Fakultas Informatika
School of Computing
Telkom University

Tugas Besar
Analisis Kompleksitas Algoritma

"PERBANDINGAN RUNNING TIME BUBBLE SORT DENGAN MERGE SORT"

Nama anggota kelompok :

Daniel Septyadi (1301180009)

Muhammad Haidir Ali (1301180205)

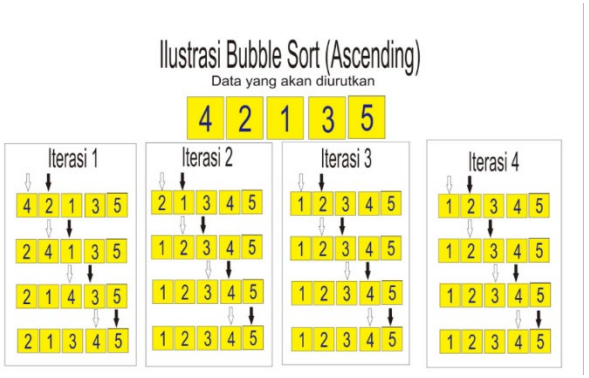
Gilang Muhamad Rizky (1301180286)

Randika Dwi Maulana Rasyid (1301180361)

Penjelasan Program

- **Bubble Sort**

Bubble sort merupakan algoritma yang berjalan secara berulang ulang dan dalam perulangannya melakukan perbandingan data dan menukar posisi dari data yang dibandingkan sampai tidak ada lagi data yang ditukar pada setiap iterasi.



Gambar 1. Ilustrasi Bubble sort

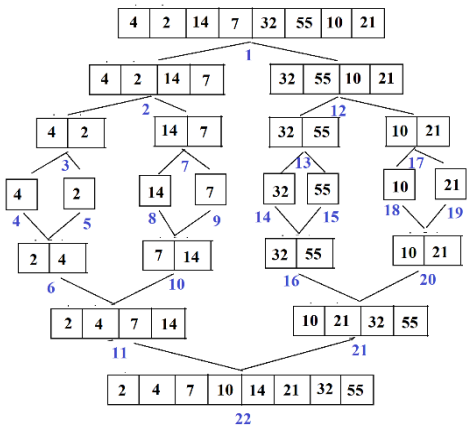
```
BubbleSort(array)
for i=length(array)-1 downto 0 do
        for j=1 to i do
                if array[j-1] > array[j] then
                swap(array[j-1],array[j])
                end if
        end for
end for
```

Gambar 2. Pseudo code dari algoritma Bubble sort

Penjelasan Program

- **Merge Sort**

Merge sort merupakan algoritma sorting yang bekerja dengan cara membagi sebuah struktur data menjadi lebih dari satu sub-bagian, lalu setiap anggota pada setiap sub-bagian akan dibandingkan dan posisinya akan ditukar jika memenuhi syarat.



Gambar 1. Ilustrasi Marge sort

```
MergeSort(array,low,high)
        mid : integer
        if (low<high) then
                        mid <- (low+high)/2
                        MergeSort(array,low,mid)
                        MergeSort(array,mid+1,high)
                        Merge(array,low,high,mid)
        endif
```

Gambar 2. Pseudo code dari algoritma merge sort

Strategi Algoritma

Untuk hasil running time dari kedua sort yaitu merge sort dan bubble sort dengan n seperti dibawah ini, hasil ditunjukan dalam bentuk tabel.

n=10; n=100; n=1000; n=10000; n=100000.

