LAPORAN PROJEK MATA KULIAH PEMROGRAMAN DAN ALGORITMA

“Perbandingan Waktu Eksekusi Algoritma Bubble Sort, Insertion Sort, dan Quick Sort pada Data Acak Menggunakan Python Tanpa Sorting Bawaan.”

Dosen Pembimbing:
Hasanuddin Al-Habib, M.Si.

Disusun Oleh:

Moch. Abdul Karim	**(24030214010)**
Hafid Mahfud Abdullah	**(24030214024)**
Naura Fatia Divani Elitha	**(24030214038)**
Aurell Salsabila Shafa Fredella	**(24030214047)**
Kamaulina Fanni Wahyuningsih	**(24030214088)**
Elvina Ahyaruniswah	**(24030214096)**

Program Studi Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Negeri Surabaya
Tahun 2025

Daftar Isi

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pengurutan data adalah proses penting dalam bidang ilmu komputer yang berfungsi untuk mengatur data dalam urutan tertentu, sehingga memudahkan proses pencarian dan pengolahan lebih lanjut. Algoritma pengurutan yang umum digunakan meliputi Bubble Sort, Insertion Sort, dan Quick Sort, yang memiliki karakteristik dan efisiensi berbeda dalam hal waktu eksekusi. Bubble Sort dan Insertion Sort adalah algoritma sederhana yang mudah diimplementasikan, namun memiliki kompleksitas waktu $O(n^2)$ sehingga kurang efisien untuk dataset berukuran besar. Sebaliknya, Quick Sort yang menggunakan pendekatan divide and conquer memiliki kompleksitas rata-rata $O(n \log n)$ dan terbukti lebih cepat serta efisien dalam pengurutan dataset besar (Conference UT, 2024)

Pengujian empiris pada dataset acak (n=100, 500, 1000) menunjukkan Quick Sort konsisten tercepat (0.000075s-0.002588s), memvalidasi keunggulan praktisnya. Penelitian ini membandingkan waktu eksekusi ketiga algoritma menggunakan Python tanpa fungsi sorting bawaan, bertujuan memberikan data empiris untuk pemilihan algoritma optimal berdasarkan efisiensi waktu.

1.2. Rumusan Masalah

1. Bagaimana cara mengimplementasikan Bubble Sort, Quick Sort, dan Insertion Sort dalam program?
2. Seberapa besar perbedaan waktu eksekusi dari ketiga algoritma tersebut?
3. Algoritma manakah yang paling efisien dalam menangani dataset kecil maupun sedang?

1.3. Tujuan

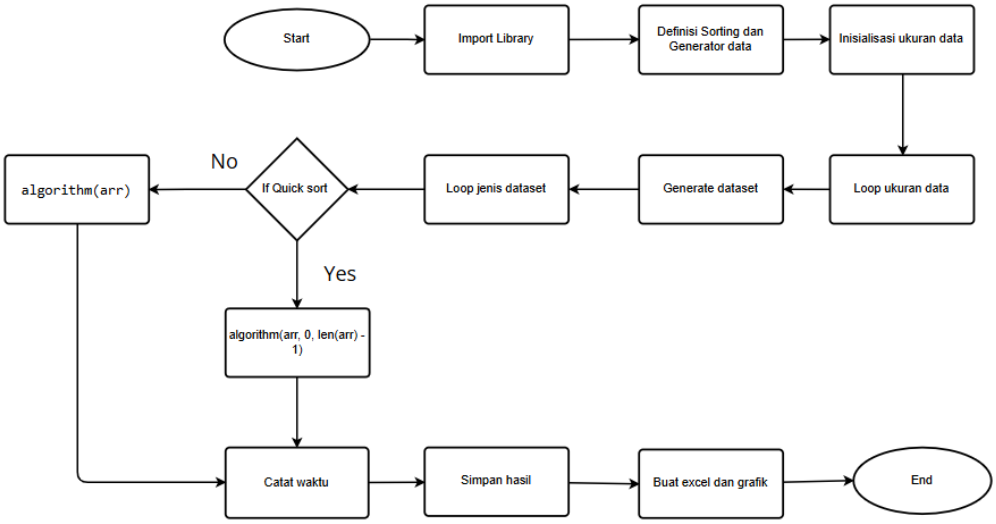
1. Mengimplementasikan tiga algoritma sorting: Bubble Sort, Quick Sort, dan Insertion Sort dalam pemrograman.
2. Menguji performa masing-masing algoritma dengan menggunakan dataset acak.
3. Membuat tabel perbandingan untuk menguji perbedaan tiga sorting algoritma.
4. Menganalisis perbandingan ketiga sorting algoritma berdasarkan kompleksitas dan waktu eksekusi.

BAB II
ANALISIS DAN PERANCANGAN

2.1. Analisis Kebutuhan Aplikasi

Aplikasi membutuhkan generator dataset (numbers, ASCII, datetime), time akurat (time.time()), dan output tabel dan grafik. Pengujian pada n=100, 500, dan 1000 untuk merepresentasikan skala kecil-sedang-besar.

