Nama : Annisa Nugraheni

NIM : L200180066

Kelas : C

MODUL 6

PENGURUTAN LANJUTAN

PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

Soal-Soal Untuk Mahasiswa

1. Ubahlah kode mergeSort dan quicksort di atas agar bisa mengurutkan list yang berisi object-object MhsTIF yang sudah kamu buat di modul 2. Uji programmu secukupnya.

   Jawab:

   Berikut adalah screenshot program yang saya buat:

L200180066_Algostruk_Modul 6_Tugas.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas.py (3.7.0)

File Edit Format Run Options Window Help

```
def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah -1)
        quicksortbantu(A,titikbelah+1,akhir)

def partisi(A,awal,akhir):
    nilaipivot = A[awal].nim
    penandakiri = awal + 1
    penandakanan = akhir
    selesai = False

    while not selesai:
        while penandakiri <= penandakanan and A[penandakiri].nim <= nilaipivot:
            penandakiri +=1
        while A[penandakanan].nim >= nilaipivot and penandakanan >= penandakiri :
            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

print("##########################################################################")
cek(Daftar)
print("##########################################################################")
print("MERGESORT")
mergesort(Daftar)
cek(Daftar)
print("##########################################################################")
print("QUICKSORT")
quicksort(Daftar)

cek(Daftar)
```

Ln: 33 Col: 40

Berikut adalah program yang saya buat:

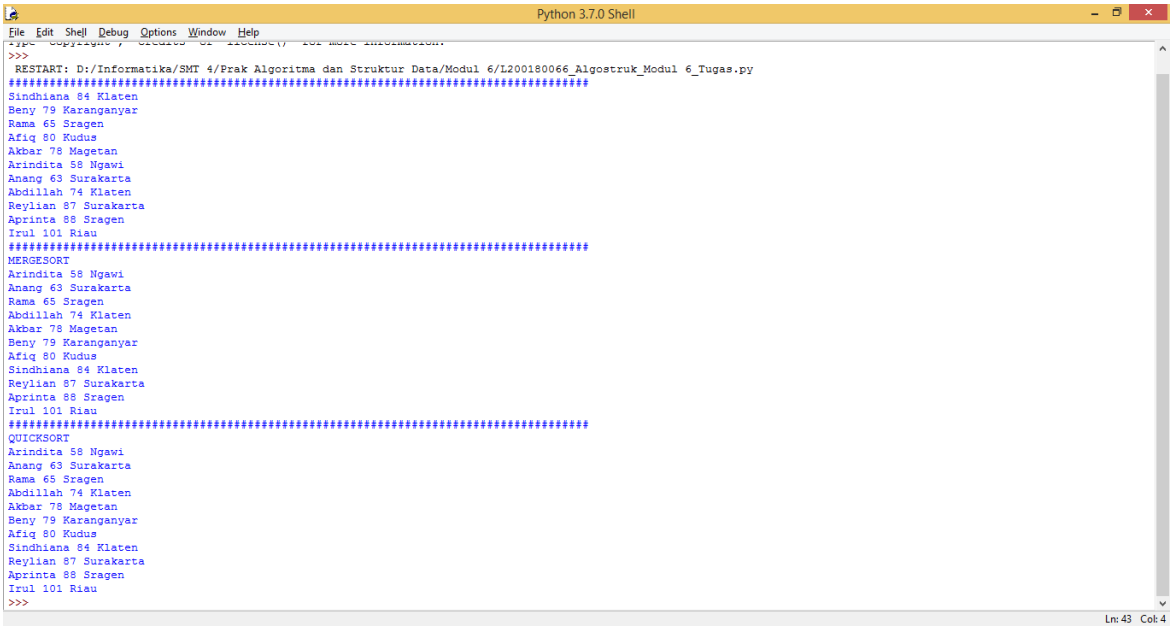
```
###NO 1
class MhsTIF(object):
    def __init__(self,nama,nim,tinggal,us):
        self.nama = nama
        self.nim = nim
        self.tinggal = tinggal
        self.us = us
    def __str__(self):
        return str(self.nama," ",self.nim," ",self.tinggal)



Daftar = [
MhsTIF("Sindhiana", 84, "Klaten", 265000),
MhsTIF("Beny", 79, "Karanganyar", 255000),
MhsTIF("Rama", 65, "Sragen", 260000),
MhsTIF("Afiq", 80, "Kudus", 280000),
MhsTIF("Akbar", 78, "Magetan", 270000),
MhsTIF("Arindita", 58, "Ngawi", 270000),
MhsTIF("Anang", 63, "Surakarta", 250000),
MhsTIF("Abdillah", 74, "Klaten", 260000),
```

```
MhsTIF("Reylian", 87, "Surakarta", 250000),
MhsTIF("Aprinta", 88, "Sragen", 265000),
MhsTIF("Irul", 101, "Riau", 300000)]

def cek(Daftar):
    for i in Daftar:
        print(i.nama,i.nim,i.tinggal)



##mergeSort
def mergesort(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].nim < separuhkanan[j].nim :
                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

##quickSort
def quicksort(A):
    quicksortbantu(A,0,len(A)-1)

def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah -1)
        quicksortbantu(A,titikbelah+1,akhir)

def partisi(A,awal,akhir):
    nilaipivot = A[awal].nim
    penandakiri = awal + 1
    penandakanan = akhir
    selesai = False

    while not selesai:
        while penandakiri <= penandakanan and A[penandakiri].nim <= nilaipivot:
            penandakiri +=1
        while A[penandakanan].nim >= nilaipivot and penandakanan >= penandakiri :
            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
```