Jumlah data (n)	Running Time (microsecond)	
	Bubble Sort	Merge Sort
10	23.0	29.0
100	1499.0	585.0
1000	114619.0	4527.0
10000	11529591.0	62962.01
100000	1315844195.0	823086.0

Fungsional Program

```
#Bubble sort Function
def bubbleSort(list):
    temp = []
    for i in range(len(list)-1, 0, -1):
        for j in range(i):
            if list[j] < list[j+1]:
                temp = list[j]
                list[j] = list[j+1]
                list[j+1] = temp
    return list
```

Gambar 1. Bubble sort function

```
#Merge sort Function
def mergeSort(_list):
    if len(_list) < 2:
        return _list
    else:
        get_center=len(_list)//2
        left=_list[:get_center]
        right=_list[get_center:]
        mergeSort(left)
        mergeSort(right)
        i=0;j=0;k=0
        while i < len (left) and j < len (right):
            if left[i]>right[j]:
                _list[k]=left[i]
                i=i+1
            else:
                _list[k]=right[j]
                j=j+1
            k=k+1
        while i < len (left):
            _list[k]=left[i]
            i=i+1
            k=k+1

        while j < len (right):
            _list[k]=right[j]
            j=j+1
            k=k+1
```

Gambar 2. Marge sort function

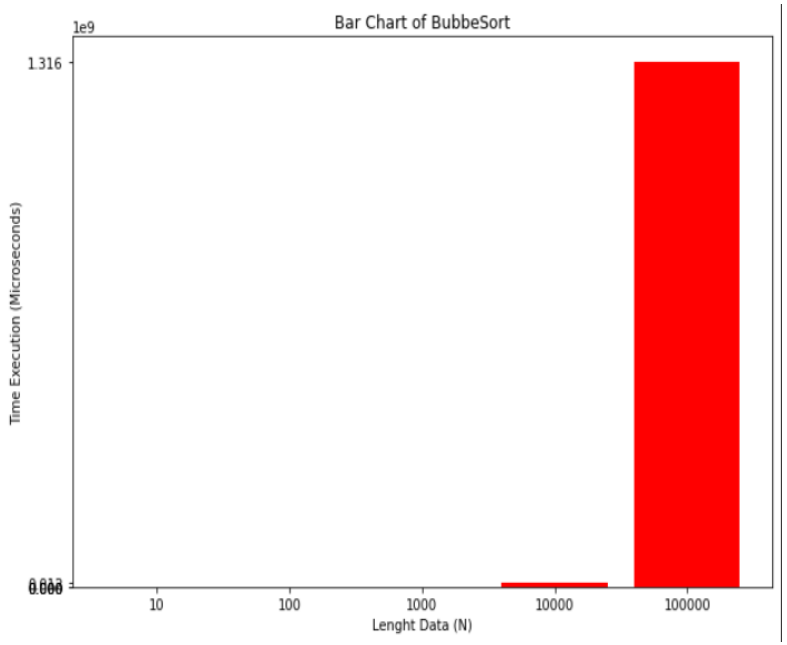
```
def cariAngka(N):
    # seed random number generator
        seed(1)
        angka = []
        for i in range(N):
            value = randint(0, N)
            angka.append(value) #tangkap random number dalam list
        return angka
        #print("Angka acak adalah sebagai berikut : \n",angka)
```

Gambar 3. Function cari angka

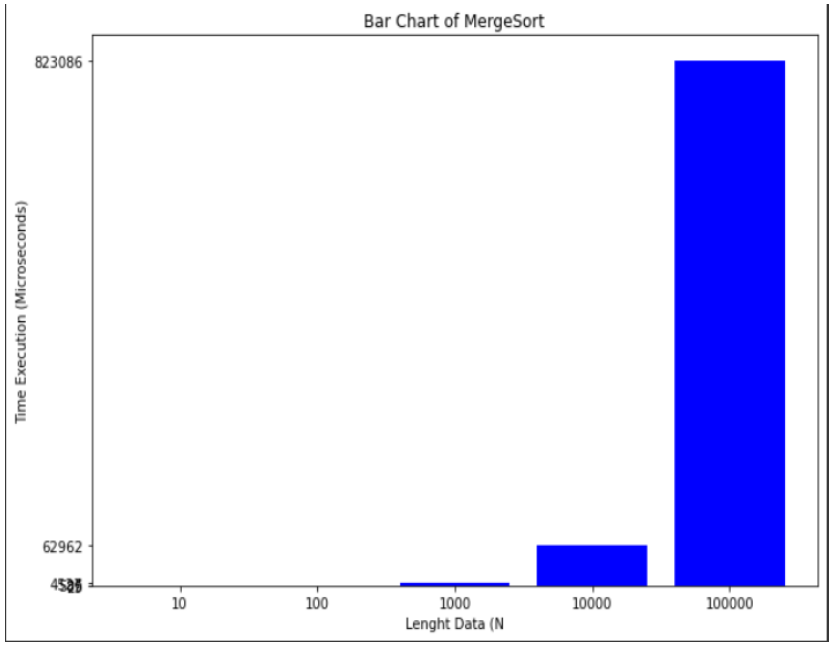
```
def getIteration():
    iteration = int(input("Jumlah Iteration: "))
    iteration_list = []
    for i in range(iteration):
        iteration_list.append(int(input("Input Batas Iteration ke "+str(i+1)+" : ")))

    return iteration_list
```

