ANALISIS PERBANDINGAN ALGORITMA *BUBBLE SORT*, *MERGE SORT*, DAN *QUICK SORT* DALAM PROSES PENGURUTAN KOMBINASI ANGKA DAN HURUF

Anisya Sonita[1], Febrian Nurtaneo[2]

[1,2]Program Studi Informatika, Fakultas Teknik, Universitas Muhammadiyah Bengkulu

Jl. Bali PO BOX 118. Telp (0736) 227665, Fax (0736) 26161, Bengkulu 38119

[1]anisyasonita@gmail.com
[2]febriannurtaneo@gmail.com

Abstrak: Peran algoritma dalam perangkat lunak atau pemprograman begitu penting, sehingga perlu memahami konsep dasar algoritma. Begitu banyak logika pemprograman yang telah diciptakan, untuk kasus yang umum dan juga khusus. Seiring berkembangnya kemajuan di bidang informatika dan teknologi, tuntutan untuk menemukan metode pemecahan masalah secara lebih tepat, efektif dan kuat menjadi sebuah kebutuhan, terutama untuk permasalahan klasik. Salah satu masalah klasik di bidang tersebut adalah pengurutan data. Pada penelitian ini akan di analisa perbandingan algortima pengurutan data, yaitu: *bubble sort, merge sort,* dan *quick sort* untuk mendapatkan waktu proses yang baik dalam proses pengurutan kombinasi angka dan huruf.
Kata kunci: Algoritma Pengurutan, Bubble Sort, Merge Sort, Quick Sort

Abstract: The role of algorithms in software or programming is so important, so it is necessary to understand the basic concept of the algorithm. So much logic programming that has been created, for the general case and also specific. As the development advances in the field of information and technology, the demand to find a method of solving the problem more precise, effective and powerful become a necessity, especially for classical problems. One of the classic problems in the field of information and computing are data sorting. In this research will be in a comparative analysis of the sorting algorithm bubble sort, merge sort, and quick sort to get a good process in the prossess of sorting a combination of numbers and letters.
Keywords: Sorting Algorithm, Bubble Sort, Merge Sort, Quick Sort

I. PENDAHULUAN

Pengaksesan data yang lebih baik, kuat, dan cepat memerlukan pengolahan data yang lebih baik pula. Salah satu jenis pengolahan data yang menjadi permasalahan klasik adalah pengurutan data bilangan bulat (*integer*). Pengurutan data memegang peranan penting yang banyak dipertimbangkan agar keseluruhan permasalahan (terutama mengenai pengolahan data) menjadi lebih baik dan lebih cepat untuk diselesaikan. Sehingga menghasilkan data yang akurat [1].

Pengurutan data atau *sorting* merupakan salah satu jenis operasi penting dalam pengolahan data. Hampir setiap saat dalam kehidupan sehari- hari sering dijumpai permasalahan-permasalahan yang harus diselesaikan dengan melibatkan operasi pengurutan data. Begitu pentingnya operasi tersebut, sehingga sampai saat ini telah banyak dikembangkan metode-metode pengurutan data dan mungkin akan tetap bermunculan metode-metode baru.

Pengurutan data memegang peranan penting yang banyak dipertimbangkan agar keseluruhan permasalahan (terutama mengenai pengolahan data) menjadi lebih baik dan lebih cepat untuk diselesaikan, sehingga menghasilkan data yang akurat. Masalah pengurutan merupakan masalah yang sering muncul dalam pemrograman komputer. Banyak pengurutan yang berbeda algoritma telah dikembangkan dan ditingkatkan untuk membuat pengurutan lebih cepat. Algoritma pengurutan adalah algoritma yang menyimpan suatu *list* pada suatu urutan tertentu, biasanya membesar atau mengecil, dan digunakan untuk mengurutkan angka ataupun huruf. Efisiensi pada pengurutan ini diperlukan untuk mengoptimalkan kecepatan pemprosesan. Semakin efisien suatu algoritma, maka pada saat dieksekusi dan dijalankan akan menghabiskan waktu yang lebih cepat dan bisa menerima lebih banyak masukan dari *user*.

II. LANDASAN TEORI

A. *Algoritma Sorting*

Algoritma *sorting* adalah kumpulan langkah-langkah penyelesaian dalam suatu masalah dengan metode tertentu, sedangkan *sorting* didefinisikan sebagai pengurutan sejumlah data berdasarkan nilai kunci tertentu untuk mengurutkan nilai dari yang terkecil (*ascending*) atau sebaliknya (*descending*) [2].

B. *Kompleksitas Algoritma*

Kompleksitas suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk mendapatkan hasil yang diinginkan.

Hal-hal yang mempengaruhi kompleksitas waktu:

1. Jumlah masukan data untuk suatu algoritma (n).
2. Waktu yang dibutuhkan unutk menjalankan algoritma tersebut.
3. Ruang memori yang dibutuhkan untuk menjalankan algoritma yang berkaitan dengan struktur data dari program.

Kompleksitas mempengaruhi performa atau kinerja dari suatu algoritma. Kompleksitas dibagi menjadi 3 jenis yaitu, *worst case*, *best case*, dan *average case*. Masing-masing jenis kompleksitas ini menunjukkan kecepatan atau waktu yang dibutuhkan algoritma untuk mengeksekusi sejumlah kode.

C. *Algoritma Bubble Sort*

Bubble sort merupakan salah satu jenis *sorting*. Ide dari algoritma ini adalah mengulang proses pembandingan antara tiap-tiap elemen *array* dan menukarnya apabila urutannya salah. Pembandingan elemen-elemen ini akan terus diulang hingga tidak perlu dilakukan penukaran lagi. Algoritma ini termasuk dalam golongan algoritma *comparison sort*, karena menggunakan perbandingan dalam operasi antar elemennya[3].

Langkah-langkah dalam pengurutan dalam *bubble sort*, sebagai berikut: misalkan kita mempunyai sebuah *array* dengan elemen-elemen “ 4 2 5 3 9”. Proses yang akan terjadi apabila menggunakan algoritma *bubble sort* adalah sebagai berikut:

Pass pertama

(**4 2** 5 3 9) menjadi (**2 4** 5 3 9)
(2 **4 5** 3 9) menjadi (2 **4 5** 3 9)
(2 4 **5 3** 9) menjadi (2 4 **3 5** 9)
(2 4 3 **5 9**) menjadi (2 4 3 **5 9**)

Pass kedua

(**2 4** 3 5 9) menjadi (**2 4** 3 5 9)
(2 **4 3** 5 9) menjadi (2 **3 4** 5 9)
(2 3 **4 5** 9) menjadi (2 3 **4 5** 9)
(2 3 4 **5 9**) menjadi (2 3 4 **5 9**)

Pass ketiga

(**2 3** 4 5 9) menjadi (**2 3** 4 5 9)
(2 **3 4 5** 9) menjadi (2 **3 4** 5 9)

(2 3 **4 5** 9) menjadi (2 3 **4 5** 9)

(2 3 4 **5 9**) menjadi (2 3 4 **5 9**)

Dilihat dari proses di atas, sebenarnya pada pass kedua, langkah kedua, *array* telah terurut. Namun algoritma tetap dilanjutkan hingga pass kedua berakhir. *Pass* ketiga dilakukan karena definisi terurut dalam algoritma *bubble* sort adalah tidak ada satupun penukaran pada suatu pass, sehingga pass ketiga dibutuhkan untuk memverikasi keurutan *array* tersebut.

