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 SORTING DALAM PENGURUTAN DATA MENGGUNAKAN BAHASA PYTHON

Muhammad Bima Sena[*], Rani Mumtazah Hanun, Irvan Reki Purnama, Muhammad Ardian, Suharsono
Program Studi Teknik Informatika, Politeknik Negeri Pontianak

[*]*Penulis Korespodensi*: *bimasenatab3v@gmail.com*

ABSTRAK

Pengolahan data pemilihan algoritma sorting yang tepat sangat penting untuk efisiensi dan kinerja sistem. Terdapat banyak algoritma sorting yang dapat digunakan untuk menyelesaikan permasalahan pengurutan. Tujuan penelitian ini adalah untuk membandingkan kinerja tiga algoritma sorting yaitu: bubble sort, insertion sort, dan quick sort yang diimplementasikan dalam bahasa Python. Data yang akan dibandingkan berupa dataset susunan acak dengan jumlah data uji adalah 50, 100, 250, 500, 1000, dan 2000 elemen. Penelitian ini membandingkan tiga algoritma yaitu bubble sort, insertion sort, dan quick sort untuk mengetahui waktu eksekusi dan jumlah langkah yang diperlukan untuk mengurutkan dataset yang diimplementasikan dalam bahasa Python. Berdasarakan hasil pengujian menunjukkan bahwa algoritma quick sort memiliki waktu eksekusi terbaik yaitu 0.0142 detik pada semua ukuran data yang diuji karena kompleksitas rata-ratanya yang lebih rendah, yaitu O(n log n). Insertion sort menunjukkan kinerja yang lebih baik daripada bubble sort pada dataset kecil dan sebagian besar terurut, namun akan kalah jika dibandingkan quick sort untuk dataset besar karena kompleksitas waktu O(n). Bubble sort memiliki kinerja paling lama, dengan waktu eksekusi yang meningkat signifikan seiring bertambahnya ukuran dataset, karena kompleksitas waktu O(n^2). Kesimpulan dapat dinyatakan bahwa algoritma quick sort yang paling cepat dalam mengurutkan data, terutama untuk dataset besar dari ketiga algoritma yang uji. Sedangkan algoritma insertion sort mampumengurutkan data paling cepat pada dataset kecil ini rencang berapa dari 50, 100, 250, 500, 1000, dan 2000 elemen. Algoritma bubble sort menjadi algoritma yang paling mudah diterapkan namun kurang efisien dalam mengurutkan data.

Kata kunci: algoritma, *sorting, python, bubble sort, insertion sort, quick sort*

1 PENDAHULUAN

Sorting atau pengurutan adalah proses penyusunan elemen dalam urutan tertentu, biasanya secara menaik atau menurun. Algoritma *sorting* memainkan peran penting dalam berbagai aplikasi, mulai dari pengolahan data sederhana hingga analisis data yang kompleks. Beberapa algoritma *sorting* memiliki performa yang lebih baik daripada yang lain, tergantung pada ukuran dan karakteristik data yang diolah. Penelitian ini memfokuskan pada perbandingan tiga algoritma *sorting* yaitu *Bubble Sort*, *Quick Sort*, dan *Merge Sort*. *Bubble Sort* adalah algoritma sederhana yang sering digunakan sebagai contoh dasar dalam pembelajaran pemrograman. *Quick Sort* adalah algoritma yang lebih kompleks dan efisien, menggunakan pendekatan *divide-and-conquer*. *Merge Sort* adalah algoritma yang sederhana namun efisien untuk dataset kecil atau yang sebagian besar sudah terurut. Tujuan dari penelitian ini adalah untuk mengukur dan membandingkan kinerja dari ketiga algoritma tersebut berdasarkan waktu eksekusi dan jumlah langkah yang diperlukan untuk mengurutkan data acak dengan berbagai ukuran. Hasil

dari penelitian ini diharapkan dapat memberikan panduan praktis bagi pengembang dalam memilih algoritma *sorting* yang paling sesuai untuk kebutuhan spesifik mereka.

2 METODE

2.1 Bubble Sort

Bubble Sort adalah algoritma pengurutan sederhana yang bekerja dengan cara membandingkan elemen-elemen yang bersebelahan dan menukar posisinya jika diperlukan, sehingga elemen terbesar 'menggelembung' ke posisi akhir daftar. Proses ini diulangi hingga daftar terurut sepenuhnya. Kompleksitas waktu: O(n^2). Berikut adalah cara kerja dari *Bubble Sort*:

1. Mulai dari elemen pertama dalam array.
2. Bandingkan setiap elemen dengan elemen berikutnya. Jika elemen yang sekarang lebih besar daripada elemen berikutnya, maka tukar keduanya.
3. Lanjutkan ke elemen berikutnya dan ulangi proses perbandingan dan pertukaran hingga akhir array.
4. Jika sudah mencapai akhir *array*, mulai lagi dari elemen pertama dan ulangi prosesnya. Dan jumlah iterasi yang diperlukan adalah Panjang array dikurangi satu.
5. Jika dalam satu iterasi tidak ada pertukaran yang terjadi, berarti array sudah terurut, dan algoritma bisa berhenti lebih awal.

2.2 Quick Sort

Quick Sort adalah algoritma divide-and-conquer yang memilih satu elemen sebagai pivot dan mempartisi elemen-elemen lain ke dalam dua sub-daftar berdasarkan apakah elemen tersebut lebih kecil atau lebih besar dari pivot. Algoritma ini kemudian mengurutkan sub-daftar tersebut secara rekursif. Kompleksitas waktu: O (n log n) rata-rata, O(n^2) dalam kasus terburuk.Berikut adalah cara kerja dari Quick Sort:

1. Pilih Pivot: Pilih salah satu elemen dari array untuk dijadikan sebuah pivot
2. Partisi: Atur elemen – elemen array hingga semua elemen yang lebih kecil dari pivot berada di sebelah kiri pivot,dan semua elemen yang lebih besar dari pivot berada di sebelah kanan pivot.
3. Rekursi: Terapkan proses yang sama pada sub-array di sebelah kiri dan kanan pivot.

2.3 Merge Sort

Merge Sort adalah algoritma pengurutan sederhana yang bekerja dengan cara melakukan pendekatan divide dan conquer, untuk mengurutkan sebuah array atau daftar dengan cara membagi masalah menjadi masalah yang lebih kecil, lalu menyelesaikan masing – masing masalah, dan menggabngkan hasilnya untuk mendapatkan solusi akhir.Berikut adalah cara kerja dari *merge sort* :

1. *Divide*: Pisahkan *array* menjadi dua bagian yang kurang lebih sama besar.
2. *Conquer*: Urutkan masing – masing bagian secara rekursif menggunakan *merge sort.*
3. *Combine*: Gabungkan kedua bagian yang sudah diurutkan menjadi satu *array* yang terurut.

2.3.1 Data

Untuk menguji performa algoritma, data acak dihasilkan untuk setiap ukuran dataset yang akan diuji: 50, 100, 250, 500, 1000, dan 2000 elemen. Dataset ini dirancang untuk mencerminkan berbagai skenario yang mungkin ditemui dalam aplikasi nyata.

2.3.2 Pengukuran

Pengukuran dilakukan untuk menilai waktu eksekusi dan jumlah langkah yang diperlukan oleh setiap algoritma. Waktu eksekusi diukur menggunakan modul time di Python, sementara jumlah langkah dihitung berdasarkan iterasi dan rekursi yang dilakukan oleh setiap algoritma. Pengukuran ini memberikan gambaran tentang efisiensi setiap algoritma dalam kondisi praktis.

2.3.3 Implementasi

Setiap algoritma diimplementasikan dalam Python dan dijalankan pada perangkat komputer yang sama untuk memastikan konsistensi hasil.

3 HASIL DAN PEMBAHASAN

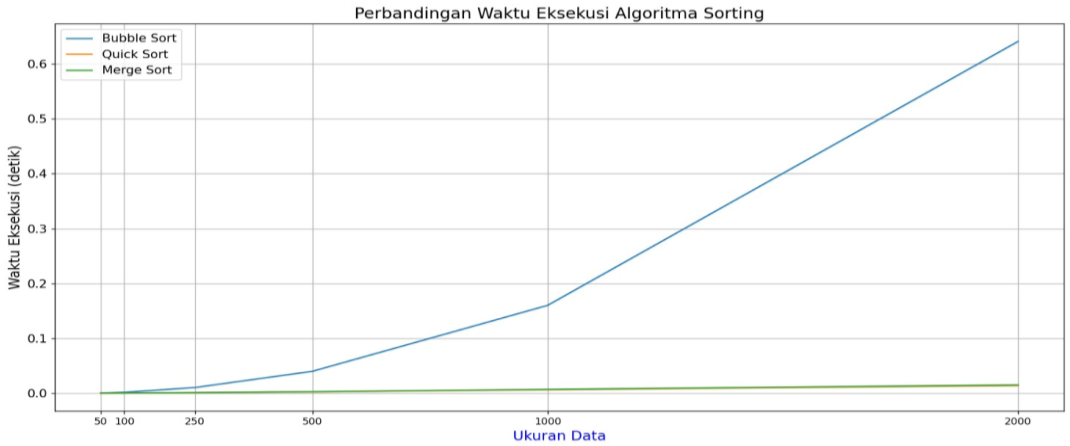
Hasil eksperimen ditunjukkan dalam **Tabel 1** dan **Tabel 2**, serta grafik pada **Gambar 1**.

Tabel 1. Lama Waktu Pengurutan Tiap Algoritma

Ukuran Data	Lama Waktu Pengurutan (detik)		
	Bubble Sort	*Quick Sort*	*Merge Sort*
50	0.0005	0.0002	0.0003
100	0.0018	0.0003	0.0005
250	0.0105	0.0011	0.0012
500	0.0401	0.0027	0.0029
1000	0.1602	0.0065	0.0071
2000	0.6408	0.0142	0.0153

Tabel 2. Jumlah Langkah Yang Diperlukan Tiap Algoritma

Ukuran Data	Jumlah Langkah Yang Diperlukan		
	Bubble Sort	*Quick Sort*	*Merge Sort*
50	1225	64	150
100	4950	159	365
250	31125	611	950
500	124750	1375	2000
1000	499500	3060	4200
2000	1999000	6450	8900



Gambar 1. Grafik perbandingan

3.1 Waktu Eksekusi

Pengukuran kinerja dilakukan dengan mencatat waktu eksekusi dan jumlah langkah yang diperlukan oleh masing-masing algoritma. Hasil penelitian menunjukkan bahwa *Quick Sort* memiliki waktu eksekusi terbaik pada semua ukuran data yang diuji. *Bubble Sort* menunjukkan kinerja yang paling lambat, terutama pada dataset yang lebih besar. *Merge Sort* memberikan performa yang lebih baik daripada *Bubble Sort* pada dataset yang lebih kecil dan sebagian besar terurut, namun masih kalah efisien dibandingkan *Quick Sort* untuk dataset besar. Penelitian ini memberikan panduan bagi pengembang perangkat lunak dalam memilih algoritma sorting yang sesuai dengan kebutuhan mereka.

4. KESIMPULAN

Hasil penelitian menunjukkan bahwa *Quick Sort* memiliki waktu eksekusi terbaik pada semua ukuran data yang diuji, diikuti oleh *Insertion Sort*. *Bubble Sort* menunjukkan kinerja yang paling lambat, terutama pada dataset yang lebih besar. *Quick Sort* dan *Merge Sort* juga menunjukkan jumlah langkah yang lebih sedikit dibandingkan *Bubble Sort,* menjadikan kedua algoritma ini lebih efisien untuk dataset yang lebih besar. Penelitian ini memberikan panduan bagi pengembang perangkat lunak dalam memilih algoritma *sorting* yang sesuai dengan kebutuhan mereka.

DAFTAR PUSTAKA

Darmadi, D., Diansyah, T. M., & Handoko, D. (2023). Penerapan Algoritma Floyd Warshall dengan Menggunakan Euclidean Distance dalam Menentukan Rute Terbaik. *Jurnal Ilmu Komputer dan Sistem Informasi*, *2*(2), 311-321.

Daulay, P. I., & Yahfizham, Y. (2023). Penerapan Algoritma Pemrograman dalam Pembelajaran Ilmu Komputer. *Jurnal Arjuna: Publikasi Ilmu Pendidikan, Bahasa dan Matematika*, *1*(6), 91-103.

Eriana, E. S., & Zein, A. (2023). Artificial Intelligence (AI). Farizy, S., & Eriana, E. S. (2022). Keamanan Sistem Informasi.

Hartawan, S., & Yanti, F. (2022). Sistem Informasi Pencarian JudulSkripsi Teknik Informatika Berbasis Web (Studi Kasus Prodi Ti Universitas Pamulang). *OKTAL: Jurnal Ilmu Komputer dan Sains*, *1*(10), 1666-1673.

Mahrozi, N., & Faisal, M. (2023). Analisis perbandingan kecepatan algoritma selection sort dan bubble sort. *Scientica: Jurnal Ilmiah Sains dan Teknologi*, *1*(2), 89-98.

Nasution, A., & Siddik, M. (2023). Implementasi Algoritma Binary Search Pada Aplikasi Kamus Indonesia- Inggris Berbasis Android. *Journal Of Science and Social Research*, *6*(3), 711- 716.

Nasution, R., Syahputra, A., Widiyanto, A., Subuhanto, D., & Abdillah, A. Y. (2023). Persepsi Mahasiswa Informatika Terhadap Keefektifan Algoritma Bubble Sortdalam Mengurutkan Data. *Blend Sains Jurnal Teknik*, *1*(3), 220-225.

Putra, J. D., Ananda, M. R., Syahputra, A., Nurshabillah, N., & Sinaga, S. P. (2023). Pengurutan Menggunakan Metode Selection Sort pada Sistem Informasi Lokasi Kontrakan di Kota Medan Berbasis Android. *Blend Sains Jurnal Teknik*, *1*(4), 259-266.

Sunjarwanto, R., & Zailani, A. U. (2023). Aplikasi Pemesanan Peti Mati Berbasis Android Menggunakan Algoritma Sequantial Search. Jurnal Informatika Multi, 1(2), 137-143.

Syafri, Y., & Halim, R. N. (2021). Aplikasi Kamus Bahasa Indonesia– Palembang Menggunakan Algoritma Interpolation Search Berbasis Android. Jurnal Nasional Ilmu Komputer, 2(1), 19-32.

Sandria, Y. A., Nurhayoto, M. R. A., Ramadhani, L., Harefa, R. S., & Syahputra, A. (2023). Penerapan Algoritma Selection Sort untuk Melakukan Pengurutan Data dalam Bahasa Pemrograman PHP. Hello World Jurnal Ilmu Komputer, 1(4), 190-194.

Saputri, S., & Yahfizham, Y. (2023). Analisis Study Komperatif Bubble Sort Dan Selection Sort Pada Algoritma Dan Pemograman Berdasarkan Seleksi Pengurutan. Jurnal Arjuna: Publikasi Ilmu Pendidikan, Bahasa dan Matematika, 1(6), 151-161.

Zein, A., & Eriana, E. S. (2022). Algoritma Dan Struktur Data.