IMPLEMENTASI ALGORITMA *BUBBLE SORT* DAN *SELECTION SORT* MENGGUNAKAN *ARRAYLIST* MULTIDIMENSI PADA PENGURUTAN DATA MULTI PRIORITAS

[1]Roma Rio Sitepu, [2] Machudor Yusman & [3] Febi Eka Febriansyah

[1]Jurusan Ilmu Komputer FMIPA Unila
[2]Jurusan Ilmu Komputer FMIPA Unila
[3]Jurusan Ilmu Komputer FMIPA Unila

Abstract

Ordering is a process of sorting data with a certain rule, so it will be arranged regularly in accordance with the rules. Basically, there are two kinds of commonly used sequencing rules: ascending and descending. In sorting data that consists of many criteria, there will be priority criteria on which the data is sorted. Priority criteria are sorted by initial criteria which are the most preferred criteria until the last criterion which is an additional criterion. In this study, researchers developed an algorithm that can help sort data that has many priorities. The algorithm used is Bubble Sort and Selection Sort, developed using multidimensional ArrayList to sort multi-priority data. The test was conducted on five times experiment and the data sorted by descending. The data used consist of four fictitious data and one real data taken from the list of all criteria of PPA 2017 scholarship recipient of MIPA faculty, University of Lampung. From the results of the tests performed, the best solution is found in the Selection Sort algorithm if compared with the Bubble Sort algorithm for multi priotity data sorting.

Keywords: *arraylist, bubble sort, selection sort, multi priority, multidimensional.*

1. Pendahuluan

Utami [4] pengurutan (*sorting*) merupakan suatu proses mengurutkan data dengan suatu aturan tertentu, sehingga tersusun secara teratur sesuai dengan aturan tersebut. Pada dasarnya ada dua macam aturan pengurutan yang biasa digunakan yaitu *ascending* (proses pengurutan data dari data yang paling kecil sampai data yang paling besar.) dan *descending (*proses mengurutkan data dari yang paling besar sampai data yang paling kecil.*)*. Proses yang terjadi dalam pengurutan data adalah proses perbandingan data dan pertukaran data.

Jika data terdiri dari banyak kriteria, maka akan ada kriteria prioritas yang menjadi dasar data tersebut diurutkan. Contohnya dalam kasus perangkingan, dimana kriteria jumlah total menjadi satu-satunya prioritas yang menjadi dasar pengurutan data. Kriteria jumlah total biasanya didapatkan dengan menjumlahkan nilai dari kriteria-kriteria lainnya. Permasalahan akan muncul ketika pada proses perangkingan, kriteria jumlah total yang merupakan satu-satunya prioritas memiliki nilai yang sama. Hal ini menjadi masalah karena akan sulit melakukan perangkingan secara adil jika hanya memiliki satu prioritas. Mengatasi permasalahan ini dibutuhkan prioritas lain agar data dapat terurut secara adil, dan perlu dibuatkan suatu algoritma yang dapat mengurutkan data multi kriteria yang memiliki lebih dari satu prioritas.

2. Metodologi

Dalam penelitian ini diimplementasikan algoritma *Bubble Sort* dan *Selection Sort* menggunakan *ArrayList* multidimensi untuk pengurutan data berdasarkan prioritas. Kemudian dibandingkan algoritma *Bubble Sort* dan *Selection Sort* berdasarkan waktu eksekusi dan jumlah perulangan, untuk mendapatkan mana algoritma yang paling baik diantara kedua algoritma tersebut.

2.1 *Bubble Sort*

Kristanto [3] pengurutan yang menggunakan algoritma *Bubble Sort* dilakukan dengan cara membandingkan elemen yang diseleksi dengan elemen yang berikutnya. Jika elemen yang diseleksi lebih besar dari elemen berikutnya maka elemen tersebut ditukar. Setiap langkah dari algoritma ini seolah-olah menggeser satu per satu elemen dari kanan ke kiri. Jika barisan bilangan tidak disusun horizontal melainkan vertikal, maka terlihat seperti gelembung-gelembung (*bubble*) yang naik dari dasar akuarium. Oleh karena itu algoritma ini disebut *Bubble Sort*.

