# LATIHAN DAN TUGAS PRAKTIKUM
# MODUL 6
# PRAKTIKUM ALGORITMA DAN STRUKTUR DATA


DISUSUN OLEH:

Nama : Syifaul Qolbi Auliya' Darojat

NIM : L200200141

KELAS : F


## UNIVERSITAS MUHAMMADIYAH SURAKARTA

Latihan 1

```
def gabungkanDuaListUrut(A,B):
    la = len(A); lb = len(B)
    C = list()
    i = 0; j = 0

    while i < la and j < lb:
        if A[1] < B[j]:
            C.append(A[i])
            i += 1
        else :
            C.append(B[j])
            j += 1
    while i < la:
        C.append(A[i])
        i += 1
    while j < lb:
        C.append(B[j])
        j += 1
    return C
```

Hasil :

```
>>> P = [2,8,15,23,37]
>>> Q = [4,6,15,20]
>>> R = gabungkanDuaListUrut(P, Q)
>>> print(R)
[4, 6, 2, 8, 15, 23, 37, 15, 20]
>>>
```

Latihan 2

```
def mergeShort(A):
    if len(A) > 1:
        mid = len(A) // 2
        separuhKiri = A[:mid]
        separuhKanan = A[mid:]

        mergeShort(separuhKiri)
        mergeShort(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
```

Hasil :

```
>>> alist = [54,26,93,17,77,31,44,55,20]
>>> mergeShort(alist)
>>> print(alist)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

Latihan 3

```python
def quickSort(A):
    quickSortBantu(A, 0, len(A) - 1)

def quickShortBantu(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]

    penandaKiri = awal + 1
    penandaKanan = akhir

    selesai = False
    while not selesai:
        while penandaKiri <= penandaKanan and\
              A[penandaKiri] <= nilaiPivot:
            penandaKiri = penandaKiri + 1

        while A[penandaKanan] >= nilaiPivot and\
              penandaKanan >= penandaKiri:
            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
```

Soal Tugas :

**Nomer 1**

```
class MhsTIF(object):
    def __init__(self,nama,nim,tinggal,us):
        self.nama = nama
        self.nim = nim
        self.tinggal = tinggal
        self.us = us

c0 = MhsTIF('Ika', 'L20019001', 'Sukoharjo', 240000)
c1 = MhsTIF('Budi', 'L20019003', 'Sragen', 230000)
c2 = MhsTIF('Ahmad', 'L20019002', 'Surakarta', 250000)
c3 = MhsTIF('Chandra', 'L20019004', 'Surakarta', 235000)
c4 = MhsTIF('Eka', 'L20019006', 'Boyolali', 240000)
c5 = MhsTIF('Fandi', 'L20019005', 'Salatiga', 250000)
c6 = MhsTIF('Deni', 'L20019007', 'Klaten', 245000)
c7 = MhsTIF('Galuh', 'L20019009', 'Wonogiri', 245000)
c8 = MhsTIF('Janto', 'L20019008', 'Klaten', 245000)
c9 = MhsTIF('Hasan', 'L20019011', 'Karanganyar', 270000)
c10 = MhsTIF('Khalid', 'L20019010', 'Purwodadi', 265000)
      You, last month • modul 5 dan 6 …
Daftar=[c0,c1,c2,c3,c4,c5,c6,c7,c8,c9]
```

Mergeshort

```
###=====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] < 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

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

A = []
for x in Daftar:
    A.append(x.nim)

print("=====Merge Sort=====")
mergeSort(A)
for i in convert(A, Daftar):
    print (i.nama,i.nim,i.tinggal,i.us)
print()
```