2.2. Diagram Alur



2.3. Sketsa Desain Antarmuka

1. Tampilan Awal Program

   Pada saat program dijalankan, sistem akan langsung melakukan proses pengujian algoritma sorting tanpa memerlukan input dari pengguna. Program menampilkan judul hasil perbandingan algoritma sorting pada layar sebagai penanda bahwa proses pengujian sedang berlangsung.

   Sketsa Tampilan Awal:

   HASIL PERBANDINGAN SORTING

2. Tampilan Tabel Hasil di *Console*

   Setelah proses pengujian selesai, sistem menampilkan tabel perbandingan waktu eksekusi algoritma sorting di console. Tabel ini berisi informasi nomor pengujian, ukuran data, jenis dataset, serta waktu eksekusi dari masing-masing algoritma.

   Sketsa Tampilan Tabel:

   No | Data Size | Dataset Type | Bubble Sort | Insertion Sort | Quick Sort

3. Keluaran Berupa File

   Selain ditampilkan pada layar, hasil pengujian juga disimpan dalam bentuk file, yaitu:

   1. File Excel (hasil_sorting.xlsx) yang berisi tabel lengkap hasil perbandingan algoritma sorting.

2. File grafik (.png) yang menunjukkan perbandingan waktu eksekusi algoritma sorting berdasarkan ukuran dan jenis data. Seluruh grafik disimpan dalam folder hasil_grafik.

Sketsa Struktur Folder Output:

```
project_sorting
 │
 ├── hasil_sorting.xlsx
 └── hasil_grafik
   ├── random_numbers.png
   ├── ascii_codes.png
   └── datetime_objects.png
```

BAB III
IMPLEMENTASI

3.1. Penjelasan Kode Program

1. Program ini menggunakan beberapa library pendukung untuk menunjang proses pengujian dan analisis algoritma sorting.

```
import random
import string
import datetime
import time
import os
import pandas as pd
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
```

Library *random* dan *string* digunakan untuk menghasilkan data uji secara acak. Library datetime digunakan untuk membentuk data bertipe waktu. Library *time* digunakan untuk mengukur waktu eksekusi setiap algoritma sorting. Library *os* digunakan untuk mengelola direktori penyimpanan hasil grafik. Library *pandas* digunakan untuk menyimpan hasil pengujian ke dalam tabel dan mengekspornya ke file Excel. Library *matplotlib* digunakan untuk menampilkan serta menyimpan grafik perbandingan performa algoritma sorting.

2. Program ini mengimplementasikan tiga algoritma sorting, yaitu Bubble Sort, Insertion Sort, dan Quick Sort.
   a. Bubble Sort

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

Fungsi *bubble_sort(arr)* bekerja dengan membandingkan dua elemen yang berdekatan dan menukarnya jika urutannya tidak sesuai. Proses ini dilakukan secara berulang hingga seluruh elemen berada dalam urutan yang benar.

Bubble Sort memiliki kompleksitas waktu $O(n^2)$ pada kasus terburuk.

   b. Insertion Sort

```
def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr
```

Fungsi *insertion_sort(arr)* mengurutkan data dengan cara menyisipkan setiap elemen ke posisi yang tepat pada bagian array yang sudah terurut sebelumnya. Algoritma ini efektif untuk data berukuran kecil.

Kompleksitas waktu Insertion Sort pada kasus terburuk adalah $O(n^2)$.

c. Quick Sort

```
def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

def partition(arr, low, high):
    pivot = arr[high]
    i = low - 1
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1
```

Quick Sort diimplementasikan menggunakan metode rekursif dengan fungsi *partition()* sebagai pembagi array berdasarkan elemen pivot. Data yang lebih kecil dari pivot diletakkan di sebelah kiri, sedangkan data yang lebih besar di sebelah kanan.

Quick Sort memiliki kompleksitas waktu rata-rata O(n log n).

3. Pembuatan data set uji

```
def generate_random_numbers(n):
    return [random.randint(1, 10000) for _ in range(n)]

def generate_random_ascii(n):
    return [ord(random.choice(string.ascii_letters)) for _ in range(n)]

def generate_random_datetimes(n):
    base = datetime.datetime(2025, 1, 1)
    return [base + datetime.timedelta(days=random.randint(0, 365)) for _ in range(n)]
```

Program menghasilkan tiga jenis dataset sebagai bahan pengujian:

a. Bilangan Acak, dihasilkan menggunakan fungsi *generate_random_numbers(n)*.
b. Kode ASCII, dihasilkan menggunakan fungsi *generate_random_ascii(n)*.
c. Objek Tanggal dan Waktu, dihasilkan menggunakan fungsi *generate_random_datetimes(n)*.

Setiap dataset dibuat dengan ukuran data 100, 500, dan 1000 elemen untuk melihat pengaruh ukuran data terhadap waktu eksekusi algoritma sorting.