D. *Algoritma Quick sort*

Quicksort merupakan Algoritma *Sorting* yang dikembangkan oleh C.A.R Hoare pada tahun1960 yang secara kasus rata-rata, membuat pengurutan O(*n* log *n*) untuk mengurutkan *n* item. Algoritma ini juga dikenal sebagai *Partition-Exchange Sort* atau disebut sebagai *Sorting* pergantian pembagi. Pada kasus terburuknya, algoritma ini membuat perbandingan O(n^2), walaupun kejadian seperti ini sangat langka. *Quick sort* sering lebih cepat dalam praktiknya dari pada algoritma *O(n log n)* yang lainnya. Dan juga, urutan dan referensi lokalisasi memori quicksort bekerja lebih baik dengan menggunakan cache CPU, jadi keseluruhan *sorting* dapat dilakukan hanya dengan ruang tambahan *O(log n*) [4].

Pada masalah pengurutan data bilangan bulat (*integer*) secara terindeks pada suatu *list* atau *array* dari bilangan yang paling besar sampai ke bilangan yang paling kecil atau sebaliknya. Tidak hanya dapat diterapkan pada pengindeksan bilangan saja, namun juga untuk pengindeksan huruf (abjad) dari A ke Z atau sebaliknya. Algoritma ini sangat baik diterapkan pada kasus pengindeksan kumpulan kata *(library sort utility)* atau kumpulan bilangan atau kombinasinya.

Algoritma ini mengikuti langkah– langkah sebagai berikut :

1. *Divide*

Memilah rangkaian data menjadi dua sub-rangkaian A[p…q-1] dan A[q+1…r] dimana setiap unsur A[p…q-1] adalah kurang dari atau sama dengan A[q] dan setiap unsur pada A[q+1…r] adalah lebih besar atau sama dengan unsur pada A[q]. A[q] disebut sebagai unsur *pivot*. Perhitungan pada unsur q merupakan salah satu bagian dari prosedur pemisahan.

2. *Conquer*

Mengurutkan unsur pada subrangkaian secara rekursif Pada algoritma *quicksort*, langkah ”kombinasi” tidak dilakukan karena telah terjadi pengurutan unsur – unsur pada sub-*array*. *Quicksort* termasuk pada pendekatan sulit membagi, mudah menggabung (*hard split/easy join*).

Cara pemilihan *pivot*:

1) *Pivot* = unsur pertama/unsur terakhir/unsur tengah tabel

2) *Pivot* dipilih secara acak dari salah satu unsur tabel.

Pivot = unsur median tabel

Algoritma ini terdiri dari 4 langkah utama:

1. Jika struktur data terdiri dari 1 atau 0 elemen yang harus diurutkan, kembalikan struktur data tersebut itu apa adanya.
2. Ambil sebuah elemen yang akan digunakan sebagai *pivot point* (*point* poros). (biasanya elemen yang paling kiri)
3. Bagi struktur data menjadi dua bagian, satu dengan elemen-elemen yang lebih besar dari pada *pivot point*, dan yang lainnya dengan elemen-elemen yang lebih kecil dari pada *pivot point.*
4. Ulangi algoritma secara rekursif terhadap kedua paruh struktur data.

E. *Algoritma Merge Sort*

Merge sort adalah metode pengurutan yang menggunakan pola *divide* and *conquer*. Strateginya adalah dengan membagi sekelompok data yang akan diurutkan menjadi beberapa kelompok kecil terdiri dari maksimal dua nilai untuk dibandingkan dan digabungkan lagi secara keseluruhan [5].

Langkah-langkah Kerja dalam *Merge sort*:

1. *Divide*

Memilah elemen – elemen dari rangkaian data menjadi dua bagian dan mengulangi pemilahan hingga satu elemen terdiri maksimal dua nilai.

2. *Conquer*

Mengurutkan masing-masing elemen.

3. Kombinasi

Mengkombinasikan dua bagian tersebut secara rekursif untuk mendapatkan rangkaian data berurutan.

Proses rekursi berhenti jika mencapai elemen dasar. Hal ini terjadi bilamana bagian yang akan diurutkan menyisakan tepat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah terurut sesuai rangkaian.