Yahya [5] *Bubble Sort* merupakan cara pengurutan yang sederhana. Konsep dari ide dasarnya adalah seperti "gelembung air" untuk elemen struktur data yang semestinya berada pada posisi awal. Cara kerja 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 *Bubble Sort* ditunjukkan pada Kode 1.

```
BubbleSort(data,n):
//data adalah array dengan n data
       FOR i ← 0 TO n-2
              FOR j ← 0 TO n-(i+2)
                            IF data[j]<data[j+1]
                                   temp←data[j]

              data[j]←data[j+1]
                                   data[j+1]←temp
                            END IF
              END FOR
       END-FOR
```

Kode 1 Algoritma Bubble Sort

Kadir [2] kunci pengurutan secara *ascending* terletak pada perintah IF data[j]>data[j+1], sementara untuk pengurutan secara *descending*, perintahnya menjadi IF data[j]<data[j+1].

Berdasarkan algoritma *Bubble Sort*, jumlah pembandingan maksimal selama pengurutan ditunjukkan pada persamaan (1):

$$(n-1)+(n-2)+(n-3)+...+2+1=\sum_{i=1}^{n-1} i=\frac{n(n-1)}{2} \qquad (1)$$

Efisiensi ini berlaku untuk *best case*, *worst case* maupun *average case*. Untuk n = 5 maka jumlah pembandingan data sebanyak ((5x4)/2) = 10 kali. Dengan demikian jika dinyatakan dengan Big O maka kompleksitas *Bubble Sort* berupa O(n(n-1)/2) atau sama dengan O(n^2). *Best case* terjadi kalau data yang diurutkan sudah dalam keadaan urut.

Algoritma *Bubble Sort* menggunakan *ArrayList* multidimensi ditunjukan pada Kode 2.

```
BubbleSort(data,n):
       FOR i ← 0 TO data.size()-1
              FOR j ← 0 TO data.size()-(i+1)
                     FOR k← 0 TO data.get(j)size()-1
       hasil=data.get(j).get(k)- data.get(j+1).get(k)
                                   IF hasil !=0
                                          IF hasil<0
                                   // Lakukan Penukaran
                                          Swap(data,j,j+1)
                                          END IF
                                          BREAK
                                   END-IF
                            END FOR
                     END-FOR
       END-FOR
```

dim

Kode 2 Algoritma Buble Sort Menggunakan ArrayList Multidimensi

Pada Kode 2 terdapat 3 buah perulangan yaitu perulangan i,j,dan k. Perulangan "j" berada di dalam perulangan "i" dan perulangan "k" berada di dalam perulangan "j", perulangan seperti ini disebut sering disebut dengan perulangan bersarang (nested loop). Perulangan i dimulai dari 0 dan i akan terus melakukan perulangan sampai data.size()-1. Perulangan j dimulai dari 0 dan j akan terus melakukan perulangan sampai data.size()-(i+1). Perulangan k dimulai dari 0 dan k akan terus melakukan perulangan sampai data.get(j)size()-1.

Di dalam perulangan k terdapat dua buah kondisi, yaitu kondisi jika nilai hasil!=0 dan kondisi jika nilai hasil<0. Nilai "hasil" didapat dari pengurangan nilai data baris ke j dan kolom ke k yang dikurangkan dengan data baris j+1 dan kolom ke k. Jika kondisi 1 tidak terpenuhi maka akan kembali ke looping k. Jika kondisi 1 terpenuhi dan kondisi 2 tidak terpenuhi maka progeram akan kembali ke looping j. Jika kedua kondisi tersebut terpenuhi maka artinya nilai dari data j lebih kecil dari nilai data j+1 dan akan dilakukan pertukaran posisi antara data j dengan data j+1. Setelah pertukaran data maka program kembali ke perulangan j. Proses tersebut akan terus berulang hingga seluruh proses perulangan selesai.

2.2 *Selection Sort*

Yahya [5] *Selection Sort* adalah suatu metode pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya sampai ke elemen yang terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisisnya dan langsung ditukar. Konsep proses *Selection Sort* adalah mencari (memilih) nilai terkecil dan menukarnya dengan elemen paling awal (paling kiri) pada setiap tahap. Proses Sort dilakukan tahap per tahap.

Agustia [1] konsep *Selection Sort* secara *descending* adalah melakukan pemilihan dari suatu nilai yang terbesar dan kemudian menukarnya dengan elemen paling awal, lalu membandingkan dengan elemen yang sekarang dengan elemen berikutnya sampai dengan elemen terakhir, perbandingan dilakukan terus sampai tidak ada lagi pertukaran data. Algoritma *Selection Sort* ditunjukkan pada Kode 3.

```
Selection Sort(data,n):
//data adalah array dengan n data
      FOR i ← 0 TO n-2
            awal=i
             FOR j ← i+1 TO n-(i+2)
                   IF (data [awal] > data[j])
                         awal = j;
                         END IF
                   END FOR
                    temp←data[i]
                     data[i]←data[awal]
                     data[awal]←temp
               END FOR
```

Kode 3 Algoritma Selection Sort

Kompleksitas waktu pada *Selection Sort* diukur dengan jumlah perbandingan. Pada tahap pertama terdapat N-1 perbandingan. Pada tahap kedua terdapat N-2 perbandingan. Pada tahap ketiga terdapat N-3 perbandingan, dan seterusnya. Dengan demikian, total perbandingan ditunjukkan pada persamaan (2):

$$(n-1)+(n-2)+(n-3)+...+2+1 \quad (2)$$

Oleh karena itu, kompleksitas waktu berupa $O(N^2)$.

Worst case diperoleh ketika data dalam keadaan urut terbalik. Perbandingan yang terjadi N(N-1)/2. Dengan demikian, *worst case* berupa $O(N^2)$ (Kadir, 2011). Algoritma *Selection Sort* menggunakan *ArrayList* multidimensi ditunjukan pada Kode 4.

```
SelectionSort(data,data.size()):
      FOR i ← 0 TO data.size()-1
            awal=i
            FOR j ← i+1 TO data.size()-1
                  FOR k← 1 TO data.get(j).size -1
hasil=data.get(awal).get(k)- data.get(j).get(k)
                              IF hasil !=0
                                    IF hasil<0
                              // Lakukan Penukaran
                                    awal=j
                                    END IF
                                    BREAK
                              END-IF
                        END FOR
               END-FOR
                    Swap(data,i,awal);
      END-FOR
```

Kode 4 Algoritma Selection Sort Menggunakan ArrayList Multidimensi

Pada Kode 4 terdapat 3 perulangan yaitu perulangan i,j,dan k. Perulangan i dimulai dari 0 dan i akan terus melakukan perulangan sampai data.size()-1. Perulangan j dimulai dari i+1 dan j akan terus melakukan perulangan sampai data.size()-1. Perulangan k dimulai dari 1 dan k terus melakukan perulangan sampai data.get(j).size -1.

Di dalam perulangan k terdapat inisialisasi dari nilai "hasil" yang didapat dari pengurangan nilai data baris ke awal dan kolom ke k yang dikurangkan dengan data baris j dan kolom ke k. Terdapat juga dua buah kondisi, yaitu kondisi jika nilai hasil!=0 dan kondisi jika nilai hasil<0. Jika kondisi pertama tidak terpenuhi maka proses perulangan kembali ke perulangan k. Jika kondisi pertama terpenuhi dan kondisi kedua tidak terpenuhi maka proses perulangan kembali menuju perulangan j. Jika kedua kondisi tersebut terpenuhi maka dapat diartikan bahwa nilai baris ke awal lebih kecil dari nilai baris ke j. Hal itu juga mengartikan bahwa baris ke j lebih prioritas dari baris ke awal, maka dilakukan penukaran posisi awal dengan j. Setelah proses perulangan j selesai dan saat menuju kembali ke perulangan i, dilakukan pertukaran nilai antara data baris ke i dengan data baris ke awal.

3. Pembahasan

Beberapa tahapan yang sudah dilakukan didapatkan bahwa untuk melakukan penelitian ini dibutuhkan data untuk melakukan pengujian.

3.1 Kebutuhan Data

Data yang digunakan pada pernelitian ini adalah data fiktif dan data daftar seluruh kriteria calon penerima beasiswa PPA 2017 fakultas MIPA Universitas Lampung. Data fiktif digunakan untuk menunjukkan bahwa algoritma sorting yang dibuat telah sesuai dan tidak terjadi kesalahan dalam pengurutan multi prioritas. Data fiktif juga digunakan untuk membandingkan jumlah waktu eksekusi dan jumlah perulangan algoritma *Bubble Sort* dan *Selection Sort*, sehingga didapatkan algoritma terbaik diantara kedua algoritma tersebut. Pengurutan data dilakuan secara *descending* berdasarkan prioritas.

Data daftar seluruh kriteria calon penerima beasiswa PPA digunakan untuk memperlihatkan implimentasi algoritma sorting multi prioritas dalam kasus real. Terdapat 5 jenis data yang akan diuji, 4 jenis data fiktif dan 1 data daftar seluruh kriteria calon penerima beasiswa PPA 2017 fakultas MIPA Universitas Lampung. Pengujian dilakukan sebanyak 3 kali pada setiap data dengan menggunakan algoritma *Bubble Sort* dan *Selection Sort*. Rincian jumlah record pengujian data ditunjukkan pada Tabel 1.

Tabel 1 Rincian Jumlah Record Pengujian Data

Nama Data	Jumlah	Jumlah Kolom
Data 1	50	5
Data 2	100	5
Data 3	50	8
Data 4	100	8
Real	136	6

3. 2. Hasil Perbandingan Algoritma *Bubble Sort* dan *Selection Sort* Pada Data Multi Prioritas

Setelah melakukan perbandingan algoritma *Bubble Sort* dan *Selection Sort*, didapatkan hasil pengujian yang ditunjukkan pada Tabel 2 dan Tabel 3:

Tabel 2 Hasil Pengujian Data Menggunakan Algoritma Bubble Sort

Data	Iterasi	*Bubble Sort* 1	*Bubble Sort* 2	*Bubble Sort* 3	Rata- Rata
Data 1	3603	0.006323297	0.006671106	0.006598123	0.006531
Data 2	9420	0.011505663	0.011945272	0.011592333	0.011681
Data 3	3922	0.006061584	0.006508035	0.006049601	0.006206
Data 4	12137	0.012203563	0.012297643	0.012295933	0.012266
Real	18504	0.038297853	0.037805218	0.040544932	0.038882

Tabel 3 Hasil Pengujian Data Dengan Algoritma Selection Sort

Data	Iterasi	Selection 1	Selection 2	Selection 3	Rata-Rata
Data 1	2019	0.004476483	0.004545475	0.004596791	0.004544
Data 2	7504	0.008675209	0.008710007	0.008988318	0.008791
Data 3	2537	0.004520958	0.004468501	0.004651528	0.004547
Data 4	10269	0.009678807	0.009579595	0.009807097	0.009688
Real	15204	0.029765679	0.031172883	0.031292621	0.030743

Diagram Perbandingan Jumlah Iterasi Algoritma *Bubble Sort* dan *Selection Sort* ditunjukkan pada Figure 1.

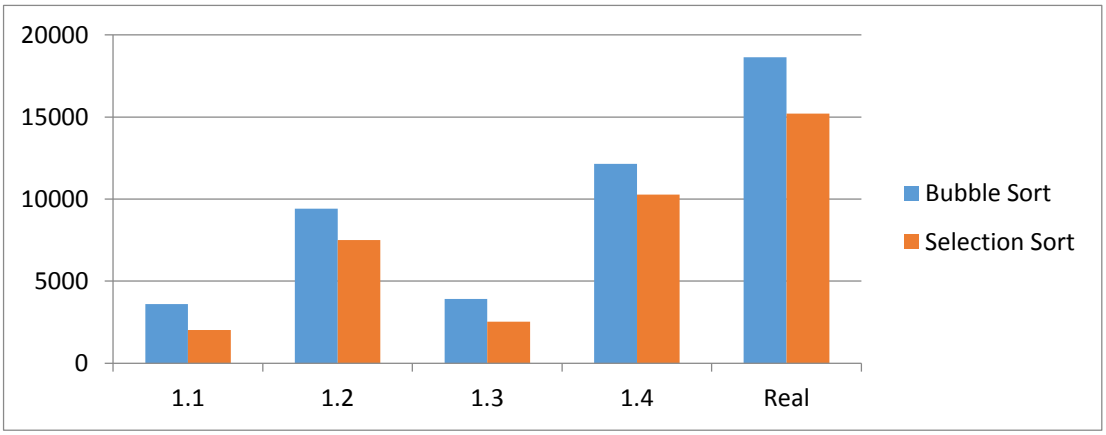

Figure 1 Diagram Perbandingan Jumlah Iterasi Algoritma *Bubble Sort* dan *Selection Sort*

Diagram Perbandingan Waktu Eksekusi Algoritma *Bubble Sort* dan *Selection Sort* ditunjukkan pada Figure 2.

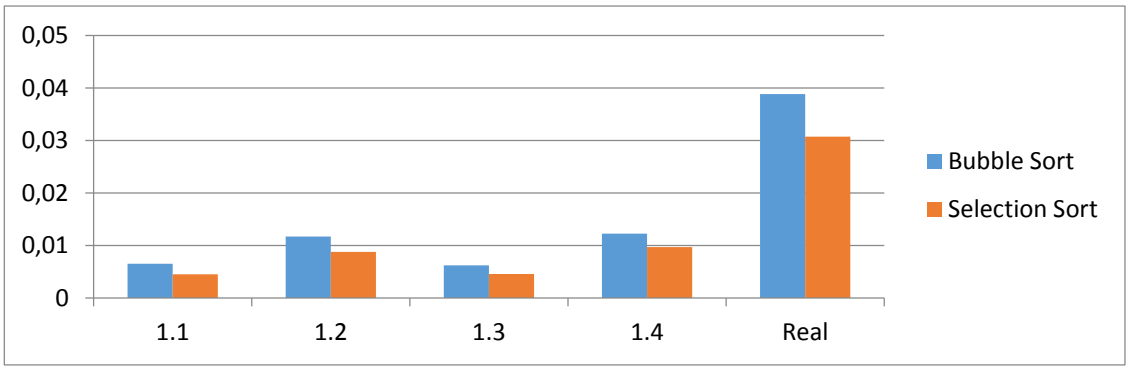

Figure 2 Diagram Perbandingan Waktu Eksekusi Algoritma *Bubble Sort* dan *Selection Sort*

Berdasarkan Figure 1 dan 2 jumlah iterasi *Selection Sort* pada setiap pengujian data lebih sedikit jika dibandingkan dengan *Bubble Sort*. Waktu Eksekusi *Selection Sort* pada setiap pengujian data juga lebih cepat jika dibandingkan dengan waktu eksekusi *Bubble Sort*.

4. Simpulan

Berdasarkan hasil penelitian yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Algoritma Bubble Sort dan Selection Sort dapat diimplementasikan menggunakan ArrayList multidimensi.

2. Dengan menggunakan algoritma Bubble Sort dan Selection Sort yang diimplementasikan menggunakan ArrayList multidimensi, dapat mengurutkan data multi kriteria yang memiliki lebih dari satu kriteria prioritas .

3. Implementasi algoritma Bubble Sort dan Selection Sort menggunakan ArrayList multidimensi dapat digunakan untuk mengurutkan data daftar mahasiswa yang layak menerima beasiswa PPA.

4. Jumlah iterasi Selection Sort pada setiap pengujian data lebih sedikit jika dibandingkan dengan Bubble Sort, dan waktu eksekusi Selection Sort pada setiap pengujian data juga lebih cepat jika dibandingkan dengan waktu eksekusi Bubble Sort.

5. Referensi

[1] Agustia, Panny. 2016. Analisis Perbandingan Kompleksitas Algoritma Pengurutan Nilai (Sorting). *Jurnal Evolusi* Vol. IV No 2.
[2] Kadir,Abdul. 2011. *Konsep dan Implementasi Struktur Data dalam Pemrograman Delphi*. ed. **1** Yogyakarta: Andi.
[3] Kristanto,Andri.2009. *Struktur Data dengan C++*. Yogyakarta: Graha Ilmu.
[4] Utami, E., Raharjo, S., & Sukrisno. 2007. *Struktur Data Konsep & Implementasinya dalam Bahasa C & Free Pascal di GNU/LINUX*. Yogyakarta: Graha Ilmu.
[5] Yahya, Sofyansyah, Y. 2014. Analisa Perbandingan Algoritma Bubble Sort dan Selection Sort Dengan Perbandingan Eksponensial, *Jurnal Pelita Informatika Budi Darma*. Vol. VI No 3