INFORMATION MANAGEMENT FOR EDUCATORS AND PROFESSIONALS
Vol. 3, No. 1, Desember 2018, 95 - 106
E-ISSN: 2548-3331

Algoritma Pengurutan Data *(Sorting) Dengan* Metode *Insertion Sort* dan *Selection Sort*

Endang Retnoningsih [1,*]

[1] Sistem Informasi; STMIK Bina Insani; Jl. Siliwangi No.6 Rawa Panjang Bekasi Bekasi Timur 17114 Indonesia, Telp. (021) 824 36 886 / (021) 824 36 996. Fax. (021) 824 009 24; e-mail: endang.retnoningsih@binainsani.ac.id

* Korespondensi: e-mail: endang.retnoningsih@binainsani.ac.id

Diterima: 17 November 2018; Review: 21 November 2018; Disetujui: 29 November 2018

Cara sitasi: Retnoningsih E. 2018. Algoritma Pengurutan Data *(Sorting)* Dengan Metode *Insertion Sort dan Selection Sort*. Information Management For Educators And Professionals. 3 (1): 95 - 106.

Abstrak: Pengurutan merupakan hal yang tidak bisa dipisahkan dari dunia komputer. Adanya kebutuhan terhadap proses pengurutan memunculkan bermacam-macam metode pengurutan yang bertujuan untuk memperoleh metode pengurutan yang optimal. Dengan menggunakan algoritma yang baik dapat menghasilkan program yang efisien dari segi waktu dan hasil yang dicapai. Data terkadang berada dalam bentuk yang tidak berpola ataupun dengan pola tertentu yang diinginkan. Secara umum ada dua jenis pengurutan data yaitu model urut naik (ascending) yang mengurutkan data dari yang mempunyai nilai terkecil sampai terbesar dan model urut turun (descending) yang mengurutkan data dari yang mempunyai nilai terbesar sampai terkecil. Perbandingan pengurutan data (sorting) dalam sebuah array (L) menggunakan dua algoritma dengan prinsip kerja yang berbeda. Pada penelitian ini membandingkan algoritma metode selection sort menggunakan prinsip pertukaran elemen dalam proses sorting, dan metode insertion sort menggunakan prinsip geser dan sisip elemen dalam proses sorting. Tidak ada algoritma terbaik untuk semua keadaan, kadang kala sebuah algoritma sangat efisien ketika jumlah datanya sedikit, namun kinerjanya menjadi berkurang ketika jumlah data ditambahkan atau meningkat. Hasil dari penelitian adalah metode *insertion sort* lebih unggul pada jumlah data yang sedikit, sedangkan metode *selection sort* lebih unggul pada jumlah data yang lebih banyak.

Kata kunci: Algoritma, Larik, Insertion Sort, Selection Sort, Visual Basic.

Abstract: *Ordering is something that cannot be separated from the computer world. The need for the sorting process gives rise to a variety of sorting methods that aim to obtain the optimal sorting method. Using a good algorithm can produce programs that are efficient in terms of time and results achieved. Data sometimes is in a form that is not patterned or with a certain pattern desired. In general there are two types of sorting data, namely the ascending model which sorts data from the smallest to the largest and the descending sequence model which sorts data from the largest to the smallest. Comparison of sorting in an array (L) uses two algorithms with different working principles. In this study comparing algorithms the selection sort method uses the principle of element exchange in the sorting process, and the insertion sort method uses the principle of sliding and inserting elements in the sorting process. There is no best algorithm for all situations, sometimes an algorithm is very efficient when the amount of data is small, but its performance decreases when the amount of data is added or increased. The results of the study are that the insertion sort method is superior to the small amount of data, while the selection sort method is superior to a larger amount of data.*

Keywords*: Algorithms, Array, Insertion Sort, Selection Sort, Visual Basic.*

E-ISSN: 2548-3331; 95 – 106

1. Pendahuluan

Algoritma merupakan bagian terpenting dan tidak dapat dipisahkan dari pemrograman. Sebelum membuat suatu program aplikasi, hal pertama yang perlu dipahami adalah algoritma atau prosedur pemecahan masalah. Beberapa hal yang perlu diperhatikan dalam merancang algoritma antara lain 1) Algoritma berisi deskripsi langkah-langkah penyelesaian masalah, 2) Setiap algoritma ditulis dalam bentuk *pseudocode*, 3) *Pseudocode* yang dibuat mempunyai kemiripan dengan bahasa pemrograman umum, 4) *Pseudocode* dalam bentuk notasi algoritmik harus diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih, 4) Algoritma yang dibuat diterjemahkan kedalam notasi bahasa pemrograman seperti pendeklarasian variabel, pemilihan tipedata, aturan sintaks, dan lainnya [Ramadhani, 2015].

Algoritma pengurutan (sorting) meletakkan elemen data kedalam kumpulan data urutan tertentu, proses pengurutan yang sebelumnya data disusun acak menjadi tersusun teratur menurut aturan tertentu [Saputro and Khasanah, 2018]. Pengurutan merupakan hal yang tidak bisa dipisahkan dari dunia komputer. Sekarang ini *Google* dikenal sebagai mesin pencari terbesar di dunia. Dalam hitungan detik dapat diperoleh informasi yang diinginkan. Adanya kebutuhan terhadap proses pengurutan memunculkan bermacam-macam metode pengurutan yang bertujuan untuk memperoleh metode pengurutan yang optimal [Sitorus and Sembiring, 2012].

Pengurutan data *(sorting)* didefinisikan sebagai proses untuk menyusun kembali himpunan objek dengan menggunakan aturan tertentu. Tujuannya adalah untuk mendapatkan kemudahan dalam pencarian dari suatu himpunan. Kelebihan suatu data yang terurut adalah mudah untuk dicek apabila ada data yang hilang.

