**Jurnal Inovasi Informatika dan Bisnis Digital (JIIBD)**

*Website: https://ioinformatic.org/index.php/JIIBD*

***Juli 2026. Vol. 1. No. 3***

# Penerapan Algoritma Insertion Sort dalam Pengurutan Data Menggunakan Bahasa Pemrograman PHP

**Wina Azizah [1*], Syahirah Mubtasimah Siregar[2]**

*[1,2]STMIK Kaputama, Binjai, Indonesia*
*winaaazizah@gmail.com[1*], mubtasimahsyahirah@gmail.com[2]*

**Abstrak**

Algoritma pengurutan merupakan salah satu aspek fundamental dalam ilmu komputer yang berperan penting dalam meningkatkan efisiensi proses pengolahan dan pencarian data. Data yang tersusun secara teratur akan mempermudah berbagai proses komputasi, terutama dalam sistem yang membutuhkan akses data cepat dan akurat. Salah satu algoritma pengurutan yang sederhana dan mudah dipahami adalah Insertion Sort, yang termasuk dalam paradigma Decrease and Conquer. Algoritma ini bekerja dengan cara mengambil satu elemen data, kemudian menyisipkannya ke posisi yang tepat pada bagian data yang telah terurut sebelumnya secara bertahap hingga seluruh data tersusun dengan baik.

Penelitian ini bertujuan untuk memahami konsep dasar algoritma Insertion Sort, menjelaskan mekanisme kerjanya, serta mengimplementasikannya dalam bahasa pemrograman PHP. Metode yang digunakan dalam penelitian ini meliputi studi literatur, perancangan algoritma, implementasi program, serta pengujian menggunakan data array. Hasil implementasi menunjukkan bahwa algoritma Insertion Sort mampu mengurutkan data secara ascending dengan benar dan stabil. Selain itu, algoritma ini tidak memerlukan memori tambahan yang besar karena proses pengurutan dilakukan langsung pada data yang ada.

Berdasarkan analisis kompleksitas waktu, algoritma Insertion Sort memiliki kompleksitas terbaik sebesar O(n) ketika data sudah dalam keadaan hampir terurut. Namun, pada kondisi rata-rata dan terburuk, kompleksitas waktunya mencapai $O(n^2)$, sehingga kurang efisien untuk pengolahan data dalam jumlah besar. Oleh karena itu, algoritma ini lebih cocok digunakan untuk data berukuran kecil atau sebagai dasar pembelajaran konsep algoritma pengurutan..

***Kata Kunci**: Insertion Sort, Decrease and Conquer, algoritma pengurutan, kompleksitas waktu, PHP*

## 1. Pendahuluan

Perkembangan teknologi informasi menuntut pengolahan data yang cepat dan efisien. Salah satu proses penting dalam pengolahan data adalah pengurutan (sorting). Proses ini bertujuan untuk menyusun data dalam urutan tertentu, baik secara ascending maupun descending, sehingga mempermudah pencarian dan analisis data.

Terdapat berbagai algoritma sorting yang telah dikembangkan, mulai dari yang sederhana hingga kompleks. Salah satu algoritma sederhana yang sering digunakan dalam pembelajaran dasar adalah *Insertion Sort*. Algoritma ini bekerja dengan cara menyisipkan elemen ke posisi yang tepat pada bagian array yang sudah terurut sebelumnya.

Insertion Sort memiliki keunggulan dalam kemudahan implementasi dan efisiensi penggunaan memori. Namun, algoritma ini memiliki kelemahan dalam hal waktu eksekusi ketika digunakan pada data berukuran besar.

.