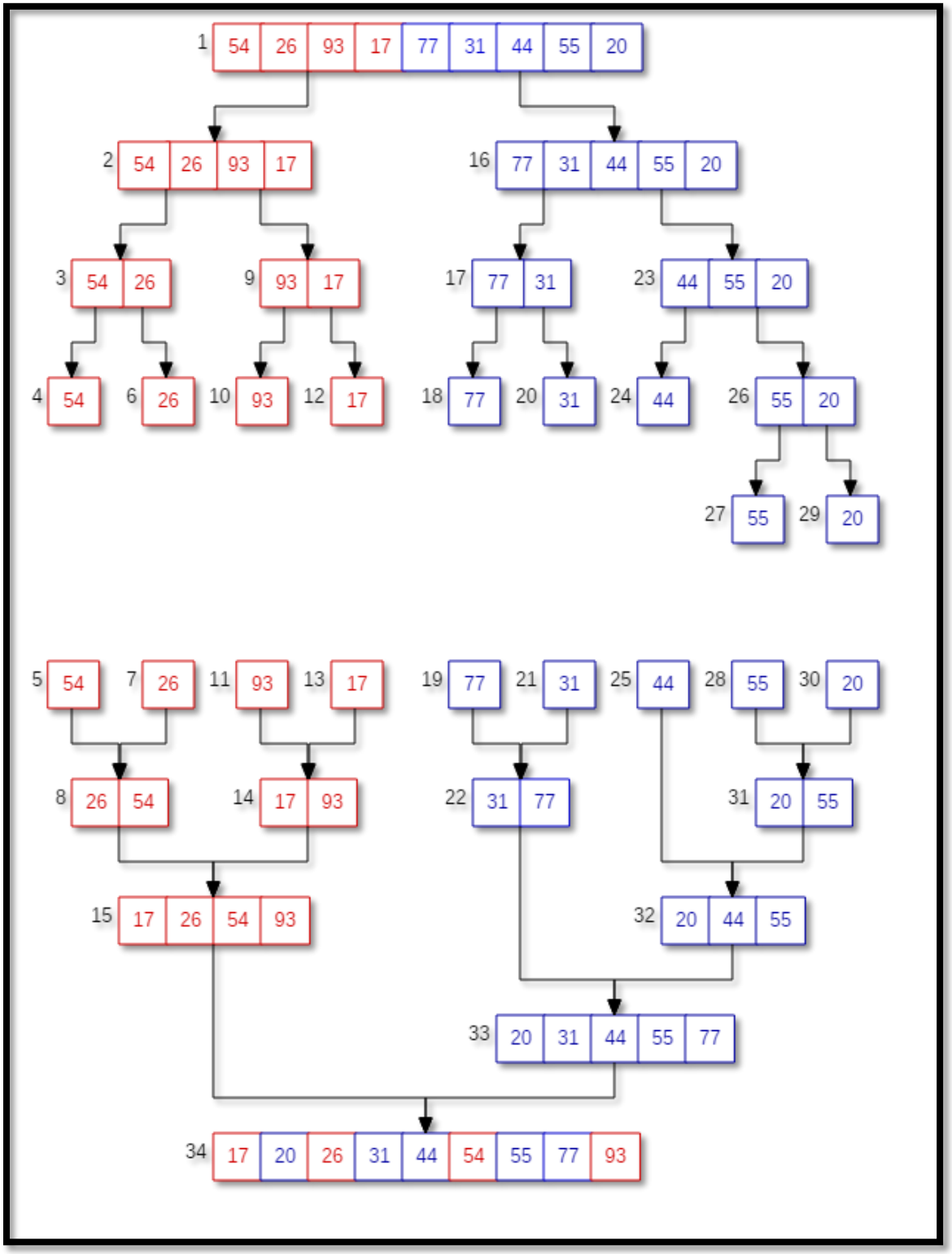
```
    return penandakanan

print("##################################################################
##################")
cek(Daftar)
print("##################################################################
##################")
print("MERGESORT")
mergesort(Daftar)
cek(Daftar)
print("##################################################################
##################")
print("QUICKSORT")
quicksort(Daftar)

cek(Daftar)
```

Berikut adalah screenshot hasil ketika program dijalankan:



2. Memakai bolpen merah atau biru, tandai dan beri nomor urut eksekusi proses pada Gambar 6.1 dan 6.2, dengan mengacu pada output di halaman 59.

Jawab:



3. Uji kecepatan. Ujilah mergeSort dan quicksort diatas (bersama metode sort yang kamu pelajari sebelumnya).

Jawab:

Berikut adalah screenshot program yang saya buat:

NO3_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO3_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
###Praktikum Algoritma dan Struktur Data
###Modul 6
###Pengurutan Lanjutan
###Soal-Soal Untuk Mahasiswa

###Nama : Annisa Nugraheni
###NIM : L200180066
###Kelas : C


###NO 3

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

def swap(A, p, q):
    tmp = A[p]
    A[p] = A[q]
    A[q] = tmp

def cariPosisiYangTerkecil(A, dariSini, sampaiSini):
    posisiYangTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
        if A[i] < A[posisiYangTerkecil]:
            posisiYangTerkecil = i
    return posisiYangTerkecil

def bubbleSort(S):
    n = len(S)
    for i in range (n-1):
        for j in range (n-i-1):
            if S[j] > S[j+1]:
                swap(S,j,j+1)
    return S

def selectionSort(S):
    n = len(S)
    for i in range(n-1):
        indexKecil = cariPosisiYangTerkecil(S, i, n)
        if indexKecil != i:
```

Ln: 137 Col: 0

NO3_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO3_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
def selectionSort(S):
    n = len(S)
    for i in range(n-1):
        indexKecil = cariPosisiYangTerkecil(S, i, n)
        if indexKecil != i:
            swap(S, i, indexKecil)
    return S

def insertionSort(S):
    n = len(S)
    for i in range(1, n):
        nilai = S[i]
        pos = i
        while pos > 0 and nilai < S[pos -1]:
            S[pos] = S[pos-1]
            pos = pos - 1
        S[pos] = nilai
    return S

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]
```

