L200180195

G

MODUL 6

- 1. Ubah kode mergeSort dan quickSort agar bias mengurutkan list yang berisi object-object mhsTIF
 - mergeSort

```
- 15 DI-
  3 1 MergeSort.py - E:/0000000 UNI STUFFS/PRAK ALGOSTRUK/L2001801
  File Edit Format Run Options Window Help
  class MhsTIF():
     def __init__(self, nim):
          self.nim = nim
     def __str__(self):
         return str(self.nim)
  m0 = MhsTIF(10)
 ml = MhsTIF(51)
 m2 = MhsTIF(2)
 m3 = MhsTIF(18)
 m4 = MhsTIF(4)
 m5 = MhsTIF(31)
 m6 = MhsTIF(13)
  m7 = MhsTIF(5)
  m8 = MhsTIF(23)
 m9 = MhsTIF(64)
 m10 = MhsTIF(29)
  m0.next = m1
  ml.next = m2
 m2.next = m3
 m3.next = m4
 m4.next = m5
 m5.next = m6
 m6.next = m7
  m7.next = m8
  m8.next = m9
 m9.next = m10
  def mergeSort(A):
      #print("Membelah
      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):</pre>
            if separuhkiri[i] < separuhkanan[j]:</pre>
                 A[k] = separuhkiri[i]
                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):</pre>
             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])
{\tt Daftar = [m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10]}
A = []
for x in Daftar:
    A.append(x.nim)
print("MERGE SORT")
mergeSort(A)
for x in convert(A, Daftar):
   print (x.nim)
```

```
RESTART: E:/000000
MERGE SORT
2
4
5
10
13
18
23
29
31
51
64
```

quickSort

```
while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
class MhsTIF():
                                                                                        penandakanan = penandakanan - 1
    def __init__(self, nim):
          self.nim = nim
                                                                                    if penandakanan < penandakiri:</pre>
                                                                                        selesai = True
                                                                                    else:
     def __str__(self):
                                                                                        temp = A[penandakiri]
          return str(self.nim)
                                                                                        A[penandakiri] = A[penandakanan]
A[penandakanan] = temp
m0 = MhsTIF(10)
ml = MhsTIF(51)
                                                                               temp = A[awal]
A[awal] = A[penandakanan]
m2 = MhsTIF(2)
m3 = MhsTIF(18)
                                                                               A[penandakanan] = temp
m4 = MhsTIF(4)
                                                                                return penandakanan
m5 = MhsTIF(31)
                                                                            def quickSortBantu(A, awal, akhir):
m6 = MhsTIF(13)
                                                                               if awal < akhir:
   titikBelah = partisi(A, awal, akhir)
   quickSortBantu(A, awal, titikBelah-1)
   quickSortBantu(A, titikBelah+1, akhir)</pre>
m7 = MhsTIF(5)
m8 = MhsTIF(23)
m9 = MhsTIF(64)
m10 = MhsTIF(29)
                                                                            def quickSort(A):
                                                                               quickSortBantu (A, 0, len(A)-1)
m0.next = m1
ml.next = m2
                                                                           def convert(arr, obj):
m2.next = m3
                                                                               hasil=[]
m3.next = m4
                                                                               for x in range (len(arr)):
                                                                                    for i in range (len(arr)):
    if arr[x] == obj[i].nim:
        hasil.append(obj[i])
m4.next = m5
m5.next = m6
m6.next = m7
                                                                               return hasil
m7.next = m8
m8.next = m9
                                                                           Daftar = [m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10]
m9.next = m10
                                                                           A = []
for x in Daftar:
                                                                              A.append(x.nim)
def partisi(A, awal, akhir):
    nilaipivot = A[awal]
                                                                           print("QUICK SORT")
                                                                           quickSort(A)
     penandakiri = awal + 1
                                                                            for x in convert(A, Daftar):
    print (x.nim)
     penandakanan = akhir
     selesai = False
     while not selesai:
          while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
               penandakiri = penandakiri + 1
```