```
=====Merge Sort=====
Ika L20019001 Sukoharjo 240000
Ahmad L20019002 Surakarta 250000
Budi L20019003 Sragen 230000
Chandra L20019004 Surakarta 235000
Fandi L20019005 Salatiga 250000
Eka L20019006 Boyolali 240000
Deni L20019007 Klaten 245000
Janto L20019008 Klaten 245000
Galuh L20019009 Wonogiri 245000
Hasan L20019011 Karanganyar 270000
```

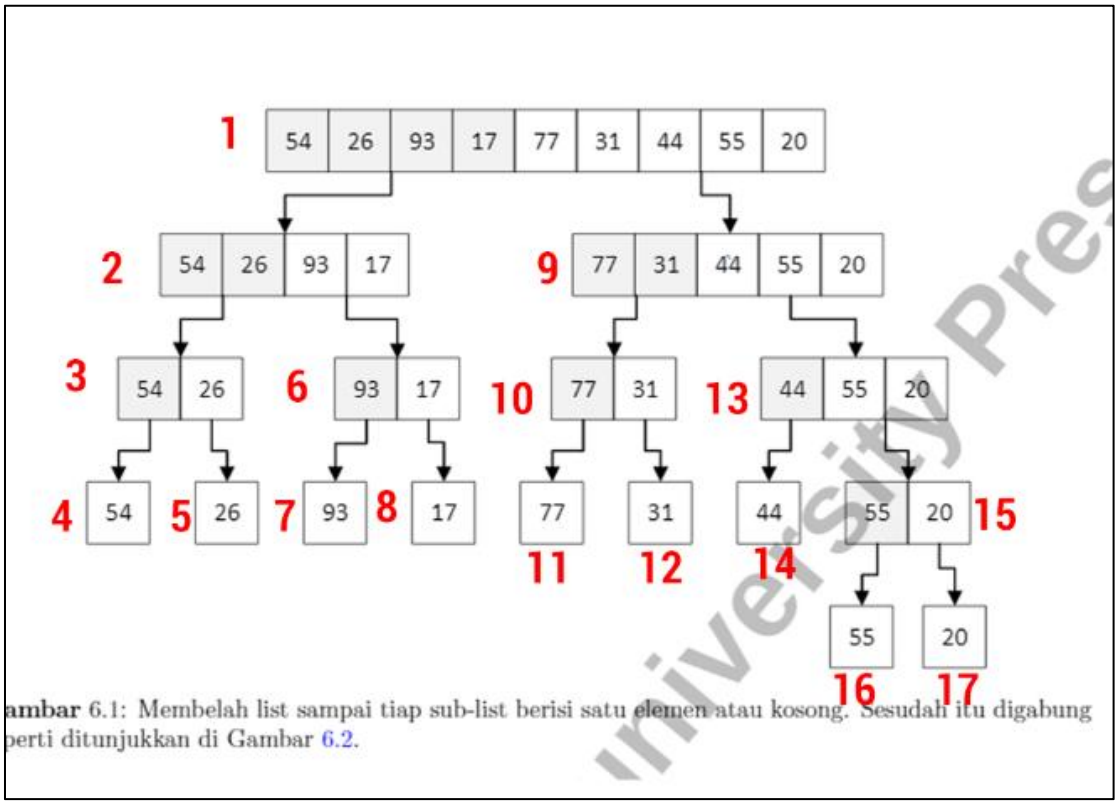
Quickshort

```
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 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


A = []
for x in Daftar:
    A.append(x.nim)

print("=====Quick Sort=====")
quickSort(A)
for i in convert(A, Daftar):
    print(i.nama, i.nim, i.tinggal, i.us)
```