Ln: 137 Col: 0

NO3_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO3_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```
        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 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:
```

Ln: 137 Col: 0

NO3_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO3_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```
            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)

daftar = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]

print (bubbleSort(daftar))
print (selectionSort(daftar))
print (insertionSort(daftar))
mergeSort(daftar)
print (daftar)
quickSort(daftar)
print (daftar)

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_bub = k[:]
u_sel = k[:]
u_ins = k[:]
u_mrg = k[:]
u_qck = k[:]

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));
```

Ln: 137 Col: 0

Berikut adalah program yang saya buat:

###NO 3

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

def swap(A, p, q):
    tmp = A[p]
    A[p] = A[q]
```

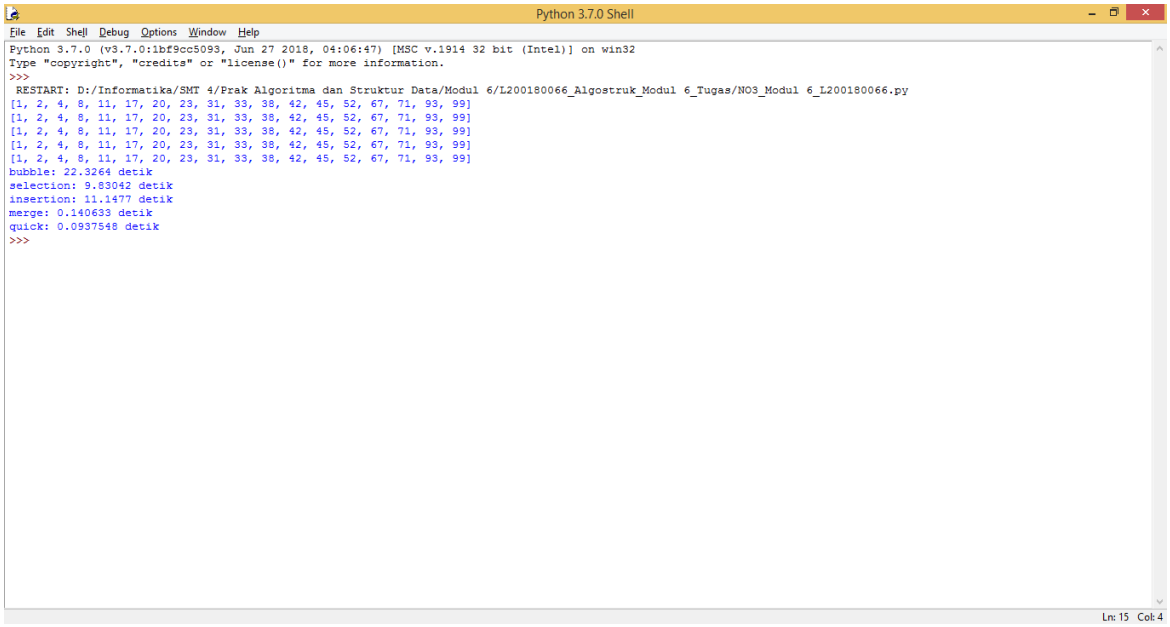
```
    A[q] = tmp

def cariPosisiYangTerkecil(A, dariSini, sampaiSini):
    posisiYangTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
        if A[i] < A[posisiYangTerkecil]:
            posisiYangTerkecil = i
    return posisiYangTerkecil

def bubbleSort(S):
    n = len(S)
    for i in range (n-1):
        for j in range (n-i-1):
            if S[j] > S[j+1]:
                swap(S,j,j+1)
    return S

def selectionSort(S):
    n = len(S)
    for i in range(n-1):
        indexKecil = cariPosisiYangTerkecil(S, i, n)
        if indexKecil != i:
            swap(S, i, indexKecil)
    return S

def insertionSort(S):
    n = len(S)
    for i in range(1, n):
        nilai = S[i]
        pos = i
        while pos > 0 and nilai < S[pos -1]:
            S[pos] = S[pos-1]
            pos = pos - 1
        S[pos] = nilai
```

```
    return S

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 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)

daftar = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]

print (bubbleSort(daftar))
print (selectionSort(daftar))
print (insertionSort(daftar))
mergeSort(daftar)
print (daftar)
quickSort(daftar)
print (daftar)

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_bub = k[:]
u_sel = k[:]
u_ins = k[:]
u_mrg = k[:]
u_qck = k[:]