III. METODELOGI PENELITIAN

Metode penelitian dalam penyusunan penelitian ini adalah metode pengumpulan data. Metode ini bertujuan untuk mendukung dalam memperoleh informasi yang dibutuhkan dalam rangka mencapai tujuan penelitian. Tujuan yang dimaksudkan dalam bentuk hipotesis merupakan jawaban sementara terhadap pertanyaan penelitian, yang merupakan elemen penting dalam penelitian. Teknik pengumpulan data yang benar akan menghasilkan hasil perbandingan dari apa yang diteliti sebelumnya.

Adapun metode pengumpulan yang diterapkan peneliti :

1. Studi Laboratorium

Yaitu studi laboratorium dimana dalam aplikasi dan mempraktekkan langsung aplikasi tersebut.

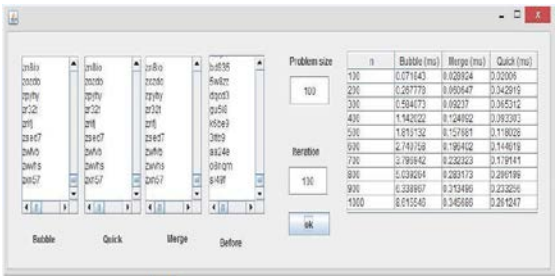
2. Studi Pustaka

Yaitu pengumpulan data yang berhubungan dengan pemrograman Java baik itu melalui buku maupun melalui jaringan internet.

IV. HASIL DAN PEMBAHASAN

A. *Tampilan Aplikasi*

Menu utama merupakan Tampilan awal dari program Analisis Perbandingan Algoritma Pengurutan *Bubble sort* dan *Quick sort* dalam proses pengurutan kombinasi angka dan huruf saat aplikasi dijalankan. Berikut ini tampilan utamanya:



Gambar 1. Tampilan aplikasi

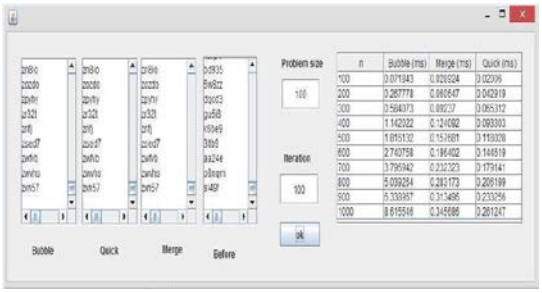
Menu program yang terdapat pada menu utama program ini terdiri dari:

1. *Text Area Bubble* yang digunakan untuk menampilkan hasil data yang diurutkan oleh *Bubble sort.*
2. *Text Area Quick* yang digunakan untuk menampilkan hasil data yang diurutkan oleh *Quick sort.*
3. *Text Area Merge* yang digunakan untuk menampilkan hasil data yang diurutkan oleh *Merge sort.*
4. *Text Area Before* yang digunakan untuk menampilkan data acak sebelum diurutkan.

5. *Text Field Problem Size* yang digunakan untuk masukkan *input* besaran nilai masalah yang diingikan.
6. *Text Field Iteration* yang digunakan untuk masukkan *input* besaran nilai iterasi yang diinginkan.
7. *Button Ok* yang digunakan untuk proses pengurutan.
8. *Table* yang digunakan untuk menampilkan hasil dari proses pengurutan yang berupa waktu.

B. *Implementasi Aplikasi*

Pertama akan di *input* nilai iterasi dan nilai problem *size*nya pada angka 100. Untuk lebih jelasnya akan tampilkan seperti pada gambar dibawah ini:



Gambar 2. Proses pengurutan data

Seperti yang terlihat pada gambar diatas, proses pengurutan menggunakan algoritma *Bubble sort* memiliki waktu 0,071843 sedangkan waktu yang digunakan algoritma *Quick sort* adalah 0,02006 dan waktu algoritma *merge sort* adalah 0,028924. Jadi berdasarkan perbandingan diatas, algoritma *Quick sort* lebih cepat dari pada algoritma *Bubble sort* dan *merge sort* untuk nilai *problem size* dan *iteration* 100.

