LAPORAN TUGAS

PEMROGRAMAN BERORIENTASI OBJEK

”Tugas 4”



Oleh :

AL FATONI NUGROHO PUTRA (06111840000003)

ADRIANUS BAGAS TANTYO DANANJAYA (06111840000102)

Kelas C

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA KOMPUTASI DAN SAINS DATA

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2019

1. SOAL

Pada tugas ini akan ditampilkan soal dan cara yang digunakan untuk mengerjakan soal-soal yang ada.

1. Penjelasan Soal

Implementasikan bubblesort dengan 10000 data, bandingkan insertion dan bubble.

Tampilkan running time sorting(waktu sorting) beserta grafik tiap beberapa data

1. Metode penyelesaian

Hal pertama yang dilakukan adalah membuat method untuk mengurutkan data. Ada 2 metode yang digunakan, yaitu metode bubble sort dan insertion sort. Bubble sort adalah metode pengurutan dengan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan. Insertion sort adalah sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan.

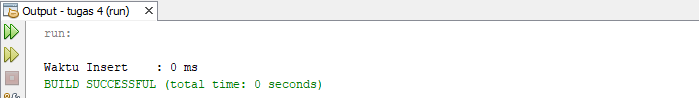
SOURCE CODE

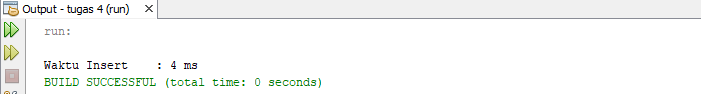
Program penyelesaian masalah tersebut yaitu:

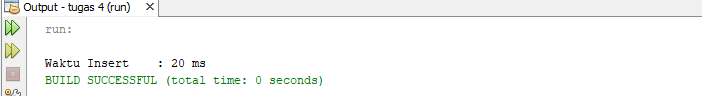
1. Class insertion\_sort

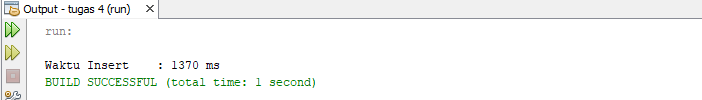
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | import java.util.Random;  public class insertion\_sort {  public void insertion\_sort(int[]list){  for(int i=1;i<list.length;i++){  int currentElement=list[i];  int k;  for(k=i-1;k>=0 && list[k]>currentElement;k--){  list[k+1]=list[k];  }  list[k+1]=currentElement;  }  }  public void infoList(int[]list){  for(int i=0;i<list.length;i++){  System.out.print(" "+list[i]);  }  }  public void bubbleSort(int[] list) {  boolean needNextPass = true;  for (int k = 1; k < list.length && needNextPass; k++) {  // Array may be sorted and next pass not needed  needNextPass = false;  for (int i = 0; i < list.length - k; i++) {  if (list[i] > list[i + 1]) {  // Swap list[i] with list[i + 1]  int temp = list[i];  list[i] = list[i + 1];  list[i + 1] = temp;  needNextPass = true; // Next pass still needed  }  }  }    }  }  class main{  public static void main(String[] args) {  int[]list=new int[100000];  Random input=new Random();  for(int i=0;i<list.length;i++){  list[i]=input.nextInt(100000);  }  insertion\_sort i= new insertion\_sort();  long starttime=System.currentTimeMillis();  i.insertion\_sort(list);  System.out.println("");  long endtime=System.currentTimeMillis();  long elapsedtime=endtime-starttime;  System.out.println("Waktu Insert : "+String.valueOf(elapsedtime)+" ms");  }  }  class bubble{  public static void main(String[] args) {  int[]list=new int[100000];  Random input=new Random();  for(int i=0;i<list.length;i++){  list[i]=input.nextInt(100000);  }  insertion\_sort i= new insertion\_sort();  long starttime=System.currentTimeMillis();  i.bubbleSort(list);  System.out.println("");  long endtime=System.currentTimeMillis();  long elapsedtime=endtime-starttime;  System.out.println("Waktu Bubble : "+String.valueOf(elapsedtime)+" ms");  }  } |

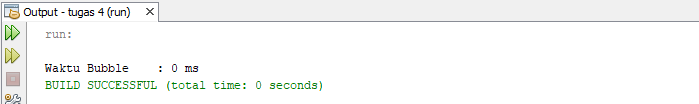
1. RUNNING PROGRAM

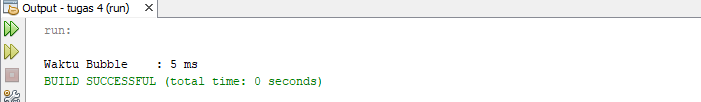
* Insertion sort 100 data

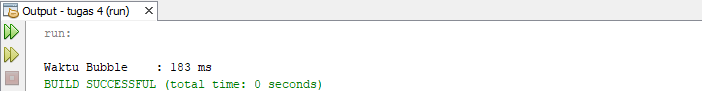
* Insertion sort 1000 data
* Insertion sort 10000 data



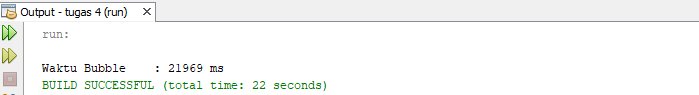
* Insertion sort 100000 data
* Bubble sort 100 data



* Bubble sort 1000 data
* Bubble sort 10000 data



* Bubble sort 100000 data



1. Grafik

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Banyaknya data | Runtime Bubble Sort  (ms) | Runtime Insertion Sort  (ms) | Rata-rata runtime Bubble sort | Rata-rata runtime Insertion sort |
| 100 | 0 | 0 | 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 1000 | 5 | 9 | 10,2 | 7,2 |
| 5 | 11 |
| 13 | 8 |
| 10 | 4 |
| 18 | 4 |
| 10000 | 224 | 24 | 201,8 | 23 |
| 198 | 18 |
| 173 | 27 |
| 204 | 26 |
| 210 | 20 |
| 100000 | 21969 | 1370 | 21619,6 | 1407,8 |
| 21547 | 1441 |
| 21521 | 1383 |
| 21577 | 1459 |
| 21484 | 1386 |

Data yang ditampilkan di grafik adalah rata-rata dari 5 runtime dari setiap jumlah data yang ditentukan.

* + Insertion sort

100 data = 0 ms

1000 data=7,2 ms

10000 data=23 ms

100000 data=1407,8 ms

* + Bubble sort

100 data=0 ms

1000 data=10,2 ms

10000 data=201,8 ms

100000 data=21619,6 ms

Dari data yang ditampilkan di grafik, dapat dinyatakan bahwa semakin besar jumlah data yang diurutkan, maka semakin signifikan perbedaan waktu runtimenya. Insertion sort memiliki rata-rata waktu runtime yang lebih cepat dari bubble sort.