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

- 2. Menulis pakai bolpen merah dan biru
- 3. Uji kecepatan

```
return S
from time import time as detak
from random import shuffle as kocok
                                                              def mergeSort(A):
import time
                                                                  #print("Membelah
                                                                  if len(A) > 1:
def swap(A, p, q):
                                                                     mid = len(A) // 2
    tmp = A[p]
                                                                      separuhkiri = A[:mid]
    A[p] = A[q]
                                                                      separuhkanan = A[mid:]
    A[q] = tmp
                                                                     mergeSort(separuhkiri)
def cariPosisiYangTerkecil(A, dariSini, sampaiSini):
                                                                      mergeSort (separuhkanan)
    posisiYangTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
                                                                      i = 0; j=0; k=0
       if A[i] < A[posisiYangTerkecil]:</pre>
                                                                      while i < len(separuhkiri) and j < len(separuhkanan):</pre>
            posisiYangTerkecil = i
                                                                          if separuhkiri[i] < separuhkanan[j]:</pre>
    return posisiYangTerkecil
                                                                              A[k] = separuhkiri[i]
                                                                              i = i + 1
def bubbleSort(S):
                                                                          else:
    n = len(S)
                                                                             A[k] = separuhkanan[j]
    for i in range (n-1):
                                                                              j = j + 1
       for j in range (n-i-1):
                                                                          k=k+1
            if S[j] > S[j+1]:
                swap(S,j,j+1)
                                                                      while i < len(separuhkiri):</pre>
    return S
                                                                          A[k] = separuhkiri[i]
                                                                          i = i + 1
def selectionSort(S):
                                                                          k=k+1
   n = len(S)
    for i in range(n-1):
                                                                      while j < len(separuhkanan):</pre>
        indexKecil = cariPosisiYangTerkecil(S, i, n)
                                                                          A[k] = separuhkanan[j]
        if indexKecil != i:
                                                                          j = j + 1
           swap(S, i, indexKecil)
                                                                          k=k+1
    return S
                                                                  #print("Menggabungkan",A)
def insertionSort(S):
                                                              def partisi(A, awal, akhir):
    n = len(S)
                                                                  nilaipivot = A[awal]
    for i in range(1, n):
       nilai = S[i]
                                                                  penandakiri = awal + 1
        pos = i
                                                                  penandakanan = akhir
        while pos > 0 and nilai < S[pos -1]:</pre>
           S[pos] = S[pos-1]
                                                                  selesai = False
           pos = pos - 1
                                                                  while not selesai:
        S[pos] = nilai
```

```
while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
            penandakiri = penandakiri + 1
        while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
           penandakanan = penandakanan - 1
        if penandakanan < penandakiri:</pre>
            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:</pre>
        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]
print (bubbleSort(daftar))
print (selectionSort(daftar))
print (insertionSort(daftar))
mergeSort(daftar)
print (daftar)
quickSort (daftar)
print (daftar)
k = [[i] \text{ 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));
```

```
[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]

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

bubble: 4.29523 detik

selection: 1.75247 detik
```

