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

PERBANDINGAN ALGORITMA *SELECTION SORT*, *SHELL SORT*, DAN *MERGE SORT* PADA DATA SAMPLING NUMERIK MENGGUNAKAN MATPLOTLIB

Riki, Miftah Faridz, Tri Syamdi Hidayah, Suharsono*
Program Studi Teknik Informatika, Politeknik Negeri Pontianak, Pontianak

**Penulis korespondensi: suhar2006@gmail.com*

ABSTRAK

Sorting merupakan proses pengurutan sejumlah data berdasarkan nilai kunci tertentu. Data setiap elemen dari sebuah larik (senarai) yang belum terurut menjadi urutan yang telah ditentukan. Pengurutan dapat dapat dilakukan dengan dua cara yaitu dari nilai terkecil ke nilai terbesar (*ascending*) atau sebaliknya (*descending*). Dalam kehidupan sehari-hari terdapat cara tertantu dalam melakukan pengurutan. Dalam pemrograman terdapat beberapa algoritma *sorting*, dimana masing-masing algoritma tersebut memiliki keunggulan masing-masing dilihat dari tingkat kompleksitas, skala data yang dikerjakan, hingga kecepatan pengurutan data. Pada penelitian ini menggunakan tiga algoritma *sorting*, yaitu *selection sort*, *merge sort*, dan *shell sort* yang akan membandingkan kecepatan pengurutan pada data numerik menggunakan bahasa pemrograman Python. Data yang akan diurutkan berupa data numerik yang urutannya teracak dan terurut. Jumlah data yang digunakan yaitu: 1000 sampai dengan 10000. Waktu eksekusi setiap algoritma sorting di ukur menggunakan *library timeit* dalam satuan *millisecond* (ms). Hasil perbandingan algoritma tersebut divisualisasikan dalam bentuk grafik menggunakan *library* Matplotlib. Pengujian pada data acak dengan 1000 sampel, *shell sort* lebih cepat dibandingkan *merge sort* dan *selection sort* dengan pengurutan 1.22 ms, namun pada data yang jumlah data yang besar (10000) *merge sort* lebih cepat dibandingkan *shell sort* dan *selection sort*. Pada data terurut menunjukkan hasil yang berbeda dengan data acak dimana *Shell Sort* lebih cepat dari *merge sort* dan *selection sort* dengan waktu eksekusi 0.59 ms untuk 1000 sampel, dan 10.62 ms untuk 10000 sampel. Dapat disimpulkan bahwa algoritma *Shell Sort* lebih cepat dalam mengurutkan data numerik baik data acak atau urut.

Kata kunci: *Sorting*, Python, Matplotlib, Algoritma, Perbandingan

1 PENDAHULUAN

Dalam ilmu komputer, istilah kata algoritma sering muncul pada setiap penyelesaian masalah. Algoritma adalah urutan langkah-langkah yang disusun secara sistematis dalam menyelesaikan masalah-masalah tertentu. Terdapat banyak jenis algoritma, salah satunya yaitu algoritma pengurutan data. Algoritma pengurutan atau yang biasanya disebut algoritma sorting digunakan untuk mengubah susunan elemen-elemen menjadi sesuai dengan urutan yang telah ditentukan. Elemen-elemen dapat diurutkan secara naik (*ascending*) maupun menurun (*descending*). Pengurutan berfungsi agar data-data dapat dikelola dengan mudah dan efisien. Terdapat banyak macam algoritma sorting antara lain, *bubble sort, selection sort, insertion sort, quick sort, shell sort, merge sort, heap sort, counting sort,* dan lain sebagainya. Setiap algoritma sorting memiliki keunggulan masing-masing baik dari segi kecepatan hingga kompleksitas ruang. Kecepatan pengurutan atau kompleksitas waktu adalah salah satu kriteria untuk menentukan algoritma mana yang dapat dikatakan bagus dan efisien. Python adalah bahasa pemrograman multiguna dengan perancangan yang berfokus pada tingkat keterbacaan kode. Sebagai bahasa pemrograman, python tentu memiliki kosa kata atau kata kunci *(Keyword*), dan aturan tersendiri yang jelas berbeda dengan bahasa pemrograman lain. Python diklaim sebagai bahasa

pemrograman yang menggabungkan kapabilitas, kemampuan, dengan sintaks kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustakan standar yang besar serta komprehensif (David, 2019). Bahasa pemrograman python merupakan bahasa pemrograman tingkat tinggi yang banyak digunakan untuk analisis data karena terdapat banyak pustaka yang tersedia untuk membantu dalam pengolahan data, seperti Pandas, NumPy, Matplotlib, Seaborn, dan lain-lain. Python juga menyediakan pustaka untuk pengukuran waktu seperti time, timeit, cProfile dan profile yang sering digunakan untuk keperluan benchmarking, profilling, serta pengukuran waktu eksekusi. Kompleksitas waktu adalah waktu yang dibutuhkan oleh algoritma untuk menjalankan dan menyelesaikan tugas sementara stabilitas memeriksa apakah suatu algoritma mempertahankan urutan data input dan output sebelum dan sesudah penyortiran (Esau Taiwo dkk., 2020).

Pada penelitian ini hanya digunakan tiga jenis algoritma sorting, yaitu *selection sort*, *merge sort*, dan *shell sort* untuk dilakukan perbandingan kecepatan pengurutan dengan objek data numerik menggunakan bahasa pemrograman Python. Kecepatan pengurutan diukur menggunakan *library* timeit yang telah tersedia pada Python. Kemudian hasil catatan waktu pengurutan akan di visualisasikan dalam bentuk diagram garis menggunakan *library* Matplotlib.

