Nama : Amartya Bintang Wijat Ranti
NIM : L200180193
Kelas : G
Mata Kuliah : Praktikum Algoritma Struktur Data

Modul 6 : Pengurutan lanjutan

1. Ubahlah kode mergeSort dan quickSort diatas agar bisa mengurutkan list yang berisi object-object mhsTIF yang sudah dibuat di Modul 2.

No1.py - E:/Materi Kuliah/Semester 4/Praktikum Alg

File Edit Format Run Options Window Help

```
from Latihan import *
from Mahasiswa import *

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

def urutkanQuick():
    A = []
    for x in Daftar:
        A.append(x.NIM)
    print("Quick Sort")
    quickSort(A)
    for x in convert(A, Daftar):
        print (x.NIM)

def urutkanMerge():
    A = []
    for x in Daftar:
        A.append(x.NIM)
    print("\nMerge Sort")
    mergeSort(A)
    for x in convert(A, Daftar):
        print (x.NIM)

urutkanQuick()
urutkanMerge()
```

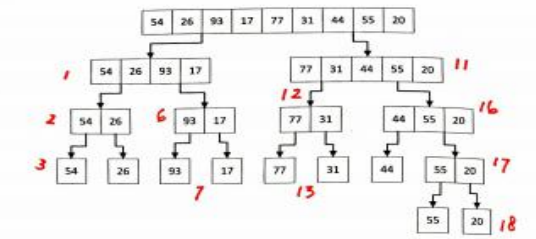
Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

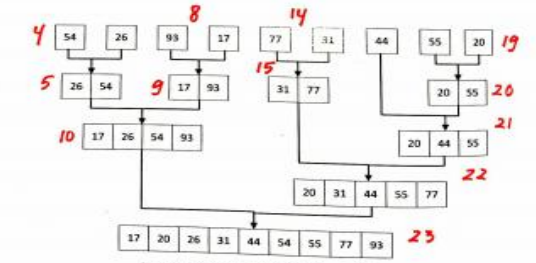
```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 201
4)] on win32
Type "copyright", "credits" or "license()"
>>>
 RESTART: E:/Materi Kuliah/Semester 4/Prakt
l6/No1.py
Quick Sort
173
179
187
188
190
192
193
194
195
204
210
211

Merge Sort
173
179
187
188
190
192
193
194
195
204
210
211
>>>
```

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

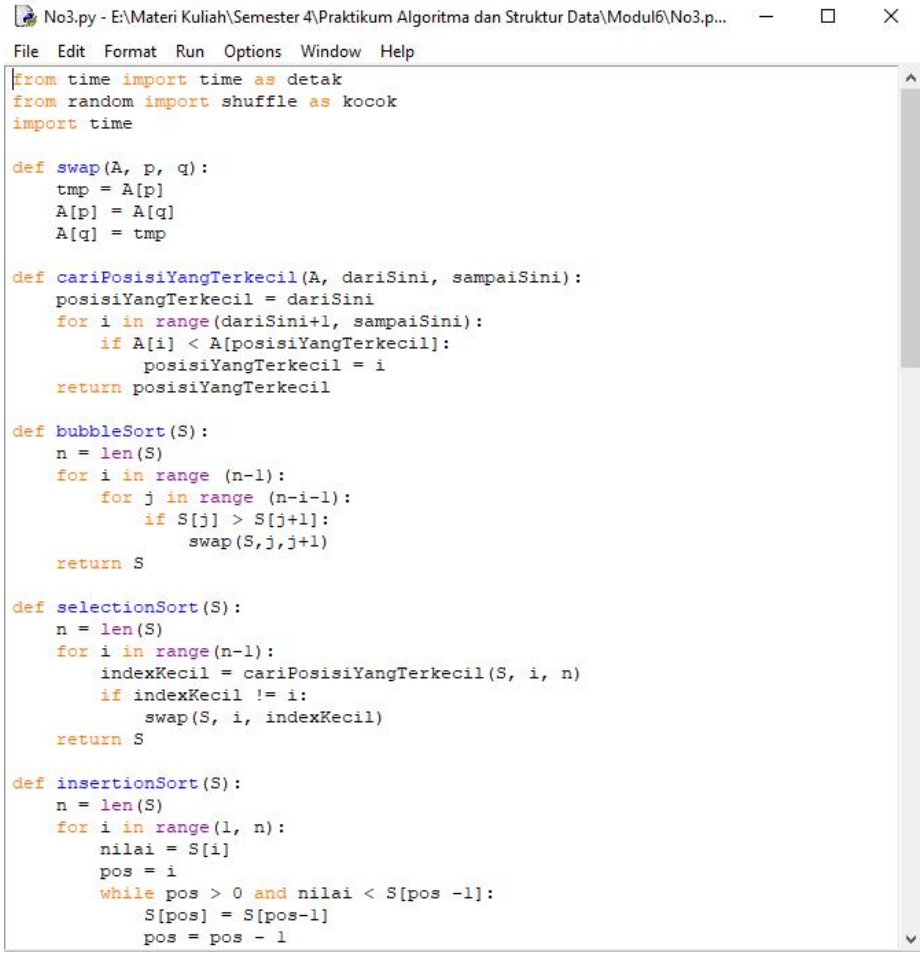


Gambar 6.1: Membelah list sampai tiap sub-list berisi satu elemen atau kosong. Sesudah itu digabung seperti ditunjukkan di Gambar 6.2.



