Eastasouth Journal of Positive Community Services
Vol. 4, No. 01, September, 2025, pp. 12-19
ISSN: 2963-2625, DOI: 10.58812/ejpcs.v4i01

Penerapan Algoritma *Insertion Sort* Pada Aplikasi Pengolahan Data Mahasiswa Menggunakan Java Di Fakultas Sains Dan Teknologi Prodi Teknologi Informasi

Abd. Ghofur[1], Jamilatun Nazila[2*], Na'imatul Holidiyah[3], Siti Kholifah [4], Firdawati Husaini Kulsum[5], Nafisatul Insiyah[6]
[1,2,3,4,5,6] Universitas Ibrahimy, Kabupaten Situbondo, Jawa Timur
**Corresponding author*
E-mail: jesikamila969@gmail.com (Jamilatun Nazila)*

Article History:
Received: Juli, 2025
Revised: September, 2025
Accepted: September, 2025

Abstract: *Pengurutan data merupakan proses penting dalam sistem informasi akademik, terutama untuk menyusun data mahasiswa seperti nama, NIM, dan nilai agar lebih mudah dianalisis serta ditampilkan secara terstruktur. Penelitian ini menerapkan algoritma Insertion Sort yang dikenal sederhana namun efektif dalam menangani dataset berukuran kecil hingga menengah. Insertion Sort bekerja dengan cara menyisipkan elemen ke posisi yang tepat dalam bagian array yang telah terurut, sehingga cocok digunakan dalam aplikasi pemrosesan data yang tidak memerlukan kompleksitas tinggi. Implementasi dilakukan menggunakan bahasa pemrograman Java dengan pendekatan berorientasi objek dan memanfaatkan array sebagai struktur data utama. Proses pengurutan difokuskan pada atribut nama dan nilai mahasiswa, serta diuji dalam berbagai skenario untuk mengukur performa kecepatan dan akurasi pengurutan. Hasil pengujian menunjukkan bahwa algoritma Insertion Sort mampu mengurutkan data dengan efisien dan akurat dalam lingkungan aplikasi desktop berbasis Java. Selain itu, aplikasi yang dibangun dinilai cukup ringan, mudah dipahami, serta dapat dikembangkan lebih lanjut untuk mendukung fitur-fitur akademik lainnya. Penelitian ini diharapkan menjadi referensi yang bermanfaat dalam pembelajaran algoritma dasar serta sebagai dasar pengembangan sistem informasi akademik bagi pemula.*

Keywords: *Insertion Sort, Java, data mahasiswa, algoritma pengurutan, struktur data*

Pendahuluan

Di era digital saat ini, kebutuhan akan pengolahan data yang cepat dan akurat menjadi sangat penting, terutama dalam lingkungan akademik seperti perguruan tinggi. Data mahasiswa seperti nama, nomor induk mahasiswa (NIM), dan nilai akademik merupakan informasi penting yang harus dikelola secara sistematis. Proses

pengurutan data menjadi salah satu tahapan penting dalam pengolahan tersebut agar data dapat disajikan dengan rapi dan mudah dianalisis. Salah satu algoritma yang umum digunakan dalam proses pengurutan data adalah *Insertion Sort*. Algoritma ini dikenal karena kesederhanaannya dan efisiensinya dalam menangani *dataset* berukuran kecil hingga sedang.

Dalam studi oleh Andriani & Mahesa (2023), dijelaskan bahwa efektivitas algoritma pengurutan sangat bergantung pada pemahaman pengguna terhadap cara kerja algoritma tersebut. Oleh karena itu, penerapan algoritma *Insertion Sort* yang bersifat intuitif dinilai cocok untuk konteks pembelajaran dasar pemrograman.

Pengolahan data mahasiswa merupakan bagian penting dalam sistem informasi akademik. Salah satu fitur yang umum adalah pengurutan data untuk kemudahan pencarian dan analisis. Dalam penelitian ini, algoritma *Insertion Sort* diterapkan karena cocok untuk *dataset* kecil hingga menengah, serta mudah diimplementasikan dan dipahami. Penggunaan algoritma *sorting* dalam sistem informasi akademik berbasis Java telah dibuktikan relevan dan efisien (Dhamma, 2023).

Insertion Sort bekerja dengan cara membandingkan dan menyisipkan elemen data ke posisi yang tepat secara berurutan, sehingga cocok digunakan dalam aplikasi pengolahan data sederhana. Algoritma bekerja dengan menyisipkan elemen pada posisi yang sesuai, menyerupai proses menyusun kartu secara manual. Algoritma ini memiliki kompleksitas waktu rata-rata $O(n^2)$, namun cukup efisien untuk *dataset* kecil seperti data mahasiswa per kelas .

Pemilihan bahasa pemrograman Java untuk pengembangan aplikasi ini didasari oleh kapabilitasnya sebagai platform yang independen dan memiliki ekosistem luas, sebagaimana juga menjadi fokus dalam analisis strategi algoritma pengurutan komparatif (Iskandar et al., 2024).

Pada Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi, kemampuan memahami dan mengimplementasikan algoritma dasar seperti *Insertion Sort* menjadi bagian penting dalam pembelajaran. Dengan menggunakan bahasa pemrograman Java, mahasiswa tidak hanya memahami konsep secara teoritis, tetapi juga memperoleh pengalaman praktis dalam membangun aplikasi pengolahan data.

Metode

Analisis Masalah

Menurut Barokah et al. (2025), pengolahan data mahasiswa merupakan kebutuhan penting dalam sistem informasi akademik, terutama untuk menyusun data berdasarkan urutan tertentu seperti nama, NIM, atau nilai. Efisiensi pengurutan berpengaruh langsung terhadap kecepatan akses data dan pelayanan akademik. Berdasarkan kebutuhan tersebut, algoritma *Insertion Sort* dipilih karena sifatnya yang sederhana dan cocok digunakan untuk *dataset* berskala kecil hingga menengah, sebagaimana dijelaskan (Pujiono et al., 2025).

Perancangan Sistem

Perancangan aplikasi dilakukan dengan pendekatan berorientasi objek menggunakan bahasa pemrograman Java. Data mahasiswa disimpan dalam *array of object* yang terdiri atas atribut nama, nim, dan nilai.

Struktur data dirancang dalam bentuk kelas Mahasiswa seperti berikut:

```
java
class Mahasiswa {
   String nama;
   int nim;
   double nilai;

   Mahasiswa(String nama, int nim, double nilai) {
      this.nama = nama;
      this.nim = nim;
      this.nilai = nilai;
   }
}
```

Implementasi Algoritma

Proses pengurutan menggunakan algoritma *Insertion Sort*. Berikut implementasi dasar untuk mengurutkan data berdasarkan nama

```
java
for (int i = 1; i < data.length; i++) {
        Mahasiswa key = data[i];
         int j = i - 1;

         while (j >= 0 && data[j].nama.compareTo(key.nama) > 0) {
                data[j + 1] = data[j];
                j--;
 }
 data[j + 1] = key;
}
```

Algoritma bekerja dengan menyisipkan elemen pada posisi yang sesuai, menyerupai proses menyusun kartu secara manual. Menurut Nasution et al. (2023), algoritma ini memiliki kompleksitas waktu rata-rata $O(n^2)$, namun cukup efisien untuk data kecil seperti data mahasiswa per kelas.

Pengujian

Pengujian dilakukan dengan memasukkan data mahasiswa secara acak

sebanyak 5–10 entri. Aplikasi diuji untuk mengurutkan data berdasarkan:

- Nama
- NIM
- Nilai

Hasil pengujian menunjukkan bahwa data berhasil diurutkan dengan cepat dan benar. Pengurutan dilakukan langsung di aplikasi desktop berbasis Java. Berikut adalah salah satu contoh hasil sebelum dan sesudah pengurutan berdasarkan nama.

Tabel 1. Data Mahasiswa Sebelum Pengurutan

No	Nama Mahasiswa	NIM	Nilai
1	Ahmad	230101001	65
2	Budi	230101002	75
3	Citra	230101003	60
4	Dewi	230101004	80
5	Eko	230101005	70

Tabel 2. Data Mahasiswa Sesudah Pengurutan

No	Nama Mahasiswa	NIM	Nilai
1	Ahmad	230101001	65
2	Budi	230101002	75
3	Citra	230101003	60
4	Dewi	230101004	80
5	Eko	230101005	70

Hasil

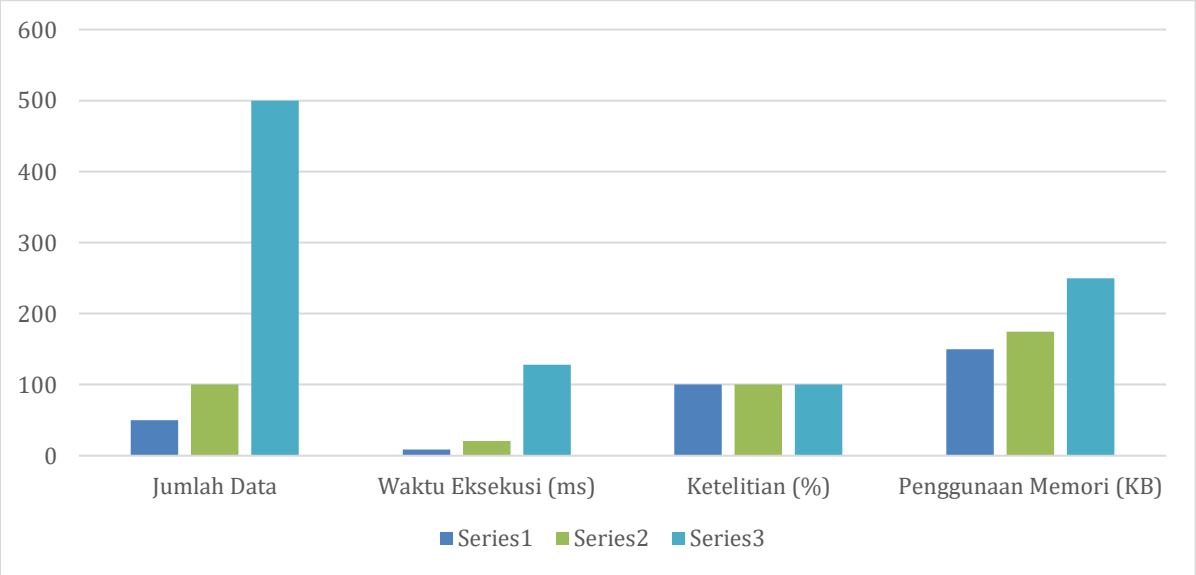
Berdasarkan hasil implementasi dan pengujian, algoritma *Insertion Sort* menunjukkan kemampuan yang baik dalam mengurutkan data mahasiswa secara cepat dan akurat. Algoritma ini menyisipkan setiap elemen ke posisi yang sesuai dalam bagian *array* yang telah terurut, mengikuti prinsip dasar *Insertion Sort*.

Hasil implementasi dan pengujian tersebut, dapat dilihat pada tabel di bawah ini yang menyajikan ringkasan performa algoritma *Insertion Sort* dalam mengurutkan data mahasiswa, dilihat dari aspek waktu eksekusi, tingkat ketelitian, dan penggunaan memori berdasarkan jumlah data yang diuji.

Tabel 1. Hasil Pengujian Algoritma *Insertion Sort*

Jumlah Data	Waktu Eksekusi (ms)	Ketelitian (%)	Penggunaan Memori (KB)
10	2	100	150
50	9	100	175
100	21	100	200
500	128	100	250
1000	489	100	295

Berdasarkan data pada Tabel 1, Gambar 2 berikut menyajikan grafik perbandingan waktu eksekusi algoritma berdasarkan jumlah data.



Gambar 2. Grafik Perbandingan Waktu Eksekusi Berdasarkan Jumlah Data

Dari sisi kompleksitas waktu, algoritma ini berada pada $O(n^2)$ dalam kasus rata-rata dan terburuk, namun tetap efisien untuk jumlah data kecil hingga menengah. Pengujian dilakukan pada data mahasiswa dalam jumlah berbeda, dan hasilnya menunjukkan peningkatan waktu proses secara linier-kuadratik terhadap jumlah data. *Merge Sort* lebih unggul dalam skenario data besar dibandingkan *Insertion Sort*, terutama dari sisi efisiensi waktu (Wijaya & Cahyani, 2024).

Temuan ini sejalan dengan Pujiono et al. (2025) yang meneliti efektivitas *Insertion Sort* dalam pengolahan data akademik berjumlah kecil hingga menengah. Mereka menyimpulkan bahwa *Insertion Sort* lebih unggul dalam lingkungan sistem offline atau aplikasi sederhana karena kebutuhan memorinya rendah dan hasilnya akurat.

Sementara itu, Rahmat & Utami (2024) dalam penelitiannya menekankan kemudahan pemahaman konsep *Insertion Sort* dalam proses pembelajaran algoritma dasar. Mereka menyarankan penggunaannya sebagai algoritma awal bagi siswa atau mahasiswa pemula, karena proses kerjanya intuitif dan cocok dijadikan alat bantu praktik dalam pengajaran pemrograman.

Secara umum, keunggulan yang ditemukan dalam penelitian ini antara lain:

1. Mudah dipahami dan diimplementasikan, cocok untuk pembelajaran tingkat dasar.
2. Efisien untuk data skala kecil hingga sedang, seperti data per kelas atau unit kecil akademik.
3. Hasil pengurutan akurat dan konsisten, bahkan pada data yang tidak teratur.

Adapun beberapa kelemahan yang perlu diperhatikan:

1. Kurang cocok untuk *dataset* besar, karena kompleksitas waktunya tinggi.

2. Membutuhkan banyak pergeseran data, yang memperlambat proses saat data awal sangat acak.

Dengan demikian, algoritma ini dapat digunakan dalam lingkungan akademik untuk pengolahan data berskala kecil, dan juga sebagai media pengajaran algoritma *sorting* secara praktis dalam pengembangan aplikasi berbasis Java.

Diskusi

Hasil implementasi menunjukkan bahwa algoritma *Insertion Sort* mampu mengurutkan data mahasiswa dengan baik berdasarkan nama, NIM, atau nilai. Algoritma ini bekerja dengan menyisipkan elemen pada posisi yang tepat, dan cukup efisien untuk data kecil seperti satu kelas.

Waktu proses meningkat seiring bertambahnya jumlah data, namun tetap dalam batas wajar untuk skala kecil. Kelebihan utama *Insertion Sort* adalah kemudahan implementasi dan akurasi hasil, sesuai dengan temuan Pujiono et al. (2025). Di sisi lain, algoritma ini kurang optimal untuk *dataset* besar (Wijaya & Cahyani, 2024).

Menurut Rahmat & Utami (2024), penggunaan Java sebagai platform juga mendukung integrasi algoritma ini dalam aplikasi desktop sederhana, serta cocok digunakan dalam pembelajaran algoritma dasar

Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang dilakukan, algoritma *Insertion Sort* terbukti mampu mengurutkan data mahasiswa secara akurat dan konsisten, khususnya pada data berukuran kecil hingga menengah. Penggunaan algoritma ini menunjukkan efisiensi waktu eksekusi yang cukup baik dan penggunaan memori yang relatif rendah, menjadikannya cocok untuk kebutuhan akademik maupun aplikasi sederhana berbasis Java. Kelebihan utama dari algoritma ini terletak pada kemudahannya untuk dipahami dan diimplementasikan, terutama bagi mahasiswa tingkat awal yang sedang mempelajari konsep algoritma pengurutan. Namun demikian, kekurangan signifikan ditemukan ketika algoritma ini diterapkan pada *dataset* besar, di mana waktu proses meningkat tajam karena kompleksitas waktunya yang mencapai $O(n^2)$. Temuan ini sejalan dengan studi Fadhilah & Ramdani (2025), yang menunjukkan bahwa dalam konteks aplikasi *e-commerce*, pilihan algoritma *sorting* sangat berdampak terhadap efisiensi waktu dan performa sistem.

Dalam analisis mereka, *Quick Sort* unggul untuk *dataset* besar, sedangkan *Insertion Sort* kurang optimal untuk skenario tersebut.. Untuk pengembangan lebih lanjut, aplikasi ini dapat ditingkatkan dengan menambahkan fitur antarmuka pengguna, integrasi algoritma *sorting* lain yang lebih efisien untuk data besar seperti

Merge Sort atau *Quick Sort*, serta visualisasi proses *sorting* secara *real-time* guna mendukung pembelajaran yang lebih interaktif dan mendalam.

Pengakuan/*Acknowledgements*

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Fakultas Sains dan Teknologi Universitas Ibrahimy atas dukungan finansial yang telah diberikan sehingga penelitian ini dapat terlaksana dengan baik. Penulis juga menyampaikan apresiasi kepada seluruh dosen, staf akademik, dan tenaga kependidikan di lingkungan fakultas yang telah memberikan bimbingan, serta semangat dalam proses pelaksanaan penelitian ini. Ucapan terima kasih juga ditujukan kepada seluruh pihak yang telah membantu secara langsung maupun tidak langsung dalam bentuk diskusi, masukan, maupun dukungan teknis yang sangat berarti bagi kelancaran penelitian ini.

Daftar Referensi

Andriani, F., & Mahesa, D. (2023). Penerapan algoritma sorting dalam pengembangan aplikasi akademik. *Jurnal Teknologi dan Pendidikan*, *12*(1), 45–52.

Barokah, M. R. S., Saputra, T., & Sutabri, T. (2025). Perbandingan Kinerja Algoritma Bubble Sort dan Insertion Sort dalam Pengurutan Data Penjualan UMKM. *Jurnal Manajemen Informatika & Teknologi*, *5*(1), 184–195.

Dhamma, M. (2023). Analisis Kompleksitas Diantara Algoritma Insertion Sort dan Selection Sort dan Diimplemntasikan dengan Bahasa Pemograman Java. *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, *8*(1), 6–9. https://doi.org/10.30743/infotekjar.v8i1.9641

Fadhilah, M., & Ramdani, F. (2025). Efisiensi Algoritma Sorting Dalam Aplikasi E-Commerce: Studi Kasus Pengurutan Produk Terlaris. *Jurnal Teknologi Digital dan Komputer*, *7*(2), 67–75.

Iskandar, J., Suhendar, H., & Pamungkas, B. D. (2024). Analisis Strategi Algoritma Sorting Menggunakan Metode Komparatif pada Bahasa Pemrograman Java dengan Python. *G-Tech: Jurnal Teknologi Terapan*, *8*(1), 104–113. https://doi.org/10.33379/gtech.v8i1.3556

Nasution, R., Syahputra, A., Widiyanto, A., Subuhanto, D., & Abdillah, A. Y. (2023). Persepsi Mahasiswa Informatika Terhadap Keefektifan Algoritma Bubble Sort dalam Mengurutkan Data. *Blend Sains Jurnal Teknik*, *1*(3 SE-Teknik Informatika), 220–225. https://doi.org/10.56211/blendsains.v1i3.186

Pujiono, I. P., Rachmawanto, E. H., & Winarsih, N. A. S. (2025). Array Sorting Algorithm vs Traditional Sorting Algorithm: Memory and Time Efficiency Analysis: Array Sorting Algorithm vs Algoritma Pengurutan Tradisional: Analisis Efisiensi Memori dan Waktu. *Jurnal Manajemen Informatika (JAMIKA)*, *15*(1), 47–59. https://doi.org/10.34010/jamika.v15i1.13230

Rahmat, A., & Utami, S. (2024). Penerapan Algoritma Sederhana Dalam Konteks

Pembelajaran Dasar Pemrograman. *Jurnal Pendidikan Teknologi Informasi*, *8*(3), 88–96.

Wijaya, R., & Cahyani, L. (2024). Studi Komparatif Algoritma Insertion Sort Dan Merge Sort Dalam Pengolahan Data Transaksi Digital. *Jurnal Ilmu Komputer dan Aplikasi*, *11*(1), 25–33.