Larik *(Array)*

Array adalah kumpulan data untuk menyimpan item bertipe data sama, biasanya pada pengurutan data adalah data dengan tipe sama. Setiap data disimpan dalam alamat memori yang berbeda yang disebut elemen *array* . Komponen-komponen dari *array* antara lain nama *array,* nilai *array,* indeks *array,* jenis *array* [Situmorang, 2016]. Adapun proses yang dapat dilakukan dalam sebuah *a*rray antara lain memasukkan elemen data, mencari elemen data, menghapus lemen data, mencetak atau menampilkan hasil dari proses pengolahan data, menyisipkan elemen data, mencari posisi elemen data tertentu dan proses mengurutkan elemen data [Situmorang, 2016].

Pengurutan Data *(Sorting)*

Data terkadang berada dalam bentuk yang tidak berpola ataupun dengan pola tertentu yang diinginkan. Tidak ada algoritma terbaik untuk semua keadaan, kadang kala sebuah algortitma sangat efisien ketika jumlah datanya sedikit, namun kinerjanya menjadi berkurang ketika jumlah data ditambahkan atau meningkat. Meskipun memiliki kemampuan komputasi yang lebih tinggi, namun jika menggunakan algoritma yang kurang efisien, maka akan membutuhkan waktu lebih lama. Sehingga untuk memecahkan permasalahan diperlukan sebuah algoritma yang efektif dan efisien agar persoalan komputasi serta terbatasnya alokasi memori dapat diatasi.

Secara umum ada dua jenis pengurutan data yaitu [Ramadhani, 2015]: 1) Model urut naik *(ascending)* yang mengurutkan data dari yang mempunyai nilai terkecil sampai terbesar. 2) Model urut turun *(descending)* yang mengurutkan data dari yang mempunyai nilai terbesar sampai terkecil. Jika N buah data disimpan didalam sebuah *array* Nilai, maka pengurutan *ascending* berarti menyusun elemen *array* sedemikian hingga:

$$NILAI[0] \leq NILAI[1] \leq NILAI[2] \leq ... \leq NILAI[N-1] \quad (1)$$

Sumber: Sitorus and Sembiring (2012)

Sedangkan pengurutan *descending* berarti menyusun elemen *array* sedemikian hingga:

$$NILAI[0] \geq NILAI[1] \geq NILAI[2] \geq ... \geq NILAI[N-1] \quad (2)$$

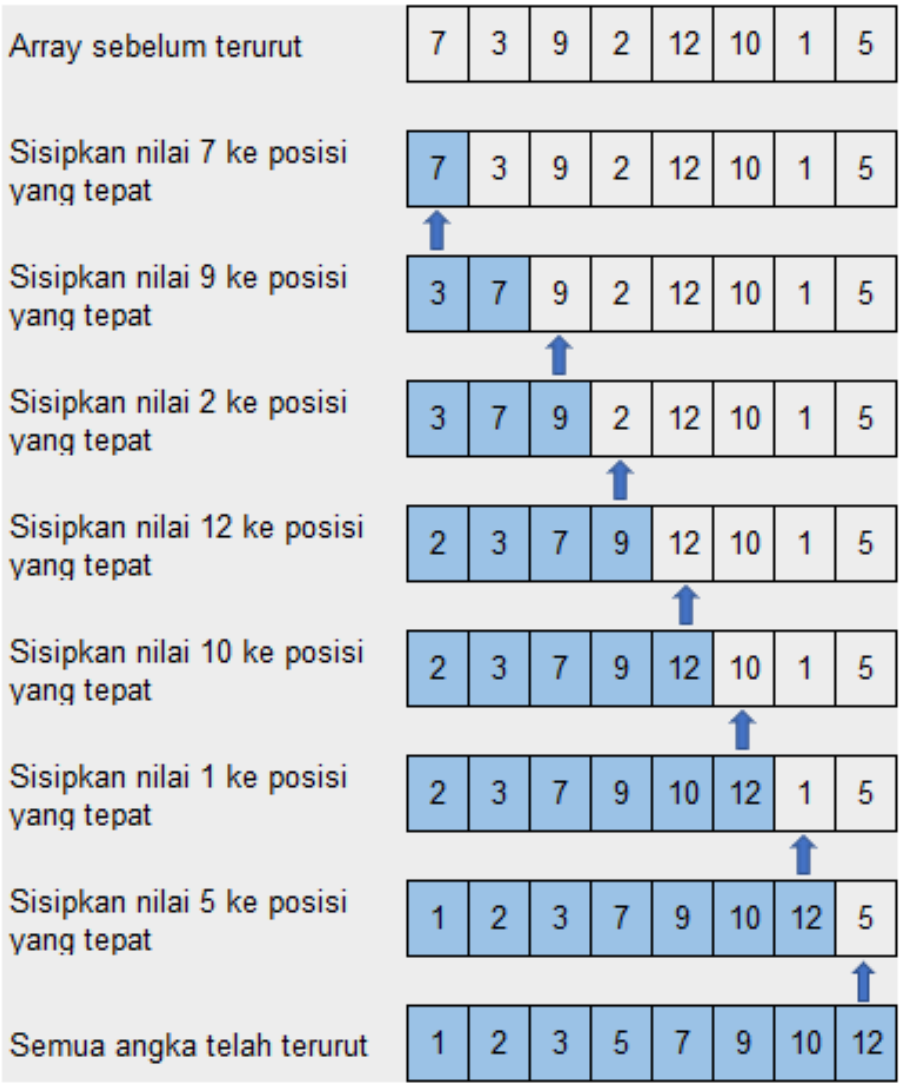
Sumber: Sitorus and Sembiring (2012)

Algoritma pengurutan data yang sering ditemukan dalam literatur komputer antara lain *bubble sort, selection sort, insertion sort, heap sort, shell sort, quick sort, merge sort, radix sort,*

dan tree sort [Munir, 2011]. Semua algoritma pengurutan selalu melakukan operasi perbandingan data untuk menemukan posisi urutan yang tepat. Berdasarkan tempat penyimpanan data, *sorting* dibedakan antara *external sort* dan *internal sort* [Sjukani, 2012]. *External sort* bila datanya berada dalam media *external*, atau *external storage* seperti *hardisk*. *Internal sort* bila datanya ada dalam *internal storage* atau *memory* komputer.

Algoritma *Insertion Sort*

Algoritma *insertion sort,* adalah metode pengurutan dengan cara menyisipkan elemen data pada posisi yang tepat. Pencarian posisi yang tepat dilakukan dengan melakukan pencarian berurutan didalam barisan elemen, selama pencarian posisi yang tepat dilakukan pergeseran elemen [Sitorus and Sembiring, 2012]. Pengurutan *insertion sort* sangat mirip dengan konsep permainan kartu, bahwa setiap kartu disisipkan secara berurutan dari kiri ke kanan sesuai dengan besar nilai kartu tersebut, dengan syarat apabila sebuah kartu disisipkan pada posisi tertentu kartu yang lain akan bergeser maju atau mundur sesuai dengan besaran nilai yang dimiliki [Ramadhani, 2015]. Proses pengurutan data dalam sebuah a*rray* menggunakan algoritma *insertion sort* tersebut dapat dilihat pada gambar 1, berikut:



Sumber: Ramadhani (2015).

Gambar 1. *Insertion Sort*

Pada gambar 1, terlihat pergeseran *array* dilakukan dari *i=1* yang kemudian dibandingkan dengan *array* yang berada disebelah kiri. Apabila *array* kedua lebih kecil dari *array* pertama, akan dilakukan penukaran. Selanjutnya dengan *looping* secara perlahan indeks dari *array*

E-ISSN: 2548-3331; 95 – 106

sebelah kanan dibandingkan lagi dengan *array* sebelah kiri yang sudah tersusun rapi sehingga apabila ditemukan nilai sebelah kanan lebih kecil dari nilai sebelah kiri yang sudah tersusun rapi, pergeseran dan penukaran akan dilakukan.

Algoritma *Selection Sort*

Algoritma *selection sort* sering juga disebut dengan metode maksimum atau minimum. Metode maksimum karena didasarkan pada pemilihan data atau elemen maksimum sebagai dasar pengurutan. Konsepnya dengan memilih elemen maksimum kemudian mempertukarkan elemen maksimum tersebut dengan elemen paling akhir untuk urutan *ascending* dan elemen pertama untuk *descending.*

Algoritma *selection sort* disebut juga dengan metode minimum karena didasarkan pada pemilihan elemen minimum sebagai dasar pengurutan. Konsepnya dengan memilih elemen minimum kemudian mempertukarkan elemen minimum dengan elemen paling akhir untuk urutan *ascending* dan elemen pertama untuk urutan *descending.* Proses yang dilakukan oleh algoritma *selection sort* adalah mengambil nilai terbesar dari susunan data dan menggantikannya dengan data yang paling kanan [Ramadhani, 2015].

Proses pengurutan data dalam sebuah a*rray* menggunakan algoritma *selection sort* tersebut dapat dilihat pada gambar 2, berikut:

Sumber: Ramadhani (2015)

Gambar 2. Metode *Selection Sort* (*Ascending)*

Pada gambar 2, terlihat proses yang dilakukan oleh algoritma *selection sort* adalah mengambil nilai terbesar dari susuna *array* dan menggantikannya dengan *array* yang paling kanan. Algoritma *selection sort* selalu menganggap nilai pada index pertama sebagai nilai maksimum *(max=0)*, kemudian membandingkan nilai dari indeks pertama tersebut *(array[max])* dengan nilai-nilai pada indeks berikutnya. Apabila nilai dari indeks yang dibandingkan lebih tinggi dari nilai indeks pertama, nilai tersebut diambil sebagai nilai *max (max=i).* Pembandingan tersebut dilakukan sebanyak ukuran *array* atau sejumlah elemen data yang akan diurutkan.

Visual Basic.*Net* merupakan *core* dari pembuatan aplikasi berbasis *.Net,* model untuk *development* dimana *platform* dan aplikasi bisa dibuat dan dijalankan tanpa bergantung pada alat (*device*) yang dipakai [Bernardo et al., 2015]. Pada penelitian ini Visual Basic *.Net* digunakan sebagai antar muka dalam perbandingan algoritma *insertion sort* dan *selection sort.*

2. Metode Penelitian

Perbandingan pengurutan data *(sorting)* dalam sebuah *array (L)* menggunakan dua algoritma dengan prinsip kerja yang berbeda. Metode *selection sort* menggunakan prinsip pertukaran elemen dalam proses *sorting*, sedangkan metode *insertion sort* menggunakan prinsip geser dan sisip elemen dalam proses *sorting* [Munir, 2011].

Metode *insertion sort,* secara *ascending* urutan langkah secara garis besar untuk setiap *pass i=2,…, n* yaitu *y* ← *L[i]*, sisipkan y pada tempat yang sesuai diantara *L[1] … L[i].* Selanjutnya langkah rincian pada setiap *Pass* yaitu a) ***Pass2,*** Elemen *y = L[2]* harus cari tempat yang tepat di dalam L[1..2] dengan cara menggeser elemen L[1..1] ke kanan (atau ke bawah) bila L[1..1] lebih besar daripada L[2]. b) ***Pass3,*** Elemen *y = L[3]* harus cari tempat yang tepat di dalam L[1..3] dengan cara menggeser elemen L[1..2] ke kanan (atau ke bawah) bila L[1..2] lebih besar daripada L[3]. c) Seterusnya sampai dengan ***Pass n***, Elemen *y = L[n]* harus cari tempat yang tepat di dalam L[1..n] dengan cara menggeser elemen L[1..n-1] ke kanan (atau ke bawah) bila L[1..n-1] lebih besar daripada L[n].

Metode *selection sort,* secara *ascending* urutan langkah secara garis besar yaitu *JumlahPass=n-1*, untuk setiap *Pass i=1,2,..,JumlahPass* lakukan pencarian elemen, pertukaran *maks* dengan elemen *ke-n,* mengurangi nilai *n* satu. Selanjutnya langkah rincian pada setiap *Pass* yaitu a) ***Pass1,*** cari elemen maksimum di dalam *L[1..n],* pertukarkan elemen maksimum dengan elemen *L[n]*, ukuran larik yang belum terurut = *n-1. b*) ***Pass2,*** cari elemen maksimum di dalam *L[1..n-1],* pertukarkan elemen maksimum dengan elemen *L[n-1]*, ukuran larik yang belum terurut = *n-16.* c) ***Pass3,*** cari elemen maksimum di dalam *L[1..n-2],* pertukarkan elemen maksimum dengan elemen *L[n-2]*, ukuran larik yang belum terurut = *n-3.* d) Seterusnya sampai dengan ***Pass n-1,*** cari elemen maksimum di dalam *L[1..2],* pertukarkan elemen maksimum dengan elemen *L[2]*, ukuran larik yang belum terurut = *1.* e) Setelah *Pass n-1*, elemen yang tersisa adalah *L[1],* tidak perlu diurutkan lagi karena hanya satu-satunya.

3. Hasil dan Pembahasan

Perbandingan pengurutan data *(sorting)* dalam penelitian dilakukan dengan mementukan jangkauan *(range)* data yang akan diuji adalah sejumlah sepuluh elemen data, lima puluh elemen data dan lima ratus elemen data.

3.1. Pseudocode Sorting

Pseudocode mempunyai hubungan dengan penulisan bahasa pemrogrman sehingga proses penerjemahan lebih mudah ke kode program yang digunakan. Tidak ada aturan baku dalam membuat *pseudocode.* Pada setiap satu kali langkah *n* disebut juga dengan satu kali *pass n,* pengurutan terdapat proses mencari elemen data maksimum (*ascending)* atau elemen data minimum (*descending)* dan proses pertukaran dua buah elemen data dalam *array.*

Pseudocode Insertion Sort

Algoritma *insertion sort* proses mencari elemen data maksimum (*ascending)* selengkapnya seperti pada gambar 3 berikut:

E-ISSN: 2548-3331; 95 – 106

```
Procedure InsertionSort1 (input/output L : LarikInt, input n:integer)
L[1] ≤ L[2] ≤ … ≤ L[n]

DEKLARASI
i,j,y  : integer

ALGORITMA
for i ← 2 to n do
  y ← L[i]
  j ← i-1
  while (j ≥ 1)and (not ketemu) do
    if y ‹L[j] then
       L[j+1] ← L[j]
       j ← j-1
    else
       ketemu ← true
    endif
  endwhile
  {j‹1 or ketemu}
   L[j+1] ← y
endfor
```

Sumber: Hasil Penelitian (2018)

Gambar 3 *Pseudocode Insertion Sort (Ascending)*

Pseudocode Selection Sort

Algoritma *selection sort* proses mencari elemen data maksimum (*ascending)* selengkapnya seperti pada gambar 4 berikut:

```
Procedure SelectionSort1 (input/output L : LarikInt, input n:integer)
L[1] ≤ L[2] ≤ … ≤ L[n]

DEKLARASI
i,j,imaks,maks,temp : integer

ALGORITMA
for i ← n downto 2 do
  imaks ← 1
  maks ← L[1]

  for j ← 2 to i do
       if L[j] > L[maks] then
           imaks ← j
           maks ← L[j]
       endif
  endfor

  temp ← L[i]
  L[i] ← L[maks]
  L[imaks] ← temp
endfor
```

Sumber: Hasil Penelitian (2018)

Gambar 4. *Pseudocode Selection Sort (Ascending)*

3.2 Pengurutan Data *(Sorting)* Menggunakan *Visual Basic .Net*

Visual Basic *.Net* merupakan pemrograman berorientasi objek *(OOP)* yang dapat digunakan untuk merancang antar muka dalam perbandingan algoritma. *OOP* bermaksud untuk memecahkan masalah pemrograman dengan pola pikir manusia. Program yang dibuat didesain dengan membuat prosedur-prosedur yang sederhana. Pada *OOP* terdapat objek yang memiliki informasi dari sebuah atribut data yang berada dalam sebuah *class.* Berikut adalah *class – class* dalam pengurutan data.

```
Public Class insertion
Dim stopwatch As New Diagnostics.Stopwatch
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
Me.Close()
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Me.stopwatch.reset()
        If data.Text = "" Then
            MsgBox("Tolong Input Jumlah Data Terlebih Dahulu")
            data.Focus()
        Else
            Me.stopwatch.Start()
               hasil.Text = "Data : "
            hasil2.Text = "Data : "
            Dim input As Integer
            input = (data.Text)
            Dim inputLength As Integer
            inputLength = Int(input)
            Dim arr(inputLength) As Integer
            Randomize()
            If data.Text >= 100 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((999 * Rnd()) + 1))
                Next
            ElseIf data.Text >= 1000 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((9999 * Rnd()) + 1))
                Next
               ElseIf data.Text >= 10000 Then
                For index As Integer = 0 To inputLength - 1
               arr(index) = CInt(Int((99999 * Rnd()) + 1))
                Next
            ElseIf data.Text >= 100000 Then
                For index As Integer = 0 To inputLength - 1 '
                    arr(index) = CInt(Int((999999 * Rnd()) + 1))
                Next
            ElseIf data.Text >= 1000000 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((1300000 * Rnd()) + 1))
                Next
```

Sumber: Hasil Penelitian (2018)

Gambar 5 (a) . Algoritma *Insertion Sort* Menggunakan Visual Basic *.Net*

Gambar 5 (a) dan gambar 5 (b) merupakan kesatuan penulisan algoritma metode *insertion sort* kedalam *class* pada menggunakan bahasa pemrograman Visual Basic *.Net.*

```
            ElseIf data.Text < 100 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((99 * Rnd()) + 1))
                Next
            Else
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((99999 * Rnd()) + 1))
                Next
               End If
               If hasil.Text.Trim <> Nothing Then
                For index2 As Integer = 0 To inputLength - 1
                    hasil.Text &= (arr(index2) & "   ")
                Next
                    Dim i, j As Integer
                For i = 1 To inputLength - 1 Step 1
                    Dim pick_item As Integer = arr(i)
                    Dim inserted As Integer = 0
                    j = i - 1
                    While (j >= 0 And inserted <> 1)
                        If (pick_item < arr(j)) Then
                            arr(j + 1) = arr(j)
                            j -= 1
                            arr(j + 1) = pick_item
                        Else : inserted = 1
                               End If
                    End While
                Next
                For i = 0 To inputLength - 1
                    hasil2.Text &= (arr(i) & "  ")
                Next
                Me.stopwatch.Stop()
            End If
            data.Clear()
            data.Focus()
        End If
       waktu.Text = Me.stopwatch.Elapsed.Seconds.ToString("00") & "," &
       Me.stopwatch.Elapsed.Milliseconds.ToString("000")
    End Sub
End Class
```

Sumber: Hasil Penelitian (2018)

Gambar 5 (b) . Algoritma *Insertion Sort* Menggunakan Visual Basic *.Net*

E-ISSN: 2548-3331; 95 – 106

```
Public Class selection
Dim stopwatch As New Diagnostics.Stopwatch
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
Me.Close()
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Me.stopwatch.Reset()
        If data.Text = "" Then
            MsgBox("Tolong Input Jumlah Data Terlebih Dahulu")
            data.Focus()
        Else
            Me.stopwatch.Start()
            hasil.Text = "Data : "
            hasil2.Text = "Data : "
            Dim t, j As Integer
            Dim input As Integer
            input = (data.Text)
            Dim inputLength As Integer
            inputLength = Int(input)
            Dim arr(inputLength) As Integer Randomize()
            If data.Text >= 100 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((999 * Rnd()) + 1))
                Next
            ElseIf data.Text >= 1000 Then
                For index As Integer = 0 To inputLength - 1
               arr(index) = CInt(Int((9999 * Rnd()) + 1))
                Next
           ElseIf data.Text >= 10000 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((99999 * Rnd()) + 1))
                Next
           ElseIf data.Text >= 100000 Then
                For index As Integer = 0 To inputLength - 1
               arr(index) = CInt(Int((999999 * Rnd()) + 1))
                Next
            End If
```

Sumber: Hasil Penelitian (2018)

Gambar 6 (a). Algoritma *Selection Sort* Menggunakan *Visual Basic .Net*

Gambar 6 (a) dan gambar 6 (b) merupakan kesatuan penulisan algoritma metode *selection sort* kedalam *class* pada menggunakan bahasa pemrograman Visual Basic *.Net.*

```
            ElseIf data.Text >= 1000000 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((1300000 * Rnd()) + 1))
                Next
            ElseIf data.Text < 100 Then
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((99 * Rnd()) + 1))
                Next
            Else
                For index As Integer = 0 To inputLength - 1
                    arr(index) = CInt(Int((99999 * Rnd()) + 1))
                Next
                    If hasil.Text.Trim <> Nothing Then
                For index2 As Integer = 0 To inputLength - 1
                    hasil.Text &= (arr(index2) & "    ")
                Next
                For i = 0 To inputLength - 2
                    For j = i + 1 To inputLength - 1
                        If arr(i) > arr(j) Then
                            t = arr(i)
                            arr(i) = arr(j)
                            arr(j) = t
                        End If
                    Next
                Next
                For i = 0 To inputLength - 1
                    hasil2.Text &= (arr(i) & "  ")
                Next
            End If
            data.Clear()
            data.Focus()
            Me.stopwatch.Stop()
        End If
       waktu.Text = Me.stopwatch.Elapsed.Seconds.ToString("00") & "," &
       Me.stopwatch.Elapsed.Milliseconds.ToString("000")
    End Sub
End Class
```

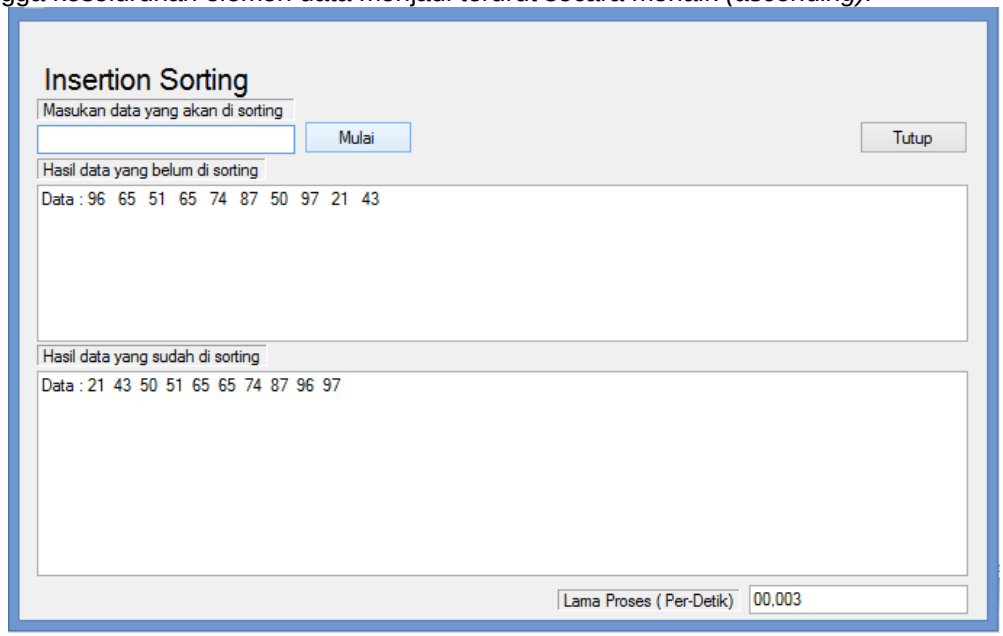
Sumber: Hasil Penelitian (2018)

Gambar 6 (b). Algoritma *Selection Sort* Menggunakan *Visual Basic .Net*

3.3. Hasil Uji Coba Elemen Data Acak

Pada penelitian ini dilakukan pengujian terhadap *algoritma insertion* sort dan *selection sort* dengan memasukkan data yang akan *disorting* ke dalam *text box* sejumlah elemen data. Misalnya, sejumlah 10 elemen data, kemudian klik mulai maka akan ditampilkan data acak pada *text box* hasil data yang belum di sorting data pada *text box* hasil data yang sudah di sorting. Pada Pengujian juga ditampilkan pada *text box* waktu yang diperlukan oleh algoritma dalam mengurutkan elemen data (proses *sorting*).

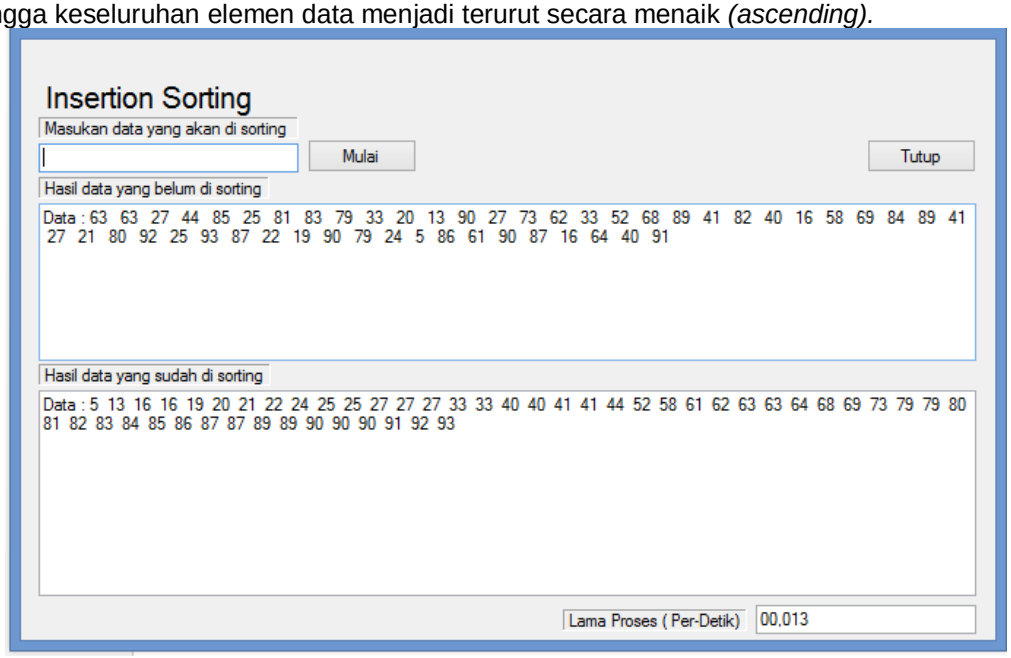
Gambar 7 berikut menunjukkan pengujian menggunakan algoritma *insertion sort*, elemen yang diuji sejumlah sepuluh data secara acak, menghasilkan waktu 00.003 detik sehingga keseluruhan elemen data menjadi terurut secara menaik *(ascending).*



Sumber: Hasil Penelitian (2018)

Gambar 7. Uji Coba Sepuluh Data Menggunakan *Insertion Sort*

Gambar 8 berikut menunjukkan pengujian menggunakan algoritma *insertion sort*, elemen yang diuji sejumlah sepuluh data secara acak, menghasilkan waktu 00.013 detik sehingga keseluruhan elemen data menjadi terurut secara menaik *(ascending).*

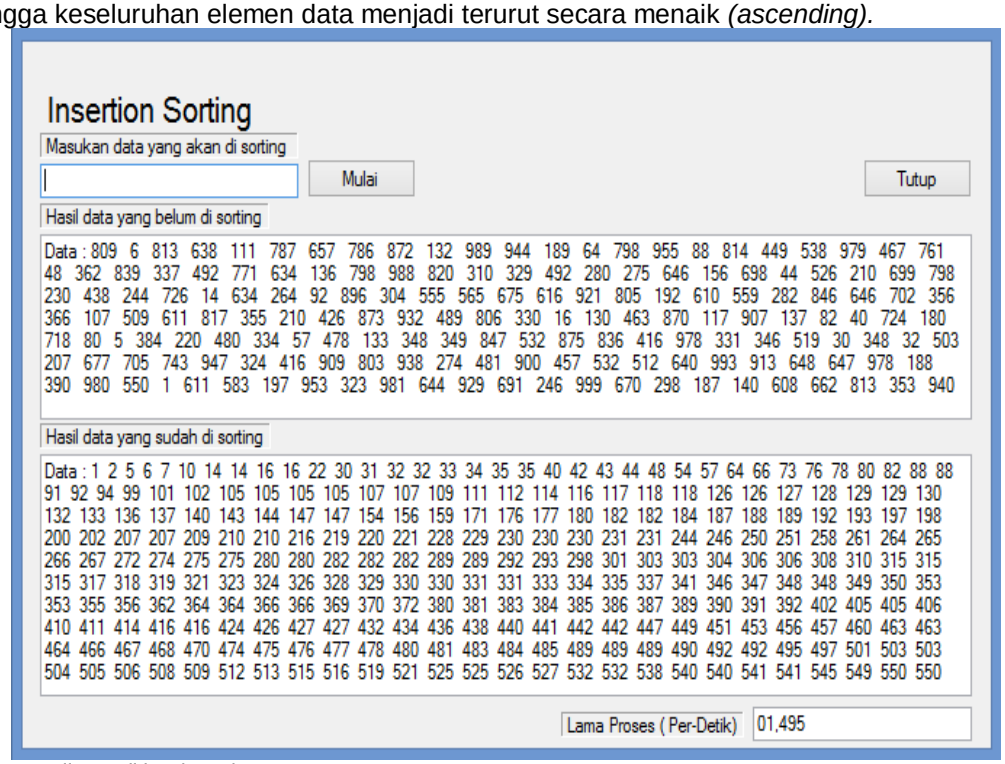


Sumber: Hasil Penelitian (2018)

Gambar 8. Uji Coba Lima Puluh Data Menggunakan *Insertion Sort*

E-ISSN: 2548-3331; 95 – 106

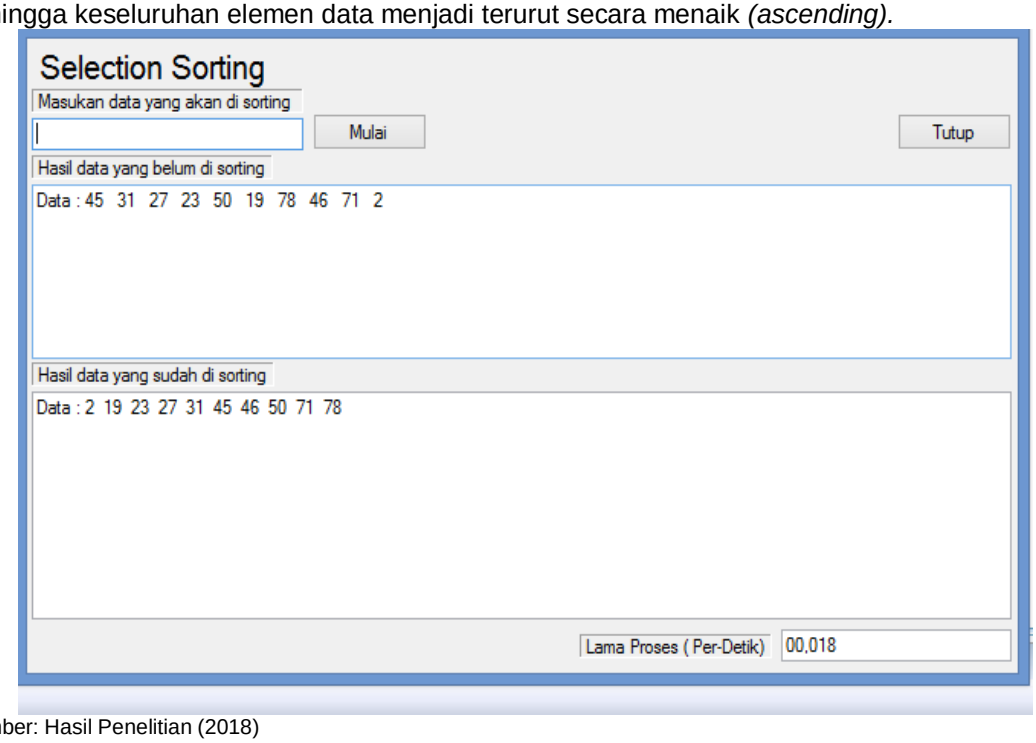
Gambar 9 berikut menunjukkan pengujian menggunakan algoritma *insertion sort*, elemen yang diuji sejumlah sepuluh data secara acak, menghasilkan waktu 01.495 detik sehingga keseluruhan elemen data menjadi terurut secara menaik *(ascending).*



Sumber: Hasil Penelitian (2018)

Gambar 9. Uji Coba Lima Ratus Data Menggunakan *Insertion Sort*

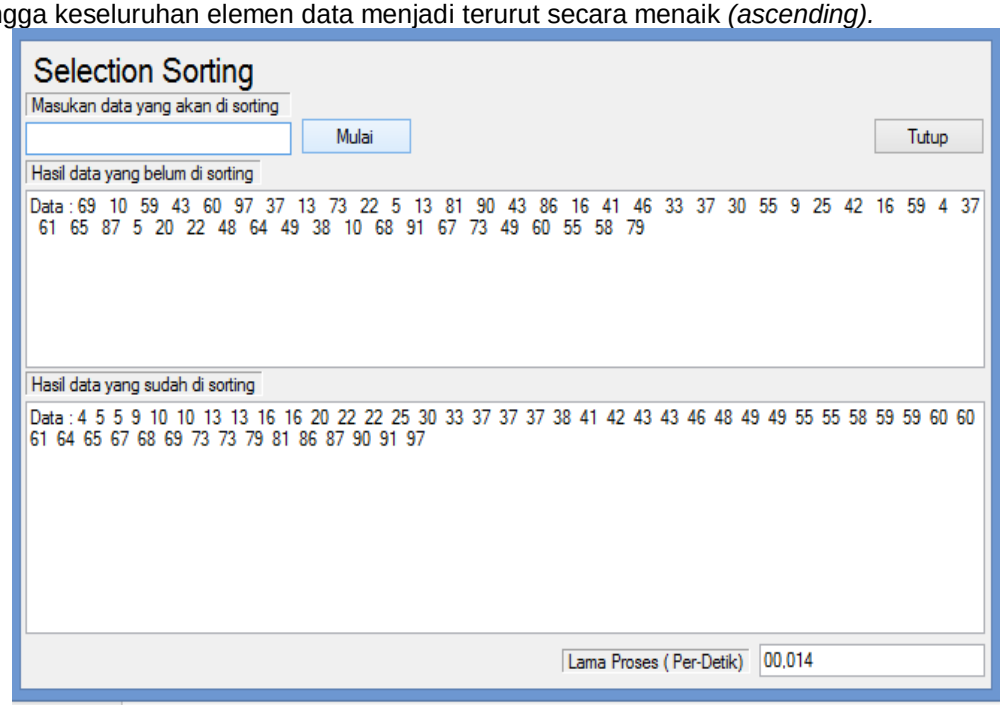
Gambar10 berikut menunjukkan pengujian menggunakan algoritma *selection sort*, elemen yang diuji sejumlah sepuluh data secara acak, menghasilkan waktu 00.018 detik sehingga keseluruhan elemen data menjadi terurut secara menaik *(ascending).*



Sumber: Hasil Penelitian (2018)

Gambar 10. Uji Coba Sepuluh Data Menggunakan *Selection Sort*

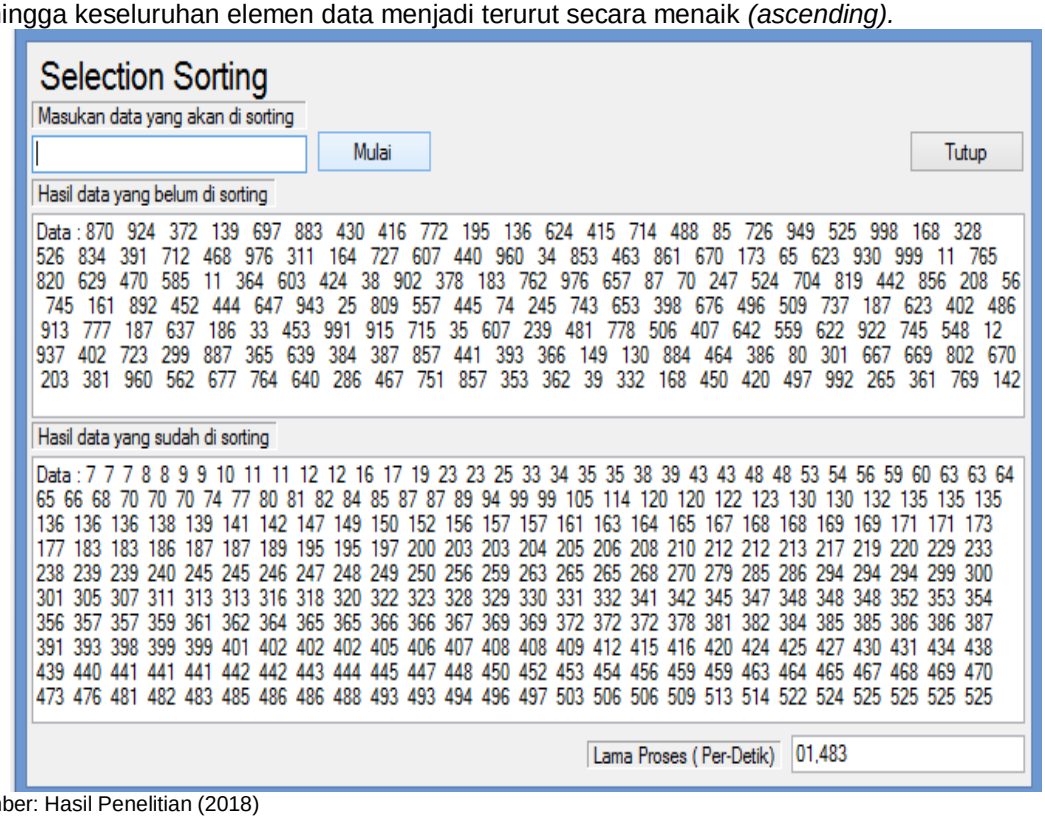
Gambar 11 berikut menunjukkan pengujian menggunakan algoritma *selection sort*, elemen yang diuji sejumlah lima puluh data secara acak, menghasilkan waktu 00.014 detik sehingga keseluruhan elemen data menjadi terurut secara menaik *(ascending).*



Sumber: Hasil Penelitian (2018)

Gambar 11. Uji Coba Lima Puluh Data Menggunakan *Selection Sort*

Gambar 12 berikut menunjukkan pengujian menggunakan algoritma *selection sort*, elemen yang diuji sejumlah lima ratus data secara acak, menghasilkan waktu 01.483 detik sehingga keseluruhan elemen data menjadi terurut secara menaik *(ascending).*



Sumber: Hasil Penelitian (2018)

Gambar 12. Uji Coba Lima Ratus Data Menggunakan *Selection Sort*

E-ISSN: 2548-3331; 95 – 106

Tabel 1 menunjukkan hasil keseluruhan pengujian terhadap elemen data dan waktu yang diperlukan oleh algoritma *insertion sort* dan *selection sort* dalam melakukan *sorting.*

Tabel 1. Waktu Yang Dihasilkan Pada Pengujian Algoritma

Algoritma	10 Elemen Data	50 Elemen Data	500 Elemen Data
Insertion Sort	00.003 detik	00.013 detik	01.495 detik
Selection Sort	00.018 detik	00.014 detik	01.483 detik

Sumber: Hasil Penelitian (2018)

4. Kesimpulan

Dengan algoritma yang baik dapat dihasilkan sebuah program yang efisien dari segi waktu dan hasil yang dicapai. Sebuah algoritma sangat efisien ketika jumlah datanya sedikit, namun kinerjanya menjadi berkurang ketika jumlah data ditambahkan atau meningkat. Hasil dari penelitian adalah metode *insertion sort* lebih unggul pada jumlah data yang sedikit yaitu pada sejumlah sepuluh elemen data membutuhkan waktu 00.003 detik, lima puluh elemen data membutuhkan waktu 00.013 detik, sedangkan metode *selection sort* lebih unggul pada jumlah data yang lebih banyak yaitu pada sejumlah lima ratus elemen data membutuhkan waktu 01.083 detik.

Referensi

Bernardo, Mesterjon, Zulita LN. 2015. Implementasi Metode Selection Sort Untuk Menentukan Nilai Prestasi Siswa Kelas 3 dan Kelas 4 SD Negeri 107 Selumba. Jurnal Media Infotama 11(1): 91–100.

Munir R. 2011. Algoritma & Pemrograman Dalam Bahasa Pascal dan C. Bandung: Informatika Bandung.

Ramadhani. 2015. Dasar Algoritma dan Struktur Data dengan Bahasa JAVA. Yogyakarta: CV Andi Offset.

Saputro FE, Khasanah FN. 2018. Teknik Selection Sort dan Bubble Sort Menggunakan Borland C ++. Jurnal Mahasiswa Bina Insani. 2 (2): 136–145.

Sitorus L, Sembiring DJ. 2012. Konsep dan Implementasi Struktur Data dengan C++. Yogyakarta: CV Andi Offset.

Situmorang H. 2016. Simulasi Pengurutan Data Dengan Algoritma Heap Sort. Jurnal Mahajana Informasi 1 (2): 20–30.

Sjukani M. 2012. Struktur Data (Algoritma dan Struktur Data 2), Edisi 5. Jakarta: Mitra Wacana Media.