Nama : Malik

Muhammad

NIM : L200180072

Kelas : C

Modul 6

Laporan Praktikum

6.1 Menggabungkan dua list yang sudahurut

Berikut adalah screenshoot dari program yang saya buat :

```
'''
def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos - 1]:          #-> cari posisi yang tepat
            A[pos] = A[pos-1]                          # dan geser ke kanan terus
            pos = pos -1                               # nilai - nilai yang lebih besar
        A[pos] = nilai                                 #-> pada posisi ini tempatkan nilai elemen ke i

def gabungkanDuaListUrut(A,B):
    C = A + B; insertionSort(C); return C
```

Berikut adalah screenshoot saat program dijalankan :

```
= RESTART: C:/Users/user/Documents/Tugas/ASD
>>> P = [2,8,15,23,37]
>>> Q = [4,6,15,20]
>>> R = gabungkanDuaListUrut(P,Q)
>>> print(R)
[2, 4, 6, 8, 15, 15, 20, 23, 37]
>>>
```

Halaman 56

Berikut adalah screenshoot dari program yang saya buat :

Berikut adalah screenshoot saat program dijalankan :

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

    #Gabungkan keduanya sampai salah satu kosong
    while i < la and j < lb:
        if A[i] < B[j]:
            C.append(A[i])
            i += 1
        else:
            C.append(B[j])
            j += 1

    while i < la:                       #Jika A mempunyai sisa
        C.append(A[i])                  #Tumpukkan ke list baru itu
        i += 1                          #satu demi satu

    while j < lb:                       #Jika B mempunyai sisa
        C.append(B[j])                  #Tumpukkan ke list baru itu
        j += 1                          #satu demi satu

    return C

= RESTART: C:/Users/user/Documents/Tugas,
>>> P = [2,8,15,23,37]
>>> Q = [4,6,15,20]
>>> R = gabungkanDuaListUrut(P,Q)
>>> print(R)
[2, 4, 6, 8, 15, 15, 20, 23, 37]
>>>
```

6.2 MergeSort

Penyegaran: Fungsi Rekursif

Berikut adalah screenshoot dari program yang saya buat :

```
def faktorial(a):
    if a==1:                                    #base case
        return 1
    else:                                       #recursive case
        return a*faktorial(a-1)
```

Berikut adalah screenshoot saat program dijalankan :

```
========== RESTART: C:
>>> faktorial(1)
1
>>> faktorial(7)
5040
>>> faktorial(10)
3628800
>>> faktorial(4)
24
>>>
```

Halaman 59

Berikut adalah screenshoot dari program yang saya buat:

Berikut adalah screenshoot saat program dijalankan :

```
def mergeSort(A):
    #print("Membelah A ", A)
    if len(A) > 1:
        mid = len(A) // 2                   #Membelah list
        separuhKiri = A[:mid]               #Slicing ini langkah yang expensive sebenarnya
        separuhKanan = A[mid:]              #Bisakah kamu membuatnya lebih baik ?

        mergeSort(separuhKiri)              #Ini rekursi. Memanggil lebih lanjut mergeSort
        mergeSort(separuhKanan)             #Untuk separuhKiri dan separuhKanan

        #Di bawah ini adalah proses penggabungan
        i = 0; j = 0; k = 0
        while i < len(separuhKiri) and j < len(separuhKanan):
            if separuhKiri[i] < separuhKanan[j]:    #while-loop ini
                A[k] = separuhKiri[i]               #menggabungkan kedua list, yakni
                i = i + 1                           #separuhKiri dan separuhKanan,
            else:                                   #sampai salah satu kosong
                A[k] = separuhKanan[j]              #Perhatikan kesamaan strukturnya
                j = j + 1                           #dengan proses penggabungan
            k = k + 1                               #dua list urut

        while i < len(separuhKiri):         #Jika separuhKiri mempunyai sisa
            A[k] = separuhKiri[i]           #tumpukkan ke A
            i = i + 1                       #satu demi satu
            k = k + 1

        while j < len(separuhKanan):        #Jika separuhKanan mempunyai sisa
            A[k] = separuhKanan[j]          #tumpukkan ke A
            j = j + 1                       #satu demi satu
            k = k + 1
    #print("Menggabungkan ", A)

========== RESTART: C:/Users/u
>>> alist = [54,26,93,17,77,31,44,55,20]
>>> mergeSort(alist)
>>> print(alist)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