Tujuan penelitian ini yaitu untuk mengetahui waktu yang diperlukan oleh algoritma *Selection sort*, *Shell Sort*, dan *Merge sort* dalam penyelesaian masalah pengurutan data. Selain itu, penelitian ini juga bertujuan untuk mengetahui algoritma yang paling cepat dalam pengurutan data di antara algoritma *Insertion sort*, *Shell sort*, dan *Merge sort*. Dengan adanya penelitian ini, diharapkan dapat membantu dalam pemilihan algoritma *sorting* yang akan digunakan pada penyelesaian masalah-masalah pengurutan data.

2 METODE

2.1 Bentuk Penelitian

Metode penelitian yang digunakan adalah metode penelitian komparatif dengan pendekatan kuantitatif. Menurut (Hafni Sahir, 2021) metode penelitian komparatif adalah metode penelitian dengan sifat meneliti hubungan dengan pengamatan langsung pada faktor yang diduga sebagai penyebab sebagai pembanding. Menurut pendapat (Arsyam & Tahir, 2021) tujuan penelitian komparatif adalah untuk melihat perbedaan dua atau lebih situasi, peristiwa, kegiatan, atau program yang sejenis atau hampir sama yang melibatkan semua unsur atau komponennya. Pada penelitian ini terdapat tiga jenis algoritma *sorting* yang digunakan dan penelitian ini berfungsi membandingkan kecepatan algoritma *sorting*.

2.2 Pengumpulan Data

Pada penelitian ini, data yang digunakan adalah data sampling numerik mulai dari 1 sampai 10000 dalam format Microsoft Excel yang terdiri dari dua kolom, yaitu kolom pertama berisi data acak dan kolom kedua berisi data sudah terurut dari kecil ke terbesar (*ascending*).

2.3 Pengolahan Data

Menurut (Salim & Haidir, 2019) pengolahan data adalah suatu proses untuk mendapatkan data dari setiap variabel penelitian yang siap dianalisis. Penelitian ini digunakan algoritma *selection sort*, *Shell Sort*, dan *merge sort* untuk melakukan pengolahan data yang akan divisualisasikan dengan grafik. Perangkat yang digunakan adalah laptop dengan spesifikasi AMD Ryzen 5 5600U *with* Radeon Graphics, RAM 16 GB, SSD M.2 WDC PC SN530 512 GB, dan GPU NVIDIA GeForce RTX 3050.

2.3.1 *Selection Sort*

Selection sort adalah suatu algoritma pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya sampai ke elemen yang terakhir (Wijaya & Wira Harjanti, 2024). *Selection sort* bekerja dengan mencari atau memilih nilai terkecil atau terbesar dan menukarnya dengan elemen awal paling kiri pada setiap tahap. Algoritma ini dikenal dengan algoritma pemilihan dimana melakukan pemilihan elemen minimum ataupun maksimum.

```
SelectionSort(array)
    n = length of array
    for i = 0 to n - 1
        minIndex = i
        for j = i + 1 to n - 1
            if array[minIndex] > array[j]
                minIndex = j
        swap array[i] with array[minIndex]
    return array
```

Gambar 1. Pseudocode Selection Sort

2.3.2 Merge Sort

Metode *merge sort* menggunakan konsep *devide* dan *conquer* yang membagi data S dalam dua kelompok yaitu S1 dan S2 yang tidak beririsan (*disjoint*). Proses pembagian data dilakukan secara rekursif sampai data tidak dapat dibagi lagi atau dengan kata lain data dalam sub bagian menjadi tunggal (Anggreani dkk., 2020). Cara kerjanya dengan membagi *array* menjadi dua bagian yang lebih kecil, mengurutkan setiap bagian secara rekursif, dan kemudian menggabungkan dua bagian yang telah diurutkan tersebut menjadi satu *array* yang terurut.

```
MergeSort(array)
    if length of array > 1
        mid = length of array // 2
        L = array[0:mid]
        R = array[mid:length of array]

        MergeSort(L)
        MergeSort(R)

        i = j = k = 0

        while i < length of L and j < length of R
            if L[i] < R[j]
                array[k] = L[i]
                i = i + 1
            else
                array[k] = R[j]
                j = j + 1
            k = k + 1

        while i < length of L
            array[k] = L[i]
            i = i + 1
            k = k + 1

        while j < length of R
            array[k] = R[j]
            j = j + 1
            k = k + 1
```

Gambar 2. Pseudocode *Merge Sort*

2.3.3 *Shell Sort*

Shell sort adalah algoritma penyortiran yang termasuk dalam kategori algoritma penyortiran berbasis perbandingan. Ini ditemukan oleh Donald Shell pada tahun 1959, dan ini merupakan perpanjangan dari algoritma penyortiran penyisipan (Yanti & Emi Sita Eriana, 2024). Algoritma *shell sort* mengurutkan elemen-elemen array dengan jarak tertentu (gap), kemudian secara bertahap mengurangi gap dan mengurutkan elemen dengan jarak yang lebih kecil, sampai akhirnya menggunakan jarak 1 untuk penyortiran akhir, yang menghasilkan *array* yang terurut.

```
ShellSort(array)
    n = length of array
    gap = n // 2

    while gap > 0
        for i = gap to n - 1
            temp = array[i]
            j = i
            while j >= gap and array[j - gap] > temp
                array[j] = array[j - gap]
                j = j - gap
            array[j] = temp
        gap = gap // 2
    return array
```

Gambar 3. Pseudocode Shell Sort

2.4 Pengukuran Waktu

Pengukuran waktu dilakukan menggunakan *library* timeit pada bahasa pemrograman Python. *Library* timeit dapat mengukur waktu eksekusi lebih akurat dengan cara kerja mengulangi eksekusi kode berkali-kali dan mengambil waktu rata-rata, sehingga dapat mengurangi pengaruh fluktuasi sementara pada hasil pengukuran.

2.5 Visualisasi

Visualisasi data adalah ilustrasi grafis untuk representasi bergambar data dengan penggunaan desain ilustrasi yang terintegrasi. Teknik visualisasi data populer yang terdiri dari *scatter plot*, *bar chart*, *pie chart*, dan *line chart*, banyak digunakan di bidang ilmu data, pemodelan matematika, dan penelitian komputasi (Sial dkk., 2021). Matplotlib adalah pustaka grafis untuk paket visualisasi data dengan Python yang mencakup aspek integral dalam tumpukan ilmu data python dan mudah didukung dengan NumPy, Pandas, dan pustaka terkait lainnya (Mieko Suparwanto dkk., 2023). Matplotlib merupakan pustaka Python cross-platform untuk membuat grafik 2 dimensi dengan kualitas tinggi. Matplotlib dapat digunakan dalam script Python, Interpreter Python dan iPython, server, dan 6 GUI toolkit. Dengan Matplotlib, membuat plots, histograms, spectra, bar charts, errorchards, scatterplots, akan lebih mudah dilakukan (David, 2019).Visualisasi pada penelitian ini dibuat menggunakan *library* matplotlib dalam bentuk grafik garis yang datanya diambil dari hasil pengurutan setiap algoritma sorting pada data acak dan urut.

3 HASIL DAN PEMBAHASAN

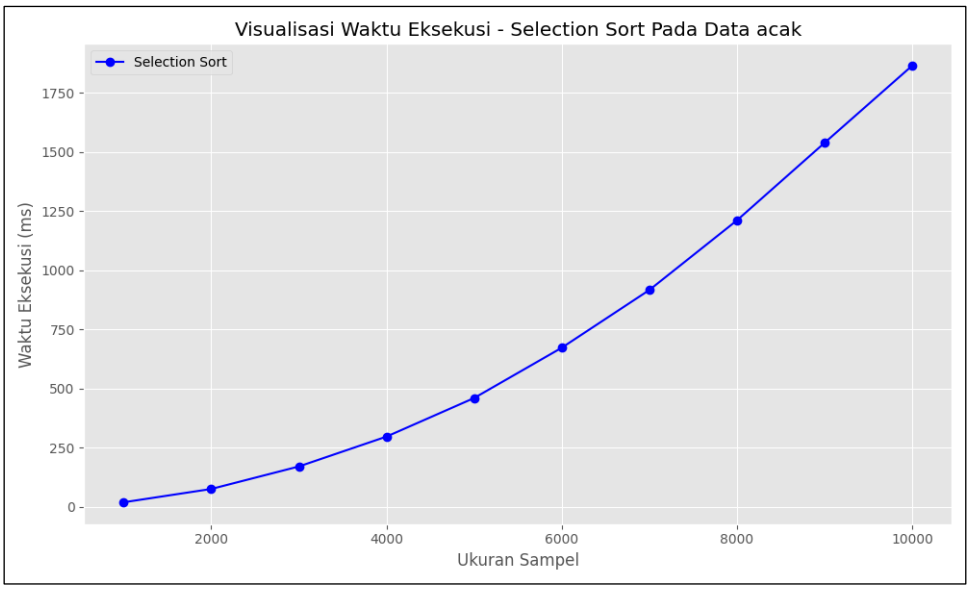
3.1 Hasil Pengujian Selection Sort

Pada pengujian algoritma *selection sort* untuk data acak dengan data terkecil (1000) diperoleh rata-rata waktu pengurutan dari lima kali percobaan yaitu 18.85 ms. Untuk data data terbesar (10000) diperoleh 1864.86 ms dengan sepuluh ribu data. Waktu eksekusi setiap sampel dapat dilihat pada tabel yang disajikan pada **Tabel 1**.

Tabel 1. Waktu Eksekusi Algoritma *Selection sort* Pada Data Acak

Ukuran Sampel	Percobaan ke-1 (ms)	Percobaan ke-2 (ms)	Percobaan ke-3 (ms)	Percobaan ke-4 (ms)	Percobaan ke-5 (ms)	Rata-rata Waktu (ms)
1000	19.53	18.68	18.48	18.68	18.88	18.85
2000	75.54	75.50	75.35	75.29	74.14	75.16
3000	169.72	168.88	170.52	172.40	169.97	170.30
4000	296.63	294.05	295.35	291.83	305.02	296.58
5000	458.18	455.62	461.27	460.09	462.66	459.56
6000	668.88	659.64	665.82	672.76	693.82	672.18
7000	906.96	905.33	945.91	915.71	907.67	916.31
8000	1198.63	1192.15	1251.71	1197.92	1213.65	1210.81
9000	1531.94	1498.72	1567.82	1542.80	1551.95	1538.65
10000	1869.38	1869.80	1875.81	1858.58	1850.73	1864.86

Berdasarkan **Tabel 1,** dapat dilihat bahwa pengurutan data acak dengan algoritma *selection sort* menghasilkan waktu rata-rata yang kenaikannya stabil. Pada ukuran sampel data terkecil (1000) waktu terbaik diperoleh sebesar 18,48 ms dan waktu terburuk sebesar 19,53 ms. Sedangkan pada ukuran sampel data terbesar (10000) waktu terbaik diperoleh sebesar 1850,73 ms dan waktu terburuk diperoleh sebesar 1875,81 ms. Visualisasi grafik garis waktu rata-rata setiap ukuran sampel dapat dilihat pada **Gambar 4**.



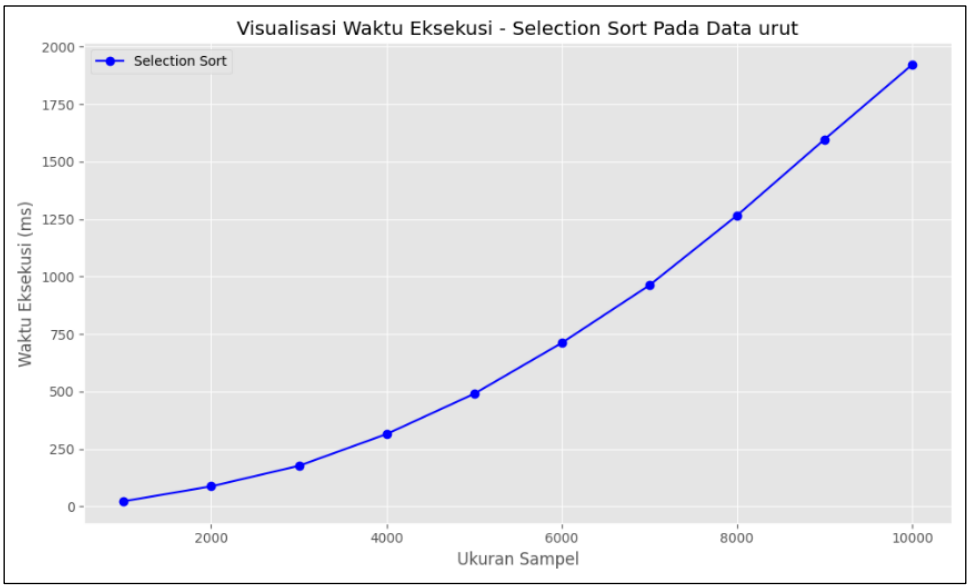
Gambar 4. Visualisasi Waktu Eksekusi *Selection sort* Pada Data Acak

Untuk data yang sudah urut, pengurutan menggunakan algoritma *selection sort* dengan lima kali pengulangan untuk data terkecil (1000) menghasilkan rata-rata waktu eksekusi 22.18 ms. Pada data terbesar (10000) dihasilkan rata-rata waktu eksekusi selama 1922.37 ms. Waktu eksekusi setiap sampel data dapat dilihat pada **Tabel 2**.

Tabel 2. Waktu Eksekusi Algoritma *Selection sort* Pada Data Urut

Ukuran Sampel	Percobaan ke-1 (ms)	Percobaan ke-2 (ms)	Percobaan ke-3 (ms)	Percobaan ke-4 (ms)	Percobaan ke-5 (ms)	Rata-rata Waktu (ms)
1000	21.34	19.85	21.72	28.22	19.80	22.18
2000	90.05	86.10	98.99	80.92	80.29	87.27
3000	172.05	170.58	179.01	180.66	180.11	176.48
4000	316.73	312.68	309.88	313.77	319.49	314.51
5000	495.54	483.94	490.71	488.67	490.32	489.84
6000	710.84	714.27	712.06	702.06	717.38	711.32
7000	959.34	959.87	964.33	965.76	961.43	962.15
8000	1279.57	1265.26	1270.43	1259.45	1255.49	1266.04
9000	1600.79	1599.90	1606.81	1594.77	1584.63	1597.38
10000	1935.65	1913.18	1926.13	1918.00	1918.88	1922.37

Dapat dilihat pada **Tabel 2**, bahwa waktu pengurutan algoritma *selection sort* pada data yang sudah terurut menghasilkan waktu yang lebih lama dibandingkan pengurtan dengan data acak. Pengurutan data yang sudah terurut dengan *selection sort* dapat memakan waktu yang lebih lama dibandingkan dengan pengurutan data yang teracak karena meskipun data sudah terurut, *selection sort* tetap melakukan pertukaran meskipun mungkin tidak ada perubahan nyata karena elemen yang dipilih sudah berada di tempat yang benar. Visualisasi dalam bentuk grafik garis dapat dilihat pada **Gambar 5**.



Gambar 5. Visualisasi Waktu Eksekusi *Selection sort* Pada Data Urut

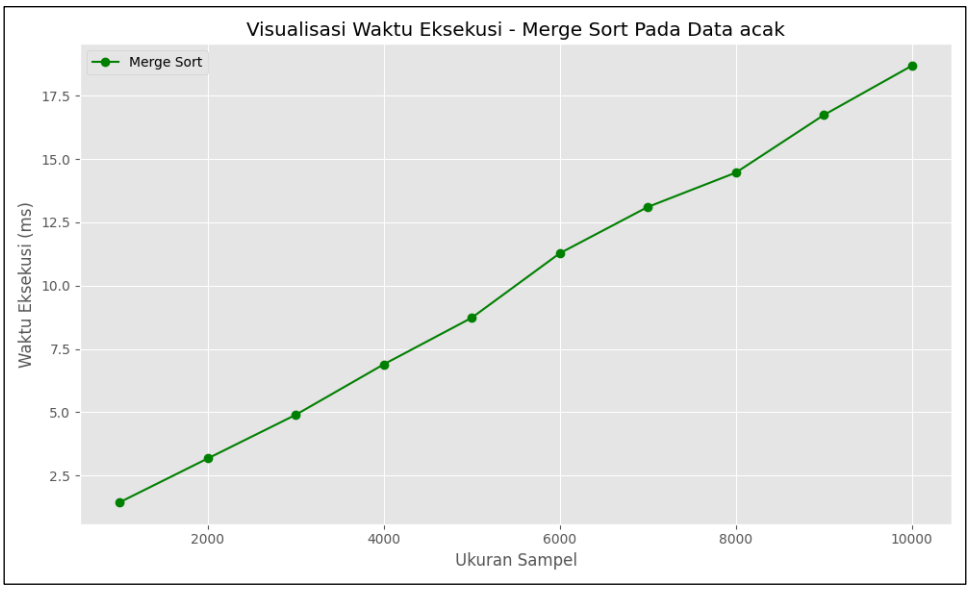
3.2 Hasil Pengujian Merge Sort

Pada pengujian algoritma *merge sort* dengan 1000 data acak, diperoleh waktu rata-rata dari lima kali pengulangan yaitu selama 1.44 ms. Sedangkan pada data terbesar (10000) diperoleh waktu selama 18.69. Dalam pengurutan data terbesar, waktu terbaik yang diperoleh *merge sort* adalah 18.44 ms. Waktu eksekusi setiap ukuran sampel dapat dilihat pada **Tabel 3**.

Tabel 3. Waktu Eksekusi Algoritma *Merge sort* Pada Data Acak

Ukuran Sampel	Percobaan ke-1 (ms)	Percobaan ke-2 (ms)	Percobaan ke-3 (ms)	Percobaan ke-4 (ms)	Percobaan ke-5 (ms)	Rata-rata Waktu (ms)
1000	1.42	1.60	1.39	1.39	1.39	1.44
2000	3.47	3.26	3.03	3.03	3.07	3.17
3000	5.00	4.93	4.84	4.88	4.82	4.89
4000	6.64	6.80	6.56	7.59	6.85	6.89
5000	8.88	8.56	8.64	8.63	8.96	8.73
6000	10.69	10.94	10.46	11.12	13.19	11.28
7000	14.06	13.69	12.45	12.51	12.84	13.11
8000	14.42	15.01	14.33	14.26	14.31	14.47
9000	16.58	16.32	16.87	17.23	16.74	16.75
10000	18.57	18.68	18.44	19.33	18.44	18.69

Berdasarkan data pada **Tabel 3**, terlihat bahwa terjadi kenaikan waktu eksekusi sebesar 2,55 ms pada pengurutan dengan ukuran sampel 6000 data. Penurunan waktu ekseskusi kembali turun sebesar 1,36 ms pada pengurutan data di antara ukuran sampel 7000 data dengan 8000 data. waktu rata-rata eksekusi dapat divisualisasikan dalam bentuk grafik garis seperti pada **Gambar 6**.



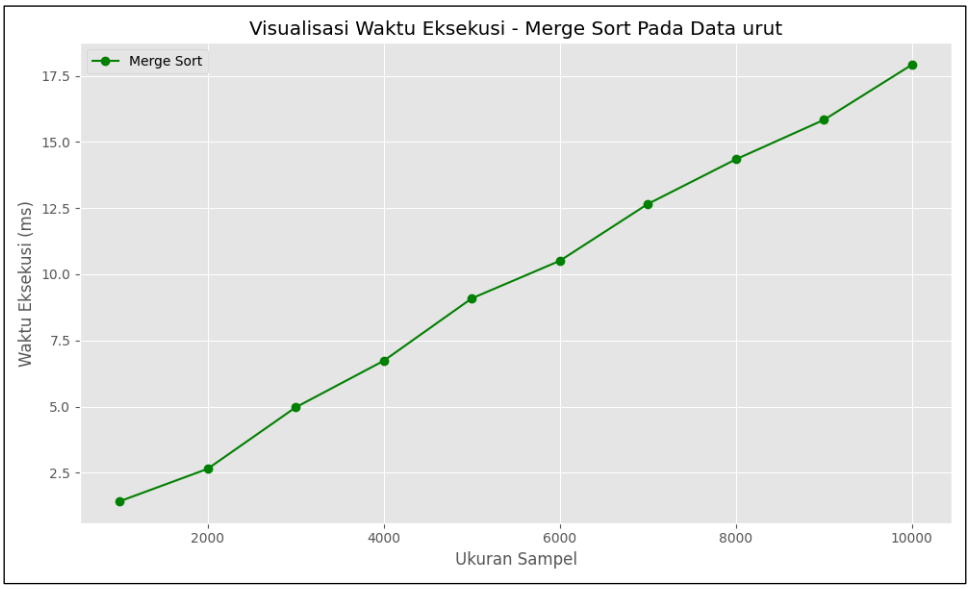
Gambar 6. Visualisasi Waktu Eksekusi *Merge sort* Pada Data Acak

Dengan data yang sudah terurut, algoritma *merge sort* menghasilkan rata-rata waktu yang tidak jauh berbeda dengan pengurutan data acak. Waktu yang dihasilkan dengan pengujian 1000 data adalah 1.42 ms. Sedangkan pada pengujian dengan data terbesar (10000) *merge sort* membutuhkan waktu selama 17.92 ms. Hasil waktu eksekusi dari setiap sampel dengan data yang sudah terurut dapat dilihat pada **Tabel 4**.

Tabel 4. Waktu Eksekusi Algoritma *Merge sort* Pada Data Urut

Ukuran Sampel	Percobaan ke-1 (ms)	Percobaan ke-2 (ms)	Percobaan ke-3 (ms)	Percobaan ke-4 (ms)	Percobaan ke-5 (ms)	Rata-rata Waktu (ms)
1000	1.38	1.86	1.23	1.45	1.19	1.42
2000	2.66	2.77	2.58	2.59	2.64	2.65
3000	4.39	4.08	5.21	6.25	4.93	4.97
4000	6.21	6.81	7.64	6.43	6.56	6.73
5000	8.12	9.84	8.79	7.75	10.94	9.09
6000	9.38	9.96	9.46	11.88	11.87	10.51
7000	12.09	12.26	11.62	14.09	13.22	12.66
8000	14.67	12.83	14.17	14.14	15.90	14.34
9000	14.40	17.08	16.98	15.19	15.51	15.83
10000	15.93	18.50	20.33	17.40	17.43	17.92

Berdasarkan hasil pada **Tabel 4**, terlihat bahwa perbedaan waktu eksekusi setiap ukuran sampel mulai dari 1000 data hingga 10000 data mengahsilkan kenaikan waktu yang stabil. hasil waktu rata-rata pada pengurutan data urut dari setiap ukuran sampel dapat divisualisasikan seerti pada **Gambar 7**.



Gambar 7. Visualisasi Waktu Eksekusi *Merge sort* Pada Data Urut

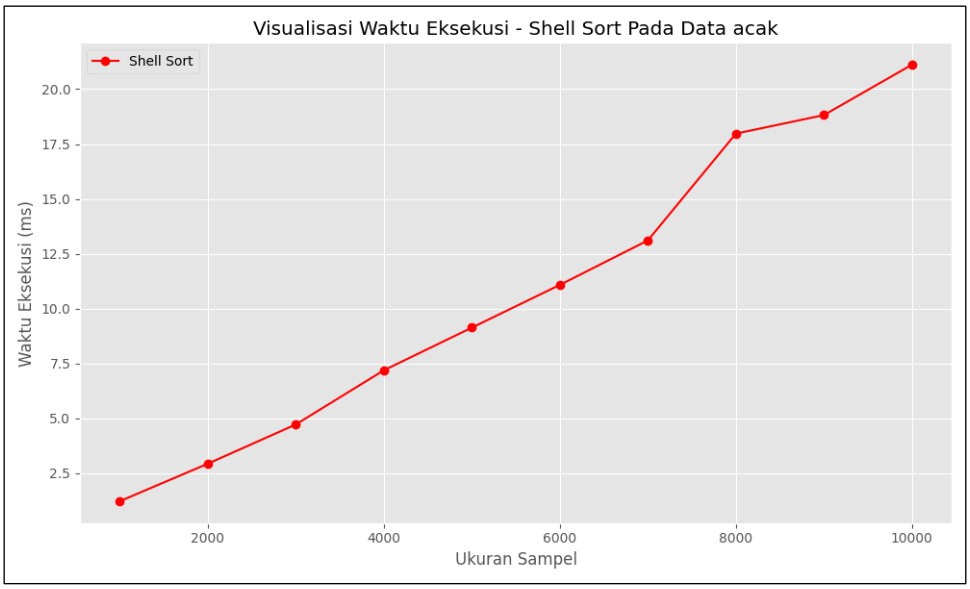
3.3 Hasil Pengujian *Shell Sort*

Pengurutan data acak dengan *shell sort* pada ukuran sampel sebanyak 1000 data dengan lima kali pengulangan, menghasilkan waktu rata-rata selama 1.22 ms. Pada ukuran sampel sebanyak 10000 dengan lima kali pengulangan, *shell sort* menghasilkan waktu rata-rata selama 21.11 ms.

Tabel 5. Waktu Eksekusi Algoritma *Shell Sort* Pada Data Acak

Ukuran Sampel	Percobaan ke-1 (ms)	Percobaan ke-2 (ms)	Percobaan ke-3 (ms)	Percobaan ke-4 (ms)	Percobaan ke-5 (ms)	Rata-rata Waktu (ms)
1000	1.28	1.28	1.18	1.19	1.18	1.22
2000	2.87	3.19	2.94	2.82	2.83	2.93
3000	4.68	4.79	4.71	4.72	4.71	4.72
4000	7.10	7.76	7.13	7.03	6.92	7.19
5000	9.02	9.18	9.17	9.37	8.92	9.13
6000	11.24	11.03	11.09	10.94	11.10	11.08
7000	13.70	13.30	12.83	12.64	13.08	13.11
8000	16.80	20.76	16.82	16.77	18.72	17.97
9000	19.04	18.60	18.61	19.24	18.61	18.82
10000	21.70	21.05	20.91	20.98	20.93	21.11

Shell sort mengalami kenaikan selisih waktu yang cukup signifikan dari pengurutan dengan 7000 data ke 8000 data sebesar 4,86 ms. Kenaikan selisih waktu dikarenakan kompleksitas waktu rata-rata dari *shell sort* bervariasi tergantung pada *gap sequence*. Visualisasi waktu rata-rata setiap ukuran sampel dapat dilihat pada **Gambar 8**.



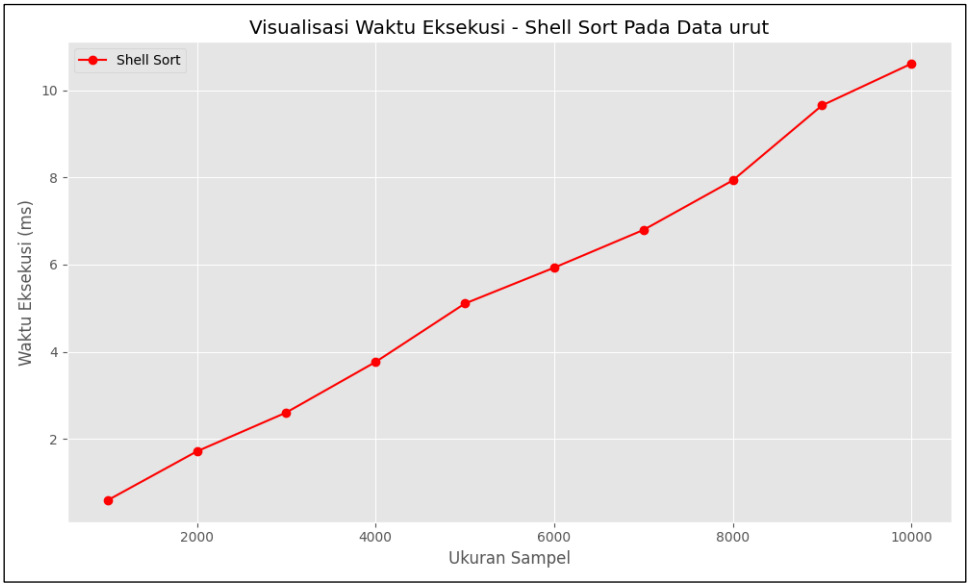
Gambar 8. Visualisasi Waktu Eksekusi *Shell Sort* Pada Data Acak

Pada pengurutan data yang terurut dengan 5 kali pengulangan eksekusi, *Shell Sort* menghasilkan waktu terbaik pada 1000 data urut yaitu 0.56 ms dan waktu terburuk 0.65 ms dengan waktu rata-rata 0.59 ms. Pada data terbesar (10000) waktu terbaik diperoleh 8.91 ms dan waktu terburuk 12.07 ms dengan waktu rata-rata 10.62 ms. Waktu eksekusi setiap ukuran sampel dapat dilihat pada **Tabel 6**.

Tabel 6. Waktu Eksekusi Algoritma *Shell Sort* Pada Data Urut

Ukuran Sampel	Percobaan ke-1 (ms)	Percobaan ke-2 (ms)	Percobaan ke-3 (ms)	Percobaan ke-4 (ms)	Percobaan ke-5 (ms)	Rata-rata Waktu (ms)
1000	0.65	0.57	0.57	0.56	0.58	0.59
2000	1.31	1.47	1.40	1.37	3.01	1.71
3000	3.33	2.36	2.48	2.37	2.46	2.60
4000	4.98	3.59	3.00	3.18	4.07	3.76
5000	4.39	6.87	4.10	4.62	5.55	5.11
6000	5.07	5.57	7.63	5.21	6.18	5.93
7000	6.43	7.49	6.11	6.85	7.10	6.80
8000	8.36	8.24	7.63	7.35	8.09	7.93
9000	8.96	9.65	8.97	11.48	9.23	9.66
10000	8.91	11.00	10.81	10.29	12.07	10.62

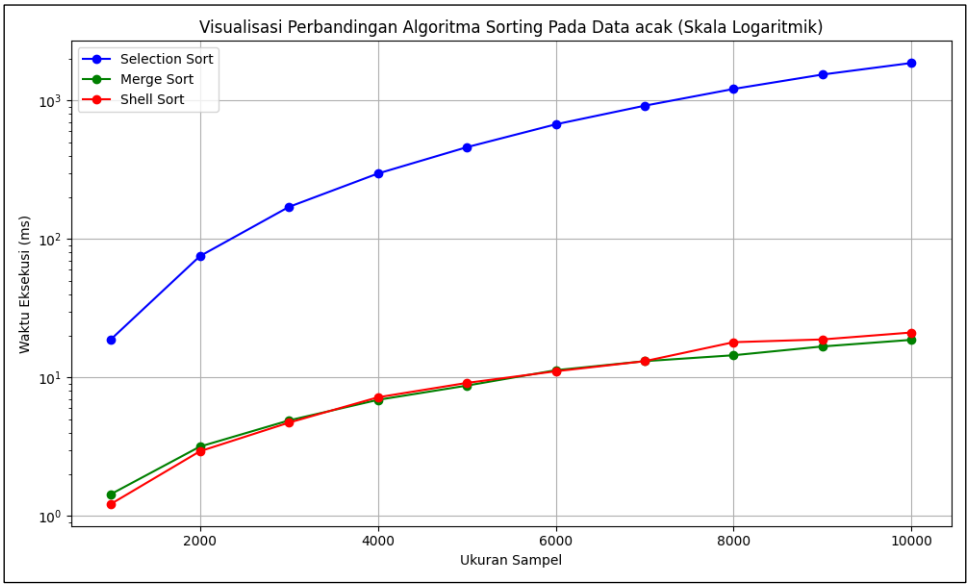
Selisih waktu rata-rata yang dihasilkan *shell sort* dengan data urut terlihat tidak stabil. Terjadi penurunan selisih waktu pada ukuran data 6000 dan terjadi kenaikan pada ukuran data 9000. Visualisasi waktu rata-rata setiap ukuran sampel data disajikan pada **Gambar 9**.



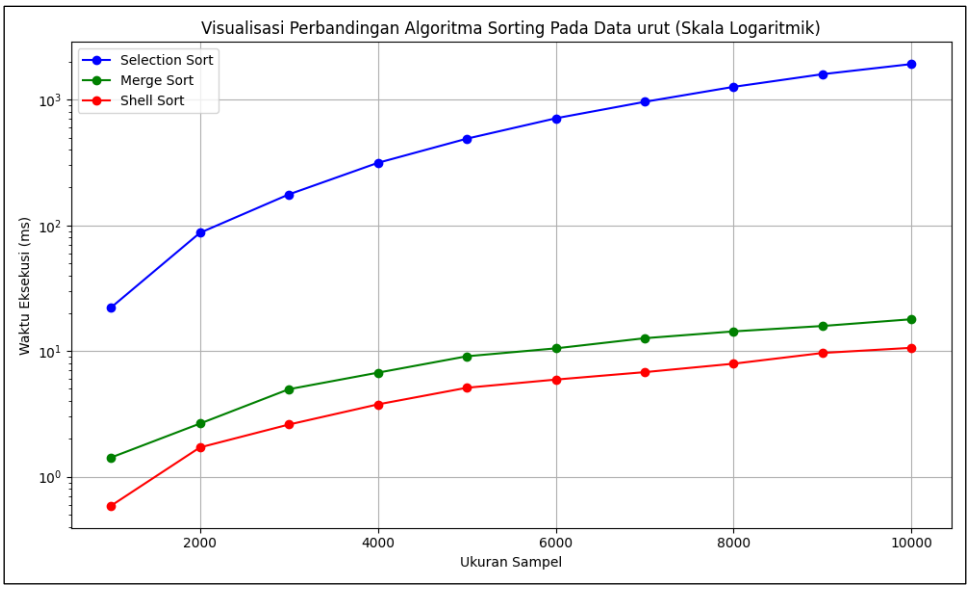
Gambar 9. Visualisasi Waktu Eksekusi *Shell Sort* Pada Data Urut

3.4 Perbandingan Visual

Pengujian pada penelitian ini menggunakan bahasa pemrograman Python dengan code editor Visual Studio Code untuk menjalankan program yang menghasilkan keluaran waktu sorting setiap algoritma. Kecepatan waktu pengurutan diukur dengan lima kali pengulangan kemudian dihasilkan waktu rata-rata dari setiap jumlah sampel pada algoritma sorting yang dilakukan. Visualisasi dilakukan dengan mengambil waktu rata-rata dari setiap ukuran sampel. Karena perbedaan waktu eksekusi yang tidak terlalu signifikan antara *merge sort* dan *Shell Sort*, sehingga visualisasi menggunakan skala biasa menyebabkan kesulitan dalam melihat perbedaannya. Visualisasi menggunakan skala logaritmik dapat menyederhanakan representasi data yang mencakup rentang nilai yang sangat besar atau sangat kecil, dengan demikian perbedaan waktu eksekusi dapat dilihat dengan lebih jelas. Gambar disajikan pada **Gambar 10** dan **Gambar 11**.



Gambar 10. Perbandingan Visual Algoritma Sorting Pada Data Acak Dalam Skala Logaritmik



Gambar 11. Perbandingan Visual Algoritma Sorting Pada Data Urut Dalam Skala Logaritmik

Terlihat pada **Gambar 10**, waktu pengurutan data acak menggunakan algoritma *merge sort* lebih cepat mulai dari data ke 8000 dan *shell sort* lebih cepat di awal hingga data ke 6000. Sedangkan pada **Gambar 11**, terlihat bahwa pengurutan pada data urut dengan algoritma *shell sort* lebih cepat dari pada *merge sort* dan *selection sort*. Algoritma selection sort tetap lambat pada kedua jenis data.

4 KESIMPULAN

Setelah dilakukan pengukuran waktu eksekusi pada algoritma *selection sort*, *merge sort*, dan *Shell Sort*, diperoleh hasil yang menunjukkan bahwa *Shell Sort* lebih baik dalam pengurutan data acak dan urut dengan rata-rata waktu 1.22 ms pada seribu data acak, 21.11 ms pada sepuluh ribu data acak, 0.59 ms pada seribu data urut, dan 10.62 ms pada sepuluh ribu data urut. Algoritma *merge sort* memiliki kecepatan yang hampir sama dengan *Shell Sort*, meskipun perbedaannya tidak terlalu signifikan, *Shell Sort* masih lebih cepat dalam penanganan pengurutan data acak dan urut. Algoritma *selection sort* bekerja lebih lama dari dibandingkan *merge sort* dan *Shell Sort*, dengan waktu rata-rata yang diperlukan pada pengurutan seribu data acak selama 18.85 ms dan 22.18 ms pada seribu data urut. Penulis sadar bahwa masih terdapat kekurangan pada artikel ini, penulis berharap agar artikel ini dapat ditambahkan pengukuran kompleksitas ruang oleh penulis selanjutnya.

UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa, karena rahmatnya penelitian yang berjudul "PERBANDINGAN ALGORITMA *SELECTION SORT*, *SHELL SORT*, DAN *MERGE SORT* PADA DATA SAMPLING NUMERIK MENGGUNAKAN MATPLOTLIB" dapat terselesaikan. Penulis juga berterima kasih kepada seluruh pihak yang telah membantu dalam pembuatan penelitian ini baik dalam bentuk dukungan, konsultan, perizinan, dan lain-lain, sehingga dapat selesai dengan sebagaimana mestinya.

DAFTAR PUSTAKA

Anggreani, D., Wibawa, A. P., Purnawansyah, P., & Herman, H. (2020). Perbandingan Efisiensi Algoritma Sorting dalam Penggunaan Bandwidth. *ILKOM Jurnal Ilmiah*, *12*(2), 96–103. https://doi.org/10.33096/ilkom.v12i2.538.96-103

Arsyam, M., & Tahir, M. Y. (2021). Ragam Jenis Penelitian dan Perspektif. *Jurnal STAI DDI Makssar*. http://jurnal.staiddimakassar.ac.id/index.php/aujpsi

David, F. (2019). VISUALISASI DATA DALAM BENTUK 3 DIMENSI DENGAN MENGGUNAKAN BAHASA PEMROGRAMAN PYTHON. *Seminar Nasional PIMIMD-5, ITP, Padang*. https://doi.org/10.21063/PIMIMD5.2019.1

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*, 2532–2540. https://doi.org/10.1016/j.procs.2020.04.274

Mieko Suparwanto, B., Studi Teknik Informatika Jurusan Teknik Elektro, P., Negeri Pontianak, P., Jendral Ahmad Yani Pontianak Kalimantan Barat, J., & Artikel, R. (2023). Visualisasi Data Dari Data Ketidakhadiran Mahasiswa Menggunakan Pemrograman Python. *JURNAL TEKNOLOGI INFORMASI*, *02*(02), 103–113. http://jurnal.utu.ac.id/JTI

Sial, A. H., Rashdi, S. Y. S., & Khan, Dr. A. H. (2021). Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python. *International Journal of Advanced Trends in Computer Science and Engineering*, *10*(1), 277–281. https://doi.org/10.30534/ijatcse/2021/391012021

Wijaya, S., & Wira Harjanti, T. (2024). PERBANDINGAN ALGORITMA SORTING DENGAN MENGGUNAKAN BAHASA PEMOGRAMAN JAVASCRIPT DALAM PENGGUNAAN WAKTU KOMPUTASI DAN PENGGUNAAN MEMORI. *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, *8*(2), 294–302. https://www.jdoodle.com/execute-nodejs-

Salim, H., & Haidir. (2019). *Penelitian Pendidikan: Metode, Pendekatan, dan Jenis* (I. S. Azhar, Ed.; 1 ed., Vol. 1). Kencana (Divisi PrenadaMedia Group).

Yanti, F., & Emi Sita Eriana, Mk. (2024). *ALGORITMA SORTING DAN SEARCHING* (A. M. L. Eri Setiawan, Ed.; 2024 ed., Vol. 1). CV. EUREKA MEDIA AKSARA.