Gambar 4. Function get iteration

Visualisasi Dari running hasil kedua algoritma

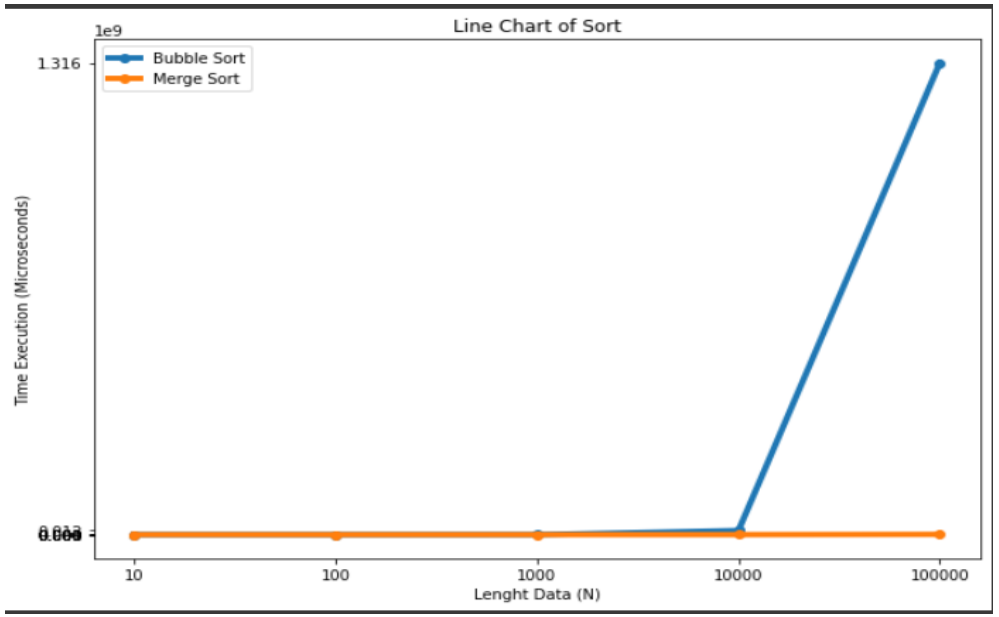


Gambar 1. Running Time Algoritma Bubble Sort



Gambar 2. Running Time Algoritma Merge Sort

Output Program



Gambar 2. Pseudo code dari algoritma merge sort

Berdasarkan line chart pada gambar dapat terlihat perbedaan running time yang sangat signifikan saat N (jumlah data) yang ingin diurutkan sebanyak 100.000 data.

Referensi

[1] Astrachan, O. (2003). Bubble sort: an archaeological algorithmic analysis. *ACM Sigcse Bulletin*, *35*(1), 1-5.

[2] Min, W. (2010, July). Analysis on bubble sort algorithm optimization. In *2010 International forum on information technology and applications* (Vol. 1, pp. 208-211). IEEE.

[3] Abidin. T, Zamanhuri. I. Struktur Data dan Algoritma Metode Pengurutan Mege Sort.

Terima Kasih