```
=====Quick Sort=====
Ika L20019001 Sukoharjo 240000
Ahmad L20019002 Surakarta 250000
Budi L20019003 Sragen 230000
Chandra L20019004 Surakarta 235000
Fandi L20019005 Salatiga 250000
Eka L20019006 Boyolali 240000
Deni L20019007 Klaten 245000
Janto L20019008 Klaten 245000
Galuh L20019009 Wonogiri 245000
Hasan L20019011 Karanganyar 270000
```

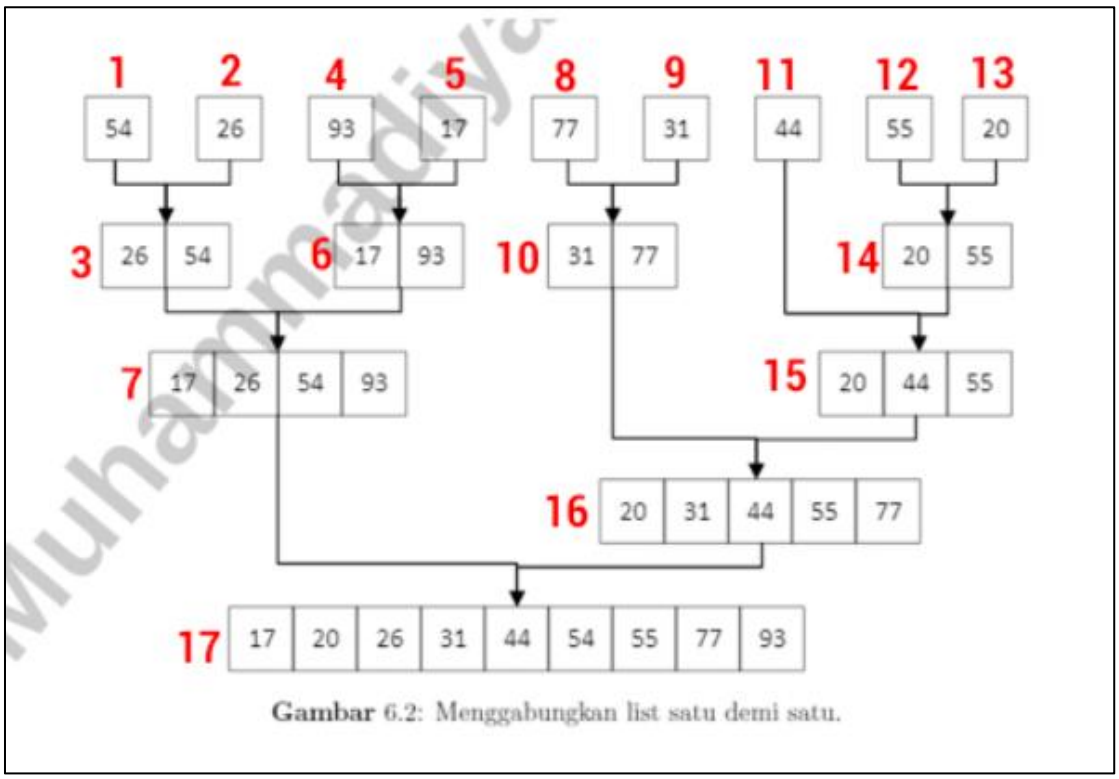
**Nomer 2**

Mergeshort




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

Quickshort




**Gambar** 6.2: Menggabungkan list satu demi satu.

**Nomer 3**

```
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):
    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



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)



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)
```

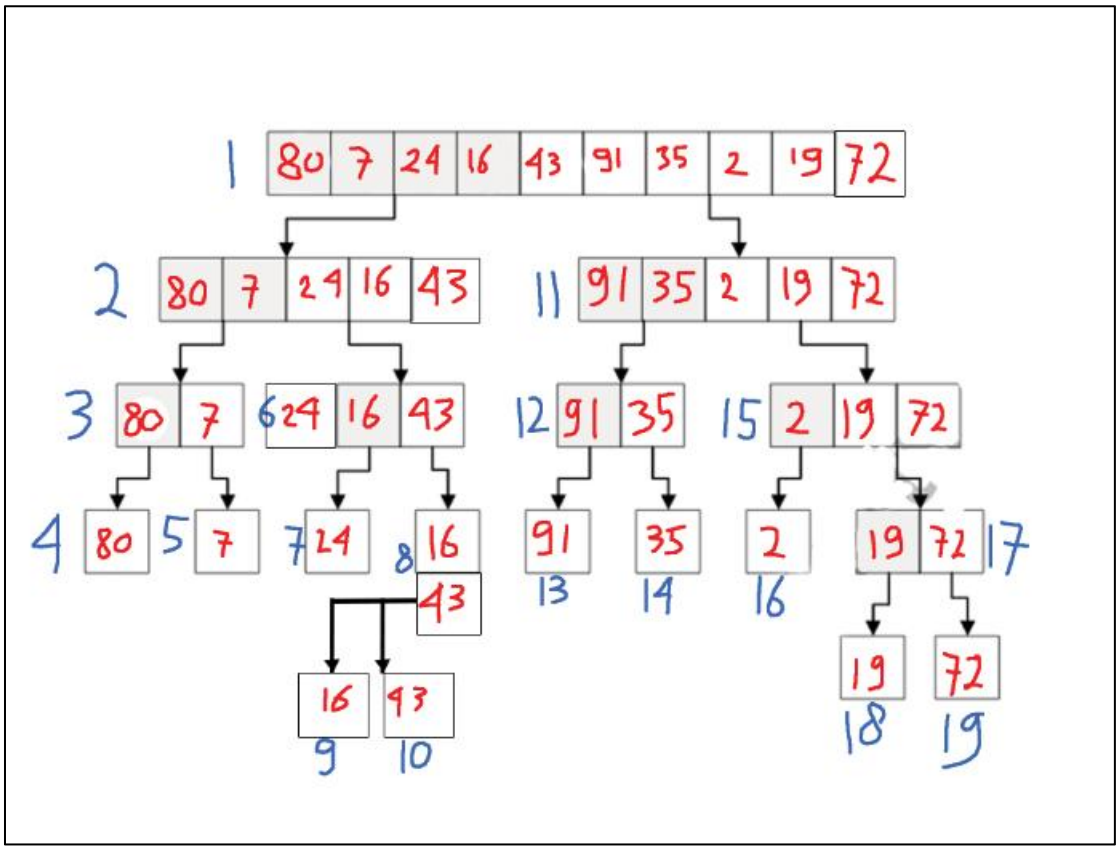
```
ak = detak()
print("quick: %g detik" % (ak-aw))
```

Hasil :

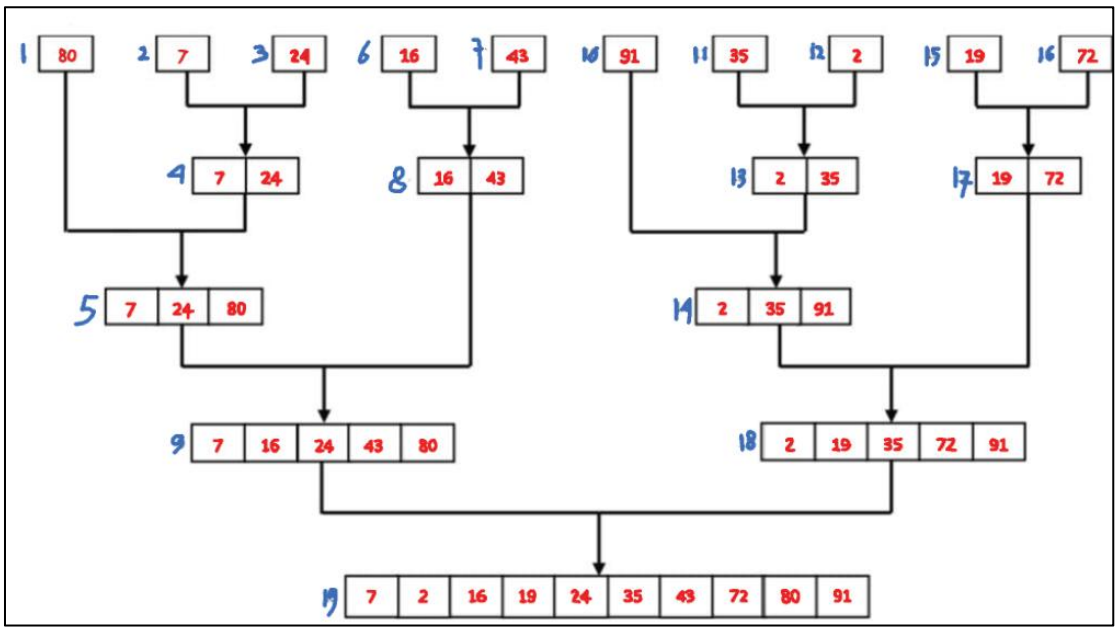
```
$ "C:/Program Files/Python39/
bubble: 3.81754 detik
selection: 1.57184 detik
insertion: 1.90853 detik
merge: 0.025949 detik
quick: 0.0190446 detik
```

**Nomer 4**

Mergeshort



Quickshort


1 80
2 7
3 24
6 16
7 43
10 91
11 35
12 2
15 19
16 72
4 7 24
8 16 43
13 2 35
17 19 72
5 7 24 80
14 2 35 91
9 7 16 24 43 80
18 2 19 35 72 91
19 7 2 16 19 24 35 43 72 80 91

**Nomer 5**

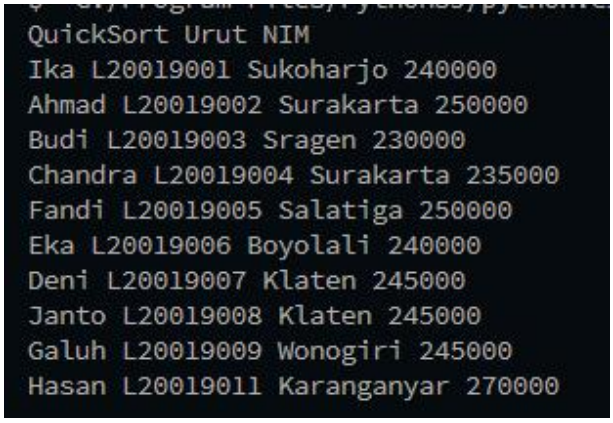
```
def mergeSort2(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergeSort2(A, awal, mid)
        mergeSort2(A, mid+1, akhir)