## 2. Tinjauan Pustaka

Penelitian yang dilakukan oleh Sitorus membahas implementasi algoritma *Insertion Sort* dalam proses pengurutan bilangan positif menggunakan pendekatan Flowgorithm. Penelitian ini menekankan pentingnya algoritma sorting dalam pembelajaran dasar pemrograman, khususnya dalam membantu mahasiswa memahami konsep logika algoritmik secara bertahap. Hasil penelitian menunjukkan bahwa *Insertion Sort* sangat efektif digunakan sebagai media pembelajaran karena langkah-langkahnya sederhana dan mudah divisualisasikan [1]. Selain itu, penelitian ini juga menjelaskan bahwa algoritma Insertion Sort bekerja dengan cara menyisipkan elemen ke posisi yang tepat pada bagian array yang telah terurut sebelumnya. Pendekatan ini membuat algoritma ini sangat cocok untuk data yang ukurannya kecil atau hampir terurut. Namun, dalam implementasi nyata, performa algoritma ini akan menurun ketika digunakan pada dataset yang besar karena kompleksitas waktunya yang bersifat kuadratik, dengan demikian, penelitian ini mendukung bahwa Insertion Sort masih relevan sebagai dasar pembelajaran algoritma, meskipun tidak selalu optimal untuk kebutuhan industri yang menangani data dalam skala besar.

Pujiono melakukan penelitian mengenai perbandingan efisiensi waktu dan penggunaan memori dari berbagai algoritma sorting, termasuk Insertion Sort, menggunakan bahasa pemrograman Java. Penelitian ini menunjukkan bahwa perkembangan teknologi informasi menuntut

pengolahan data yang lebih cepat dan efisien, sehingga pemilihan algoritma yang tepat menjadi sangat penting dalam sistem komputasi modern [2].

Hasil penelitian menunjukkan bahwa Insertion Sort memiliki keunggulan dalam penggunaan memori karena termasuk algoritma *in-place*, sehingga tidak memerlukan ruang tambahan yang besar. Namun, dari sisi waktu komputasi, algoritma ini kalah dibandingkan algoritma lain seperti Quick Sort dan Merge Sort, terutama ketika jumlah data semakin besar. Hal ini disebabkan oleh banyaknya proses perbandingan dan pergeseran elemen yang terjadi selama proses sorting.
Penelitian ini memperkuat pemahaman bahwa Insertion Sort lebih cocok digunakan pada kondisi tertentu, seperti dataset kecil atau hampir terurut, dibandingkan dengan penggunaan umum pada data besar.

Studi perbandingan tujuh algoritma sorting menggunakan bahasa pemrograman JavaScript dengan fokus pada waktu komputasi dan penggunaan memori. Dalam penelitian ini, berbagai algoritma seperti Merge Sort, Quick Sort, dan Insertion Sort diuji menggunakan data acak dalam jumlah tertentu untuk mengetahui performanya secara empiris [3]. Hasil penelitian menunjukkan bahwa Insertion Sort memiliki performa yang cukup baik dalam kondisi tertentu, bahkan dalam beberapa pengujian menunjukkan waktu yang relatif efisien dibandingkan algoritma lain untuk dataset kecil. Namun, ketika jumlah data meningkat, algoritma ini mengalami penurunan performa yang signifikan dibandingkan algoritma berbasis divide-and-conquer seperti Quick Sort.
Penelitian ini menunjukkan bahwa efisiensi algoritma sangat bergantung pada karakteristik data yang digunakan. Oleh karena itu, pemilihan algoritma harus mempertimbangkan ukuran dan kondisi data agar mendapatkan hasil yang optimal.
Analisis komparatif terhadap beberapa algoritma sorting yang sering digunakan dalam competitive programming, termasuk Insertion Sort. Penelitian ini menekankan bahwa sorting merupakan operasi dasar yang sangat penting dalam berbagai aplikasi komputer, mulai dari pengolahan data hingga sistem pencarian [4]. Dalam hasil penelitiannya, ditemukan bahwa setiap algoritma memiliki karakteristik dan efisiensi yang berbeda tergantung pada kondisi data. Insertion Sort memiliki keunggulan dalam kesederhanaan dan stabilitas, namun kurang unggul dalam efisiensi waktu untuk dataset besar. Sebaliknya, algoritma seperti Quick Sort dan Merge Sort lebih unggul dalam hal kompleksitas waktu. penelitian ini memberikan gambaran bahwa tidak ada algoritma yang paling sempurna untuk semua kondisi, sehingga pemilihan algoritma harus disesuaikan dengan kebutuhan dan karakteristik data yang diolah.

Analisis kompleksitas antara algoritma Insertion Sort dan Selection Sort dengan implementasi menggunakan bahasa pemrograman Java. Penelitian ini bertujuan untuk memahami performa algoritma melalui pengujian pada kondisi terbaik, rata-rata, dan terburuk [5].
Hasil penelitian menunjukkan bahwa Insertion Sort memiliki kompleksitas waktu terbaik sebesar O(n) ketika data sudah hampir terurut, sedangkan pada kondisi rata-rata dan terburuk mencapai $O(n^2)$. Hal ini menunjukkan bahwa performa algoritma sangat dipengaruhi oleh kondisi awal data. Selain itu, dibandingkan dengan Selection Sort, Insertion Sort cenderung lebih efisien dalam beberapa kasus karena jumlah pergeseran data yang lebih adaptif.

Decrease and Conquer adalah paradigma algoritma yang menyelesaikan masalah dengan cara mengurangi ukuran masalah secara bertahap hingga ditemukan solusi akhir.
Contoh Algoritma yang menggunakan metode ini:

a) Binary Search
b) Insertion Sort
c) DFS
d) Euclidean Algoritma
e) Topological Sorting
f) Exponentiaton by Squaring

Karakteristik metode ini :

a) Mengurangi ukuran masalah sedikit demi sedikit
b) Solusi sebelumnya membantu Solusi berikutnya.
c) Lebih efesien disbanding brute force.
d) Cocok untuk algoritma pencarian dan pengurutan.

**Tabel 1:** Perbedaan dengan Dvide and Conquer

| **Decrease and Conquer** | **Divide and Conquer** |
|---|---|
| Mengurangi masalah bertahap | Membagi masalah menjadi banyak bagian |
| Satu submasalah bertahap | Banyak submasalah |
| Contoh : Insertion Sort | Contoh : Merge Sort |

**Tabel 2:** Perbedaan dengan Brute Force

| **Decrease and Conquer** | **Brute Force** |
|---|---|
| Menggunakan Strategi | Mencoba semua kemungkinan |
| Lebih Efesien | Lebih lambat |
| Hemat Waktu | Membutuhkan waktu besar |

Contoh Penerapan dalam kehidupan nyata:
Contoh sederhana Insertion Sort adalah saat seseorang menyusun kartu remi.
Kartu pertama dianggap sudah urut, kartu berikutnya disisipkan ke posisi yang benar. proses dilakukan terus hingga semua kartu tersusun rapi

# 3. Metodologi

## 3.1. Pengertian Insertion Sort

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara mengambil satu elemen lalu menyisipkannya ke posisi yang tepat pada bagian data yang sudah terurut.
Algoritma ini mudah dipahami dan cocok digunakan untuk data kecil.

## 3.2. Cara kerja Insertion Sort Contoh Data

```
Plain text

[5, 3, 4, 1, 2]
```

Langkah 1:
Data pertama dianggap sudah urut.

```
Plain text

[5] [3,4,1,2]
```

Langkah 2:
Ambil angka 3, lalu bandingkan dengan 5.
Karena 3 lebih kecil dari 5,maka tukar posisi.
Hasil:

```
Plain text

[3,5,4,1,2]
```

Langkah 3:
Ambil angka 4.
Bandingkan dengan 5 (Geser 5)
Bandingkan dengan 3 (Letakkan setelah 3).
Hasil:

```
Plain text

[3,4,5,1,2]
```

Langkah 4:
Ambil angka 1.
Geser semua angka yang lebih besar.
Hasil;

```
Plain text

[1,3,4,5,2]
```

Langkah 5:
Ambil angka 2
Geser angka yang lebih besar.
Hasil akhir ;

```
Plain text

[1,2,3,4,5]
```

**Tabel 3:** Ilustrasi Proses Insertion Sort

| Iterasi | Kondisi Array |
|---|---|
| Awal | 5 3 4 1 2 |
| 1 | 3 5 4 1 2 |
| 2 | 3 4 5 1 2 |

| | |
|---|---|
| 3 | 1 3 4 5 2 |
| 4 | 1 2 3 4 5 |

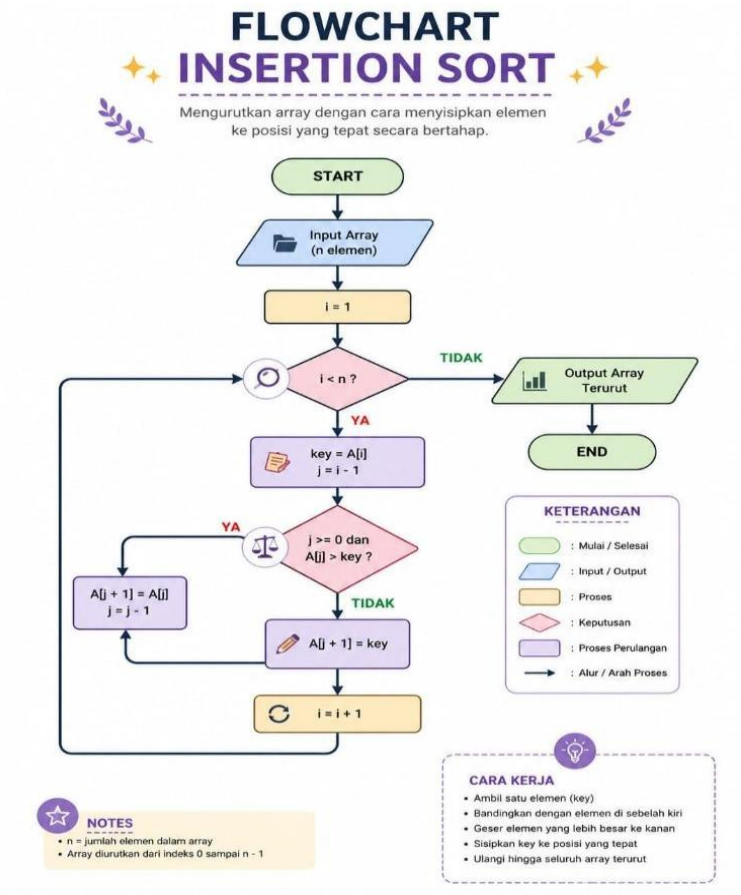
### 3.3. Pseudocode Insertion Sort

```
InsertionSort(A)
for i = 1 sampai n-1
     key = A[i]
     j = i - 1

     while j >= 0 dan A[j] > key
          A[j+1] = A[j]
          j = j - 1

     A[j+1] = key
```

### 3.4. Flowchart Sederhana




### 3.5. Analisis Kompleksitas Waktu

**Best Case (kasus terbaik)**
Best Case terjadi Ketika data sudah terurut.
Contoh:

```
Plain text

[1,2,3,4,5]
```

Pada kondisi ini :

- Tidak ada pertukaran data
- Hanya dilakukan satu kali pengecekan setiap iterasi.

Kompleksitas waktunya:

$$O(n)$$

Artinya : waktu bertambah secara linear mengikuti jumlah data.

**Wort Case (Kasus Terburuk)**
Wort case terjadi Ketika data terurut terbalik.
Contoh:

```
Plain text

[5, 4, 3, 2, 1]
```

Pada kondisi ini :

- Semua elemen harus dibandingkan
- Semua elemen harus digeser Jumlah operasi menjadi sangkat banyak.

Maka komplesitas waktunya:

$$O(n^2)$$

**Avarage Case (Kasus rata-rata)**
Avarage case terjadi Ketika data dalam kondisi acak.
Contoh:

```
Plain text

[4, 1, 5, 2, 3]
```

Pada kondisi ini:

- Sebagai data perlu digeser
- Sebagai data tidak perlu digeser Kompleksitas waktu tetap:

$$O(n^2)$$

## 4. Implementasi Program Php

### 4.1. Program Insertion Sort Menggunakan PHP

```php
<?php

$data = [5, 3, 4, 1, 2];

for ($i = 1; $i < count($data); $i++) {

    $key = $data[$i];
    $j = $i - 1;

    while ($j >= 0 && $data[$j] > $key) {
        $data[$j + 1] = $data[$j];
        $j--;
    }

    $data[$j + 1] = $key;
}

echo "Hasil pengurutan: <br>";

foreach ($data as $nilai) {
    echo $nilai . " ";
}

?>
```

Hasil Output:

```
Hasil pengurutan:
1 2 3 4 5
```

### 4.2. Hasil Pengujian Program

Program Insertion Sort yang dibuat menggunakan bahasa pemrograman PHP telah berhasil dijalankan dengan baik. Program mampu menerima data array, melakukan proses pengurutan, dan menampilkan hasil akhir secara ascending (dari kecil ke besar).
Berdasarkan hasil pengujian tersebut, dapat disimpulkan bahwa algoritma Insertion Sort berhasil mengurutkan data sesuai dengan tujuan program.

### 4.3. Pembahasan Proses Algoritma

Pada algoritma Insertion Sort, proses pengurutan dilakukan dengan cara mengambil satu elemen kemudian membandingkannya dengan elemen sebelumnya hingga ditemukan posisi yang tepat.
Proses pengurutan dilakukan secara bertahap sebagai berikut:

**Tabel 4**: Proses

| Iterasi | Proses | Hasil Array |
|---|---|---|
| Awal | Data Awal | 5 3 4 1 2 |
| 1 | Angka 3 di bandingkan dengan 5 lalu di tukar. | 3 5 4 1 2 |
| 2 | Angka 4 disisipkan di antara 3 dan 5. | 3 4 5 1 2 |
| 3 | Angka 1 dipindahkan ke posisi paling depan. | 1 3 4 5 2 |
| 4 | Angka 2 disisipkan setelah angka 1. | 1 2 3 4 5 |

Dari proses tersebut terlihat bahwa algoritma bekerja dengan cara menyusun bagian kiri array menjadi terurut terlebih dahulu, kemudian menambahkan elemen baru ke posisi yang sesuai.

### 4.4. Analisis Hasil Pengujian

Berdasarkan hasil pengujian program, algoritma Insertion Sort memiliki performa yang cukup baik pada jumlah data kecil.
Beberapa hal yang dapat dianalisis dari hasil pengujian yaitu:

a. Program Berjalan Sesuai Logika Algoritma; Setiap elemen berhasil dibandingkan dan ditempatkan pada posisi yang benar sehingga menghasilkan array yang terurut.
b. Penggunaan Memori Efisien; Insertion Sort tidak membutuhkan array tambahan karena proses sorting dilakukan langsung pada array asli.
c. Proses Sorting Mudah Dipahami; Langkah-langkah algoritma terlihat jelas sehingga cocok digunakan untuk pembelajaran dasar algoritma pemrograman.

d. Waktu Eksekusi Bertambah Saat Data Banyak; Semakin besar jumlah data, maka proses perbandingan dan pergeseran elemen akan semakin banyak.

**Tabel 5:** Perbandingkan dengan Algoritma Lain

| Algoritma | Kompleksitas | Kelebihan |
|---|---|---|
| Insertion Sort | O ($n^2$) | Sederhana dan hemat memori. |
| Merge Sort | O (n log n) | Lebih cepat untuk data besar |
| Bubble Sort | O ($n^2$) | Mudah dipahami |
| Quick Sort | O (n log n) | Sangat cepat pada banyak kasus |

Dari tabel tersebut dapat diketahui bahwa Insertion Sort unggul dalam kesederhanaan, namun kurang efisien dibanding algoritma modern untuk data berukuran besar.

## 5. Kesimpulan

Berdasarkan hasil pembahasan dan implementasi program yang telah dilakukan, dapat disimpulkan bahwa algoritma Insertion Sort merupakan salah satu algoritma pengurutan sederhana yang menggunakan metode Decrease and Conquer. Algoritma ini bekerja dengan cara menyisipkan setiap elemen ke posisi yang tepat pada bagian array yang telah terurut sebelumnya. Proses tersebut dilakukan secara bertahap hingga seluruh data tersusun secara teratur.
Dalam implementasinya menggunakan bahasa pemrograman PHP, algoritma Insertion Sort dapat dibuat dengan cukup mudah karena struktur logikanya sederhana dan mudah dipahami. Program berhasil menerima input data, melakukan proses pengurutan, serta menampilkan hasil data yang telah terurut dengan baik. Hal ini menunjukkan bahwa algoritma Insertion Sort dapat digunakan sebagai dasar pembelajaran dalam memahami konsep sorting pada pemrograman.

Analisis kompleksitas waktu, Insertion Sort memiliki kompleksitas terbaik sebesar O(n) ketika data sudah hampir terurut. Namun pada kondisi rata-rata dan terburuk, kompleksitas waktunya mencapai O($n^2$) karena banyaknya proses perbandingan dan pergeseran elemen. Oleh sebab itu, algoritma ini kurang efisien untuk data berukuran besar, tetapi masih sangat baik digunakan pada data berukuran kecil atau hampir terurut. Selain itu, Insertion Sort memiliki beberapa kelebihan seperti mudah dipahami, mudah diimplementasikan, hemat memori,

dan stabil dalam proses pengurutan. Walaupun demikian, algoritma ini juga memiliki kelemahan yaitu waktu proses yang lebih lambat dibandingkan algoritma sorting lain seperti Merge Sort atau Quick Sort ketika jumlah data semakin banyak.

Secara keseluruhan, pembelajaran mengenai Insertion Sort memberikan pemahaman penting tentang konsep dasar algoritma pengurutan dan cara kerja metode Decrease and Conquer. Dengan memahami algoritma ini, mahasiswa dapat lebih mudah mempelajari algoritma sorting lainnya yang lebih kompleks di masa mendatang.

## Daftar Pustaka

[1] Sitorus, Z., Prayogi, D., Rizko, M. A., Suteja, A. G., & Harahap, M. R. (2024). Implementation of the Insertion Sort Algorithm to Sort Positive Integers in Ascending Order Using Flowgorithm. Journal of Information Technology, Computer Science and Electrical Engineering, 1(3), 323–328.

[2] Pujiono, I. P., Trianto, R. B., & Hana, F. M. (2024). Perbandingan Efisiensi Memori dan Waktu Komputasi Pada 7 Algoritma Sorting Menggunakan Bahasa Pemrograman Java. SIMKOM, 9(2), 218–230.

[3] Candorcare, A. S., & Subramani, K. (2026). Inserge-Sort: Rewriting Insertion-Sort. International Journal of Computer Mathematics: Computer Systems Theory.

[4] Ganapathi, P., & Chowdhury, R. (2021). Parallel Divide-and-Conquer Algorithms for Bubble Sort, Selection Sort and Insertion Sort. The Computer Journal, 65(10), 2709–2719.

[5] Mamatnabiyev, Z., & Zhaparov, M. (2024). Comparative Analysis of Sorting Algorithms Used in Competitive Programming. Journal of Emerging Technologies and Computing, 55(2), 64–70.

[6] Phillip, A., Kumar, S., & Shah, S. A. (2024). Language Aware Sorting: Empirical and ML Driven Performance Evaluation of Insertion Sort. International Journal of Computer Science.

[7] Ghofur, A., Nazila, J., Holidiyah, N., Kholifah, S., Kulsum, F. H., & Insiyah, N. (2025). Penerapan Algoritma Insertion Sort Pada Aplikasi Pengolahan Data Mahasiswa Menggunakan Java. Eastasouth Journal of Positive Community Services, 4(1), 12–19.

[8] Rodríguez, et al. (2022). A Sorting Algorithm Based on Ordered Block Insertions. Journal of Computational Science, 64, 101866.

[9] Rahmani, M. K. I. (2024). Rahmani Sort: A Novel Variant of Insertion Sort Algorithm with O(n log n) Complexity. arXiv Preprint.