Gambar 6.2: Menggabungkan list satu demi satu.

3. Uji kecepata. Ujilah mergeSort dan quickSort diatas (bersama metode sort yang kamu pelajari sebelumnya) dengan kode berikut




```
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 = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

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));
```

Hasil Run :

```
>>>
 RESTART: E:\Materi Kuliah\Semester 4\Praktikum Algoritma dan Struktur Data\Modu
l6\No3.py
bubble: 9.12455 detik
selection: 5.44228 detik
insertion: 6.93308 detik
merge: 0.0877671 detik
quick: 0.079088 detik
>>>
```

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

a) Merge sort

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

80	7	24	16	43	91	35	2	19	72

Proses 1

7	80		26	24		43	91		2	35		19	72

Proses 2

7	16	24	80		2	35	43	91		19	72

Proses 3

2	7	16	24	35	43	80	91		19	72

Proses 4

2	7	16	19	24	35	43	72	80	91

b) Quick sort

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	43	35	72	80	91
		Low	High						

					Pivot				
2	7	16	19	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 lalu mem-puss index awal dan index akhir bersama listnyabsaat kita memanggil mergeSort secara rekursif. Kamu akan perlu memisah fungsi mergeSort itu menjadi beberapa fungsi, mirip halnya dengan apa yang dilakukan algoritma quick sort

No5.py - E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modul6/No5.p...

File Edit Format Run Options Window Help

```
from Mahasiswa import *

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)

print("Sebelum diurutkan")
cetak(Daftar)
mergeSort(Daftar)
print("\nSetelah diurutkan")
cetak(Daftar)
```

Hasil Run :

```
>>>
 RESTART: E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modu
l6/No5.py
Sebelum diurutkan
Bintang, nim 193. Tinggal di Purwodadi. Uang saku Rp 240000 tiap bulannya.
Ainin, nim 195. Tinggal di Pati. Uang saku Rp 230000 tiap bulannya.
Danang, nim 204. Tinggal di Sragen. Uang saku Rp 250000 tiap bulannya.
Cecyl, nim 210. Tinggal di Surakarta. Uang saku Rp 235000 tiap bulannya.
Alfian, nim 194. Tinggal di Semarang. Uang saku Rp 240000 tiap bulannya.
Aviza, nim 187. Tinggal di Madiun. Uang saku Rp 250000 tiap bulannya.
Baity, nim 211. Tinggal di Klaten. Uang saku Rp 245000 tiap bulannya.
Ulin, nim 190. Tinggal di Madiun. Uang saku Rp 245000 tiap bulannya.
Viola, nim 173. Tinggal di Boyolali. Uang saku Rp 245000 tiap bulannya.
Riska, nim 192. Tinggal di Rembang. Uang saku Rp 270000 tiap bulannya.
Fatwa, nim 179. Tinggal di Boyolali. Uang saku Rp 230000 tiap bulannya.
Sekar, nim 188. Tinggal di Sulawesi. Uang saku Rp 300000 tiap bulannya.

Setelah diurutkan
Fatwa, nim 179. Tinggal di Boyolali. Uang saku Rp 230000 tiap bulannya.
Ainin, nim 195. Tinggal di Pati. Uang saku Rp 230000 tiap bulannya.
Cecyl, nim 210. Tinggal di Surakarta. Uang saku Rp 235000 tiap bulannya.
Alfian, nim 194. Tinggal di Semarang. Uang saku Rp 240000 tiap bulannya.
Bintang, nim 193. Tinggal di Purwodadi. Uang saku Rp 240000 tiap bulannya.
Viola, nim 173. Tinggal di Boyolali. Uang saku Rp 245000 tiap bulannya.
Ulin, nim 190. Tinggal di Madiun. Uang saku Rp 245000 tiap bulannya.
Baity, nim 211. Tinggal di Klaten. Uang saku Rp 245000 tiap bulannya.
Aviza, nim 187. Tinggal di Madiun. Uang saku Rp 250000 tiap bulannya.
Danang, nim 204. Tinggal di Sragen. Uang saku Rp 250000 tiap bulannya.
Riska, nim 192. Tinggal di Rembang. Uang saku Rp 270000 tiap bulannya.
Sekar, nim 188. Tinggal di Sulawesi. Uang saku Rp 300000 tiap bulannya.
>>>
```

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

No6.py - E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modul6/No6.p...

