Prosiding Seminar Nasional Sains dan Teknologi Seri 02
Fakultas Sains dan Teknologi, Universitas Terbuka
Vol. 1 No. 2 (2024)
e-ISSN: 3047-6569

PERBANDINGAN KINERJA ALGORITMA BUBBLE SORT, INSERTION SORT, DAN SELECTION SORT MENGGUNAKAN DATA BILANGAN NUMERIK PADA PROGRAM PYTHON

Akmal Isyqi, Gita Dwi Astuti, Adityo Dwi Saputro, Muhammad Adli Barryananda, Suharson*
Program Studi Teknik Informatika, Politeknik Negeri Pontianak, Kota Pontianak

**Penulis korespondensi: suhar2006@gmail.com*

ABSTRAK

Pengurutan merupakan suatu proses mengurutkan suatu data sehingga data tersebut dapat tersusun secara teratur. Pengurutan tersebut bertujuan untuk mengurutkan data, baik secara menaik (*ascending*) maupun menurun (*descending*). Terdapat tiga algoritma pengurutan data yaitu *Bubble sort, Insertion sort,* dan *Selection sort* yang akan digunakan pada penelitian ini. Setiap algoritma pengurutan memiliki kelebihan masing - masing dalam hal eksekusi waktu. Dalam penelitian ini akan membandingkan perbandingan terhadap tiga algoritma tersebut yang diimplementasikan menggunakan bahasa pemrograman Python. Data uji yang digunakan adalah angka acak dengan jumlah data 50, 100, 250, 300, 500 dan 650. Pengukuran waktu eksekusi diukur dalam satuan detik dengan menggunakan library *time*. Penelitian ini bertujuan untuk membandingkan kinerja dan tingkat efisiensi waktu dari ketiga algoritma *Bubble sort, Insertion sort,* dan *Selection sort* dalam mengurutkan data. Hasil dari penelitian menunjukkan bahwa pada data angka acak 650 elemen, *Bubble sort* memiliki waktu eksekusi 95,6 detik, *Insertion sort* memiliki waktu eksekusi 0,61 detik dan *Selection sort* memiliki waktu eksekusi 0,63 detik. Dari hasil penelitian yang telah dilakukan, algoritma *Bubble sort* memiliki kinerja yang kurang efisien dengan waktu eksekusi terlama. *Insertion sort* memiliki kinerja dan waktu eksekusi yang lebih baik dibandingkan *Bubble sort*. *Selection sort* memiliki kinerja yang hampir sama dengan *Bubble sort* dengan eksekusi waktu yang hampir sama dengan *Insertion sort.* Dalam hal ini algoritma *Insertion sort* memiliki kinerja yang lebih cepat dan waktu eksekusi yang lebih efisien dibandingkan *Bubble sort* dan *Insertion sort*.

Kata kunci: pengurutan, *selection sort, insertion sort, bubble sort, python*

1. PENDAHULUAN

Dalam pemrograman, algoritma memegang peranan penting dan tidak dapat dipisahkan. Dibutuhkan algoritma yang baik dalam penyelesaian sebuah masalah agar dapat terselesaikan dengan baik. Pengembangan algoritma yang efisien memegang peranan penting dalam pemrograman karena secara langsung mempengaruhi kualitas dan kinerja dari suatu program. Dalam memilih algoritma, terdapat beberapa faktor yang perlu dipertimbangkan, yakni kompleksitas waktu dan ruang. Kompleksitas waktu mengacu pada seberapa cepat algoritma tersebut berjalan seiring dengan peningkatan ukuran. Sedangkan kompleksitas ruang mengacu pada seberapa banyak memori yang dibutuhkan oleh algoritma untuk menyelesaikan masalah. Tidak hanya itu kestabilan dan kemudahan implementasi juga perlu dipertimbangkan. Dengan menggunakan algoritma yang baik dapat menghasilkan program yang efisien dari segi kompleksitas waktu dan ruang. Pengurutan data merupakan salah satu contoh penerapan yang sering digunakan dalam pemrograman.

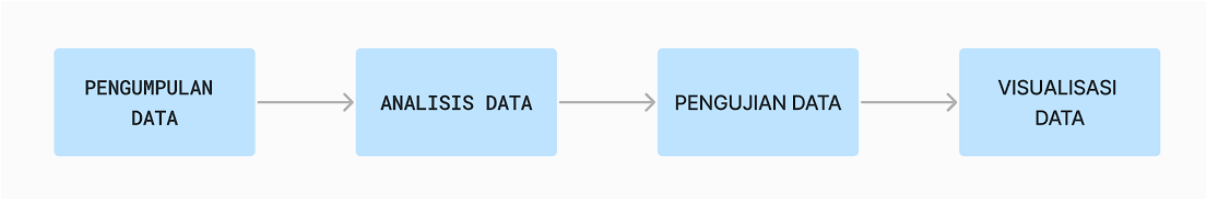
Pengurutan data atau biasa yang disebut dengan *sorting* merupakan proses penyusunan data pada suatu pola dengan cara penyusunan data sesuai dengan urutan tertentu yang dilakukan

penyortiran pada suatu data yang masih acak. Banyak jenis algoritma sorting yang telah dikembangkan dan ditingkatkan agar pengurutan data lebih cepat. Terdapat beberapa algoritma sorting pada python yang cukup populer dalam pengurutan data diantaranya adalah *Bubble sort, Selection sort, Insertion sort, Quick sort, Merge sort, Heap sort, Shell sort, Counting sort, Comb sort, Bucket sort, Radix sort.* Secara umum ada dua jenis pengurutan data yaitu pengurutan data yang pengurutannya dimulai dari nilai yang paling kecil hingga yang paling besar (*ascending)* dan pengurutan data yang pengurutannya disusun dari data yang paling besar sampai paling kecil (*descending*). Pada karya ilmiah ini akan menghasilkan perbandingan dari segi waktu pengurutan data random secara acak dengan menggunakan algoritma yaitu *Bubble sort, Selection sort* dan *Insertion sort.*

Bubble sort merupakan algoritma yang bekerja dengan membandingkan pasangan elemen berdekatan dalam daftar dan menukarkan posisinya jika elemen-elemen tersebut tidak dalam urutan yang benar, proses ini berulang – ulang hingga data yang acak sudah terurut. *Insertion sort* adalah salah satu Algoritma sorting dengan penggunaan paling sederhana, metode ini juga sering disebut sebagai metode pertengahan. Artinya, metode ini memiliki kecepatan rata - rata antara metode *Bubble sort* dan *Selection sort*. Cara kerja dari metode insertion adalah mengurutkan elemen bagian kiri hingga seluruh elemen berhasil diurutkan. *Selection sort* adalah salah satu algoritma pengurutan sederhana dengan cara memilih elemen terkecil dari data dan menukarnya dengan elemen pertama. Kemudian akan memilih elemen terkecil dari sisa data dan menukarnya dengan elemen kedua hingga seluruh data terurut.

2. METODE

Studi literatur adalah teknik untuk mendapatkan informasi dengan membaca dan mempelajari literatur dari berbagai sumber, seperti buku, jurnal, dan referensi internet. Tujuan dari metode ini adalah untuk mendapatkan pemahaman yang mendalam tentang berbagai macam algoritma pengurutan terkait dengan penelitian ini. Langkah ini penting untuk memastikan bahwa algoritma yang dipilih dalam penelitian memiliki landasan teori yang kuat dan relevan dengan permasalahan yang dibahas sehingga dapat membantu memecahkan masalah.



Gambar 1. Alur proses metode penelitian

2.1 Pengumpulan Data

Pada penelitian ini, data yang dikumpulkan merupakan data acak dengan jumlah data 50, 100, 250, 300, 500, dan 650. Data acak akan dihasilkan dengan menggunakan *library random* dan akan disimpan untuk digunakan dalam pengujian algoritma pengurutan. Proses pengumpulan data dilakukan dengan teliti untuk memastikan bahwa data mewakili berbagai ukuran dataset. Setiap langkah dalam proses ini dirancang untuk menjaga keakuratan data, sehingga hasil pengujian algoritma pengurutan sesuai dengan yang diharapkan.

2.2 Analisis Data

Tahap analisis melibatkan peninjauan data angka acak yaitu, memeriksa data acak yang telah dikumpulkan, memastikan bahwa data telah mencakup ukuran dataset yang bervariasi. Proses analisis data mencakup verifikasi dataset untuk memastikan bahwa dataset yang telah

ditentukan benar – benar acak, implementasi, dan penerapan pada algoritma pengurutan yang telah ditentukan pada setiap dataset. Setiap algoritma dianalisis pada semua ukuran dataset untuk mengukur kinerja masing – masing algoritma. Selain itu, penting untuk memastikan bahwa setiap dataset disimpan dengan benar agar dapat digunakan dalam pengujian berulang tanpa adanya perubahan atau kehilangan data.

2.3 Pengujian Data

Pengujian data menggunakan teknik pengukuran waktu eksekusi. Waktu eksekusi diukur menggunakan fungsi waktu dengan *library time* pada python. Proses pengujian meliputi inisialisasi pengukuran waktu dengan menyiapkan fungsi untuk mengukur waktu eksekusi dari setiap algoritma dan replikasi pengujian beberapa kali untuk memastikan konsistensi hasil. Selain itu, dilakukan juga analisis terhadap variasi waktu eksekusi untuk setiap algoritma pada setiap ukuran data set, sehingga dapat diperoleh gambaran yang jelas mengenai efisiensi waktu dari setiap algoritma.

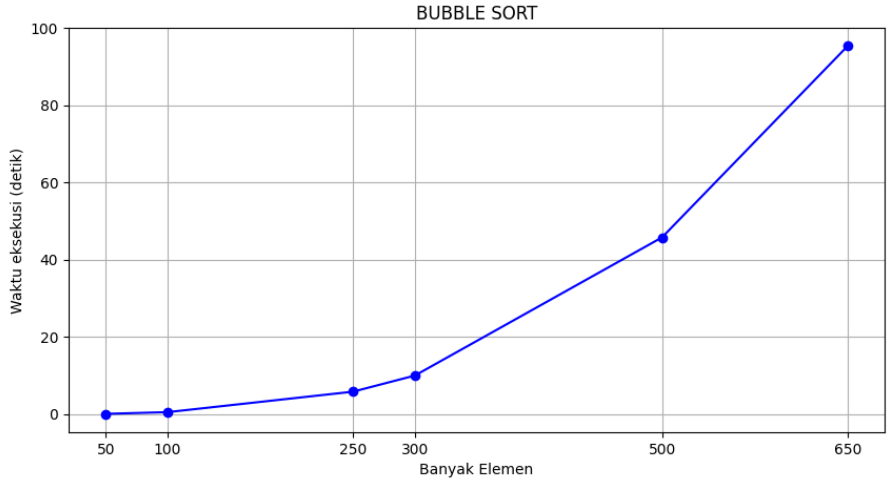
2.4 Visualisasi Data

Visualisasi data dilakukan dengan tujuan untuk memberikan gambaran mengenai efisiensi waktu dari masing – masing algoritma pengurutan dengan jelas dan mudah untuk dipahami. Dalam penelitian ini, visualisasi data menggunakan *library Matplotlib*. Langkah – langkah dalam proses visualisasi data meliputi, pengumpulan hasil pengukuran waktu eksekusi dari setiap ukuran dataset dari masing – masing algoritma dan pembuatan grafik perbandingan menggunakan *Mathplotlib.* Grafik yang dihasilkan digunakan untuk menarik kesimpulan mengenai efisiensi dan kinerja dari masing – masing algoritma, serta untuk mendukung kesimpulan penelitian.

3. HASIL DAN PEMBAHASAN

Setiap algoritma memiliki efisiensi waktu yang berbeda dan akan terlihat pada hasil pengujian ini. Kinerja dari masing – masing algoritma meningkat seiring bertambahnya dataset, serta membandingkan efisiensi dalam konteks waktu eksekusi.

3.1 *Bubble Sort*

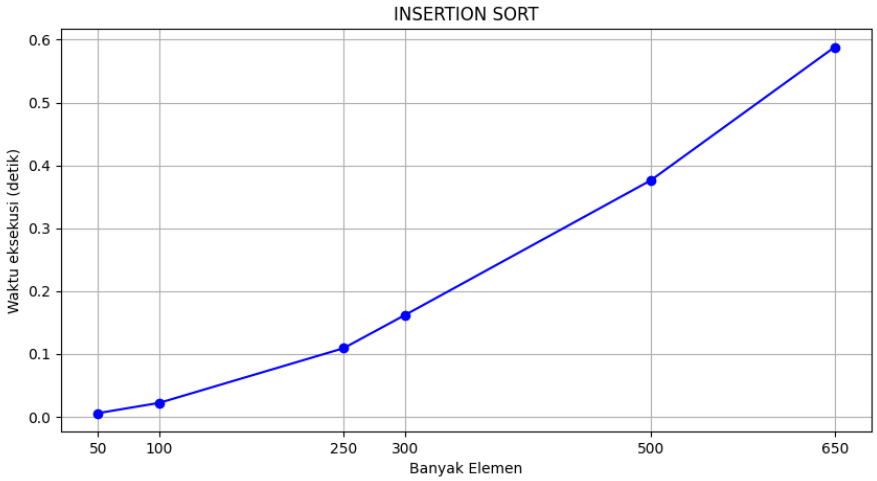


Gambar 2. Grafik *Bubble sort*

Berdasarkan grafik pada **Gambar 2**, menunjukkan jumlah waktu yang dibutuhkan algoritma *Bubble sort* untuk mengurutkan dataset dengan berbagai ukuran 50, 100, 250, 300, 500, dan

650 elemen. Dari grafik ini, terlihat bahwa waktu eksekusi *Bubble sort* meningkat seiring dengan bertambahnya ukuran dataset. Pada dataset dengan 50 elemen, waktu eksekusi sekitar 0,09 detik, tetapi pada dataset dengan 650 elemen, waktu eksekusi meningkat secara drastis menjadi sekitar 95,65 detik. Peningkatan yang signifikan ini menunjukkan bahwa *Bubble sort* kurang efisien untuk dataset besar karena banyaknya perbandingan dan pertukaran yang harus dilakukan, sesuai dengan kompleksitas waktu O(n^2) yang dimilikinya.

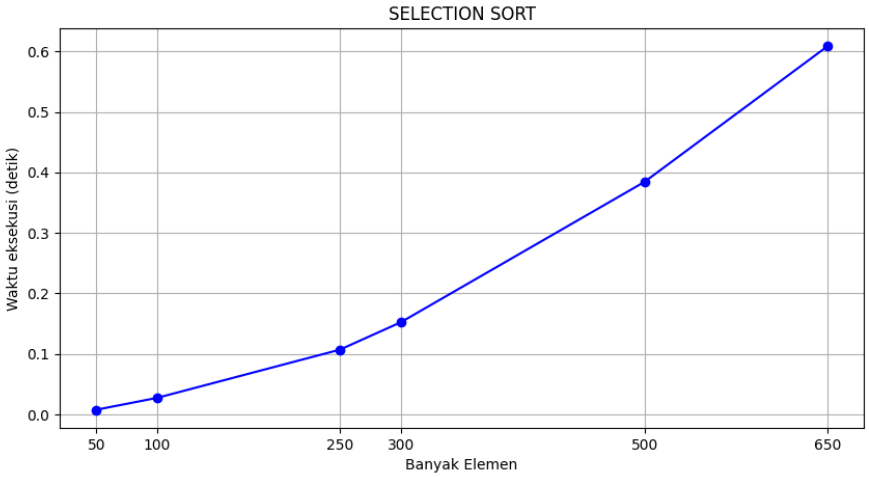
3.2 *Insertion Sort*



Gambar 3. Grafik *Insertion sort*

Berdasarkan grafik pada **Gambar 3**, menunjukkan waktu eksekusi algoritma *Insertion sort* untuk berbagai ukuran dataset. Waktu eksekusi *Insertion sort* meningkat seiring dengan bertambahnya ukuran dataset, tetapi meningkat lebih sedikit dibandingkan dengan *Bubble sort*. Untuk dataset dengan 50 elemen, waktu eksekusi sekitar 0,007 detik, dan untuk *dataset* dengan 650 elemen, waktu eksekusi sekitar 0,62 detik. *Insertion sort* memiliki kompleksitas waktu O(n^2) dalam kasus terburuk, tetapi dalam praktiknya sering kali lebih cepat, terutama untuk dataset yang hampir terurut. Grafik menunjukkan bahwa *Insertion sort* bekerja lebih baik dan lebih konsisten dibandingkan dengan *Bubble sort*, terutama pada dataset yang lebih besar.

3.3 *Selection Sort*



Gambar 4. Grafik *Selection sort*

Berdasarkan grafik pada Gambar 4, menunjukkan waktu eksekusi algoritma *Selection sort* untuk berbagai ukuran dataset. Waktu eksekusi *Selection sort* juga meningkat seiring bertambahnya ukuran dataset, tetapi peningkatannya lebih lambat dibandingkan dengan *Bubble sort*. Sebagai contoh, untuk dataset dengan 50 elemen, waktu eksekusi sekitar 0,006 detik, dan untuk dataset dengan 650 elemen, waktu eksekusi sekitar 0,63 detik. Meskipun *Selection sort* juga memiliki kompleksitas waktu O(n^2), algoritma ini lebih efisien dalam praktik dibandingkan dengan *Bubble sort* karena pertukaran data yang dilakukan lebih sedikit. Grafik menunjukkan peningkatan waktu eksekusi yang lebih stabil dan tidak secepat *Bubble sort*.

3.4 Perbandingan Waktu Eksekusi

Tabel 1. Perbandingan waktu eksekusi algoritma pengurutan (detik)

Banyak sampel	*Bubble sort*	*Selection sort*	*Insertion sort*
50	0.091944	0.006289	0.007111
100	0.447671	0.018664	0.018802
250	5.953263	0.115436	0.096004
300	10.263618	0.140338	0.152236
500	56.660067	0.530602	0.526162
650	95.654014	0.630618	0.618893

Pada **Tabel 1**, menunjukkan waktu eksekusi dari tiga algoritma pengurutan dengan berbagai sampel dataset. Berdasarkan tabel di atas, terlihat bahwa eksekusi waktu pada *Bubble sort* meningkat secara drastis dengan bertambahnya sampel, dari 0,091944 detik untuk 50 sampel menjadi 95,654014 detik untuk 650 sampel. Sebaliknya, *Selection sort* dan *Insertion sort* mengalami peningkatan waktu eksekusi yang lebih lambat. Misalnya, *Selection sort* membutuhkan 0,006289 detik untuk 50 sampel dan 0,630618 detik untuk 650 sampel. *Insertion sort* menunjukkan pola yang mirip dengan *Selection sort*, dengan waktu eksekusi 0,007111 detik untuk 50 sampel dan 0,618893 detik untuk 650 sampel.
Perbandingan ini menunjukkan bahwa untuk dataset yang lebih besar, *Bubble sort* jauh lebih lambat dibandingkan dengan *Selection sort* dan *Insertion sort*. Kedua algoritma ini lebih efisien dan memiliki peningkatan waktu eksekusi yang lebih stabil seiring bertambahnya ukuran dataset.

4. KESIMPULAN

Dari hasil penelitian yang telah dilakukan, dapat diketahui bahwa metode pengurutan menggunakan *Bubble sort* memiliki fungsi yang lebih sederhana dan mudah untuk dipahami namun memiliki efisiensi waktu yang lama. Kemudian pengurutan data menggunakan *Insertion sort* memiliki efisiensi waktu yang hampir sama dengan metode *Selection sort* akan tetapi metode Insertion sort memiliki efisiensi waktu tercepat. Dapat disimpulkan bahwa penggunaan metode *Insertion sort* dalam pengurutan data acak lebih efisien dibandingkan dengan penggunaan metode *Bubble sort*.

UCAPAN TERIMAKASIH

Penelitian ini dapat berjalan dengan baik, karena adanya bantuan dari berbagai pihak yang terlibat di dalamnya. Oleh karena itu, kami mengucapkan terimakasih yang sebesar-besarnya kepada pihak-pihak yang telah membantu kami dalam mengerjakan karya ilmiah tersebut. Kami menyadari bahwa tanpa adanya bantuan dari berbagai pihak, penelitian ini tidak dapat berjalan dengan lancar dan semestinya.

DAFTAR PUSTAKA

Abdullah, M. F., Hafiza, I., Wahyuni, R., & Syahputra, A. (2023). Penggunaan Algoritma *Bubble sort* dalam Pengurutan Nomor Induk Mahasiswa. *Hello World Jurnal Ilmu Komputer*, *2*(1), 14–19. https://doi.org/10.56211/helloworld.v2i1.206

Aryanto, R. P., Nilogiri, A., & Wardoyo, A. E. (2023). Optimasi Pengurutan Data Bilangan dengan Menggabungkan Algoritma Selection Sort Hybrid dan Bucket Sort. *Edumatic: Jurnal Pendidikan Informatika*, *7*(1), 39–48. https://doi.org/10.29408/edumatic.v7i1.12358

Astuti, Y. A. (2023). Analisis Pengujian Data Algoritma *Bubble sort*. *REMIK: Riset dan E-Jurnal Manajemen Informatika Komputer*, *7*(3), 1413–1420. https://polgan.ac.id/jurnal/index.php/remik/article/view/12594

Esau Taiwo, O., Christianah, A. O., Oluwatobi, A. N., Aderonke, K. A., & Kehinde, A. J. (2020). Comparative Study Of Two Divide And Conquer Sorting Algorithms: Quicksort and Mergesort. *Procedia Computer Science*, *171*(2019), 2532–2540. https://doi.org/10.1016/j.procs.2020.04.274

Luthfi Zulfa, M., Nurina Sari, B., & Singaperbangsa Karawang Abstract, U. (2022). Analisis Perbandingan Algoritma *Bubble sort*, Shell Sort, dan Quick Sort dalam Mengurutkan Baris Angka Acak menggunakan Bahasa Java. *Jurnal Ilmiah Wahana Pendidikan*, *2022*(13), 237–246. https://doi.org/10.5281/zenodo.6962346

Mahrozi, N., & Faisal, M. (2023). Analisis Perbandingan Kecepatan Algoritma Selection Sort Dan *Bubble sort*. *Jurnal Ilmiah Sain dan Teknologi*, *1*(2), 89–98.

Saputri, *, Pemograman, D., Seleksi, B., & Saputri, P. (2023). Analisis Study Komperatif *Bubble sort* Dan Selection Sort Pada Algoritma. *Bahasa dan Matematika*, *1*(6), 151–161. https://doi.org/10.61132/arjuna.v1i6.305

Sari, N., Gunawan, W. A., Sari, P. K., Zikri, I., & Syahputra, A. (2022). Analisis Algoritma *Bubble sort* Secara Ascending Dan Descending Serta Implementasinya Dengan Menggunakan Bahasa Pemrograman Java. *ADI Bisnis Digital Interdisiplin Jurnal*, *3*(1), 16–23. https://doi.org/10.34306/abdi.v3i1.625

Setyantoro, D., & Hasibuan, R. A. (2020). Analisis Dan Perbandingan Kompleksitas Algoritma Exchange Sort Dan Insertion Sort Untuk Pengurutan Data Menggunakan Python. *Tekinfo*, *21*(1), 48–56. https://journals.upi-yai.ac.id/index.php/TEKINFO/article/view/1139

Sunandar, E. (2021). Implementation Of *Bubble sort* Algorithm On 2 Fruit Models Of Data Selection Using The Java Program Language. *Petir*, *14*(2), 159–169. https://doi.org/10.33322/petir.v14i2.946