- 4. Diberikan list L
- Tingkatkan efisiensi mergeSort dengan tidak menggunakan operator Slice dan lalu mempass indek awal dan index akhir bersama list-nya saat kita memanggil mergeSort secara rekursif.

```
class MhsTIF():
                                                                 def mergeSort2(A, awal, akhir):
    def init (self, nama, nim, kota, us):
         self.nama = nama
                                                                    mid = (awal+akhir)//2
        self.nim = nim
                                                                     if awal < akhir:
        self.kota = kota
                                                                         mergeSort2(A, awal, mid)
        self.us = us
                                                                         mergeSort2(A, mid+1, akhir)
          str (self):
                                                                     a, f, l = 0, awal, mid+l
        s = self.nama +', NIM '+str(self.nim)\
                                                                     tmp = [None] * (akhir - awal + 1)
             +'. Tinggal di '+ self.kota \
                                                                     while f <= mid and l <= akhir:
             +'. Uang saku Rp. '+ str(self.us) \
                                                                        if A[f].ambilUangSaku() < A[l].ambilUangSaku():</pre>
             +' tiap bulannya.'
                                                                              tmp[a] = A[f]
        return s
                                                                              f += 1
    def ambilNama(self):
                                                                             tmp[a] = A[1]
        return self.nama
                                                                             1 += 1
    def ambilNim(self):
                                                                          a += 1
        return self.nim
    def ambilUangSaku(self):
                                                                     if f <= mid:
        return self.us
                                                                          tmp[a:] = A[f:mid+l]
m0 = MhsTIF("Alfa", 76, "Banyuwangi", 249000)
ml = MhsTIF("Pita", 53, "Purwokerto", 234000)
                                                                     if 1 <= akhir:
m2 = MhsTIF("Octa", 37, "Purworejo", 220000)
m3 = MhsTIF("Ila", 49, "Surakarta", 232000)
                                                                         tmp[a:] = A[1:akhir+1]
m4 = MhsTIF("Uni", 46, "Demak", 300000)
                                                                     a = 0
m5 = MhsTIF("Yeri", 31, "Cilacap", 250000)
m6 = MhsTIF("Tisa", 60, "Kutai", 245000)
                                                                     while awal <= akhir:
                                                                        A[awal] = tmp[a]
m7 = MhsTIF("Roro", 91, "Lembang", 231000)
m8 = MhsTIF("Elvi", 15, "Bogor", 289000)
                                                                          awal += 1
m9 = MhsTIF("Winda", 81, "Pontianak", 250000)
ml0 = MhsTIF("Qina", 43, "Lombok", 550000)
                                                                 def mergeSort(A):
                                                                     mergeSort2(A, 0, len(A)-1)
daftar = [m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10]
def cetak(A):
    for i in A:
        print (i)
```

```
>>> cetak(daftar)
Alfa, NIM 76. Tinggal di Banyuwangi. Uang saku Rp. 249000 tiap bulannya.
Pita, NIM 53. Tinggal di Purwokerto. Uang saku Rp. 234000 tiap bulannya.
Octa, NIM 37. Tinggal di Purworejo. Uang saku Rp. 220000 tiap bulannya.
Ila, NIM 49. Tinggal di Surakarta. Uang saku Rp. 232000 tiap bulannya.
Uni, NIM 46. Tinggal di Demak. Uang saku Rp. 300000 tiap bulannya.
Yeri, NIM 31. Tinggal di Cilacap. Uang saku Rp. 250000 tiap bulannya.
Tisa, NIM 60. Tinggal di Kutai. Uang saku Rp. 245000 tiap bulannya.
Roro, NIM 91. Tinggal di Lembang. Uang saku Rp. 231000 tiap bulannya.
Elvi, NIM 15. Tinggal di Bogor. Uang saku Rp. 289000 tiap bulannya.
Winda, NIM 81. Tinggal di Pontianak. Uang saku Rp. 250000 tiap bulannya.
Qina, NIM 43. Tinggal di Lombok. Uang saku Rp. 550000 tiap bulannya.
>>> mergeSort(daftar)
>>> cetak(daftar)
Octa, NIM 37. Tinggal di Purworejo. Uang saku Rp. 220000 tiap bulannya.
Roro, NIM 91. Tinggal di Lembang. Uang saku Rp. 231000 tiap bulannya.
Ila, NIM 49. Tinggal di Surakarta. Uang saku Rp. 232000 tiap bulannya
Pita, NIM 53. Tinggal di Purwokerto. Uang saku Rp. 234000 tiap bulannya.
Tisa, NIM 60. Tinggal di Kutai. Uang saku Rp. 245000 tiap bulannya.
Alfa, NIM 76. Tinggal di Banyuwangi. Uang saku Rp. 249000 tiap bulannya.
Winda, NIM 81. Tinggal di Pontianak. Uang saku Rp. 250000 tiap bulannya.
Yeri, NIM 31. Tinggal di Cilacap. Uang saku Rp. 250000 tiap bulannya.
Elvi, NIM 15. Tinggal di Bogor. Uang saku Rp. 289000 tiap bulannya.
Uni, NIM 46. Tinggal di Demak. Uang saku Rp. 300000 tiap bulannya.
Qina, NIM 43. Tinggal di Lombok. Uang saku Rp. 550000 tiap bulannya.
```

6. Meningkatkan efisiensi program quicksort dengan memakai metode median dari tiga untuk memilih pivot

```
class MhsTIF():
     def __init__(self, nama, nim, kota, us):
                                                                         def cetak():
          self.nama = nama
                                                                              for i in A:
          self.nim = nim
                                                                                     print(i)
         self.kota = kota
          self.us = us
     def __str__(self):
                                                                               kurang = []
          s = self.nama +', NIM '+str(self.nim) \
              +'. Tinggal di '+ self.kota \
                                                                               lebih = []
               +'. Uang saku Rp. '+ str(self.us)\
              +' tiap bulannya.'
          return s
                                                                               else:
     def ambilNama(self):
          return self.nama
     def ambilNim(self):
          return self.nim
     def ambilUangSaku(self):
          return self.us
m0 = MhsTIF("Alfa", 76, "Banyuwangi", 249000)
ml = MhsTIF("Pita", 53, "Purwokerto", 234000)
m2 = MhsTIF("Octa", 37, "Purworejo", 220000)
m3 = MhsTiF("Ila", 49, "Surakarta", 232000)
m4 = MhsTiF("Uni", 46, "Demak", 300000)
m5 = MhsTiF("Yeri", 31, "Cilacap", 250000)
m6 = MhsTIF("Tisa", 60, "Kutai", 245000)
m7 = MhsTIF("Roro", 91, "Lembang", 231000)
m8 = MhsTIF("Elvi", 15, "Bogor", 289000)
m9 = MhsTIF("Winda", 81, "Pontianak", 250000)
ml0 = MhsTIF("Qina", 43, "Lombok", 550000)
daftar = [m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10]
                                                                         quickSort(A)
                                                                         cetak()
A = [1]
for i in daftar:
    A.append(i.nama)
```

```
def quickSort(arr):
    pivotList = []
    if len(arr) <= 1:
       return arr
        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")
print("\nSetelah diurutkan")
```