File Edit Format Run Options Window Help

```
from Mahasiswa import *

def cetak(A):
    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.ambilUangSaku() < pivot.ambilUangSaku():
                kurang.append(i)
            elif i.ambilUangSaku() > pivot.ambilUangSaku():
                lebih.append(i)
            else:
                pivotList.append(i)
        kurang = quickSort(kurang)
        lebih = quickSort(lebih)
        return kurang + pivotList + lebih

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

Hasil Run :

```
>>>
 RESTART: E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modu
l6/No6.py
Sebelum diurutkan
Bintang, nim 193. Tinggal di Purwodadi. Uang saku Rp 240000 tiap bulannya.
Ainin, nim 195. Tinggal di Pati. Uang saku Rp 230000 tiap bulannya.
Danang, nim 204. Tinggal di Sragen. Uang saku Rp 250000 tiap bulannya.
Cecyl, nim 210. Tinggal di Surakarta. Uang saku Rp 235000 tiap bulannya.
Alfian, nim 194. Tinggal di Semarang. Uang saku Rp 240000 tiap bulannya.
Aviza, nim 187. Tinggal di Madiun. Uang saku Rp 250000 tiap bulannya.
Baity, nim 211. Tinggal di Klaten. Uang saku Rp 245000 tiap bulannya.
Ulin, nim 190. Tinggal di Madiun. Uang saku Rp 245000 tiap bulannya.
Viola, nim 173. Tinggal di Boyolali. Uang saku Rp 245000 tiap bulannya.
Riska, nim 192. Tinggal di Rembang. Uang saku Rp 270000 tiap bulannya.
Fatwa, nim 179. Tinggal di Boyolali. Uang saku Rp 230000 tiap bulannya.
Sekar, nim 188. Tinggal di Sulawesi. Uang saku Rp 300000 tiap bulannya.

Setelah diurutkan
Bintang, nim 193. Tinggal di Purwodadi. Uang saku Rp 240000 tiap bulannya.
Ainin, nim 195. Tinggal di Pati. Uang saku Rp 230000 tiap bulannya.
Danang, nim 204. Tinggal di Sragen. Uang saku Rp 250000 tiap bulannya.
Cecyl, nim 210. Tinggal di Surakarta. Uang saku Rp 235000 tiap bulannya.
Alfian, nim 194. Tinggal di Semarang. Uang saku Rp 240000 tiap bulannya.
Aviza, nim 187. Tinggal di Madiun. Uang saku Rp 250000 tiap bulannya.
Baity, nim 211. Tinggal di Klaten. Uang saku Rp 245000 tiap bulannya.
Ulin, nim 190. Tinggal di Madiun. Uang saku Rp 245000 tiap bulannya.
Viola, nim 173. Tinggal di Boyolali. Uang saku Rp 245000 tiap bulannya.
Riska, nim 192. Tinggal di Rembang. Uang saku Rp 270000 tiap bulannya.
Fatwa, nim 179. Tinggal di Boyolali. Uang saku Rp 230000 tiap bulannya.
Sekar, nim 188. Tinggal di Sulawesi. Uang saku Rp 300000 tiap bulannya.
>>>
```

7. Uji kecepatan keduanya dan perbandingkan juga dengan kode awalnya

No7.py - E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modul6/No7.p...

File Edit Format Run Options Window Help

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

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)

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

    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 mergeSortNew(A):
    mergeSort2(A, 0, len(A)-1)

def quickSortNew(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 = quickSortNew(kurang)
        lebih = quickSortNew(lebih)
```

```
        return kurang + pivotList + lebih

daftar = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

mergeSort(daftar)
print (daftar)
quickSort(daftar)
print (daftar)
mergeSortNew(daftar)
print (daftar)
quickSortNew(daftar)
print (daftar)

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_mrg = k[:]
u_qck = k[:]
u_mrgNew = k[:]
u_qckNew = k[:]

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));
aw=detak();mergeSortNew(u_mrgNew);ak=detak();print("merge New: %g detik" %(ak-aw));
aw=detak();quickSortNew(u_qckNew);ak=detak();print("quick New: %g detik" %(ak-aw));
```

Hasil Run :

```
>>>
 RESTART: E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modu
l6/No7.py
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
merge: 0.0745022 detik
quick: 0.0737064 detik
merge New: 0.0829716 detik
quick New: 0.0359025 detik
>>>
```

8. Buatlah versi linked-list untuk program mergeSort diatas

No8.py - E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modul6/No8.p...

File Edit Format Run Options Window Help

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

def cetak(head):
    curr = head
    while curr is not None:
        try:
            print (curr.data)
            curr = curr.tautan
        except:
            pass

a = Node(1)
b = Node(3)
c = Node(5)
d = Node(7)
e = Node(2)
f = Node(4)
g = Node(6)

a.tautan = b
b.tautan = c
c.tautan = d
d.tautan = e
e.tautan = f
f.tautan = g

def mergeSortLL(A):
    linked = A
    try:
        daftar = []
        curr = A
        while curr:
            daftar.append(curr.data)
            curr = curr.tautan
        A = daftar
```

```
    except:
        A = A

    if len(A) > 1:
        mid = len(A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]

        mergeSortLL(separuhkiri)
        mergeSortLL(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

    for x in A:
        try:
            linked.data = x
            linked = linked.tautan
        except:
            pass

mergeSortLL(a)
```

```
cetak(a)
```

Hasil Run :

```
>>>
 RESTART: E:/Materi Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modu
l6/No8.py
1
2
3
4
5
6
7
>>>
```