C. *Pengujian Aplikasi*

Pada penelitian ini digunakan metode pengujian *black box* untuk menguji aplikasi yang telah dibuat dalam segi tampilan *user interface*.

Tabel 1. Hasil Pengujian *Black Box*

No	*Requirement* yang diuji	Objek uji	Hasil Pengujian
1	*Problem Size*	Aplikasi melakukan proses pengurutan	Sesuai
2	*Iteration*	Aplikasi melakukan proses pengurutan	Sesuai

D. *Hasil Pengujian Terhadap Waktu Proses*

Hasil pengujian pada beberapa data di atas dirangkum dalam table berikut yang menunjukkan waktu prosespengurutan.

Tabel 2. Hasil Perbandingan Pada Proses *Problem Size*

No	***Problem Size***	***Itera-tion***	***Bubble sort***	***Quick sort***	***Merge sort***
			Waktu Proses (ms)		
1	10	100	0.03352	0.003911	0.005444
2	20	100	0.006146	0.005029	0.007511
3	30	100	0.010895	0.006705	0.007899
4	40	100	0.018718	0.008102	0.009081
5	50	100	0.027936	0.011174	0.024599
6	60	100	0.041346	0.012572	0.027880
7	70	100	0.055594	0.015086	0.037756
8	80	100	0.071797	0.017699	0.045112
9	90	100	0.090514	0.018997	0.058945
10	100	100	0.112026	0.021791	0.036668

Tabel 3. Hasil Perbandingan Pada Proses *Iteration*

No	***Problem Size***	***Itera-tion***	***Bubble sort***	***Quick sort***	***Merge sort***
			Waktu Proses (ms)		
1	100	10	0.108114	0.020952	0.048933
2	100	20	0.111188	0.02207	0.049998
3	100	30	0.109511	0.021231	0.051287
4	100	40	0.12795	0.025981	0.059563
5	100	50	0.112026	0.021791	0.059817
6	100	60	0.110629	0.022908	0.051222
7	100	70	0.112305	0.022628	0.052348
8	100	80	0.109791	0.021511	0.059988
9	100	90	0.11007	0.023746	0.059944
10	100	100	0.109511	0.022349	0.058898

V. PENUTUP

Pembuatan program komputer tidak terlepas dari algoritma, apalagi program yang dibuat sangat

kompleks. Program dapat dibuat dengan mengabaikan algoritma, akan tetapi program tersebut memiliki akses yang lambat atau bahkan sangat lambat dan memakai memori yang banyak. Dalam menguji suatu algoritma, dibutuhkan beberapa kriteria untuk mengukur efisiensi algoritma, kriterianya adalah memeriksa kebenaran algoritma dengan cara matematis dan menyederhanakannya.

Dari hasil analisa perbandingan Algoritma *Bubble Sort*, *Quick Sort*, dan *Merge Sort*, dapat disimpulkan bahwa Algoritma *Quick Sort* memiliki waktu yang lebih cepat dan *Bubble Sort* membutuhkan waktu komputasi yang paling lama.

REFERENSI

[1] Fauzi, Indrayana, 2005. Pebandingan Kecepatan/Waktu Komputasi Beberapa Algortima Pengurutan (*Sorting*), Institut Teknologi Bandung.

[2] Wisudawan, Wahyu Fahmy, 2007. *Kompleksitas Algoritma Sorting yang Populer Dipakai*. Bandung: Teknik Informatika, Institut Teknologi Bandung.

[3] Rheinadi, Ryan, 2009. *Analisis Algoritma Bubble sort*. Makalah IF2091 Strategi Algoritmik Tahun 2009. Program Studi Teknik Informatika, Sekolah Elektro dan Informatika, Institut Teknologi Bandung.

[4] Yahfizham, 2008. Analisis Waktu Algoritma *Quick sort* dan *Merge sort*.

[5] Hendra Saptadi, Arief . 2013, *Analisis Algoritma Insertion sort, Merge sort dan Implementasinya dalam Bahasa Pemograman C++*. Akademi Teknik Telekomunikasi Shandy Putra Purwokerto