Sebelum diurutkan HASIL: Alfa Pita Octa Ila Yeri Tisa Roro Elvi Winda Oina Setelah diurutkan Alfa Pita Octa Ila Uni Yeri Tisa Roro Elvi Winda Qina >>>

7. Uji kecepatan keduanya dan perbandingkan juga dgn kode awalnya

```
from time import time as detak
from random import shuffle as kocok
import time
                                                                        selesai = False
                                                                        while not selesai:
def mergeSort(A):
    #print("Membelah
                           ",A)
                                                                            while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:</pre>
    if len(A) > 1:
                                                                               penandakiri = penandakiri + 1
        mid = len(A) // 2
separuhkiri = A[:mid]
                                                                            while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
        separuhkanan = A[mid:]
                                                                                penandakanan = penandakanan - 1
        mergeSort(separuhkiri)
                                                                            if penandakanan < penandakiri:</pre>
        mergeSort (separuhkanan)
                                                                                selesai = True
        i = 0:i=0:k=0
                                                                                temp = A[penandakiri]
        while i < len(separuhkiri) and j < len(separuhkanan):</pre>
                                                                                A[penandakiri] = A[penandakanan]
                                                                                A[penandakanan] = temp
             if separuhkiri[i] < separuhkanan[j]:</pre>
                 A[k] = separuhkiri[i]
                 i = i + 1
                                                                        temp = A[awal]
                                                                        A[awal] = A[penandakanan]
             else:
                                                                        A[penandakanan] = temp
                A[k] = separuhkanan[j]
                 j = j + 1
                                                                        return penandakanan
             k=k+1
                                                                   def quickSortBantu(A, awal, akhir):
        while i < len(separuhkiri):</pre>
                                                                        if awal < akhir:</pre>
             A[k] = separuhkiri[i]
                                                                            titikBelah = partisi(A, awal, akhir)
             i = i + 1
                                                                            quickSortBantu(A, awal, titikBelah-1)
             k=k+1
                                                                            quickSortBantu(A, titikBelah+1, akhir)
        while j < len(separuhkanan):</pre>
                                                                   def quickSort(A):
             A[k] = separuhkanan[j]
                                                                        quickSortBantu (A, 0, len(A)-1)
             j = j + 1
             k=k+1
                                                                   def mergeSort2(A, awal, akhir):
    #print("Menggabungkan", A)
                                                                        mid = (awal+akhir)//2
                                                                        if awal < akhir:</pre>
def partisi(A, awal, akhir):
                                                                            mergeSort2(A, awal, mid)
    nilaipivot = A[awal]
                                                                            mergeSort2(A, mid+1, akhir)
    penandakiri = awal + 1
    penandakanan = akhir
```

```
a, f, l = 0, awal, mid+l
    tmp = [None] * (akhir - awal + 1)
    while f <= mid and l <= akhir:
        if A[f] < A[l]:</pre>
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[1]
           1 += 1
        a += 1
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if 1 <= akhir:</pre>
        tmp[a:] = A[1: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));
```

```
[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: 0.0312686 detik
quick: 0.0156262 detik
merge New: 0.0468693 detik
quick New: 0.0156167 detik
```

8. Buat versi linked list untuk program mergeSort di atas

```
class Node():
    def __init__(self, data, tautan=None):
                                                        if len(A) > 1:
       self.data = data
        self.tautan = tautan
def cetak(head):
   curr = head
    while curr is not None:
           print (curr.data)
           curr = curr.tautan
        except:
a = Node(1)
b = Node(3)
                                                                else:
c = Node(5)
d = Node(7)
e = Node(2)
                                                                k=k+1
f = Node(4)
g = Node(6)
a.tautan = b
b.tautan = c
c.tautan = d
                                                                k=k+1
d.tautan = e
e.tautan = f
f.tautan = g
def mergeSortLL(A):
                                                                k=k+1
    linked = A
    try:
                                                        for x in A:
       daftar = []
       curr = A
        while curr:
           daftar.append(curr.data)
           curr = curr.tautan
                                                            except:
        A = daftar
                                                               pass
    except:
       A = A
                                                   mergeSortLL(a)
                                                   cetak(a)
```

```
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):</pre>
     if separuhkiri[i] < separuhkanan[j]:</pre>
        A[k] = separuhkiri[i]
         i = i + 1
        A[k] = separuhkanan[j]
         j = j + 1
while i < len(separuhkiri):
    A[k] = separuhkiri[i]
     i = i + 1
while j < len(separuhkanan):
    A[k] = separuhkanan[j]
     j = j + 1
     linked.data = x
    linked = linked.tautan
```

HASIL: