**ANALISIS PERBANDINGAN *RUNNING TIME* ALGORITMA *HEAP SORT* DAN *INSERTION SORT* DALAM BAHASA PYTHON DAN JAVA**

Ananda Putra

(2108561001)

Program Studi Informatika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Udayana

2022

e-mail: {nandaptr.official@gmail.com}

**Abstrak**

Peran algoritma dalam perangkat lunak atau pemrograman sangat penting, sehingga perlu untuk memahami konsep dasar dari algoritma. Jadi banyak logika pemrograman yang telah dibuat, untuk kasus umum dan juga khusus. Data sequencing dapat digunakan dalam memilah nilai algoritma (pengurutan) yaitu, *insertion sort* (pengurutan oleh penyisipan) dan *heap sort* (pengurutan dengan tumpukan). Efektivitas algoritma dapat diukur dengan berapa banyak waktu dan ruang (*space* / algoritma memori) yang diperlukan untuk menjalankan algoritma. Semakin sedikit ruang dan waktu yang dibutuhkan untuk menjalankan sebuah algoritma maka algoritma tersebut dapat dikatakan semakin efisien, namun waktu dan ruang yang di dibutuhkan oleh suatu algoritma bergantung pada jumlah data yang diolah. Tulisan ini hanya membahas dan menganalisis kompleksitas waktu untuk *heap sort* dan *insertion sort* dalam bahasa python dan java.

**Kata kunci:** Java, Python, Sorting, Insertion Sort, Heap Sort, Algoritma, Kompleksitas Waktu.

**Abstract**

*The role of algorithms in software or programming is very important, so it is necessary to understand the basic concepts of algorithms. So a lot of programming logic has been created, for both general and special cases. Sequencing data can be used in sorting algorithm values ​​(sorting), namely, insertion sort (sort by insertion) and heap sort (sort by heap). The effectiveness of the algorithm can be measured by how much time and space (algorithm memory) is required to run the algorithm. The less space and time needed to run an algorithm, the more efficient the algorithm can be said to be, but the time and space needed by an algorithm depend on the amount of data processed. This paper only discusses and analyzes the time complexity for heap sort and insertion sort in python and java.*

***Keyword:*** *Java, Python, Sorting, Insertion Sort, Heap Sort, Algorithm, Time Complexity.*

**PENDAHULUAN**

Dalam beberapa konteks pemograman, algortima merupakan spesefikasi urutan langkah untuk melakukan pekerjaan tertentu. untuk menyelesaikan suatu masalah tidak cukup hanya dengan menemukan algoritma yang hasil penyelesaian masalahnya terbukti benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukan algoritma yang baik. Maka dari itu, sebuah algortima yang baik akan menggunakan algoritma yang efektif, efisien , tepat sasaran dan terstruktur. Untuk memilih algoritma yang kualifikasinya seperti itu dapat diukur dari waktu eksekusi algoritma dan kebutuhan ruang memori. Algoritma yang efisien adalah algoritma yang meminimumkan kebutuhan waktu dan ruang. Namun, kebutuhan waktu dan ruang dari suatu algoritma bergantung pada jumlah data yang diproseskan dan algoritma yang digunakan. Berdasarkan pernyataan diatas tulisan ini akan membandingkan dua buah algoritma yaitu *heap sort*  dan *insertion sort* di dua jenis bahasa yaitu python dan java untuk mengetahui algoritma mana yang lebih efisien.

**TINJAUAN PUSTAKA**

1. **Algoritma**

Menurut Saputra, dkk (2010:1) menjelaskan bahwa algoritma adalah deretan instruksi yang jelas untuk memecakan masalah, yaitu untuk memperoleh keluaran yang diinginkan dari suatu masukan.

Ada 3 definisi tentang algoritma yang dijelaskan oleh Suarga (2012:1), diantaranya :

1. Teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas tetapi tersusun secara logis dan sistematis.
2. Suatu prosedur yang jelas untuk menyelesaikan suatu persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya
3. Susunan langkah yang pasti, yang bila diikuti maka akan mentransformasi data input menjadi output yang beruapa informasi.

Algoritma merupakan suatu prosedur untuk menyelesaikan suatu masalah yang tersusun secara logis dan sistematis serta akan memperoleh data masukan menjadi keluaran yang diinginkan berupa informasi.

1. **Pengurutan Data (Sorting)**

Menurut Yahya (2014:135) Sorting adalah proses pengurutan data yang sebelumnya disusun secara acak atau tidak teratur menjadi urut dan teratur menurut suatu aturan tertentu. Biasanya pengurutan terbagi menjadi dua yaitu Ascending (pengurutan dari karakter/angka kecil ke karakter/angka besar dan Descending (pengurutan dari karakter/angka besar ke karakter/angka kecil).

Menurut Saputra, dkk (2010:1) juga menjelaskan bahwa algoritma sorting didefinisikan sebagai algoritma pengurutan sejumlah data berdasarkan nilai kunci tertentu. Pengurutan dapat dilakukan dari nilai terkecil ke nilai terbesar (ascending) atau sebaliknya (descending).

Pengurutan data (sorting) adalah suatu proses pengurutan data yang tersusun secara acak pada suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu. pengurutan ini dapat dilakukan dengan cara Ascending dan descending serta digunakan juga untuk mengurutkan data yang bertipe angka atau karakter.

**HASIL & PEMBAHASAN**

Dimulai dari pengujian algoritma “*Insertion Sort*” dalam bahasa python dengan kode seperti dibawah

import time

awal = time.time()

def InsertionSort(*val*):

    for isi in range(len(*val*)-1,0,-1):

        Max=0

        for lokasi in range(1,isi+1):

            if *val*[lokasi]>*val*[Max]:

                Max = lokasi

        temp = *val*[isi]

*val*[isi] = *val*[Max]

*val*[Max] = temp

def descending(*data*):

    for i in range(len(*data*)):

        for j in range(len(*data*)):

            if(*data*[i] > *data*[j]):

                temp = *data*[i]

*data*[i] = *data*[j]

*data*[j] = temp

words = []

infile = open("nanda\_wordlist.txt","r")

for line in infile:

    temp = line.split()

    for i in temp:

      words.append(i)

infile.close()

InsertionSort(words)

outfile = open("InsertionSort\_PY.txt", "w")

for i in words:

    outfile.writelines(i)

    outfile.writelines(" ")

outfile.close()

print("Running Time : %s detik" % (time.time() - awal))

Setelah running didapatkan output sebagai berikut:



Berlanjut ke pengujian algoritma “Heap Sort” dalam bahasa python dengan kode seperti dibawah

import time

awal = time.time()

def HeapSort(*val*):

    for start in range((len(*val*)-2)//2, -1, -1):

        siftdown(*val*, start, len(*val*)-1)

    for end in range(len(*val*)-1, 0, -1):

*val*[end], *val*[0] = *val*[0], *val*[end]

        siftdown(*val*, 0, end - 1)

    return *val*

def siftdown(*val*, *start*, *end*):

    root = *start*

    while True:

        child = root \* 2 + 1

        if child > *end*: break

        if child + 1 <= *end* and *val*[child] < *val*[child + 1]:

            child += 1

        if *val*[root] < *val*[child]:

*val*[root], *val*[child] = *val*[child], *val*[root]

            root = child

        else:

            break

words = []

infile = open("nanda\_wordlist.txt","r")

for line in infile:

    temp = line.split()

    for i in temp:

      words.append(i)

infile.close()

HeapSort(words)

outfile = open("HeapSort\_PY.txt", "w")

for i in words:

    outfile.writelines(i)

    outfile.writelines(" ")

outfile.close()

print("Running Time : %s detik" % (time.time() - awal))

Setelah running didapatkan output sebagai berikut:



Setelah diadakan pengujian 10x running untuk setiap algoritma maka didapatkan hasil sebagai berikut:

|  |  |  |
| --- | --- | --- |
| No Pengujian | Running Time Insertion Sort  (detik) | Running Time Heap Sort (detik) |
| 1 | 0.0120 | 0.0029 |
| 2 | 0.0112 | 0.0029 |
| 3 | 0.0124 | 0.0030 |
| 4 | 0.0115 | 0.0027 |
| 5 | 0.0127 | 0.0039 |
| 6 | 0.0112 | 0.0039 |
| 7 | 0.0131 | 0.0039 |
| 8 | 0.0135 | 0.0038 |
| 9 | 0.0116 | 0.0039 |
| 10 | 0.0119 | 0.0029 |

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritma | Best Case | Average Case | Worst Case |
| **Insertion Sort** | 0.0112s | 0.0120s | 0.0135s |
| **Heap Sort** | 0.0027s | 0.0033s | 0.0039s |

Berdasarkan hasil data diatas maka dapat disimpulkan *Heap Sort* menjadi algoritma yang lebih efisien daripada *Insertion Sort* ketika dijalankan pada engine python.

Selanjutnya pengujian algoritma “*Insertion Sort”* dalam bahasa java dengan kode seperti dibawah

import java.io.*\**;

import java.util.*\**;

public class InsertionSort {

    public static *void* main(String[] *args*) throws Exception {

*long* start = System.currentTimeMillis();

        File file = **new** File("nanda\_wordlist.txt");

        Scanner sc = **new** Scanner(file);

        ArrayList<String> list = **new** ArrayList<String>();

        while (sc.hasNextLine()) {

            list.add(sc.nextLine());

        }

        sc.close();

        String[] arr = list.toArray(**new** String[0]);

        insertionSort(arr);

        FileWriter writer = **new** FileWriter("Insertion\_Java.txt");

        for (String str : arr) {

            writer.write(str + System.lineSeparator());

        }

        writer.close();

*long* time = System.currentTimeMillis() - start;

        System.out.println("lama waktu (TimeMillis): " + time);

    }

    public static *void* insertionSort(String[] *arr*) {

        for (*int* i = 1; i < *arr*.length; i++) {

            String temp = *arr*[i];

*int* j = i - 1;

            while (j >= 0 && temp.compareTo(*arr*[j]) < 0) {

*arr*[j + 1] = *arr*[j];

                j--;

            }

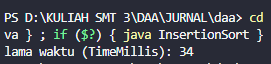
*arr*[j + 1] = temp;

        }

    }

}

Setelah running didapatkan output sebagai berikut:



Berlanjut ke pengujian algoritma “Heap Sort” dalam bahasa java dengan kode seperti dibawah

import java.io.*\**;

import java.util.*\**;

public class InsertionSort {

    public static *void* main(String[] *args*) throws Exception {

*long* start = System.currentTimeMillis();

        File file = **new** File("D:/KULIAH SMT 3/DAA/JURNAL/daa/src/nanda\_wordlist.txt");

        Scanner sc = **new** Scanner(file);

        ArrayList<String> list = **new** ArrayList<String>();

        while (sc.hasNextLine()) {

            list.add(sc.nextLine());

        }

        sc.close();

        String[] arr = list.toArray(**new** String[0]);

        insertionSort(arr);

        FileWriter writer = **new** FileWriter("Insertion\_Java.txt");

        for (String str : arr) {

            writer.write(str + System.lineSeparator());

        }

        writer.close();

*long* time = System.currentTimeMillis() - start;

        System.out.println("lama waktu (TimeMillis): " + time);

    }

    public static *void* insertionSort(String[] *arr*) {

        for (*int* i = 1; i < *arr*.length; i++) {

            String temp = *arr*[i];

*int* j = i - 1;

            while (j >= 0 && temp.compareTo(*arr*[j]) < 0) {

*arr*[j + 1] = *arr*[j];

                j--;

            }

*arr*[j + 1] = temp;

        }

    }

}

Setelah running didapatkan output sebagai berikut:



Setelah diadakan pengujian 10x running untuk setiap algoritma maka didapatkan hasil sebagai berikut:

|  |  |  |
| --- | --- | --- |
| No Pengujian | Running Time Insertion Sort  (milidetik) | Running Time Heap Sort (milidetik) |
| 1 | 33 | 33 |
| 2 | 39 | 34 |
| 3 | 36 | 31 |
| 4 | 39 | 35 |
| 5 | 41 | 38 |
| 6 | 38 | 33 |
| 7 | 36 | 32 |
| 8 | 37 | 31 |
| 9 | 37 | 32 |
| 10 | 37 | 31 |

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritma | Best Case | Average Case | Worst Case |
| **Insertion Sort** | 33ms | 37.3ms | 41ms |
| **Heap Sort** | 31ms | 33ms | 38ms |

Berdasarkan hasil data diatas maka dapat disimpulkan *Heap Sort* menjadi algoritma yang lebih efisien daripada *Insertion Sort* ketika dijalankan pada engine Java.

|  |  |  |
| --- | --- | --- |
| Algoritma | Java (avg time) | Python (avg time) |
| Insertion Sort | 37.3ms | 12ms |
| Heap Sort | 33ms | 3ms |

Berdasarkan data yang telah saya dapat maka dapat disimpulkan bahwa Heap Sort lebih efisien dari Insertion Sort dari segi time complexity & python lebih baik dalam segi efisiensi waktu daripada java dalam menjalankan Algoritma Heap Sort. Testing dilakukan dengan menggunakan (“Windows 10, Core i7 10700K, 16GB 3200Mhz DDR4, 256SSD m.2 NVME”)

**DAFTAR PUSTAKA**

Suarga. 2012. Algoritma Pemrograman. Yogyakarta: Andi.

Yahya, Sofyansyah Yusari. 2014. Analisa Perbandingan Algoritma Bubble Sort dan Selection Sort Dengan Metode Perbandingan Eksponensial. Jurnal Pelita Informatika Budi Darma, Vol : VI, No : 3, April 2014

Saputra, dkk. 2010. Analisis Algoritma Rekursif Quick Sort