

Nama : Riska Meilina Puspa

NIM : L200180192

Kelas : G

Modul 6

Praktikum Algoritma dan Struktur Data

No.1

```
from Modul6 import *
from listMahasiswa import *

def convert(arr, obj):
    hasil=[]
    for x in range (len(arr)):
        for i in range (len(obj)):
            if arr[x] == obj[i].nim:
                hasil.append(obj[i])
    return hasil

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

    print("Merge Sort")
    mergeSort(A)
    for x in convert(A, Daftar):
        print (x.nim)

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)
```

```
>>> urutkanMerge()
```

```
Merge Sort
```

```
100
```

```
130
```

```
133
```

```
143
```

```
155
```

```
177
```

```
180
```

```
189
```

```
192
```

```
199
```

```
211
```

```
>>> urutkanQuick()
```

```
Quick Sort
```

```
100
```

```
130
```

```
133
```

```
143
```

```
155
```

```
177
```

```
180
```

```
189
```

```
192
```

```
199
```

```
211
```

```
>>> |
```

No.3

```
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));
|

```

```
bubble: 15.2878 detik
selection: 6.89578 detik
insertion: 7.3101 detik
merge: 0.129496 detik
quick: 0.0290306 detik
>>> |
```

No.5

```
from listMahasiswa 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)
|
```

Sebelum diurutkan

Wosoek, nim 100. Tinggal di Sukoharjo. Uang saku Rp 240000. tiap bulannya.
Minhae, nim 133. Tinggal di Sragen. Uang saku Rp 230000. tiap bulannya.
Riska, nim 192. Tinggal di Surakarta. Uang saku Rp 250000. tiap bulannya.
Hangyul, nim 180. Tinggal di Surakarta. Uang saku Rp 235000. tiap bulannya.
Seungyoun, nim 155. Tinggal di Boyolali. Uang saku Rp 240000. tiap bulannya.
Yohan, nim 189. Tinggal di Salatiga. Uang saku Rp 250000. tiap bulannya.
Seungwoo, nim 177. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Junho, nim 143. Tinggal di Wonogiri. Uang saku Rp 245000. tiap bulannya.
Eunsang, nim 211. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Dohyun, nim 130. Tinggal di Karanganyar. Uang saku Rp 270000. tiap bulannya.
Hyeongjun, nim 199. Tinggal di Purwodadi. Uang saku Rp 265000. tiap bulannya.

Setelah diurutkan

Minhae, nim 133. Tinggal di Sragen. Uang saku Rp 230000. tiap bulannya.
Hangyul, nim 180. Tinggal di Surakarta. Uang saku Rp 235000. tiap bulannya.
Seungyoun, nim 155. Tinggal di Boyolali. Uang saku Rp 240000. tiap bulannya.
Wosoek, nim 100. Tinggal di Sukoharjo. Uang saku Rp 240000. tiap bulannya.
Eunsang, nim 211. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Junho, nim 143. Tinggal di Wonogiri. Uang saku Rp 245000. tiap bulannya.
Seungwoo, nim 177. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Yohan, nim 189. Tinggal di Salatiga. Uang saku Rp 250000. tiap bulannya.
Riska, nim 192. Tinggal di Surakarta. Uang saku Rp 250000. tiap bulannya.
Hyeongjun, nim 199. Tinggal di Purwodadi. Uang saku Rp 265000. tiap bulannya.
Dohyun, nim 130. Tinggal di Karanganyar. Uang saku Rp 270000. tiap bulannya.

>>> |

No.6

```
from listMahasiswa 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)
|
```


Sebelum diurutkan

Wosoek, nim 100. Tinggal di Sukoharjo. Uang saku Rp 240000. tiap bulannya.
Minhae, nim 133. Tinggal di Sragen. Uang saku Rp 230000. tiap bulannya.
Riska, nim 192. Tinggal di Surakarta. Uang saku Rp 250000. tiap bulannya.
Hangyul, nim 180. Tinggal di Surakarta. Uang saku Rp 235000. tiap bulannya.
Seungyoun, nim 155. Tinggal di Boyolali. Uang saku Rp 240000. tiap bulannya.
Yohan, nim 189. Tinggal di Salatiga. Uang saku Rp 250000. tiap bulannya.
Seungwoo, nim 177. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Junho, nim 143. Tinggal di Wonogiri. Uang saku Rp 245000. tiap bulannya.
Eunsang, nim 211. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Dohyun, nim 130. Tinggal di Karanganyar. Uang saku Rp 270000. tiap bulannya.
Hyeongjun, nim 199. Tinggal di Purwodadi. Uang saku Rp 265000. tiap bulannya.

Setelah diurutkan

Wosoek, nim 100. Tinggal di Sukoharjo. Uang saku Rp 240000. tiap bulannya.
Minhae, nim 133. Tinggal di Sragen. Uang saku Rp 230000. tiap bulannya.
Riska, nim 192. Tinggal di Surakarta. Uang saku Rp 250000. tiap bulannya.
Hangyul, nim 180. Tinggal di Surakarta. Uang saku Rp 235000. tiap bulannya.
Seungyoun, nim 155. Tinggal di Boyolali. Uang saku Rp 240000. tiap bulannya.
Yohan, nim 189. Tinggal di Salatiga. Uang saku Rp 250000. tiap bulannya.
Seungwoo, nim 177. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Junho, nim 143. Tinggal di Wonogiri. Uang saku Rp 245000. tiap bulannya.
Eunsang, nim 211. Tinggal di Klaten. Uang saku Rp 245000. tiap bulannya.
Dohyun, nim 130. Tinggal di Karanganyar. Uang saku Rp 270000. tiap bulannya.
Hyeongjun, nim 199. Tinggal di Purwodadi. Uang saku Rp 265000. tiap bulannya.

>>> |

No.7

```
from time import time as detik
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));|

```

```

[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: 0.183345 detik
quick: 0.0939975 detik
merge New: 0.155185 detik
quick New: 0.0281191 detik
>>> |

```

No.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(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)

```

```
1  
2  
3  
4  
5  
6  
7
```

```
>>> |
```