Berikut adalah screenshoot program saat perintah print(“Membelah ”, A) dan print(“Menggabungkan ”, A) sudah diaktifkan :

```
def mergeSort(A):
    print("Membelah A ", A)
    if len(A) > 1:
        mid = len(A) // 2                       #Membelah list
        separuhKiri = A[:mid]                   #Slicing ini langkah yang expensive sebenarnya
        separuhKanan = A[mid:]                  #Bisakah kamu membuatnya lebih baik ?

        mergeSort(separuhKiri)                  #Ini rekursi. Memanggil lebih lanjut mergeSort
        mergeSort(separuhKanan)                 #Untuk separuhKiri dan separuhKanan

        #Di bawah ini adalah proses penggabungan
        i = 0; j = 0; k = 0
        while i < len(separuhKiri) and j < len(separuhKanan):
            if separuhKiri[i] < separuhKanan[j]:    #while-loop ini
                A[k] = separuhKiri[i]               #menggabungkan kedua list, yakni
                i = i + 1                           #separuhKiri dan separuhKanan,
            else:                                   #sampai salah satu kosong
                A[k] = separuhKanan[j]              #Perhatikan kesamaan strukturnya
                j = j + 1                           #dengan proses penggabungan
            k = k + 1                               #dua list urut

        while i < len(separuhKiri):         #Jika separuhKiri mempunyai sisa
            A[k] = separuhKiri[i]           #tumpukkan ke A
            i = i + 1                       #satu demi satu
            k = k + 1

        while j < len(separuhKanan):        #Jika separuhKanan mempunyai sisa
            A[k] = separuhKanan[j]          #tumpukkan ke A
            j = j + 1                       #satu demi satu
            k = k + 1
    print("Menggabungkan ", A)
```

Berikut adalah screenshoot saat program dijalankan :

```
>>> alist = [54,26,93,17,77,31,44,55,20]
>>> mergeSort(alist)
Membelah A  [54, 26, 93, 17, 77, 31, 44, 55, 20]
Membelah A  [54, 26, 93, 17]
Membelah A  [54, 26]
Membelah A  [54]
Menggabungkan  [54]
Membelah A  [26]
Menggabungkan  [26]
Menggabungkan  [26, 54]
Membelah A  [93, 17]
Membelah A  [93]
Menggabungkan  [93]
Membelah A  [17]
Menggabungkan  [17]
Menggabungkan  [17, 93]
Menggabungkan  [17, 26, 54, 93]
Membelah A  [77, 31, 44, 55, 20]
Membelah A  [77, 31]
Membelah A  [77]
Menggabungkan  [77]
Membelah A  [31]
Menggabungkan  [31]
Menggabungkan  [31, 77]
Membelah A  [44, 55, 20]
Membelah A  [44]
Menggabungkan  [44]
Membelah A  [55, 20]
Membelah A  [55]
Menggabungkan  [55]
Membelah A  [20]
Menggabungkan  [20]
Menggabungkan  [20, 55]
Menggabungkan  [20, 44, 55]
Menggabungkan  [20, 31, 44, 55, 77]
Menggabungkan  [17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> print(alist)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

6.3 Quick Sort

Berikut adalah screenshoot dari program yang saya buat:

Berikut adalah screenshoot saat program dijalankan :

```
def partisi(A, awal, akhir):
    nilaiPivot = A[awal]                            #Disini nilaiPivot kita ambil dari elemen yang paling kiri

    penandaKiri = awal + 1                          #Posisi awal penandaKiri
    penandaKanan = akhir                            #Posisi awal penandaKanan

    selesai = False
    while not selesai:                              #loop di bawah adalah untuk mengatur ulang posisi semua elemen

        while penandaKiri <= penandaKanan and A[penandaKiri] <= nilaiPivot:     #penandaKiri bergerak ke kanan sampai ketemu suatu nilai yang
            penandaKiri = penandaKiri + 1                                       #lebih besar dari nilaiPivot

        while A[penandaKanan] >= nilaiPivot and penandaKanan >= penandaKiri:    #penandaKanan bergerak ke kiri sampai ketemu suatu nilai yang
            penandaKanan = penandaKanan - 1                                     #lebih kecil dari nilaiPivot

        if penandaKanan < penandaKiri:                                          #Kalau dua penanda sudah bersilangan
            selesai = True                                                      #selesai & lanjut ke penempatan pivot
        else:
            temp = A[penandaKiri]                                               #tapi kalau belum bersilangan
            A[penandaKiri] = A[penandaKanan]                                    #tukarlah isi yang ditunjuk oleh
            A[penandaKanan] = temp                                              #penandaKiri dan penandaKanan

    temp = A[awal]                  #Kalau acara tukar menukar posisi sudah selesai
    A[awal] = A[penandaKanan]       #kita lalu menempatkan pivot pada posisi yang tepat
    A[penandaKanan] = temp          #yakni posisi penandaKanan
                                    #posisi penandaKanan adalah juga titikBelah
    return penandaKanan             #Fungsi ini mengembalikan titikBelah ke pemanggil


def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)        #Atur elemen dan dapatkan titikBelah
        quickSortBantu(A, awal, titikBelah - 1)     #Ini rekursi untuk belah sisi kiri
        quickSortBantu(A, titikBelah + 1, akhir)    #dan belah sisi kanan

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

Ln: 118 Col: 80

```
========== RESTART: C:/Users/use
>>> A = [54,26,93,17,77,31,44,55,20]
>>> partisi(A,0,8)
5
>>> quickSort(A)
>>> print(A)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```