TUGAS 1

DESAIN DAN ANALISIS ALGORITMA

PERBANDINGAN KOMPLEKSITAS ALGORITMA PENGURUTAN BUBBLE SORT & INSERTION SORT

Dosen Pengampu: Trihastuti Yuniati, S.Kom., M.T.

Nama : Mukhamad Fatkhul Allam

Nim :19102226

Kelas : S1IF 07-MM4

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2022

I. DASAR TEORI

A. Algoritma Sorting

Algoritma Sorting adalah kumpulan langkahlangkah dalam menyelesaikan masalah dengan suatu metode tertentu. Algoritma adalah urutan instruksi yang tidak ambigu untuk memecahkan masalah [2]. Algoritma pengurutan bertujuan mengatur ulang item dari daftar yang diberikan dengan urutan tanpa mengurangi nilai atau isi di dalamnya [2]. Dalam memilih algoritma perlu memperhitungkan efisiensi dengan kompleksitas waktu dan ruang. Tujuan efisiensi adalah untuk menemukan waktu yang paling cepat dalam melakukan proses Sorting didefinisikan sebagai proses pengurutan sejumlah data yang disusun secara acak menjadi terurut dan teratur. Pengurutan ini terbagi menjadi dua yaitu ascending dan descending. Sorting didefinisikan sebagai proses pengurutan sejumlah data yang disusun secara acak menjadi terurut dan teratur. Pengurutan ini terbagi menjadi dua yaitu urutan menaik dari nilai terkecil sampai nilai terbesar (ascending) dan urutan menurun dari nilai terbesar sampai nilai terkecil (descending).

Terdapat dua macam pengurutan :

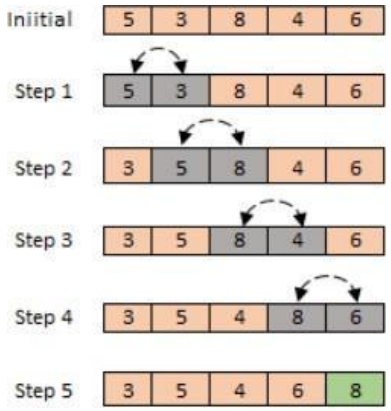
1. **Pengurutan internal (internal sort)**, yaitu pengurutan terhadap sekumpulan data yang disimpan dalam media internal komputer yang dapat diakses setiap elemennya secara langsung. Dapat dikatakan sebagai pengurutan tabel.
2. **Pengurutan eksternal (external sort)**, yaitu pengurutan data yang disimpan dalam memori sekunder, biasanya data bervolume besar sehingga tidak mampu untuk dimuat semuanya dalam memori.

B. Bubble Sort

Bubble sort adalah proses pengurutan sederhana yang bekerja dengan cara berulang kali membandingkan dua elemen data pada suatu saat dan menukar elemen data yang urutannya salah. Algoritma bubble sort merupakan algortima yang dapat dikatakan paling lama dan prosesnya sangat mudah dilakukan. Tinggal diatur apakah data yang akan dirutkan mulai dari data terkecil sampai terbesar atau dimulai dari data terbesar sampai terkecil.

Proses pengurutan dapat dilakukan dari data awal atau dari data akhir. Jika dimulai dari data awal, maka data paling awal akan dibandingkan dengan data berikutnya, jika data awal lebih besar dari data berikutnya maka tukar posisi (swap). Dan pengecekan yang sama dilakukan terhadap data yang selanjutnya sampai dengan data yang terakhir. Kebalikanya, jika dimulai dari data paling akhir maka data paling akhir akan dibandingkan dengan data didepanya, jika data paling akhir lebih kecil dari data didepanya maka tukar posisi.

Proses pada algoritma bubble sort dilakukan tahap per tahap, misalnya untuk n = 7 maka akan dilakukan (n – 1) = 6 tahap (mulai dari 0 sampai dengan n - 2).



1. Pseudocode Algoritma Bubble Sort

   Algoritma bubble sort memiliki bentuk pseudocode sebagai berikut dan berubah sedikit dikarenakan data diurutkan dari terbesar sampai terkecil:

```
// Method bubblesort berisi parameter array sebagai data yang akan di sorting
bubblesort(A[0..n - 1])
// perulangan dari index 0 sampai ke total index di kurangi 2
for i ← 0 to n - 2 do
    // perulangan dari index 0 sampai ke total index di kurangi 2 – index dari
    variabel i
    for j ← 0 to n - 2 - i do
    // jika value array dengan index dari variabel j + 1 lebih besar dari array dengan
    index variabel j maka akan di swap atau di geser
    if A[j + 1] > A[j]
        swap A[j] and A[j + 1];
```

2. Kompleksitas Bubble Sort

Algoritma di dalam bubble sort terdiri dari 2 kalang (loop) bertingkat. Kalang pertama berlangsung selama N-1 kali. Indeks untuk kalang pertama adalah Pass. Kemudian kalang tingkat kedua berlangsung dari N sampai dengan Pass+1.

Dengan demikian, proses compare yang terjadi sebanyak:

$$T(n) = (N-1) + (N-2) + \cdots + 2 + 1 = \sum N - i = \frac{N(N-1)}{2}$$

$$T$$

$T(n)$ ini merupakan kompleksitas untuk kasus terbaik ataupun terburuk. Hanya saja pada kasus terbaik, yaitu pada kasus jika struktur data telah terurut sesuai perintah, proses Pass terjadi tanpa adanya assignment. Hal ini jelas menunjukkan bahwa algoritma akan berjalan lebih cepat. Hanya saja tidak terlalu signifikan.

C. Insertion Sort

Insertion Sort merupakan sebuah teknik pengurutan dengan cara membandingkan dan mengurutkan dua data pertama pada array, kemudian membandingkan data pada array berikutnya apakah sudah berada di tempat semestinya. Algorithma insertion sort seperti proses pengurutan kartu yang berada di

tangan kita. Algorithma ini dapat mengurutkan data dari besar ke kecil (Ascending) dan kecil ke besar (Descending). Cara kerja insertion sort sebagaimana namanya. Pertama-tama, dilakukan iterasi, dimana di setiap iterasi insertion sort memindahkan nilai elemen, kemudian menyisipkannya berulang-ulang sampai ke tempat yang tepat. Begitu seterusnya dilakukan. Dari proses iterasi, seperti biasa, terbentuklah bagian yang telah di-sorting dan bagian yang belumAlgoritma ini tidak cocok untuk set data dengan jumlah besar karena kompleksitas dari algorithma ini adalah $O()$ di mana n adalah jumlah item.

Algoritma *Insertion Sort* dapat dirangkum sebagai berikut:

1. Simpan nilai T_i kedalam variabel sementara, dengan $i = 1$.
2. Bandingkan nilainya dengan elemen sebelumnya.
3. Jika elemen sebelumnya (T_{i-1}) lebih besar nilainya daripada T_i, maka tindih nilai T_i dengan nilai T_{i-1} tersebut. Decrement i (kurangi nilainya dengan 1).
4. Lakukan terus poin ke-tiga, sampai $T_{i-1} \leq T_i$.
5. Jika $T_{i-1} \leq T_i$ terpenuhi, tindih nilai di T_i dengan variabel sementara yang disimpan sebelumnya.
6. Ulangi langkah dari poin 1 di atas dengan i di-increment (ditambah satu).

1. Pseudocode Algoritma Insertion Sort

   Algoritma bubble sort memiliki bentuk pseudocode sebagai berikut:

```
    // perulangan dengan nilai awal 1 sampai nilai n kurangi 1
for i = 1 to n-1
        // menyimpan value I ke dalam variabel j
set j = i
        // menyimpan value array dengan index j ke dalam variabel t
set t = a[j]
    // melakukan perulangan jika kondisi nilai j lebih dari 0 dan nilai variabel a dengan index j-1
    lebih besar dari nilai t
repeat while j > 0 and a[j-1] > t
                // Pindah value dari array dengan index j-1 ke kanan
move a[j-1] to right
        // hentikan perulangan
end repeat
        // menyimpan nilai t kedalam array a dengan index dari value j
set a[j] = t
    // hentikan perulangan for
```

```
end for
```

2. Kompleksitas Insertion Sort

Algoritma *Insertion Sort* juga terdiri dari 2 kalang bersarang. Dimana terjadi N-1 Pass (dengan N adalah banyak elemen struktur data), dengan masing-masing *Pass* terjadi i kali operasi perbandingan. i tersebut bernilai 1 untuk *Pass* pertama, bernilai 2 untuk Pass kedua, begitu seterusnya hingga *Pass* ke N-1.

$$T(n) = 1 + 2 + \ldots + n - 1 = \sum_{i=1}^{n-i} i = \frac{n(n-1)}{2} = O(n^2)$$

II. IMPLEMENTASI

A. Bubble Sort

1. Pseudocode

- Procedure :

  //method bubble_sort dengan parameter array arr[] dan length

  ```
  bubble_sort(arr[] : integer, length : integer)
  ```

- Kamus :

  // variabel yang digunakan

  ```
  not_sorted : boolean
  j : integer
  tmp : integer
  ```

- Algoritma :

  ```
  not_sorted ←
  true j ← 0
  while(not_sorted) do
      not_sorted ←
      false j++
      for (i : integer ← 0; i < length;
            i++) tmp ← arr[i]
            arr[i] ← arr[i +
            1] arr[i + 1] ←
            tmp not_sorted ←
            true
  ```

2. Screenshoot Program

```python
# Python program for implementation of Bubble Sort
# Mukhamad Fatkhul Allam U
# 19102226
# MM 4

import numpy as np
from time import process_time

def bubbleSort(arr):
  n = len(arr)

  # Traverse through all array elements
  for i in range(n):

    # Last i elements are already in place
    for j in range(0, n-i-1):

      # traverse the array from 0 to n-i-1
      # Swap if the element found is greater
      # than the next element
      if arr[j] > arr[j+1]:
        arr[j], arr[j+1] = arr[j+1], arr[j]
```

```python
jumlah_data = int(input("masukkkan jumlah data : "));
arr = np.random.randint(0,10000,jumlah_data)
arr_1 = arr
# Driver code to test above
# arr = [64, 34, 25, 12, 22, 11, 90]

start_time1 = process_time()
bubbleSort(arr_1)
end_time1 = process_time()


print("Sorted array bubble :")

print(arr_1)
print("\n Waktu Eksekusi : ", float(end_time1-start_time1),"second")

print("\n\n Sorted array insertion  :")
```

B. Insertion Sort

1. Pseudocode

- Procedure :

```
insertionSort(arr[] : integer, size : integer)
```

- Kamus :

```
step : integer
key : integer
j : integer
```

- Algoritma :

```
for (step ← 1; step < size; step++)
     key ← arr[step]
     j ← step - 1
     while(key < arr[j]) and (j >= 0)) do
           arr[j+1] ← arr[j]
           --j
     arr[j+1] ← key
end for
```

2. Screenshoot Program

```python
# Python program for implementation of Bubble Sort
# Mukhamad Fatkhul Allam U
# 19102226
# MM 4

import numpy as np
from time import process_time

def insertionSort(arr):

  # Traverse through 1 to len(arr)
  for i in range(1, len(arr)):

    key = arr[i]

    # Move elements of arr[0..i-1], that are
    # greater than key, to one position ahead
    # of their current position
    j = i-1
    while j >=0 and key < arr[j] :
        arr[j+1] = arr[j]
        j -= 1
    arr[j+1] = key
```

```python
jumlah_data = int(input("masukkkan jumlah data : "));
arr = np.random.randint(0,10000,jumlah_data)

arr_2 = arr
# Driver code to test above
# arr = [64, 34, 25, 12, 22, 11, 90]

start_time2 = process_time()
insertionSort(arr_2)
end_time2 = process_time()

print("\n\n Sorted array insertion  :")

print(arr_2)
print("\n Waktu Eksekusi : ", float(end_time2-start_time2),"second")
```

III. PENGUJIAN

Data yang digunakan pada pengujian dengan algoritma *sorting* ini dibuat dari hasil fungi *random* pada bahasa pemrograman C++ dengan jumlah keseluruhan datanya yaitu 10.00. Data yang dibuat dengan fungsi *random* tersebut disimpan kedalam file berformat txt dengan tujuan agar data yang digunakan sama dalam proses sorting menggunakan dua metode yang berbeda.

Spesifikasi komputer yang digunakan pada pengujian ini adalah sebagai berikut:

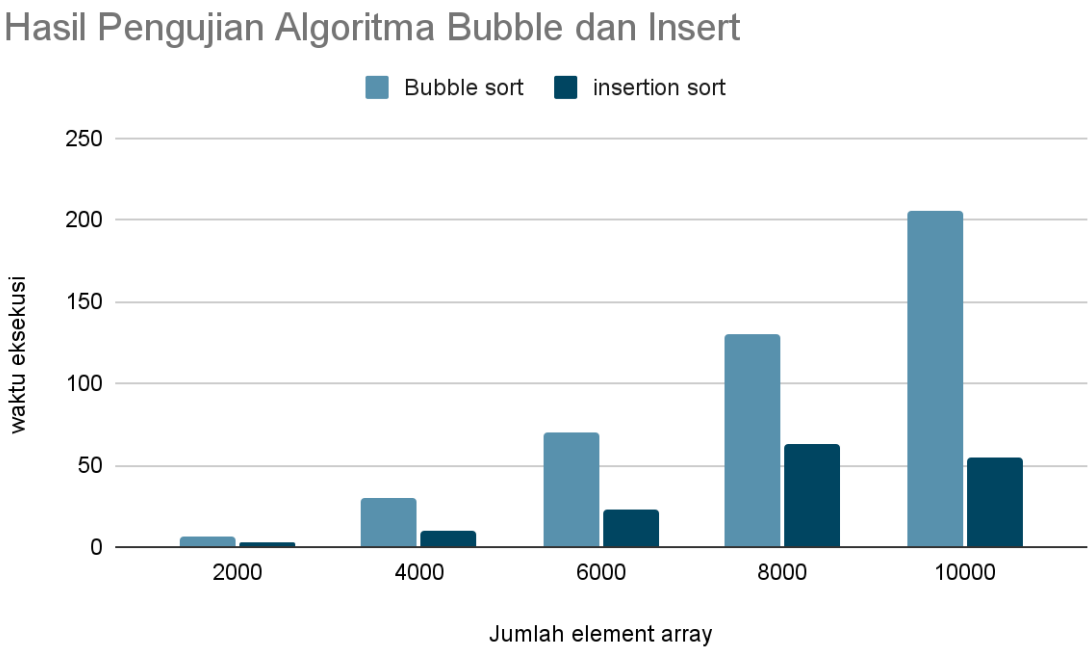
1. Processor AMD A10 9600P RADEON 5
2. RAM 8 GB
3. SSD 240GB (*Read* 520 MB/s dan *Write* 450 MB/s)

Pada proses pengujian dilakukan pengurutan atau *sorting* dengan jumlah input elemen yang berbeda, dimulai dari 2.000 elemen dengan kelipatan 2 hingga elemen akhir dengan jumlah 10.000. Hasil waktu yang didapatkan dapat dilihat pada tabel dibawah ini

Tabel 1. Hasil waktu eksekusi

n	Waktu eksekusi Bubble Sort (ms)	Waktu eksekusi Insertion Sort (ms)
2.000	7 ms	3 ms
4.000	30 ms	10 ms
6.000	70 ms	23 ms
8.000	130 ms	37 ms
10.000	205 ms	55 ms

Gambar 1. Grafik Hasil Pengujian



IV. ANALISIS HASIL PENGUJIAN

1. Bubble Sort dan Insertion sort

Algoritma di dalam bubble sort terdiri dari 2 kalang (loop) bertingkat. Kalang pertama berlangsung selama N-1 kali. Indeks untuk kalang pertama adalah *Pass*. Kemudian kalang tingkat kedua berlangsung dari N sampai dengan *Pass*+1.

Algoritma Insertion Sort juga terdiri dari 2 kalang bersarang. Dimana terjadi N-1 *Pass* (dengan N adalah banyak elemen struktur data), dengan masing-masing *Pass* terjadi i kali operasi perbandingan. i tersebut bernilai 1 untuk *Pass* pertama, bernilai 2 untuk *Pass* kedua, begitu seterusnya hingga *Pass* keDengan demikian, proses compare yang terjadi sebanyak:

$$T(n) = 1 + 2 + \ldots + n - 1 = \sum_{i=1}^{n-i} i = \frac{n(n-1)}{2} = O(n^2)$$

Tabel 2. Hasil perhitungan kompleksitas

n	$T(n) = \frac{n(n-1)}{2} = O(n^2)$	n^2
2000	1.999.000	4.000.000
4000	7.998.000	16.000.000
6000	17.997.000	36.000.000
8000	31.996.000	64.000.000
10000	49.995.000	100.000.000

Dari table di atas, untuk n yang besar pertumbuhan T(n) sebanding dengan n^2 T(n) tumbuh seperti n^2 tumbuh saat n bertambah. Kita katakan bahwa T(n) sebanding dengan n^2 dan kita tuliskan T(n) = O(n^2). Secara Kompleksitas, bubble sort dan insertion sort mempunyai Big-Oh yang sama. Walaupun begitu, insertion sort sebenarnya lebih bagus.

2. Kesimpulan

Dari hasil waktu *running* pada tabel diatas dapat disimpulkan bahwa kedua algoritma yang diuji memiliki perbedaan waktu yang signifikan. Algoritma *sorting Bubble Sort* mendapatkan waktu yang lebih lama dari algoritma *sorting Insertion Sort* sebab dalam proses pengurutan, algoritma *Bubble Sort* melakukan perbadingan terlebih dahulu dengan data selanjutnya. Hal tersebut memakan waktu yang cukup lama, maka dari itu untuk pengurutan data dalam jumlah yang banyak sangat tidak efesien.

V. REFERENSI

- Wisudawan, Wahyu Fahmy. "Kompleksitas Algoritma Sorting yang Populer Dipakai". pp. 1-8, 2006.
- S Agus Perdana Windarto, Hanny Harumy, “Belajar Dasar Algoritma Dan Pemrograman C++,” J.Chem. Inf. Model., no. Agust 2017, pp. 1-209, 2016.
- 2] hackerearth, "hackerearth," [Online]. Available: https://www.hackerearth.com/practice/algorithms/sorting/insertion-sort/tutorial/. [Accessed 10 Mei 2022].