim
        self.kota = kota
        self.us = us

    def __str__(self):
        s = self.nama +', NIM '+str(self.nim)\
            +'. Tinggal di '+ self.kota \
            +'. Uang saku Rp. '+ str(self.us)\
            +' tiap bulannya.'
        return s

    def ambilNama(self):
        return self.nama
    def ambilNim(self):
        return self.nim
    def ambilUangSaku(self):
        return self.us

c0 = MhsTIF("Ika", 10, "Sukoharjo", 240000)
c1 = MhsTIF("Budi", 51, "Sragen", 230000)
c2 = MhsTIF("Ahmad", 2, "Surakarta", 250000)
c3 = MhsTIF("Chandra", 18, "Surakarta", 235000)
c4 = MhsTIF("Eka", 4, "Boyolali", 240000)
c5 = MhsTIF("Fandi", 31, "Salatiga", 250000)
c6 = MhsTIF("Deni", 13, "Klaten", 245000)
c7 = MhsTIF("Galuh", 5, "Wonogiri", 245000)
c8 = MhsTIF("Janto", 23, "Klaten", 245000)
c9 = MhsTIF("Hasan", 64, "Karanganyar", 270000)
c10 = MhsTIF("Khalid", 29, "Purwodadi", 265000)

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
```

   Program mergeSort

```
def cetak(A):
    for i in A:
        print (i)
```

```
def mergeSort2(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergeSort2(A, awal, mid)
        mergeSort2(A, mid+1, akhir)

    a, f, l = 0, awal, mid+1
    tmp = [None] * (akhir - awal + 1)
    while f <= mid and l <= akhir:
        if A[f].ambilUangSaku() < A[l].ambilUangSaku():
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

    if f <= mid:
        tmp[a:] = A[f:mid+1]

    if l <= akhir:
        tmp[a:] = A[l:akhir+1]

    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

def mergeSort(A):
    mergeSort2(A, 0, len(A)-1)

mergeSort(Daftar)
cetak(Daftar)
```

Saat dijalankan di python shell hasilnya :

```
Budi, NIM 51. Tinggal di Sragen. Uang saku Rp. 230000 tiap bulannya.
Chandra, NIM 18. Tinggal di Surakarta. Uang saku Rp. 235000 tiap bulannya.
Eka, NIM 4. Tinggal di Boyolali. Uang saku Rp. 240000 tiap bulannya.
Ika, NIM 10. Tinggal di Sukoharjo. Uang saku Rp. 240000 tiap bulannya.
Janto, NIM 23. Tinggal di Klaten. Uang saku Rp. 245000 tiap bulannya.
Galuh, NIM 5. Tinggal di Wonogiri. Uang saku Rp. 245000 tiap bulannya.
Deni, NIM 13. Tinggal di Klaten. Uang saku Rp. 245000 tiap bulannya.
Fandi, NIM 31. Tinggal di Salatiga. Uang saku Rp. 250000 tiap bulannya.
Ahmad, NIM 2. Tinggal di Surakarta. Uang saku Rp. 250000 tiap bulannya.
Khalid, NIM 29. Tinggal di Purwodadi. Uang saku Rp. 265000 tiap bulannya.
Hasan, NIM 64. Tinggal di Karanganyar. Uang saku Rp. 270000 tiap bulannya.
```

6. Meningkatkan efisiensi quicksort dengan memakai metode median-dari-tiga untuk memilih pivotnya.

```
class MhsTIF():
    def __init__(self, nama, nim, kota, us):
        self.nama = nama
        self.nim = nim
        self.kota = kota
        self.us = us

    def __str__(self):
        s = self.nama +', NIM '+str(self.nim)\
            +'. Tinggal di '+ self.kota \
            +'. Uang saku Rp. '+ str(self.us)\
            +' tiap bulannya.'
        return s

    def ambilNama(self):
        return self.nama
    def ambilNim(self):
        return self.nim
    def ambilUangSaku(self):
        return self.us

c0 = MhsTIF("Ika", 10, "Sukoharjo", 240000)
c1 = MhsTIF("Budi", 51, "Sragen", 230000)
c2 = MhsTIF("Ahmad", 2, "Surakarta", 250000)
c3 = MhsTIF("Chandra", 18, "Surakarta", 235000)
c4 = MhsTIF("Eka", 4, "Boyolali", 240000)
c5 = MhsTIF("Fandi", 31, "Salatiga", 250000)
c6 = MhsTIF("Deni", 13, "Klaten", 245000)
c7 = MhsTIF("Galuh", 5, "Wonogiri", 245000)
c8 = MhsTIF("Janto", 23, "Klaten", 245000)
c9 = MhsTIF("Hasan", 64, "Karanganyar", 270000)
c10 = MhsTIF("Khalid", 29, "Purwodadi", 265000)

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
A = []
for i in Daftar:
    A.append(i.nama)
```

```
def cetak():
    for i in A:
        print(i)

def quickSort(arr):
    kurang = []
    pivotList = []
    lebih = []
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        for i in arr:
            if i < pivot:
                kurang.append(i)
            elif i > pivot:
                lebih.append(i)
            else:
                pivotList.append(i)
        kurang = quickSort(kurang)
        lebih = quickSort(lebih)
        return kurang + pivotList + lebih

print("Sebelum diurutkan")
cetak()
print("\nSetelah diurutkan")
quickSort(A)
cetak()
```

Hasilnya :

```
Sebelum diurutkan
Ika
Budi
Ahmad
Chandra
Eka
Fandi
Deni
Galuh
Janto
Hasan
Khalid

Setelah diurutkan
Ika
Budi
Ahmad
Chandra
Eka
Fandi
Deni
Galuh
Janto
Hasan
Khalid
```

7. Menguji kecepatan keduanya dan bandingkan dengan kode awalnya.
   Saar diuji menggunakan kode seperti di nomor 3, didapatkan hasil :

```
merge: 0.0737996 detik
quick: 0.0259743 detik
merge New: 0.0529501 detik
quick New: 0.087769 detik
```

8. Membuat linked-list untuk program mergeSort.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def appendList(self, data):
        node = Node(data)
        if self.head == None:
            self.head = node
        else:
            curr = self.head
            while curr.next != None:
                curr = curr.next
        curr.next = node

    def appendSorted(self, data):
        node = Node(data)
        curr = self.head
        prev = None

        while curr is not None and curr.data < data:
            prev = curr
            curr = curr.next

        if prev == None:
            self.head = node
        else:
            prev.next = node

        node.next = curr

    def printList(self):
        curr = self.head
        while curr != None:
            print("%d" % curr.data),
            curr = curr.next

    def mergeSorted(self, list1, list2):
        if list1 is None:
            return list2
        if list2 is None:
            return list1
```

```
        if list1.data < list2.data:
            temp = list1
            temp.next = self.mergeSorted(list1.next, list2)
        else:
            temp = list2
            temp.next = self.mergeSorted(list1, list2.next)
        return temp


list1 = LinkedList()
list1.appendSorted(14)
list1.appendSorted(13)
list1.appendSorted(5)
list1.appendSorted(19)
list1.appendSorted(9)

print("List 1 :"),
list1.printList()

list2 = LinkedList()
list2.appendSorted(30)
list2.appendSorted(15)
list2.appendSorted(1)

print("List 2 :"),
list2.printList()

list3 = LinkedList()
list3.head = list3.mergeSorted(list1.head, list2.head)

print("Merged List :"),
list3.printList()
```

Hasilnya :

```
List 1 :
5
9
13
14
19
List 2 :
1
15
30
Merged List :
1
5
9
13
14
15
19
30
```