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));
```

Berikut adalah screenshot hasil ketika program dijalankan:



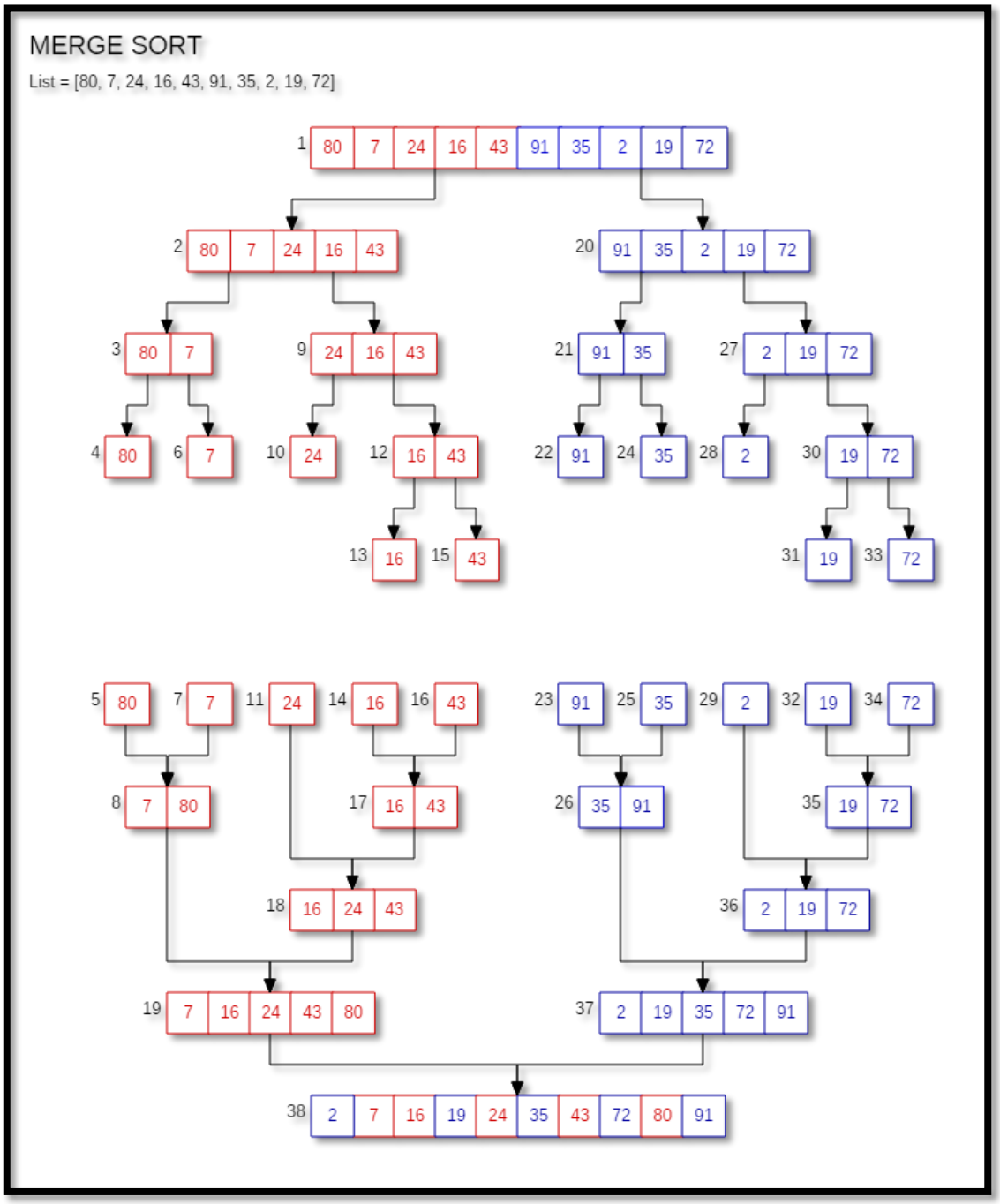
Dilihat dari hasil diatas, maka jika diurutkan dari yang paling cepat, urutannya sebagai berikut:

a. quickSort = 0.09 detik
b. mergeSort = 0.14 detik
c. selection sort = 9.83 detik
d. insertion sort = 11.14 detik
e. bubble sort = 22.32 detik

4. Diberikan list L = [80, 7, 24, 16, 43, 91, 35, 2, 19, 72], gambarlah trace pengurutan untuk algoritma
   a. Merge sort
   b. Quick sort

   Jawab:

   a. Merge sort



b. Quick sort

QUICK SORT

List L = [80, 7, 24, 16, 43, 91, 35, 2, 19, 72]

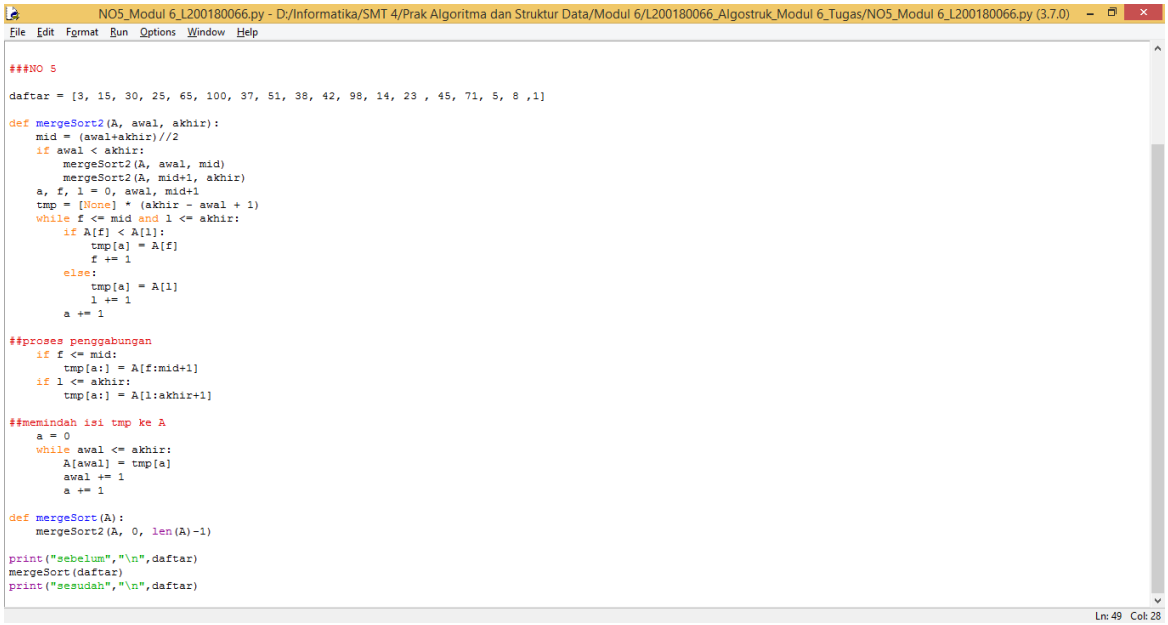
80	7	24	16	43	91	35	2	19	72

pivot									
80	7	24	16	43	91	35	2	19	72
low									high
									pivot
72	7	24	16	43	91	35	2	19	80
low									high
									pivot
72	7	24	16	43	91	35	2	19	80
					low				high
					pivot				
72	7	24	16	43	80	35	2	19	91
					low				high
								pivot	
72	7	24	16	43	19	35	2	80	91
					low			high	
pivot									
72	7	24	16	43	19	35	2	80	91
low							high		
							pivot		
2	7	24	16	43	19	35	72	80	91
low							high		
pivot									
2	7	24	16	43	19	35	72	80	91
low						high			
	pivot								
2	7	24	16	43	19	35	72	80	91
	low					high			
		pivot							
2	7	24	16	43	19	35	72	80	91
		low				high			
		pivot							
2	7	24	16	43	19	35	72	80	91
		low			high				
					pivot				
2	7	19	16	43	24	35	72	80	91
		low			high				
					pivot				
2	7	19	16	43	24	35	72	80	91
				low	high				
				pivot					
2	7	19	16	24	43	35	72	80	91
				low	high				
		pivot							
2	7	19	16	24	43	35	72	80	91
		low	high						
				pivot					
2	7	16	19	24	35	43	72	80	91
				low	high				
2	7	16	19	24	35	43	72	80	91

5. Tingkatkan efisiensi program mergeSort dengan tidak memakai operator slice (seperti A[:mid] dan A[mid:]) dan mem-pass index awal dan index akhir bersama listnya saat kita memanggil mergeSort secara rekursif. Kamu perlu memisah fungsi mergeSort itu menjadi beberapa fungsi, mirip halnya dengan apa yang dilakukan algoritma quicksort.

Jawab:

Berikut adalah screenshot program yang saya buat:




```
###NO 5

daftar = [3, 15, 30, 25, 65, 100, 37, 51, 38, 42, 98, 14, 23 , 45, 71, 5, 8 ,1]

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] < A[l]:
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

##proses penggabungan
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if l <= akhir:
        tmp[a:] = A[l:akhir+1]

##memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

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

print("sebelum","\n",daftar)
mergeSort(daftar)
print("sesudah","\n",daftar)
```

Berikut adalah program yang saya buat:

```
###NO 5

daftar = [3, 15, 30, 25, 65, 100, 37, 51, 38, 42, 98, 14, 23 , 45, 71, 5, 8 ,1]

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] < A[l]:
            tmp[a] = A[f]
            f += 1
```

```
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

##proses penggabungan
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if l <= akhir:
        tmp[a:] = A[l:akhir+1]

##memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

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

print("sebelum","\n",daftar)
mergeSort(daftar)
print("sesudah","\n",daftar)
```

Berikut adalah screenshot hasil ketika program dijalankan:

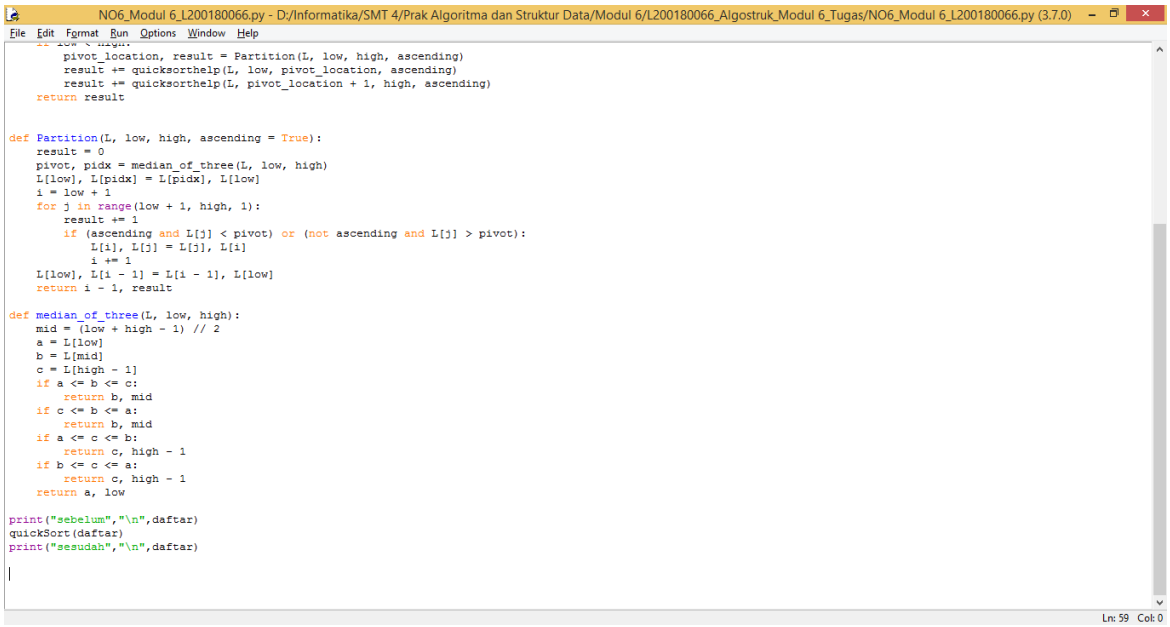
```
Python 3.7.0 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO5_Modul 6_L200180066.py
sebelum
 [3, 15, 30, 25, 65, 100, 37, 51, 38, 42, 98, 14, 23, 45, 71, 5, 8, 1]
sesudah
 [1, 3, 5, 8, 14, 15, 23, 25, 30, 37, 38, 42, 45, 51, 65, 71, 98, 100]
>>>
Ln: 9  Col: 4
```

6. Apakah kita bisa meningkatkan efisiensi program quicksort dengan memakai metode median-dari-tiga untuk memilih pivotnya ? Ubahlah kodenya dan ujilah !

Jawab:

Berikut adalah screenshot program yang saya buat:

```
NO6_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO6_Modul 6_L200180066.py (3.7.0)
File  Edit  Format  Run  Options  Window  Help
###Praktikum Algoritma dan Struktur Data
###Modul 6
###Pengurutan Lanjutan
###Soal-Soal Untuk Mahasiswa

###Nama : Annisa Nugraheni
###NIM : L200180066
###Kelas : C


###NO 6

daftar = [55,20,95,18,78,31,44,59,27]

def quickSort(L, ascending = True):
    quicksorthelp(L, 0, len(L), ascending)

def quicksorthelp(L, low, high, ascending = True):
    result = 0
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
Ln: 41  Col: 31
```




```
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
    a = L[low]
    b = L[mid]
    c = L[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low

print("sebelum","\n",daftar)
quickSort(daftar)
print("sesudah","\n",daftar)
```

Berikut adalah program yang saya buat:

```
###NO 6

daftar = [55,20,95,18,78,31,44,59,27]

def quickSort(L, ascending = True):
    quicksorthelp(L, 0, len(L), ascending)

def quicksorthelp(L, low, high, ascending = True):
    result = 0
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
```

```
    for j in range(low + 1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
    a = L[low]
    b = L[mid]
    c = L[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low

print("sebelum","\n",daftar)
quickSort(daftar)
print("sesudah","\n",daftar)
```

Berikut adalah screenshot hasil ketika program dijalankan:

```
Python 3.7.0 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO6_Modul 6_L200180066.py
sebelum
 [55, 20, 95, 18, 78, 31, 44, 59, 27]
sesudah
 [18, 20, 27, 31, 44, 55, 59, 78, 95]
>>>
Ln: 9  Col: 4
```

7. Uji kecepatan keduanya dan perbandingkan juga dengan kode awalnya !

Jawab:

Berikut adalah screenshot program yang saya buat:

```
NO7_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO7_Modul 6_L200180066.py (3.7.0)
File  Edit  Format  Run  Options  Window  Help
###Praktikum Algoritma dan Struktur Data
###Modul 6
###Pengurutan Lanjutan
###Soal-Soal Untuk Mahasiswa

###Nama : Annisa Nugraheni
###NIM : L200180066
###Kelas : C


###NO 7

def mergesort(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+=1
            else:
                A[k] = separuhkanan[j]
                j+=1
            k+=1
        while i < len(separuhkiri):
            A[k] = separuhkiri[i]
            i+=1
            k+=1
        while j< len(separuhkanan):
            A[k] = separuhkanan[j]
            j+=1
            k+=1

alist = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]

#-----------------------------------------------------------------------------------------
Ln: 152  Col: 0
```

NO7_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO7_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
alist = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]

#-----------------------------------------------------------------------------------------

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

#-----------------------------------------------------------------------------------------
```

Ln: 152 Col: 0

NO7_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO7_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
#-----------------------------------------------------------------------------------------

#merge sort terbaru
def mergesort2_5(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergesort2_5(A, awal, mid)
        mergesort2_5(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] < A[l]:
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

#-----------------------------------------------------------------------------------------

#proses penggabungan
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if l <= akhir:
        tmp[a:] = A[l:akhir+1]

#-----------------------------------------------------------------------------------------

#memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

def mergesort_5(A):
    mergesort2_5(A, 0, len(A)-1)

#-----------------------------------------------------------------------------------------
```

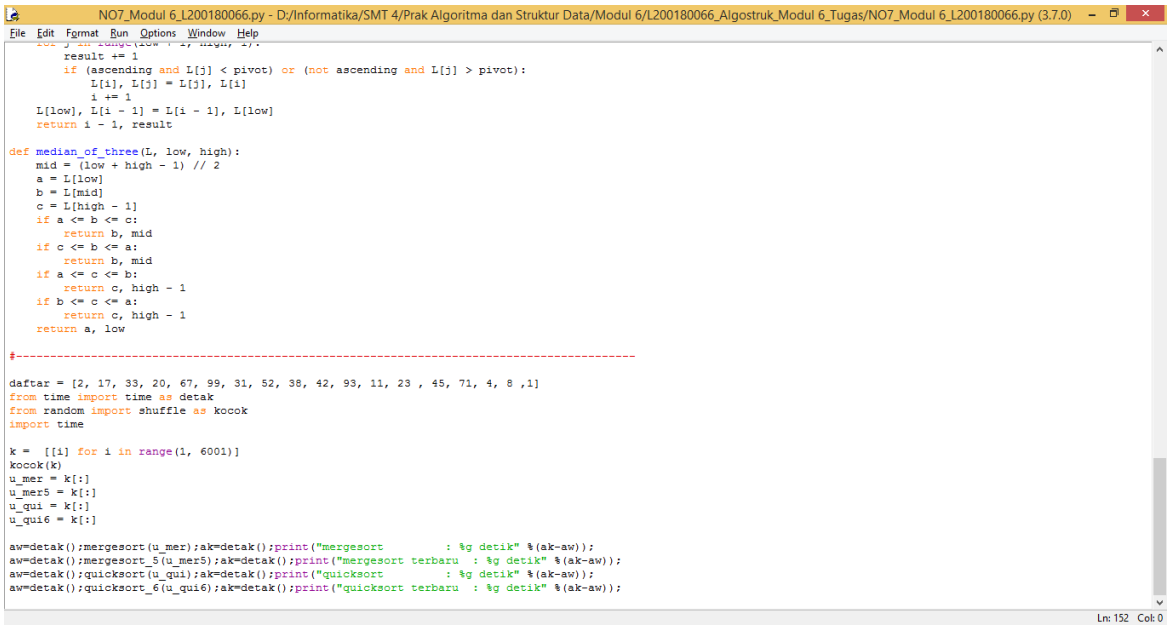
Ln: 152 Col: 0

NO7_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO7_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
#quick sort terbaru
def quicksort_6(L, ascending = True):
    quicksorthelp(L, 0, len(L), ascending)

def quicksorthelp(L, low, high, ascending = True):
    result = 0
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
    a = L[low]
    b = L[mid]
    c = L[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low
```

