

Nama : Bayu Prayitno Aji

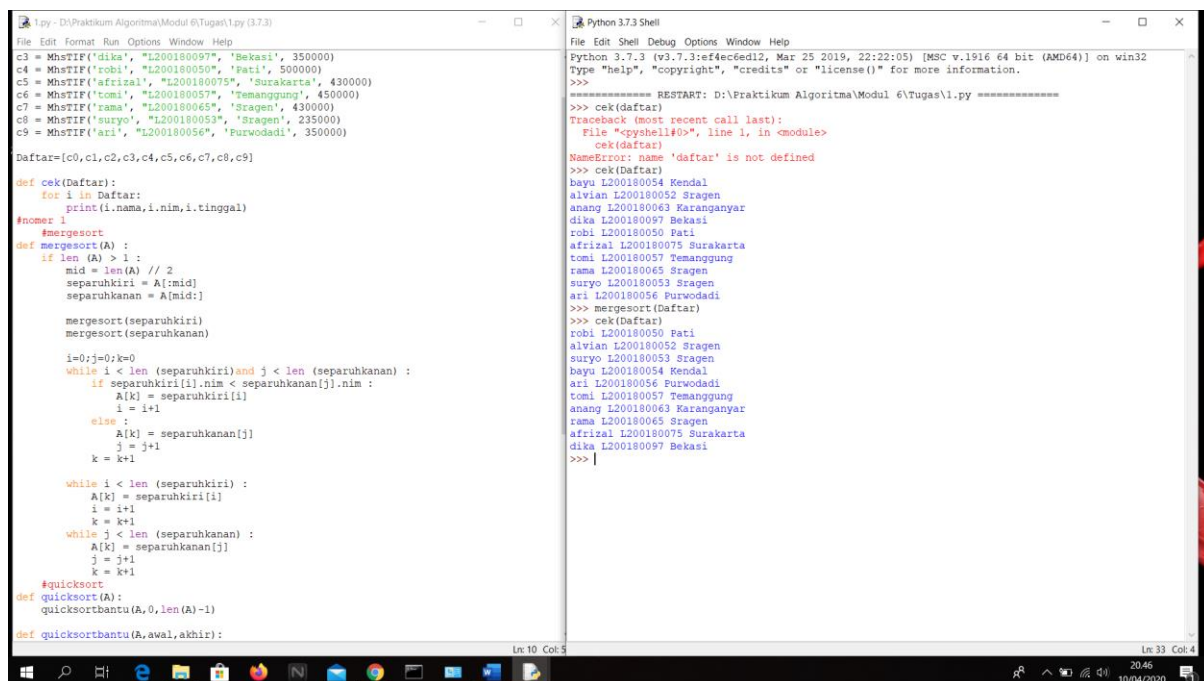
NIM : L200180054

Kelas : C

## Modul 6

- Tugas

### 1. A) Mergesort (mengurutkan mhstif)



```
1.py - D:\Praktikum Algoritma\Modul 6\Tugas\1.py (3.7.3)
File Edit Format Run Options Window Help
c3 = MhsTIF('dika', "L200180097", 'Bekasi', 350000)
c4 = MhsTIF('robi', "L200180050", 'Pati', 500000)
c5 = MhsTIF('afriзал', "L200180075", 'Surakarta', 430000)
c6 = MhsTIF('tomi', "L200180057", 'Temanggung', 450000)
c7 = MhsTIF('rama', "L200180065", 'Sragen', 430000)
c8 = MhsTIF('suryo', "L200180053", 'Sragen', 235000)
c9 = MhsTIF('ari', "L200180056", 'Purwodadi', 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)
#nomor 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
def quicksort(A):
    quicksortbantu(A,0,len(A)-1)

def quicksortbantu(A,awal,akhir):

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Praktikum Algoritma\Modul 6\Tugas\1.py =====
>>> cek(Daftar)
>>> cek(Daftar)
Traceback (most recent call last):
  File "<cpyshell>", line 1, in <module>
    cek(Daftar)
NameError: name 'daftar' is not defined
>>> cek(Daftar)
bayu L200180054 Kendal
alvian L200180052 Sragen
anang L200180063 Karanganyar
dika L200180097 Bekasi
robi L200180050 Pati
afriзал L200180075 Surakarta
tomi L200180057 Temanggung
rama L200180065 Sragen
suryo L200180053 Sragen
ari L200180056 Purwodadi
>>> mergesort(Daftar)
>>> cek(Daftar)
robi L200180050 Pati
alvian L200180052 Sragen
suryo L200180053 Sragen
bayu L200180054 Kendal
ari L200180056 Purwodadi
tomi L200180057 Temanggung
anang L200180063 Karanganyar
rama L200180065 Sragen
afriзал L200180075 Surakarta
dika L200180097 Bekasi
>>>
```

### B) Quicksort (mengurutkan mhstif)

The image shows a screenshot of a Python 3.7.3 IDE with two windows. The left window, titled '1.py - D:\Praktikum Algoritma\Modul 6\Tugas\1.py (3.7.3)', contains a Python script implementing a quicksort algorithm. The script defines a list 'A' with names and NIMs, and a function 'quicksort' that sorts the list in place. The right window, titled 'Python 3.7.3 Shell', shows the execution of the script. It displays the initial list 'Daftar' and the output of the 'quicksort' function, which is a sorted list 'Daftar'.

```
1.py - D:\Praktikum Algoritma\Modul 6\Tugas\1.py (3.7.3)
File Edit Format Run Options Window Help

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

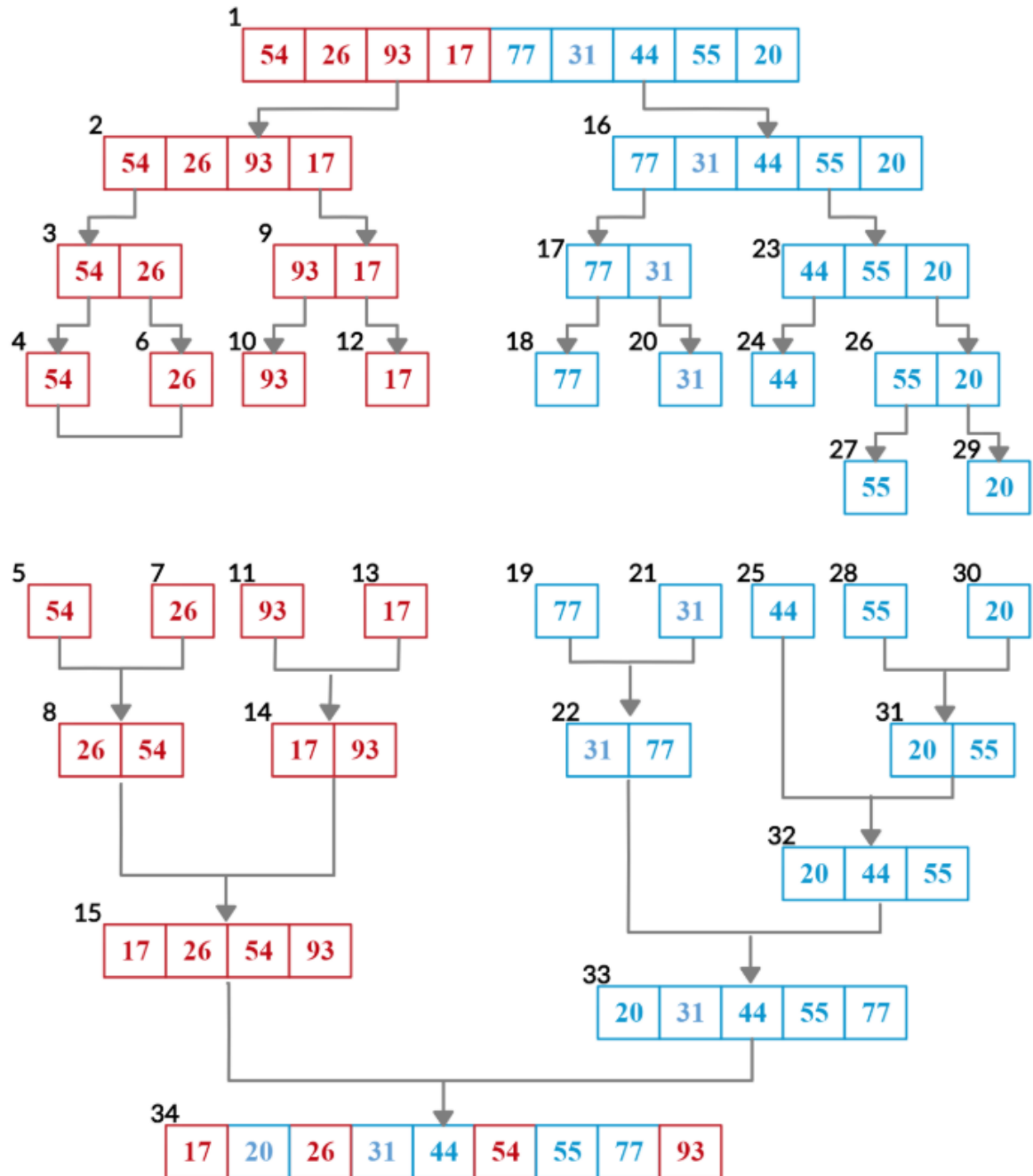
    return penandakanan

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

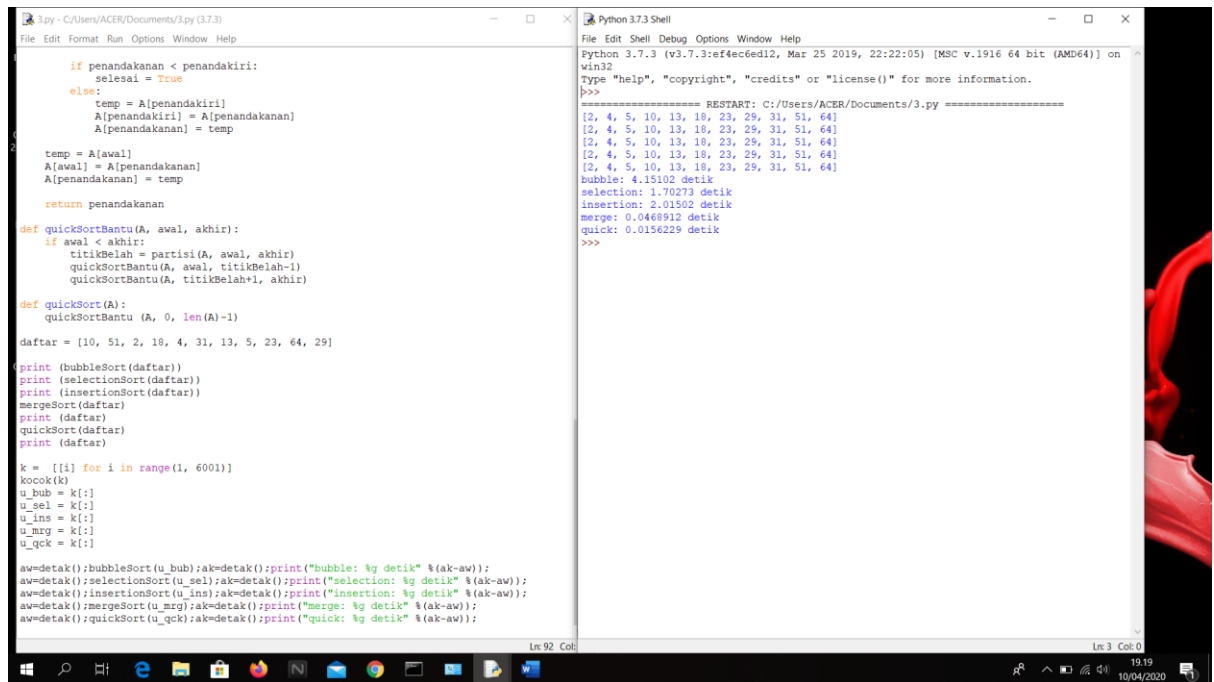
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Praktikum Algoritma\Modul 6\Tugas\1.py =====
>>> cek(Daftar)
bayu L200180054 Kendal
alvian L200180052 Sragen
anang L200180063 Karanganyar
dika L200180097 Bekasi
robi L200180050 Pati
afriзал L200180075 Surakarta
tomi L200180057 Temanggung
rama L200180065 Sragen
suryo L200180053 Sragen
ari L200180056 Purwodadi
>>> quicksort(Daftar)
>>> cek(Daftar)
robi L200180050 Pati
alvian L200180052 Sragen
suryo L200180053 Sragen
bayu L200180054 Kendal
ari L200180056 Purwodadi
tomi L200180057 Temanggung
anang L200180063 Karanganyar
rama L200180065 Sragen
afriзал L200180075 Surakarta
dika L200180097 Bekasi
>>>
```

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

## halaman 58



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



```
3.py - C:/Users/ACER/Documents/3.py (3.7.3)
File Edit Format Run Options Window Help

    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]

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[:]

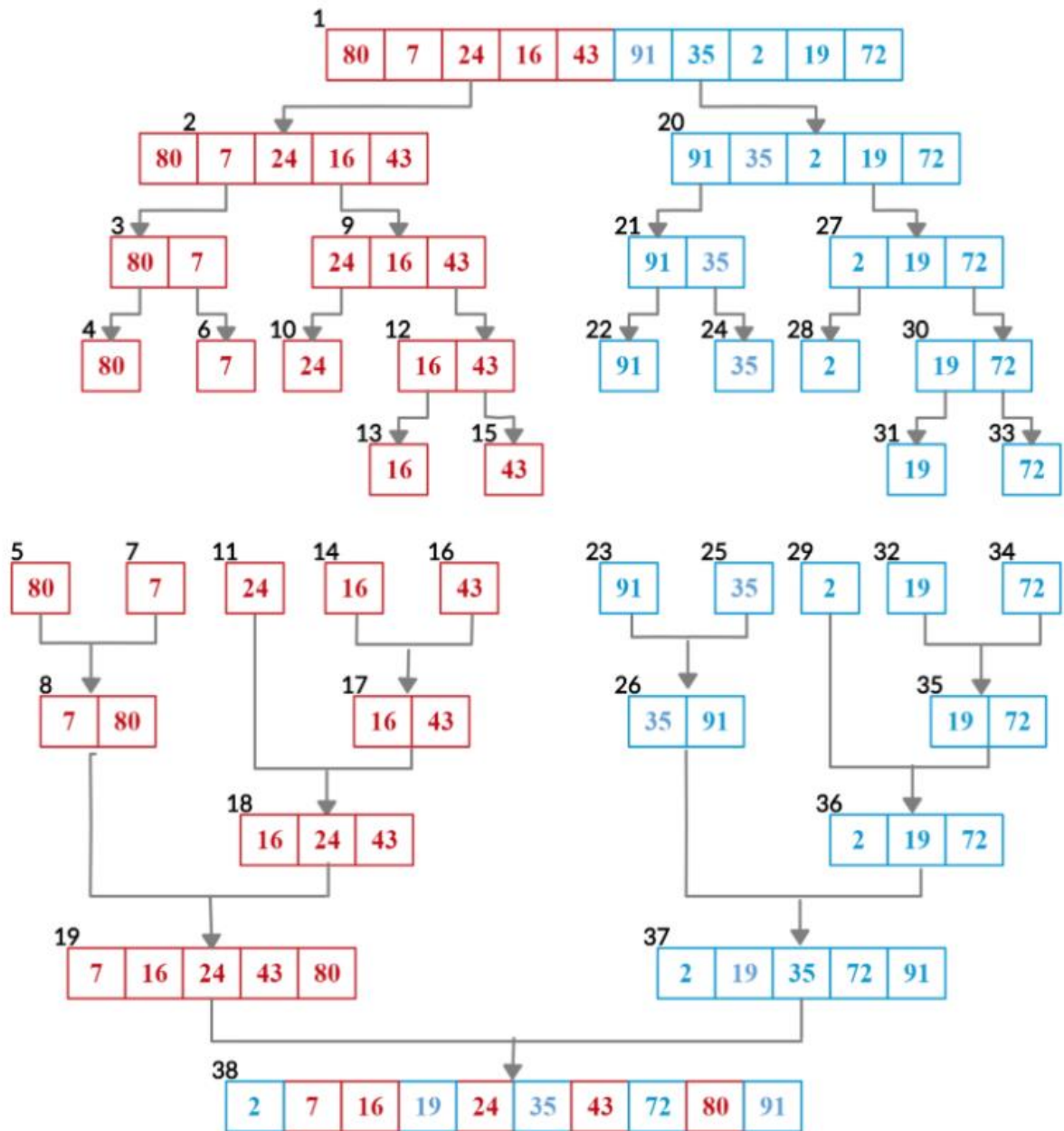
aw=detak();bubbleSort(u_bub);ak=detak();print("bubble: %g detik" %(ak-aw));
aw=detak();selectionSort(u_sel);ak=detak();print("selection: %g detik" %(ak-aw));
aw=detak();insertionSort(u_ins);ak=detak();print("insertion: %g detik" %(ak-aw));
aw=detak();mergeSort(u_mrg);ak=detak();print("merge: %g detik"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

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6d12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/ACER/Documents/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]
bubble: 4.15102 detik
selection: 1.70273 detik
insertion: 2.01502 detik
merge: 0.0468912 detik
quick: 0.0156229 detik
>>>
```

