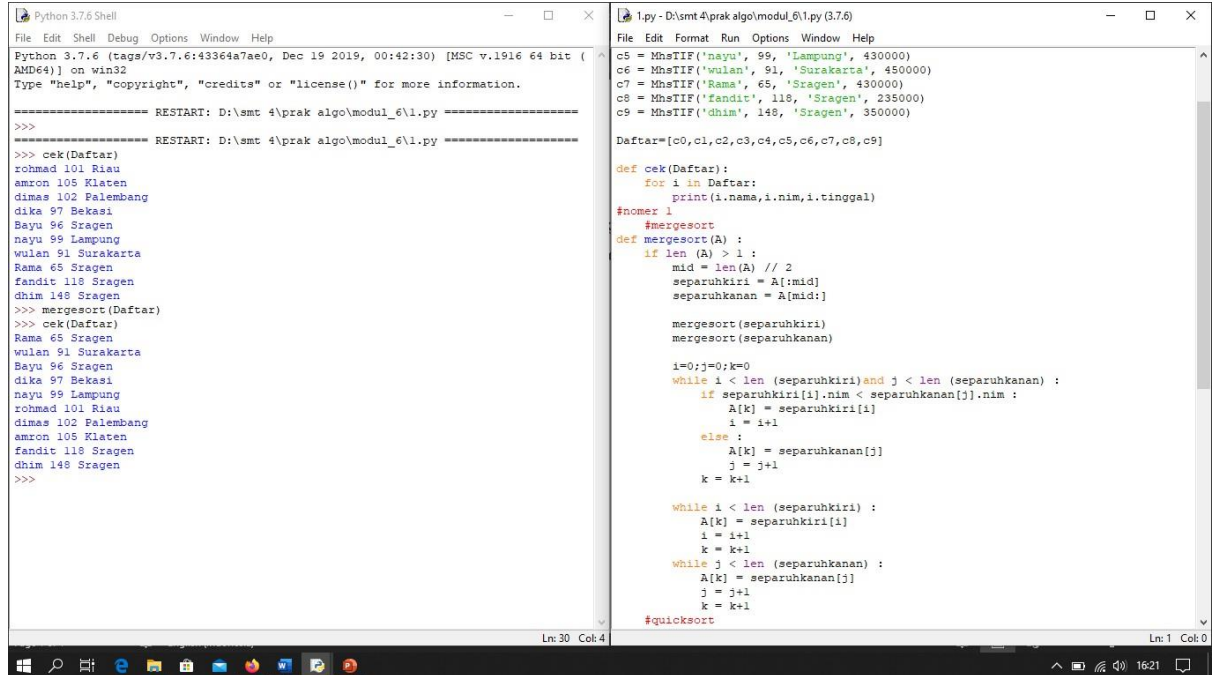


Nama : Chandika Aulia
Nim : L200180097
Kelas : D

Prak-ASD

Modul 6

1. A). Mergesort (mengurutkan MhsTif)



```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\1.py =====
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\1.py =====
>>> cek(Daftar)
rohmad 101 Riau
amron 105 Klaten
dimas 102 Palembang
dika 97 Bekasi
Bayu 96 Sragen
nayu 99 Lampung
wulan 91 Surakarta
Rama 65 Sragen
fandit 118 Sragen
dhim 148 Sragen
>>> mergesort(Daftar)
>>> cek(Daftar)
Rama 65 Sragen
wulan 91 Surakarta
Bayu 96 Sragen
dika 97 Bekasi
nayu 99 Lampung
rohmad 101 Riau
dimas 102 Palembang
amron 105 Klaten
fandit 118 Sragen
dhim 148 Sragen
>>>

1.py - D:\smt 4\prak algo\modul_6\1.py (3.7.6)
File Edit Format Run Options Window Help
c5 = MhsTIF('nayu', 99, 'Lampung', 430000)
c6 = MhsTIF('wulan', 91, 'Surakarta', 450000)
c7 = MhsTIF('Rama', 65, 'Sragen', 430000)
c8 = MhsTIF('fandit', 118, 'Sragen', 235000)
c9 = MhsTIF('dhim', 148, 'Sragen', 350000)

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

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

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

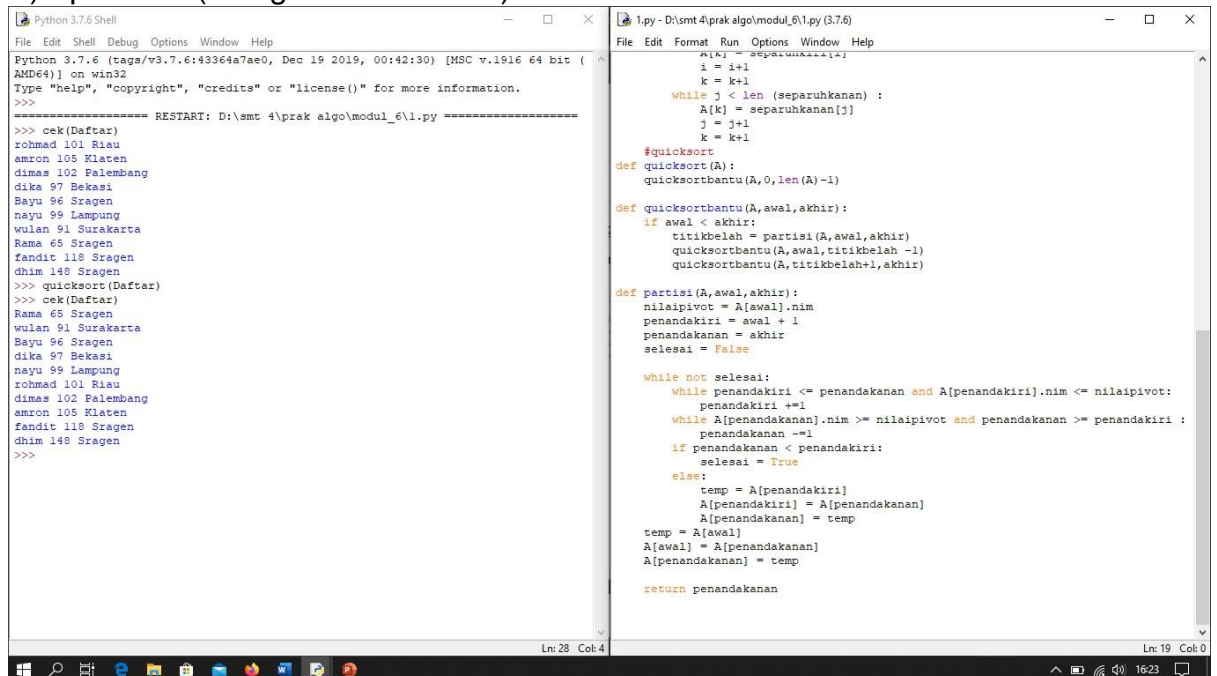
        mergesort(separuhkiri)
        mergesort(separuhkanan)

        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
```

B). quicksort (Mengurutkan MhsTif)



```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\1.py =====
>>> cek(Daftar)
rohmad 101 Riau
amron 105 Klaten
dimas 102 Palembang
dika 97 Bekasi
Bayu 96 Sragen
nayu 99 Lampung
wulan 91 Surakarta
Rama 65 Sragen
fandit 118 Sragen
dhim 148 Sragen
>>> quicksort(Daftar)
>>> cek(Daftar)
Rama 65 Sragen
wulan 91 Surakarta
Bayu 96 Sragen
dika 97 Bekasi
nayu 99 Lampung
rohmad 101 Riau
dimas 102 Palembang
amron 105 Klaten
fandit 118 Sragen
dhim 148 Sragen
>>>

1.py - D:\smt 4\prak algo\modul_6\1.py (3.7.6)
File Edit Format Run Options Window Help
A[i] = separuhkanan[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)

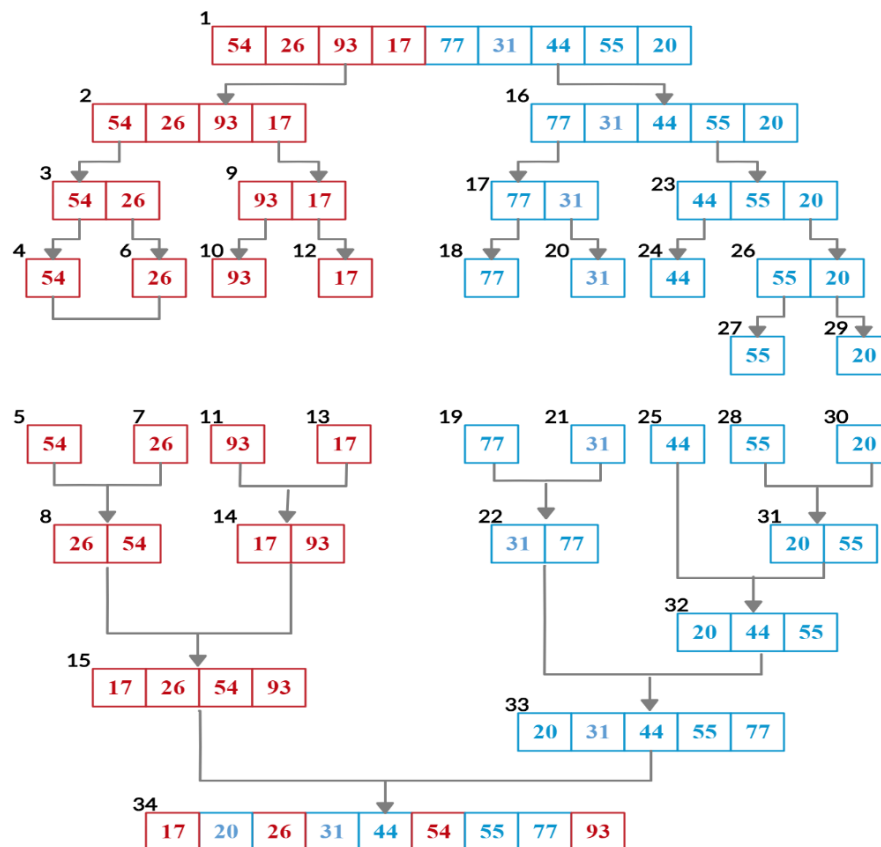
def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah-1)
        quicksortbantu(A,titikbelah+1,akhir)

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
```

2. Beri nomer urut eksekusi proses gambar 6.1 dan 6.2 mengacu pada output di halaman 59



3. Uji kecepatan bubblesort,selectionsort,insertionsort,mergesort dan quicksort

```

Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\3.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]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
bubble: 9.06801 detik
selection: 3.42748 detik
insertion: 4.90588 detik
merge: 0.0468521 detik
quick: 0.0468707 detik
>>>

3.py - D:\smt 4\prak algo\modul_6\3.py (3.7.6)
File Edit Format Run Options Window Help
#penandakanaan = temp

temp = A[awal]
A[awal] = A[penandakanaan]
A[penandakanaan] = temp

return penandakanaan

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)

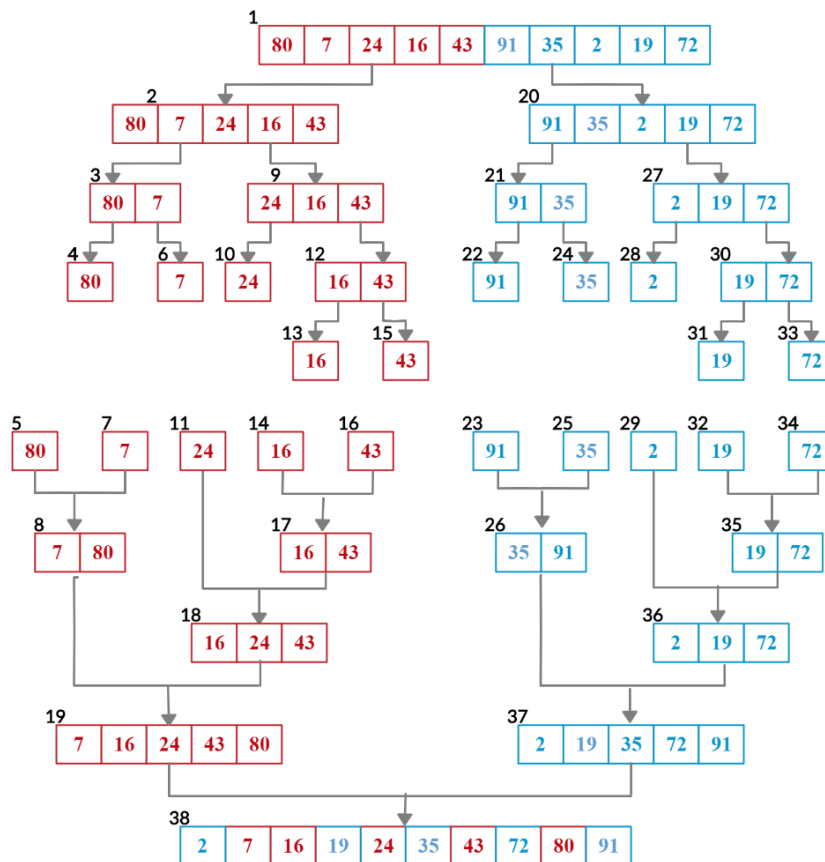
daftar = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

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)]
kacak(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));
  
```

4. A. diberikan List = [80,7,24,16,43,91,35,2,19,72] ,gambarlah trace pengurutan algoritmanya (Merge sort)



- 4 B. diberikan List = [80,7,24,16,43,91,35,2,19,72] ,gambarlah trace pengurutan algoritmanya (quickSort)

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

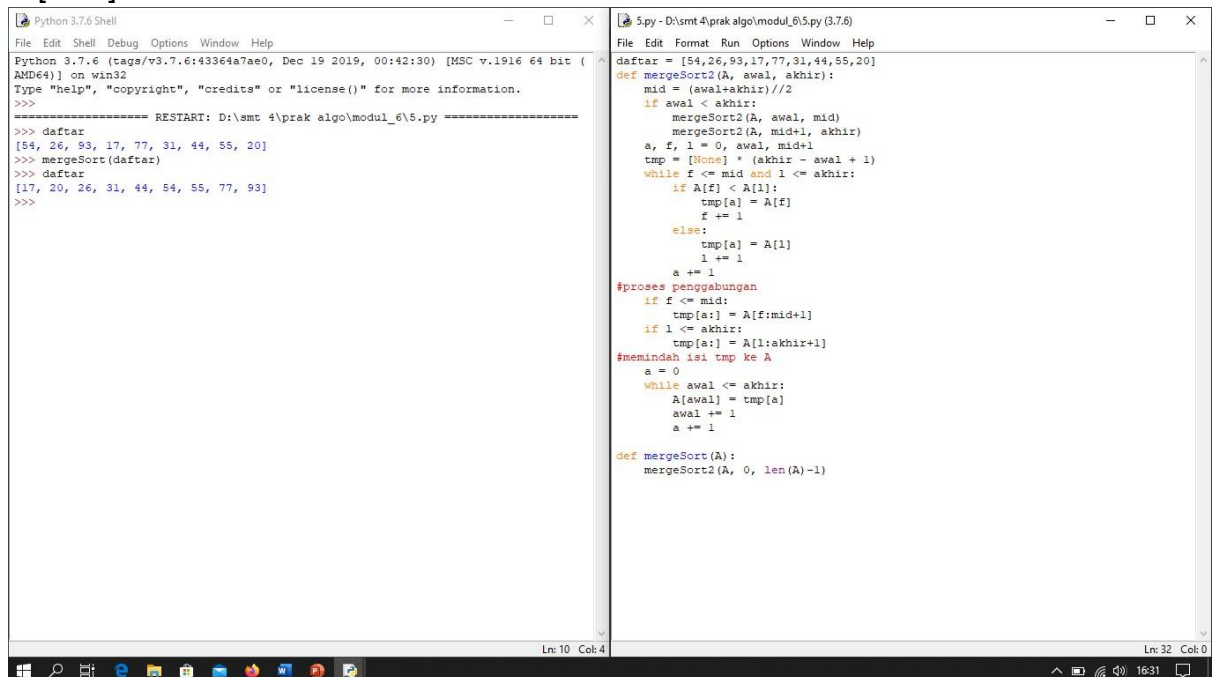
2	7	16	19	24	35	43	72	80	91
---	---	----	----	----	----	----	----	----	----

Low

High

2	7	16	19	24	35	43	72	80	91
---	---	----	----	----	----	----	----	----	----

5. Tingkatkan efisien mergesort dengan tidak memakai operator $A[:mid]$ dan $A[mid:]$

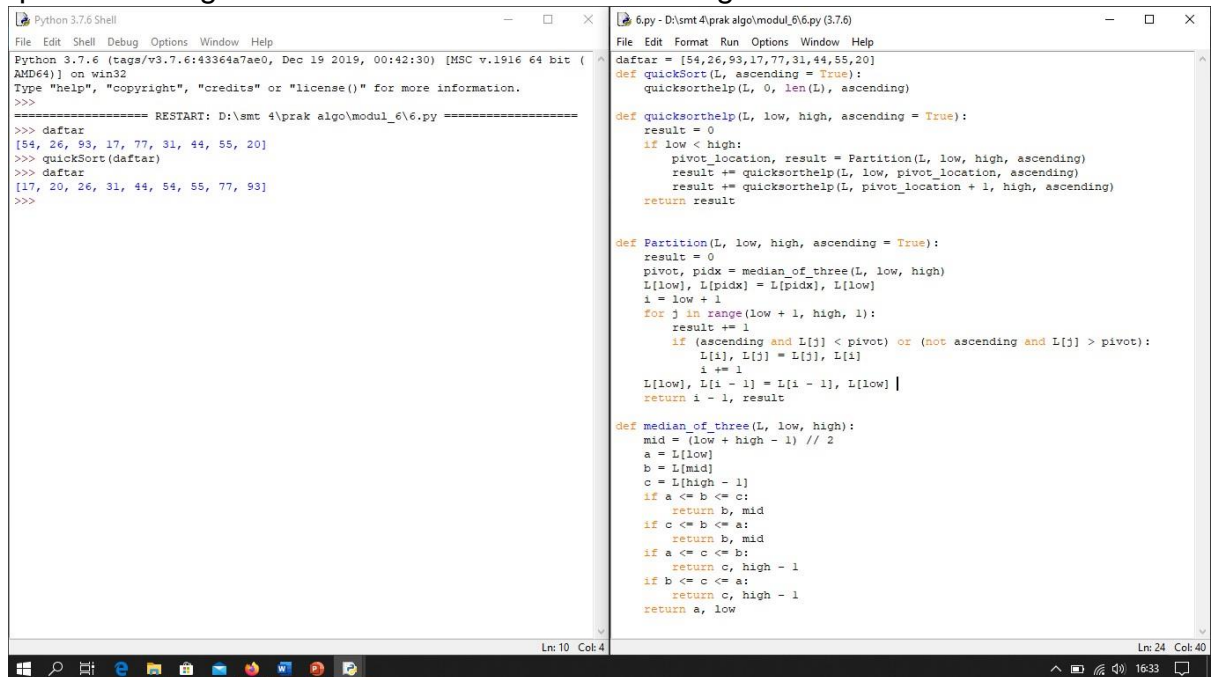


The screenshot shows two windows from a Windows desktop. The left window is a Python 3.7.6 Shell. The right window is a script editor titled '5.py - D:\smt 4\prak algo\modul_6\5.py (3.7.6)'. The shell window shows the execution of a merge sort function on a list of numbers. The script editor shows the implementation of the merge sort algorithm, which uses a recursive function 'mergeSort2' and a helper function 'proses penggabungan' to merge the sorted sub-arrays. The code is written in Python 3.7.6 syntax.

```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\5.py =====
>>> daftar
[54, 26, 93, 17, 77, 31, 44, 55, 20]
>>> mergeSort(daftar)
>>> daftar
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

```
5.py - D:\smt 4\prak algo\modul_6\5.py (3.7.6)
File Edit Format Run Options Window Help
daftar = [54,26,93,17,77,31,44,55,20]
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]
    #memindahkan 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)
```

6. quicksort dengan memakai metode median-dari-tiga



The screenshot shows two windows from a Windows desktop. The left window is a Python 3.7.6 Shell. The right window is a script editor titled '6.py - D:\smt 4\prak algo\modul_6\6.py (3.7.6)'. The shell window shows the execution of a quicksort function on a list of numbers. The script editor shows the implementation of the quicksort algorithm, which uses a recursive function 'quickSort' and a helper function 'Partition' to partition the array around a pivot. The code is written in Python 3.7.6 syntax.

```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\6.py =====
>>> daftar
[54, 26, 93, 17, 77, 31, 44, 55, 20]
>>> quickSort(daftar)
>>> daftar
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

```
6.py - D:\smt 4\prak algo\modul_6\6.py (3.7.6)
File Edit Format Run Options Window Help
daftar = [54,26,93,17,77,31,44,55,20]
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, idx = median_of_three(L, low, high)
    L[low], L[idx] = L[idx], 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
```

7. uji kecepatan program nmr 5, 6 ,mergesort (awal) dan quicksort (akhir)

The screenshot shows two windows from a Windows desktop. The left window is a Python 3.7.6 Shell with the following output:

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\7.py =====
mergesort: 0.078126 detik
mergesort terbaru: 0.0781353 detik
quicksort: 0.0468318 detik
quicksort terbaru: 0.0468888 detik
>>>
```

The right window shows a Python script named 7.py with the following code:

```
def mergesort(L, low, high):
    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 = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
from time import time as detik
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));
```

8. versi linked list mergesort

The screenshot shows two windows from a Windows desktop. The left window is a Python 3.7.6 Shell with the following output:

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_6\8.py =====
List 1 :
16
17
33
48
92
List 2 :
10
18
23
Merged List :
10
16
17
18
23
33
48
92
>>>
```

The right window shows a Python script named 8.py with the following code:

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

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

    def appendSorted(self, data):
        if self.head is None:
            self.head = Node(data)
            return
        curr = self.head
        while curr.next is not None:
            curr = curr.next
        curr.next = Node(data)

    def mergeSort(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.mergeSort(list1.next, list2)
        else:
            temp = list2
            temp.next = self.mergeSort(list1, list2.next)
        return temp

list1 = LinkedList()
list1.appendSorted(49)
list1.appendSorted(92)
list1.appendSorted(33)
list1.appendSorted(16)
list1.appendSorted(17)

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

list2 = LinkedList()
list2.appendSorted(23)
list2.appendSorted(10)
list2.appendSorted(18)

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

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

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