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


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


A = []
for x in Daftar:
    A.append(x.nim)
print("MergeSort Urut NIM")
mergeSort(A)
for i in convert(A, Daftar):
    print(i.nama, i.nim, i.tinggal, i.us)
```

Hasil :

```
MergeSort Urut NIM
Ika L20019001 Sukoharjo 240000
Ahmad L20019002 Surakarta 250000
Budi L20019003 Sragen 230000
Chandra L20019004 Surakarta 235000
Fandi L20019005 Salatiga 250000
Eka L20019006 Boyolali 240000
Deni L20019007 Klaten 245000
Janto L20019008 Klaten 245000
Galuh L20019009 Wonogiri 245000
Hasan L20019011 Karanganyar 270000
```

## Nomer 6

```python
def partisi(A, awal, akhir):
    hasil = 0
    pivot, pidx = median_dari_tiga(A, awal, akhir)
    A[awal], A[pidx] = A[pidx], A[awal]
    i = awal + 1
    for j in range(awal+1, akhir, 1):
        hasil += 1
        if (A[j] < pivot):
            A[i], A[j] = A[j], A[i]
            i += 1
    A[awal], A[i-1] = A[i-1], A[awal]
    return i - 1, hasil


def median_dari_tiga(A, awal, akhir):
    tengah = (awal+akhir-1)//2
    a = A[awal]
    b = A[tengah]
    c = A[akhir-1]
    if a <= b <= c:
        return b, tengah
    if c <= b <= a:
        return b, tengah
    if a <= c <= b:
        return c, akhir-1
    if b <= c <= a:
        return c, akhir-1
    return a, awal


def quickSortBantu(A, awal, akhir):
    hasil = 0
    if awal < akhir:
        titikBelah, hasil = partisi(A, awal, akhir)
        hasil += quickSortBantu(A, awal, titikBelah)
        hasil += quickSortBantu(A, titikBelah + 1, akhir)
    return hasil


def quickSort(A):
    quickSortBantu(A, 0, len(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])
    return hasil


A = []
for x in Daftar:
    A.append(x.nim)

print("QuickSort Urut NIM")
quickSort(A)
for i in convert(A, Daftar):
    print(i.nama, i.nim, i.tinggal, i.us,)
```




**Nomer 7**

```
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)
```

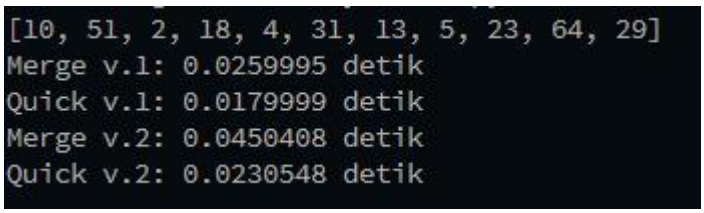
```
def quickSort(A):
    quickSortBantu(A, 0, len(A)-1)



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]
print(daftar)
mergeSort(daftar)
quickSort(daftar)
mergeSortNew(daftar)
quickSortNew(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 v.1: %g detik" % (ak-aw))
aw = detak()
quickSort(u_qck)
```

```
ak = detak()
print("Quick v.1: %g detik" % (ak-aw))
aw = detak()
mergeSortNew(u_mrgNew)
ak = detak()
print("Merge v.2: %g detik" % (ak-aw))
aw = detak()
quickSortNew(u_qckNew)
ak = detak()
print("Quick v.2: %g detik" % (ak-aw))
```

Hasil :




**Nomer 8**

```
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(80)
b = Node(7)
c = Node(24)
d = Node(16)
e = Node(43)
f = Node(91)
g = Node(35)
h = Node(2)
i = Node(19)
j = Node(72)

a.tautan = b
b.tautan = c
c.tautan = d
d.tautan = e
e.tautan = f
f.tautan = g
g.tautan = h
h.tautan = i
```

```
i.tautan = j



def mergeSortLinkedList(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:]


        mergeSortLinkedList(separuhkiri)
        mergeSortLinkedList(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
```

mergeSortLinkedList(a)

cetak(a)

Hasil :