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

## nomer 4 . merge sort



B) Diberikan List = [80,7,24,16,43,91,35,2,19,72] ,gambarlah trace pengurutan algoritmanya (quicksort)

## QuickSort

List = [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 efisien megersort dengan tidak memakai operator A[:mid] dan A[mid:]

The screenshot shows a Python 3.7.3 IDE with a file named '5.py'. The code implements a merge sort algorithm. The initial array is [54, 26, 93, 17, 77, 31, 44, 55, 20]. The code uses a recursive function 'mergeSort2' to sort the array. The execution output shows the array being sorted into [17, 20, 26, 31, 44, 54, 55, 77, 93].

```

5.py - C:/Users/ACER/Documents/5.py (3.7.3)
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]
    #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)

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/ACER/Documents/5.py =====
>>> daftar
[54, 26, 93, 17, 77, 31, 44, 55, 20]
>>> mergeSort(daftar)
>>> daftar
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
  
```

##### 6. Quicksort dengan memakai metode median-dari-tiga



```
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

Python 3.7.3 Shell
Python 3.7.3 (v3.7.3:ef4ec6d12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/ACER/Documents/6.py =====
>>> daftar
[54, 26, 93, 17, 77, 31, 44, 55, 20]
>>> quickSort(daftar)
>>> daftar
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> |
```

## 7. Uji kecepatan program nomor 5 dan 6 mergesort (awal) dan quicksort (akhir)

```
File Edit Format Run Options Window Help
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

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));

Python 3.7.3 Shell
Python 3.7.3 (v3.7.3:ef4ec6d12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/ACER/Documents/7.py =====
mergesort          : 0.0312612 detik
mergesort terbaru  : 0.0312424 detik
quicksort          : 0.0156245 detik
quicksort terbaru  : 0.0312397 detik
>>>
```

## 8. Versi linked list mergesort

```
8.py - C:/Users/ACER/Documents/8.py (3.7.3)
File Edit Format Run Options Window Help
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(48)
list1.appendSorted(92)
list1.appendSorted(33)
list1.appendSorted(16)
list1.appendSorted(17)

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

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

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

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

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

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 [v3.7.3:ef4ec6d12, Mar 25 2019, 22:22:05] [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/ACER/Documents/8.py =====
List 1 :
16
17
33
48
92
List 2 :
10
18
23
Mergesort Linked list :
10
16
17
18
23
33
48
92
>>>
```