Ln: 152 Col: 0




```
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
    a = L[low]
    b = L[mid]
    c = L[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low

#-----------------------------------------------------------------------------------------

daftar = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]
from time import time as detak
from random import shuffle as kocok
import time

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_mer = k[:]
u_mer5 = k[:]
u_qui = k[:]
u_qui6 = k[:]

aw=detak();mergesort(u_mer);ak=detak();print("mergesort          : %g detik" %(ak-aw));
aw=detak();mergesort_5(u_mer5);ak=detak();print("mergesort terbaru  : %g detik" %(ak-aw));
aw=detak();quicksort(u_qui);ak=detak();print("quicksort          : %g detik" %(ak-aw));
aw=detak();quicksort_6(u_qui6);ak=detak();print("quicksort terbaru  : %g detik" %(ak-aw));
```

Berikut adalah program yang saya buat:

###NO 7

```
def mergesort(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+=1
            else:
                A[k] = separuhkanan[j]
                j+=1
            k+=1
        while i < len(separuhkiri):
            A[k] = separuhkiri[i]
            i+=1
```

```
            k+=1
        while j< len(separuhkanan):
            A[k] = separuhkanan[j]
            j+=1
            k+=1

alist = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]

#-----------------------------------------------------------------------------------------

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

#-----------------------------------------------------------------------------------------
```

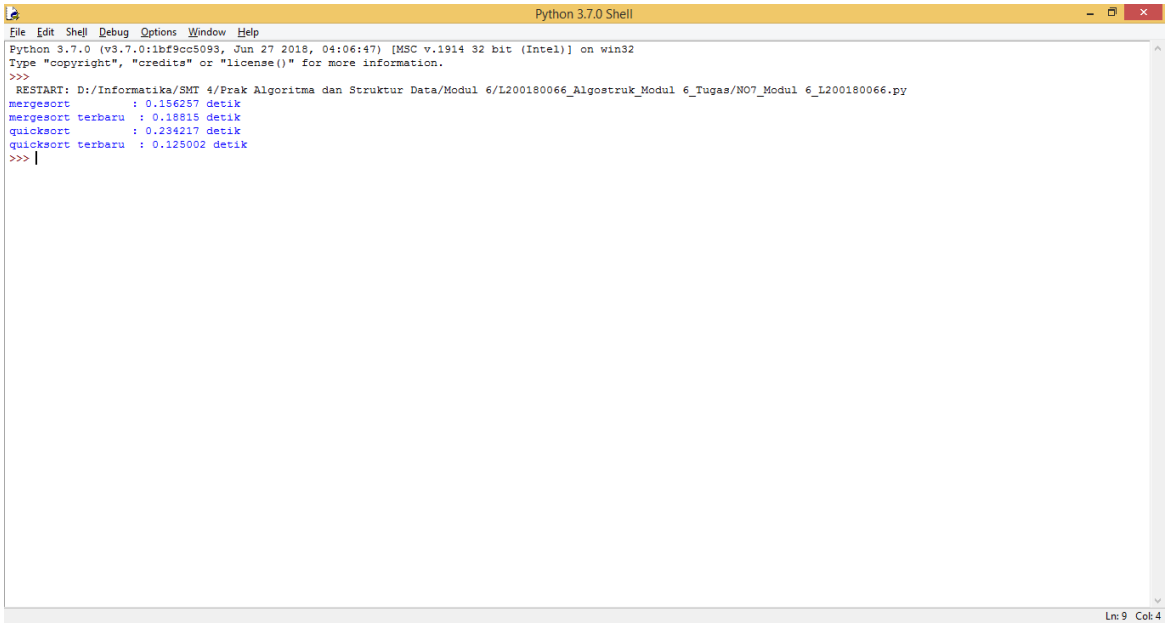
```
def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah -1)
        quicksortbantu(A,titikbelah+1,akhir)

#-----------------------------------------------------------------------------------------

def quicksort(A):
    quicksortbantu(A,0,len(A)-1)

#-----------------------------------------------------------------------------------------

#merge sort terbaru
def mergesort2_5(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergesort2_5(A, awal, mid)
        mergesort2_5(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] < A[l]:
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

#-----------------------------------------------------------------------------------------

#proses penggabungan
```

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

#-----------------------------------------------------------------------------------------

#memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

def mergesort_5(A):
    mergesort2_5(A, 0, len(A)-1)

#-----------------------------------------------------------------------------------------

#quick sort terbaru
def quicksort_6(L, ascending = True):
    quicksorthelp(L, 0, len(L), ascending)

def quicksorthelp(L, low, high, ascending = True):
    result = 0
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


def Partition(L, low, high, ascending = True):
    result = 0
```

```
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
    a = L[low]
    b = L[mid]
    c = L[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low

#-----------------------------------------------------------------------------------------

daftar = [2, 17, 33, 20, 67, 99, 31, 52, 38, 42, 93, 11, 23 , 45, 71, 4, 8 ,1]
from time import time as detak
from random import shuffle as kocok
import time

k =  [[i] for i in range(1, 6001)]
```

```
kocok(k)
u_mer = k[:]
u_mer5 = k[:]
u_qui = k[:]
u_qui6 = k[:]

aw=detak();mergesort(u_mer);ak=detak();print("mergesort        : %g detik" %(ak-aw));
aw=detak();mergesort_5(u_mer5);ak=detak();print("mergesort terbaru  : %g detik" %(ak-aw));
aw=detak();quicksort(u_qui);ak=detak();print("quicksort        : %g detik" %(ak-aw));
aw=detak();quicksort_6(u_qui6);ak=detak();print("quicksort terbaru  : %g detik" %(ak-aw));
```

Berikut adalah screenshot hasil ketika program dijalankan:



8. Buatlah versi linked-list untuk program mergeSort di atas !

Jawab:

Berikut adalah screenshot program yang saya buat:

NO8_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO8_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
###Praktikum Algoritma dan Struktur Data
###Modul 6
###Pengurutan Lanjutan
###Soal-Soal Untuk Mahasiswa

###Nama : Annisa Nugraheni
###NIM : L200180066
###Kelas : C


###NO 8

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

#-----------------------------------------------------------------------------------------

class Linked():
    def __init__(self,head = None):
        self.head = head

    def cetak(self):
        cur = self.head
        while cur != None:
            print(cur.data)
            cur = cur.next
    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)
```

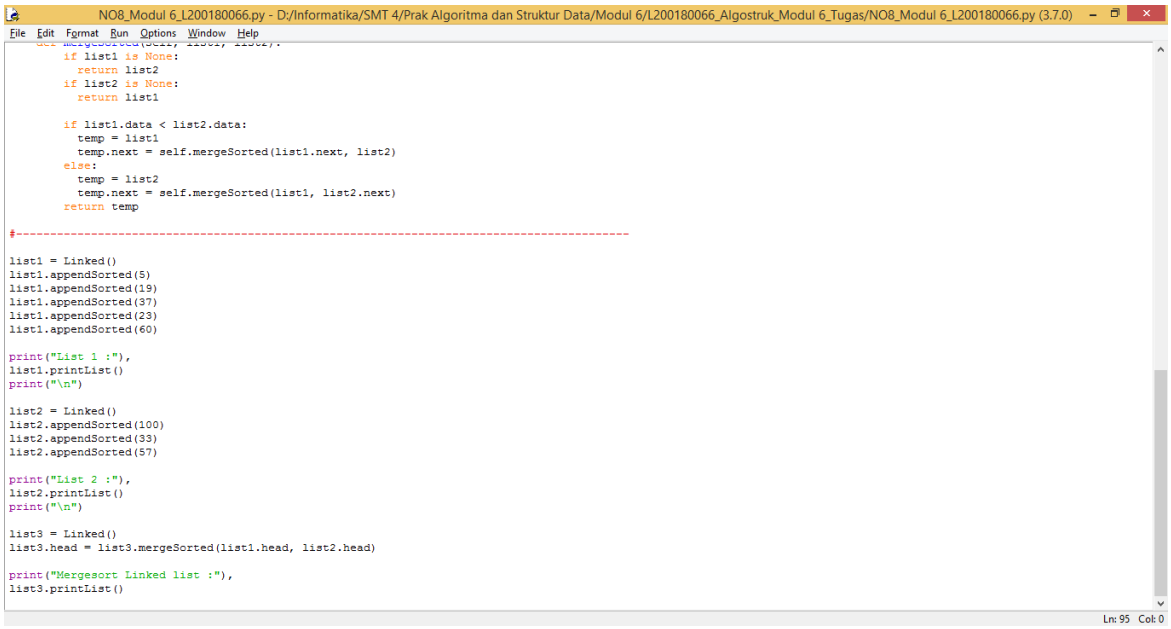
Ln: 95 Col: 0

NO8_Modul 6_L200180066.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 6/L200180066_Algostruk_Modul 6_Tugas/NO8_Modul 6_L200180066.py (3.7.0)

File Edit Format Run Options Window Help

```python
    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 = Linked()
list1.appendSorted(5)
```

Ln: 95 Col: 0



Berikut adalah program yang saya buat:

```
###NO 8


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


#-----------------------------------------------------------------------------------------


class Linked():
    def __init__(self,head = None):
        self.head = head


    def cetak(self):
        cur = self.head
        while cur != None:
            print(cur.data)
            cur = cur.next
    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 = Linked()
list1.appendSorted(5)
list1.appendSorted(19)
list1.appendSorted(37)
list1.appendSorted(23)
list1.appendSorted(60)

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

list2 = Linked()
list2.appendSorted(100)
list2.appendSorted(33)
list2.appendSorted(57)

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

list3 = Linked()
list3.head = list3.mergeSorted(list1.head, list2.head)
```

```
print("Mergesort Linked list :"),
list3.printList()
```

Berikut adalah screenshot hasil ketika program dijalankan: