ANALISIS DAN PERBANDINGAN KOMPLEKSITAS ALGORITMA EXCHANGE SORT DAN INSERTION SORT UNTUK PENGURUTAN DATA MENGGUNAKAN PYTHON

Dwipo Setyantoro, Rika Astuti Hasibuan

dwipo.setyantoro@gmail.com, rika.astuti93@gmail.com

Abstrak

Ada banyak algoritma yang dapat digunakan untuk mengurutkan data dan masing-masing mempunyai karakteristik yang berbeda-beda. Ada algoritma yang mempunyai running time yang konstan dengan berbagai urutan data (seperti Insertion Sort, Selection Sort), dan ada yang mempunyai runnng time yang sangat singkat jika data yang akan diurutkan sudah urut terlebih dahulu, seperti Exchange Sort. Penelitian ini membahas tentang implementasi dan analisis proses pengurutan data menggunakan dua algoritma yang berbeda, yaitu Insertion Sort dan Exchange/Bubble Sort. Pada tahap pertama, kedua algoritma tersebut diimplementasikan dalam bahasa pemrograman Python untuk mengurutkan sejumlah angka yang sudah ditentukan. Kemudian, program dimodifikasi agar dapat membangkitkan data sejumlah yang diperlukan yaitu sejumlah 1000, 100.000 dan 500.000 data secara acak. Kedua algoritma tersebut mengurutkan data tersebut dan hasilnya kemudian dibandingkan. Dari eksperimen yang sudah dilakukan, algoritma Insertion sort memiliki jumlah iterasi yang sama dalam setiap kondisi, sedangkan kasus terbaik dari Exchange Sort adalah hanya diperlukan satu kali iterasi jika data sudah terurut menaik.

Keyword: ***Algoritma; Insertion Sort; Exchange Sort; Phyton***

1. Pendahuluan

Sorting atau pengurutan adalah proses menyusun elemen – elemen dari masukan awal acak menjadi keluaran akhir tertata dengan urutan tertentu. Kegiatan pengurutan data dapat ditemukan pada berbagai masalah dan aplikasi. Misalkan pada kegiatan mengurutkan nomor pasien pada sebuah klinik. Agar antrian dapat berjalan dengan baik, nomor urut pasien harus diurutkan terlebih dahulu oleh petugas pendaftaran. Aplikasi pelaporan transaksi umumnya menampilkan data yang telah terurut berdasarkan tanggal transaksi atau kode field tertentu. Fungsi-fungsi statistik seperti median dan pembuatan kuartil data (*quarter*) dan percentil (*percentile*) mensyaratkan data untuk diurutkan terlebih dahulu. Pada algoritma *binary search* untuk pencarian data pun mensyaratkan data telah terurut sebelum dilakukan proses pencarian data.

Beberapa macam algoritma sorting telah dibuat karena proses tersebut sangat mendasar dan sering digunakan. Oleh karena itu, pemahaman atas algoritma – algoritma yang ada sangatlah berguna. Selain menjadi suatu aplikasi yang berdiri sendiri, pengurutan juga biasanya menjadi suatu bagian dari algoritma yang lebih besar.

Permasalahan pengurutan (sorting problem) secara formal didefinisikan sebagai berikut :

Input: Suatu urutan dari n bilangan,

Output: Suatu permutasi atau penyusunan kembali dari input sedemikian rupa sehingga pada tata urutan *ascending* (dari nilai kecil ke besar) atau pada tata urutan *descending* (dari nilai besar ke kecil).

Sebagai contoh jika diberikan masukan lima bilangan acak maka keluarannya adalah sebagaimana berikut ini:

Input :	6	2	1	4	7	
Output :	1	2	4	6	7	(*ascending*)
	7	6	4	2	1	(*descending*)

Ada banyak metode pengurutan antara lain: bubble sort, bi-directional bubble sort, selection sort, shaker sort, insertion sort, in- place merge sort, double storage merge sort, comb sort 11, shell sort, heap sort, exchange sort, merge sort, quick sort, quick sort with bubblesort, enhance quick sort, fast quick sort, radix sort algorithm, swap sort, dan lain sebagainya.

Untuk membatasi luasnya pembahasan, maka dalam penelitian ini hanya akan dibahas 2(dua) metode, yaitu Insertion Sort dan Exchange Sort. Pembahasan untuk tiap metode akan difokuskan pada cara kerja pengurutan beserta contohnya, analisa algoritma untuk kondisi terburuk (*worst case*), rata-rata (*average case*), terbaik (*best case*), implementasinya dalam bahasa Python serta pengujian waktu eksekusi untuk kedua metode tersebut.

Tujuan yang ingin dicapai dari penelitian ini adalah :

- Melakukan implementasi algoritma pengurutan Insertion Sort dan Exchange Sort dengan bahasa pemrograman Python
- Menganalisa dan membandingkan kinerja algoritma Insertion Sort dan Exchange Sort dalam proses pengurutan data.

2. Algoritma Pengurutan Insertion dan Exchange

Insertion Sort

Salah satu algoritma sorting yang paling sederhana adalah insertion sort. Insertion Sort disebut-sebut sebagai metode pertengahan. Artinya, metode ini memiliki kecepatan rata- rata antara metode primitif (bubble dan selection) dan modern (merge dan quick). Metode ini, didasarkan pada sebuah kunci yang diambil pada elemen ke-2 pada sebuah larik, lalu menyisipkan elemen tersebut jika kondisi percabangan terpenuhi. Metode penyisipan (insertion) bertujuan untuk menjadikan bagian sisi kiri larik terurutkan sampai dengan seluruh larik berhasil diurutkan.

Algoritma dan Pseudocode

Ide dari algoritma ini dapat dianalogikan seperti mengurutkan kartu.

Anggaplah bahwa terdapat sebuah meja yang berisi setumpuk kartu. Meja ini melambangkan kondisi larik sebelum diurutkan. Langkah-langkah pengurutan adalah sebagai berikut:

- Ambil kartu pertama dari meja, letakkan di tangan kiri. Misalkan : kartu 4 Keriting (*Club*)
- Ambil kartu kedua dari meja, misalkan kartu 5 Keriting. bandingkan dengan kartu yang berada di tangan kiri, kemudian letakkan pada urutan yang sesuai. Dalam hal ini, kartu dengan nomor terkecil akan diletakkan di bagian paling belakang. Karena 5 Keriting lebih besar, maka diletakkan di depan kartu 4 keriting.
- Lakukan hal yang sama pada kartu berikutnya hingga semua kartu sudah tersusun dengan urut pada tangan kiri.

Pseudocode untuk algoritma insertion sort adalah sebagai berikut:

```
Procedure InsertionSort(data[],n)
  for(i = 2; i < n; i++)
        j = i
        while ((data[j] < data[j-1])) and (j > 1)) do
              swap(data[j], data[j-1])
              j--
        endwhile
  endfor
endprocedure
```

Kompleksitas Algoritma

Kondisi terbaik (*best case*) tercapai jika data telah terurut. Hanya satu pertukaran dilakukan untuk setiap posisi i, sehingga terdapat n – 1 pertukaran, atau O (n). Sedangkan kasus terburuknya *(worst case)* tercapai jika data terurut terbalik dengan 1 + 2 + 3 + .. + n-1 pertukaran, atau O (n(n-1)/2) => O (n^2)

Exchange (Bubble) Sort

Algoritma bubble sort adalah salah satu algoritma pengurutan yang paling simple, baik dalam hal pengertian maupun penerapannya. 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 dan Pseudocode

Ide dari algoritma ini adalah menganut prinsip gelembung udara (bubble) di dalam air. Gelembung udara akan naik ke atas karena lebih ringan dibandingkan dengan air. Begitu pula dalam proses pengurutan dengan algoritma ini, bilangan yang "ringan" akan terbawa ke "pemukaan", seperti halnya dengan gelembung udara di dalam air.

Algoritma ini juga termasuk dalam golongan algoritma comparison sort, karena menggunakan pertukaran dalam operasi antar elemennya.

Pseudo code untuk algoritma exchange sort adalah sebagai berikut :

```
Procedure ExchangeSort (data[],n)
  Tukar = true
  while (Tukar) do
        Tukar = false
   for i = 2 to n do
         If (data[i] < data[i-1]) then
               Tukar = true
               Swap(data[i],data[i-1])
         endif
   endfor
  endwhile
endprocedure
```

Kompleksitas Algoritma

Kondisi terbaik (*best case*) tercapai jika data telah terurut. Tidak ada pertukaran yang dilakukan. Sedangkan kasus terburuknya *(worst case)* tercapai jika data terurut terbalik dengan 1 + 2 + 3 + .. + n-1 pertukaran, atau O (n(n-1)/2) => O (n^2)

3. Implementasi

Implementasi dilakukan dengan pemrograman Python untuk kedua algoritma tersebut dan tampilan ketika dijalankan adalah sebagai berikut:

Kode program Insertion Sort (file : TestInsertion.py)

```
# Aplikasi Pengujian Metode Insertion Sorting
# dengan 20 data
import time
data = [2,6,3,4,9,10,15,11,1,20,0,8,14,13,7,15,16,8,3,1]
# Procedure InsertionSort(data[],n)
#   for(i = 2; i < n; i++)
#         while ((data[i] < data[i-1])) and (i > 1)) do
#                   swap(data[i], data[i-1])
#                   i--

print 'Nilai awal data adalah : '
print data
print 'Proses pengurutan : '
mulai = time.time()
for i in range(1,20) :
  j = i
  while ((data[j] <data[j-1]) and (j > 0)) :
        tmp = data[j]
        data[j] = data[j-1]
        data[j-1] = tmp
        j = j - 1
  print 'Pass-',i,' ', data
akhir = time.time()
waktu = akhir - mulai
print 'Hasil akhir adalah : '
print  data
print 'Waktu mulai   = ',mulai
print 'Waktu selesai = ',akhir
print 'Lama proses   = ',waktu,' detik'
```

Program diatas dijalankan menggunakan Python dengan pemanggilan sebagai berikut :
C:\Python27>python TestInsertion.py

Dan menghasilkan tampilan sebagai berikut (dengan contoh 20 data bilangan) :

```
Nilai awal data adalah :
[2, 6, 3, 4, 9, 10, 15, 11, 1, 20, 0, 8, 14, 13, 7, 15, 16, 8, 3, 1]
Hasil akhir adalah :
[0, 1, 1, 2, 3, 3, 4, 6, 7, 8, 8, 9, 10, 11, 13, 14, 15, 15, 16, 20]
Waktu mulai   =  1395962362.59
Waktu selesai =  1395962362.59
Lama proses   =  0.00100016593933  detik
```

Hasil eksekusi program Test Insertion Sort diatas memerlukan 19 kali iterasi dengan waktu proses = 0.00100016593933 detik

Kode program Exchange Sort (file : TestExchange.py) :

```
# Aplikasi Pengujian Metode Exchange Sorting
# dengan 20 data
import time
data = [2,6,3,4,9,10,15,11,1,20,0,8,14,13,7,15,16,8,3,1]
# Procedure ExchangeSort (data[],n)
#   Tukar = true
#   while (Tukar) do
#     Tukar = false
#     for i = 2 to n do
#           If (data[i] < data[i-1]) then
#                 Tukar = true
#                 Swap(data[i],data[i-1])
#           endif
#    endfor
#   endwhile
# endprocedure

print 'Nilai awal data adalah : '
print data
print 'Proses pengurutan : '
mulai = time.time()
a = 0
tukar = True
while (tukar) :
  a += 1
  print 'Pass-',a,' : '
  tukar = False
  for i in range(1,19) :
```

```
    if (data[i] < data[i-1]) :
      tukar = True
         tmp = data[i]
         data[i] = data[i-1]
         data[i-1] = tmp
    print data
akhir = time.time()
waktu = akhir - mulai
print 'Hasil akhir adalah : '
print  data
print ‘Waktu mulai     = ‘, mulai
print ‘Waktu selesai   = ‘, akhir
print 'Lama proses     = ',waktu,' detik'
```

Program Test Exchange Sort dijalankan dengan pemanggilan :

C:\Python27>python TestInsertion.py
Yang akan menghasilkan output sebagai berikut :

Nilai awal data adalah :
[2, 6, 3, 4, 9, 10, 15, 11, 1, 20, 0, 8, 14, 13, 7, 15, 16, 8, 3, 1]
Hasil akhir adalah :
[0, 1, 1, 2, 3, 3, 4, 6, 7, 8, 8, 9, 10, 11, 13, 14, 15, 15, 16, 20]
Waktu mulai   = 1395969451.21
Waktu selesai = 1395969451.22
Lama proses   = 0.00600004196167 detik

Pada hasil eksekusi Exchange Sort dengan data yang sama digunakan pada Insertion Sort diperlukan 18 kali iterasi dengan waktu proses selama 0.00600004196167 detik

4. Pengujian

Tujuan pengujian ini adalah untuk mengetahui dan membandingkan kecepatan eksekusi perintah pengurutan data terhadap sekelompok data dengan rentang jumlah tertentu, baik pada algoritma insertion sort maupun exchange sort.

Perangkat

Pengujian dilaksanakan dengan menggunakan interpreter Python 2.7.6 [MSC v.1500 32 bit (Intel)] 32 dengan platform Windows 7 dan komputer *notebook* dengan spesifikasi:

a. Intel Intel® Core™ 2 Duo CPU T8100 @ 2.10 GHz
b. 1 GB DDR2-SDRAM.
c. 500 GB HDD.

Prosedur

Pengujian meliputi tahapan berikut ini:

a. Pengujian dengan data yang sama untuk kedua algoritma (sebanyak 20 data)

Dalam pengujian ini fungsi tampilan hasil pengurutan dimatikan

- Data acak yaitu : [2,6,3,4,9,10,15,11,1,20,0,8,14,13,7,15,16,8,3,1]

Hasil metode insertion :

```
C:\Python27>python testInsertion.py
Nilai awal data adalah :
[2, 6, 3, 4, 9, 10, 15, 11, 1, 20, 0, 8, 14, 13, 7, 15, 16, 8, 3, 1]
Hasil akhir adalah :
[0, 1, 1, 2, 3, 3, 4, 6, 7, 8, 8, 9, 10, 11, 13, 14, 15, 15, 16, 20]
Waktu mulai   =  1395972828.52
Waktu selesai =  1395972828.52
Lama proses   =  0.0  detik
```

Gambar 1. Hasil pengujian insertion sort (20 data acak)

Hasil metode Exchange :

```
C:\Python27>python testExchange.py
Nilai awal data adalah :
[2, 6, 3, 4, 9, 10, 15, 11, 1, 20, 0, 8, 14, 13, 7, 15, 16, 8, 3, 1]
Hasil akhir adalah :
[0, 1, 1, 2, 3, 3, 4, 6, 7, 8, 8, 9, 10, 11, 13, 14, 15, 15, 16, 20]
Waktu mulai   =  1395973089.39
Waktu selesai =  1395973089.39
Lama proses   =  0.0  detik
```

Gambar 2. Hasil pengujian exchange sort (20 data acak)

Karena terlalu singkatnya waktu proses, yang sebabkan sisembunyikannya kode penampilan data yang diurut, maka untuk 20 data yang ditentukan awal, baik secara acak, terurut menaik, dan terurut menurun. Menghasilkan lama proses 0.0 detik. Untuk itulah dilakukan uji coba untuk data random dengan meggunakan kode pembangkit data random.

b. Pengubahan Kode Sumber

Kode sumber diubah dengan menambahkan kode pembangkit data berupa bilangan bulat dalam rentang 1 sampai dengan 1000 secara acak

```
import random
Data = list()
for i in range(1,1000) :
   a = random.random()
   Data.append(round(a * 100))
```

Seluruh fungsi untuk menampilkan keluaran kondisi larik, baik saat sebelum maupun sesudah diurutkan, dimatikan.

c. Eksekusi Program

Program untuk pengujian insertion sort maupun merge sort kemudian dieksekusi. Penguji memasukkan angka yang menunjukkan jumlah data yang akan diurutkan. Hasil eksekusi program pengujian insertion sort:

- Ujicoba dengan 10000 data random :

```
C:\Python27>python InsertionRandom.py
Proses pengurutan InsertionSort dengan 10000 data random :
Waktu mulai   =  1395973752.53
Waktu selesai =  1395973752.53
Lama proses   =  0.000999927520752  detik
```

Gambar 3. Tampilan Pengujian Insertion Sort dengan 10.000 data

```
C:\Python27>python ExchangeRandom.py
Proses pengurutan dengan Exchange Sort pada 10000 data random :
Waktu mulai   =  1395974267.72
Waktu selesai =  1395974267.72
Lama proses   =  0.0  detik
```

Gambar 4. Tampilan Pengujian Exchange Sort dengan 10.000 data

- Ujicoba dengan 100000 data random :

```
C:\Python27>python InsertionRandom.py
Proses pengurutan InsertionSort dengan 100.000 data random :
Waktu mulai   =  1395974488.37
Waktu selesai =  1395974488.37
Lama proses   =  0.000999927520752  detik
```

Gambar 5. Tampilan pengujian Insertion Sort dengan 100.000 data

```
C:\Python27>python ExchangeRandom.py
Proses pengurutan dengan Exchange Sort pada 100.000 data random :
Waktu mulai   =  1395974401.55
Waktu selesai =  1395974401.55
Lama proses   =  0.0  detik
```

Gambar 6. Tampilan pengujian Exchage Sort dengan 100.000 data

d. Pencatatan Hasil

Data hasil pengujian selanjutnya dicatat dan dimasukkan ke dalam tabel.

Tabel 1. Tabel hasil perbandingan waktu eksekusi program

Metode Sorting	Jumlah Data	Lama Proses (detik) dengan pembulatan
Insertion	20	0.0

	1000	0.0009999
	100.000	0.0009999
Exchage	20	0.0
	1000	0.0
	100.000	0.0

5. Kesimpulan

Dari hasil implementasi maupun pengujian yang telah dilakukan dapat ditarik beberapa kesimpulan sebagaimana berikut ini:

1. Dari hasil pengujian diketahui bahwa algoritma exchange sort lebih cepat dibandingkan insertion sort untuk data yang lebih banyak, khususnya untuk jumlah data > 10000.
2. Kelemahan utama insertion sort adalah algoritma ini membutuhkan minimal passing/iterasi sebanyak n-1, sedangkan exchange sort dapat kurang dari n - 1
3. Pada kasus *best case*, algoritma exchange sort lebih unggul daripada insertion sort, sehubungan dengan kompleksitas yang lebih rendah yaitu nilai O(1) dibandingkan dengan O(n).
4. Pada kasus *worst case*, algoritma insertion sort lebih unggul terhadap exchange sort, sehubungan dengan kompleksitas yang lebih rendah yaitu nilai O (n), dibandingkan dengan O (n^2).
5. Algoritma insertion sort secara teknis lebih mudah diterapkan dibandingkan dengan exchange sort, berkaitan dengan panjangnya instruksi yang diperlukan.

Daftar Pustaka

[1] Atrinawati, L. H. Analisis Kompleksitas algoritma untuk Berbagai Macam Metode Pencarian Nilai (*Searching*) dan Pengurutan Nilai (*Sorting*) pada Tabel. Program Studi Teknik Informatika. ITB. Bandung.

[2] Drozdek, A. 2001. *Data Structures and Algorithms in C++*. Brooks/Cole Thomson Learning. California. USA.

[3] Horman, T. H., Leiserson, C. E., Rivest, R. L., dan Stein, C. 2009. *Introduction to Algorithms*. The MIT Press. Cambridge. Massachusetts London. England.

[4] Karve, S. *Insertion Sort Example*. http://www.dreamincode.net/code/snipp et279.htm. Diakses pada 18 Mei 2011.

[5] Kristanto, A. 2009. Algoritma dan Pemrograman dengan C++. Graha Ilmu. Yogyakarta.

[6] Yatini, I. dan Nasution, E. 2005. Algoritma dan Struktur Data. Graha Ilmu. Yogyakarta.

[7] Zed Shaw.2010. *Learn Python the Hard Way 3rd edition*.

[8] http://www.tutorialspoint.com/python/python_basic_syntax.htm. [online] Diakses pada Januari 2010