PERBANDINGAN ALGORITMA INSERTION SORT DENGAN MERGE SORT PADA BAHASA PEMROGRAMAN C DAN JAVA

Rifkie Primartha, ST, MT.
Jurusan Informatika, Fakultas Ilmu Komputer, Universitas Sriwijaya, Palembang
rifkie77@gmail.com

Abstrak - Algoritma pengurutan telah lama menarik para peneliti untuk mengkaji lebih dalam guna menemukan algoritma yang paling efektif dan efisien. Algoritma pengurutan saat ini telah dimanfaatkan diberbagai peralatan modern seperti mobile phone untuk mengurutkan daftar kontak. Pada penelitian ini membahas tentang kecepatan pengurutan antara algoritma Insertion Sort dengan algoritma Merge Sort menggunakan dua jenis dataset random dan descending sebanyak lima puluh ribu data. Penelitian ini juga membandingkan hasil pengurutan menggunakan bahasa pemrograman C dan bahasa pemrograman Java. Hasil dari penelitian ini menunjukkan bahwa Algortima Merge Sort membutuhkan waktu lebih singkat dibandingkan Insertion Sort dalam proses pengurutan data.

Keywords: Algoritma, Sorting, Insertion Sort, Merge Sort

1. PENDAHULUAN

Di dalam dunia pemograman terdapat banyak algoritma dengan ciri khasnya sendiri. Salah satunya yaitu algoritma pengurutan. Pengurutan dapat dilakukan secara *ascending* (kecil ke besar) maupun *descending* (besar ke kecil). Algoritma pengurutan saat ini terbagi menjadi delapan diantaranya, bubble sort, selection sort, insertion sort, shell sort, merge sort, radix sort, quick sort dan heap sort.

Tabel 1 menginformasikan tentang tingkat popularitas dari berbagai macam algoritma pengurutan (Reina & Gautama, 2013).Di dalam dunia pemograman terdapat banyak algoritma dengan ciri khasnya sendiri. Salah satunya yaitu algoritma pengurutan. Pengurutan dapat dilakukan secara *ascending* (kecil ke besar) maupun *descending* (besar ke kecil). Algoritma pengurutan saat ini terbagi menjadi delapan diantaranya, bubble sort, selection sort, insertion sort, shell sort, merge sort, radix sort, quick sort dan heap sort.

Tabel 1 menginformasikan tentang tingkat popularitas dari berbagai macam algoritma pengurutan (Reina & Gautama, 2013).

Tabel 1. Tingkat Popularitas Algoritma Pengurutan

Sort Algorithm	Hits Tahun 2000	Hits Tahun 2002
Quick Sort	26.780	80.200
Merge Sort	13.330	33.500
Heap Sort	9.830	22.960
Bubble Sort	12.400	33.800
Insertion Sort	8.450	21.870
Selection Sort	6.720	20.600
Shell Sort	4.540	8.620

Algoritma pengurutan menjadi dasar dari algoritma lainnya seperti algoritma pencarian, *patern matching*, dan lain-lain seperti yang dikatakan oleh Wang (1996) "*Sort algorithm is one of the fundamental technique in computer science because of the following reasons. First, it is the basis of many other algorithm such as searching, pattern matching, etc., and many application have been found in database system*."

Penelitian ini adalah lanjutan dari penelitian yang dilakukan oleh Reina dan Gautama, yaitu membandingkan algoritma Bubble Sort dengan Insertion Sort. Pada kesempatan ini akan membahas tentang dua macam metode pengurutan, yaitu insertion sort dan merge sort. Dengan dua algoritma pengurutan tersebut akan diterjemahkan ke dalam bahasa pemrograman C dan Java. Selanjutnya dibandingkan kecepatan proses pengurutannya menggunakan dua jenis dataset yang sama, random dan descending sebanyak lima puluh ribu dataset.

Adapun pengujian dilakukan dengan menggunakan dua jenis dataset *random* dan *descending* adalah agar memberikat hasil akurasi yang sebaik mungkin. Beberapa algoritma bekerja sangat buruk (lambat) untuk jenis dataset random seperti Bubble Sort. Penyebabnya karena banyaknya iterasi yang harus dilakukan pada semua data. Hal ini seperti yang

dikatakan oleh Astrachan (2003) "*Although popular, bubble sort is nearly universally de-rided for its poor performance on random data. This derision is justified as shown in Figure 1 where bubble sort is nearly three times as slow as insertion sort.*"

2. METODE

Pada penelitian ini digunakan dua jenis dataset, yaitu lima puluh ribu data acak/*random*, dan lima puluh ribu data yang terurut *descending* (terurut besar ke kecil). Masing-masing dataset tersebut disimpan ke dalam file ber-ektensi txt. Kedua file dataset tersebut akan dijadikan sebagai input untuk diproses menggunakan algoritma Insertion Sort dan Merge Sort pada bahasa pemrograman C dan bahasa pemrograman Java. Ouputnya adalah data yang telah terurut dari kecil ke besar beserta waktu yang ditempuh selama proses pengurutannya.

Psudocode untuk algoritma Insertion Sort adalah sebagai berikut:

```
function insertionSort(array A)
for i from 1 to length[A]-1 do
value := A[i]
j := i-1
while j >= 0 and A[j] > value do
A[j+1] := A[j]
j := j-1
done
A[j+1] = value
done
```

Sedangkan psudocode algoritma Merge Sort adalah sebagai berikut:

```
function merge(left,right)
var list result
while length(left) > 0 and
length(right) > 0
if first(left) ? first(right)
append first(left) to result
left = rest(left)
else
append first(right) to result
right = rest(right)
if length(left) > 0
append left to result
if length(right) > 0
append right to result
return result
```

Dataset random dihasilkan dengan program Java sebagai berikut:

```
import java.util.*;
class RandomNumbers {
public static void main(String[]
args) {
int i;
Random r = new Random();
for (i=1; i<=50000; i++) {
System.out.println(r.nextInt(50000)+
1);
}}}
```

Dataset descending dihasilkan dengan program Java sebagai berikut:

```
class DesNumbers {
public static void main(String[]
args) {
int i;
for (i=50000; i>0; i--) {
System.out.println(i);
}}}
```

Perangkat komputer yang digunakan adalah laptop dengan spesifikasi sebagai berikut:
Sistem Operasi : Windows 8.1 Pro 64-bit
Processor : Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz
RAM : 8 GB

3. HASIL & PEMBAHASAN

Dari psudocode di atas, algoritma diterjemahkan ke bahasa pemrograman C dan Java, hasilnya adalah sebagai berikut.

Program Insertion Sort dan Merge Sort dalam bahasa pemrograman C.

```
#include <stdio.h>
#include <time.h>

void printList();
void resetListToDescending();
void resetListToRandom();
void insertionSort(int *a, int n);
void mergeSort(int *a, int n);
void merge(int *a, int n, int m);

int list[50000];
int sizeList = sizeof list / sizeof
list[0];

int main() {
clock_t begin, end;
double time_spent;
int percobaan;
for(percobaan=1; percobaan<=5;
percobaan++) {
printf("==========\n");
printf("PERCOBAAN %d\n", percobaan);
printf("==========\n");
```

```
resetListToDescending();
begin = clock();
insertionSort(list, sizeList);
end = clock();
time_spent = (double) (end -
begin) / CLOCKS_PER_SEC;
printf("Insertion Sort for Data
Descending\t: %f detik\n",
time_spent);

resetListToRandom();
begin = clock();
insertionSort(list, sizeList);
end = clock();
time_spent = (double) (end -
begin) / CLOCKS_PER_SEC;
printf("Insertion Sort for Data
Random\t\t: %f detik\n",
time_spent);

resetListToDescending();
begin = clock();
mergeSort(list, sizeList);
end = clock();
time_spent = (double)(end - begin) /
CLOCKS_PER_SEC;
printf("Merge Sort for Data
Descending\t\t: %f detik\n",
time_spent);

resetListToRandom();
begin = clock();
mergeSort(list, sizeList);
end = clock();
time_spent = (double)(end - begin) /
CLOCKS_PER_SEC;
printf("Merge Sort for Data
Random\t\t: %f detik\n",
time_spent);
printf("\n");
}
return 0;}

void printList() {
int i;
for(i=0; i<sizeList; i++) printf("%d
", list[i]);printf("\n");}

void resetListToDescending() {
int i;
for(i=sizeList-1; i>=0; i--) {
list[sizeList-1-i] = i + 1;}}

void resetListToRandom() {
FILE *data = fopen("random.txt",
"r");
int i=0;
do{fscanf(data, "%d", &list[i]);
i++;
}while(i < sizeList);}

void insertionSort(int *a, int n) {
int i, j, t;
for (i = 1; i < n; i++) {
t = a[i];
for (j = i; j > 0 && t < a[j - 1];
j--) {
a[j] = a[j - 1];}
a[j] = t;}}

void mergeSort(int *a, int n) {
if (n < 2)
return;
int m = n / 2;
mergeSort(a, m);
mergeSort(a + m, n - m);
merge(a, n, m);}

void merge(int *a, int n, int m) {
int i, j, k;
int *x = malloc(n * sizeof (int));
for (i = 0, j = m, k = 0; k < n; k+
+){x[k] = j == n ? a[i++]: i == m
? a[j++]: a[j] < a[i] ? a[j++]:
a[i++];}for (i = 0; i < n; i++)
{a[i] = x[i];}
free(x);}
```

Program Insertion Sort dalam bahasa pemrograman Java

```
public class InsertSort {
private int[] inputNum;
public InsertSort(int[] values){
this.inputNum = values;
Sorting();}
void Sorting(){
int i, j, newValue;
for (i = 1; i < inputNum.length; i+
+) {
newValue = inputNum[i];j = i;
while (j > 0 && inputNum[j - 1] >
newValue) {
inputNum[j]=inputNum[j-1];j--;}
inputNum[j] = newValue;}}
public int[] getNum(){
return (inputNum);}}
```

Program Merge Sort dalam bahasa pemrograman Java.

```
public class MergeSort {
private int[] inputNum;
private int[] helper;
private int number;
```

```
public MergeSort(int[] values){
this.inputNum = values;
number = values.length;
this.helper = new int[number];
mergesort(0, number-1);}

private void mergesort(int low, int
high){
if(low < high){
int middle = low + (high-low)/2;
mergesort(low, middle);
mergesort(middle+1, high);
merge(low, middle, high);}}

private void merge(int low, int
middle, int high){
for(int i=low; i<=high; i++){
helper[i] = inputNum[i];}

int i=low;
int j=middle+1;
int k=low;

while(i<=middle && j<=high){
if(helper[i]<=helper[j]){
inputNum[k] = helper[i];
i++;}else{
inputNum[k] = helper[j];
j++;}k++;}
while(i<=middle){
inputNum[k]=helper[i];
k++;i++;}}

public int[] getNum(){
return (inputNum);}}
```

Program utama yang memanggil class Insertion dan Merge.

```
public class Sorting {
private static int[] numbers;
static private void InputNum(){
try{
File file=new File("random.txt");
BufferedReader read=new
BufferedReader(new
FileReader(file));
String currentLine,currentline2="";
while ((currentLine=read.readLine())
!=null){currentline2=currentline2+
currentLine+"\n";}

String[]temp=currentline2.split("\\n
");
numbers=new int[temp.length];
for (int i=0; i<temp.length;i++){
numbers[i]=Integer.parseInt(temp[i])
;}}catch (IOException e){
System.out.println(e.toString();}}

public static void main(String[]
args){
InputNum();
int[] insertValue, mergeValue;
long insertTimeStart, insertTimeEnd,
mergeTimeStart, mergeTimeEnd,
timeMerge, timeInsert;

mergeTimeStart =
System.currentTimeMillis();
MergeSort ms = new
MergeSort(numbers);
mergeTimeEnd =
System.currentTimeMillis();

insertTimeStart=System.currentTimeMi
llis();
InsertSort is=new
InsertSort(numbers);
insertTimeEnd=System.currentTimeMill
is();
timeMerge = mergeTimeEnd-
mergeTimeStart;
timeInsert=insertTimeEnd-
insertTimeStart;
insertValue = is.getNum();
mergeValue = ms.getNum();

System.out.println("Insertion
Sort");for(int i=0;
i<insertValue.length; i++){
if((i%10)==0){

System.out.println(insertValue[i]);
}else{
System.out.println(insertValue[i] +
"\t");}}
System.out.println("\nEnd of
Insertion Sort");
System.out.println("");
System.out.println("Merge Sort");

for(int i=0; i<insertValue.length;
i++){if((i%10)==0){
System.out.println(mergeValue[i]);
}else{
System.out.println(mergeValue[i]
+"\t");}}
System.out.println("\nEnd of Merge
Sort");
System.out.println("time start:
"+insertTimeStart+" milisekon");
System.out.println("time end:
"+insertTimeEnd+" milisekon");

System.out.println("Processing time
```

```
for Insertion Sort = " +
(timeInsert) + " milisekon ");
System.out.println(" ");
System.out.println("time Start:
"+mergeTimeStart+"
milisekon");System.out.println("time
End: "+mergeTimeEnd+" milisekon");
System.out.println("Processing time
for Merge Sort     = " + (timeMerge)
+ " milisekon ");}}
```

Eksekusi source code program C digunakan compiler berlisensi GNU yaitu gcc versi 4.8. Sedangkan eksekusi source code program Java menggunakan compiler JDK versi 1.7.0_79.

Dari hasil penelitian didapat data sepeti Tabel 2, yaitu waktu yang dibutuhkan menggunakan bahasa pemrograman C dan algoritma Insertion Sort untuk mengurutkan data acak selama 3,564 detik. Data descending selama 6,013 detik. Adapun dengan menggunakan algoritma Merge Sort untuk data acak selama 0,024 detik. Data descending selama 0,024 detik.

Pada Tabel 3 juga terlihat perbedaan waktu yang dibutuhkan oleh algoritma Insertion Sort dan Merge Sort dengan bahasa pemrograman Java. Waktu yang dibutuhkan menggunakan bahasa pemrograman Java dan algoritma Insertion Sort untuk mengurutkan data acak selama 0,503 detik. Data descending selama 1,22 detik. Adapun dengan menggunakan algoritma Merge Sort untuk data acak selama 0,021 detik. Data descending selama 0,028 detik.

Tabel 2. Hasil Penelitian dengan Bahasa C

Bahasa Pemrograman C		
	Data Acak	**Data Descending**
Insertion Sort	3,564000 detik	6,013000 detik
Merge Sort	0,024000 detik	0,024000 detik

Tabel 3. Hasil Penelitian dengan Bahasa Java

Bahasa Pemrograman Java		
	Data Acak	**Data Descending**
Insertion Sort	0,503 detik	1,22 detik
Merge Sort	0,021 detik	0,028 detik

Hasil yang telah didapat menunjukkan bahwa algoritma Merge Sort dapat mengurutkan data lebih cepat dibanding dengan algoritma Insertion Sort.

4. KESIMPULAN

Hasil penelitian menunjukkan algoritma Merge Sort lebih cepat dari Insertion Sort. Lamanya waktu proses pengurutan itu sangat dipengaruhi oleh cara kerja dari masing-masing algoritma. Pada Insertion Sort iterasi dilakukan satu kali pada semua data, yang dimulai dari data ke-2. Prinsip dasarnya adalah secara berulang-ulang menyisipkan setiap elemen ke dalam posisinya/tempatnya yang benar.

Sedangkan pada Merge Sort menerapkan paradigma algoritma “divide and conquer” (bagi dan atasi). Yaitu melakukan pembagian struktur data sebelum kemudian dioperasi satu per satu. Dengan membagi data menjadi lebih kecil tentu langkah-langkah pengurutan menjadi lebih singkat. Hal ini menyebabkan proses pengurutan pada Merge Sort lebih cepat dari Insertion Sort yang menerapkan iterasi langsung pada keseluruhan list data dalam jumlah besar.

Kecepatan proses pengurutan ini terbukti tidak hanya di bahasa pemrograman C, tetapi juga terbukti pada bahasa pemrograman Java.

Jika dilihat dari penelitian sebelumnya, yang dilakukan oleh Reina dan Gautama sampai dengan penelitian ini telah dilakukan, maka diantara ketiga algoritma pengurutan tersebut dari yang paling cepat adalah sebagai berikut: Merge Sort, Insertion Sort, dan Bubble Sort.

DAFTAR PUSTAKA

[1] Astrachan, Owen. (2003). Bubble Sort: An Archaeological Algorithmic Analysis. SIGCSE Bull. Computer Science Department Duke University

[2] Qin, Song. Merge Sort Algorithm. Dept of Computer Sciences, Florida Institute of Technology, Melbourne, FL 32901

[3] Reina; Gautama, Josef Bernadi. (2013). Perbandingan Bubble Sort Dengan Insertion Sort Pada Bahasa Pemrograman C dan Fortran. ComTech Vol. 4 No. 2 Desember 2013: 1106-1115

[4] Robert Sedgewick, Kevin Wayne. (2009). Algorithms in Java, 4th Edition. URL: http://www.sorting-algorithms.com

[5] Wang, Sunny Y. (1996). A New Sort Algorithm: Self-Indexed Sort. SIGPLAN Notices, 31, 28-36. DOI:10.1145/227717.227725

BIODATA PENULIS

Rifkie Primartha, memperoleh gelar sarjana teknik dari Jurusan Teknik Elektro Universitas Sriwijaya Palembang tahun 2002, dan memperoleh gelar Magister Teknik dari Universitas Gadjah Mada Yogyakarta tahun 2007. Saat ini menjadi dosen di Jurusan Informatika Fakultas Ilmu Komputer Universitas Sriwijaya Palembang.