4. Pengujian waktu eksekusi.

```
def test_sorting(algorithm, data, is_quick=False):
    arr = data.copy()
    start = time.time()
    if is_quick:
        algorithm(arr, 0, len(arr) - 1)
    else:
        algorithm(arr)
    return time.time() - start
```

Fungsi *test_sorting()* digunakan untuk mengukur waktu eksekusi setiap algoritma sorting. Pengukuran dilakukan dengan mencatat waktu sebelum dan sesudah proses sorting menggunakan fungsi *time.time()*. Selisih waktu tersebut digunakan sebagai nilai waktu eksekusi algoritma.

5. Penyajian dan Penyimpanan Hasil

Hasil pengujian disimpan dalam dua bentuk, yaitu:

1. Tabel, yang disimpan ke dalam file Excel *(hasil_sorting.xlsx)* menggunakan library pandas.
2. Grafik, yang menunjukkan perbandingan waktu eksekusi antar algoritma sorting untuk setiap jenis dataset. Grafik disimpan dalam folder hasil_grafik dalam format gambar *(.png)*.

6. Output Program

Program menampilkan tabel hasil perbandingan algoritma sorting pada layar serta menghasilkan grafik visual untuk memudahkan analisis performa algoritma berdasarkan jenis dan ukuran data.

3.2. Manual Penggunaan

1. Copy kode lengkap ke file .py
2. Jalankan python filename.py
3. Lihat tabel hasil otomatis dan grafik yang tersimpan di *file explorer*

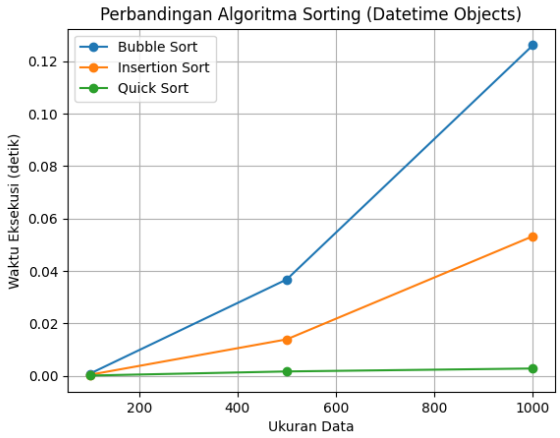
3.3. Screenshot Output

1. Screenshot hasil yang muncul pada terminal

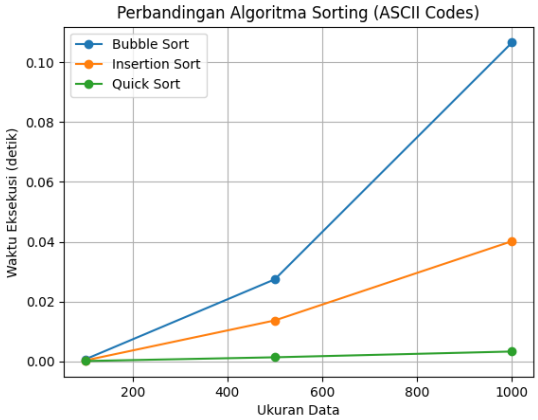
HASIL PERBANDINGAN SORTING (DALAM TABEL)

No	Data Size	Dataset Type	Bubble Sort	Insertion Sort	Quick Sort
1	100	Random Numbers	0.001438 s	0.000672 s	0.000355 s
2	100	ASCII Codes	0.001496 s	0.000935 s	0.000305 s
3	100	Datetime Objects	0.002188 s	0.000893 s	0.000303 s
3	100	Datetime Objects	0.002188 s	0.000893 s	0.000303 s
4	500	Random Numbers	0.049312 s	0.022022 s	0.001861 s
5	500	ASCII Codes	0.045192 s	0.020837 s	0.002800 s
6	500	Datetime Objects	0.055848 s	0.024347 s	0.002266 s
7	1000	Random Numbers	0.156637 s	0.052091 s	0.001735 s
8	1000	ASCII Codes	0.084142 s	0.042292 s	0.003181 s
9	1000	Datetime Objects	0.111347 s	0.042772 s	0.001772 s

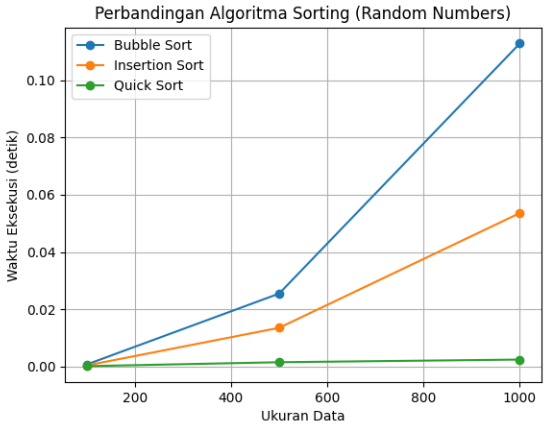
2. Grafik perbandingan sorting algoritma menggunakan dataset *datetime*



3. Grafik perbandingan sorting algoritma menggunakan dataset ASCII



4. Grafik perbandingan sorting algoritma menggunakan dataset angka acak



5. Screenshot tabel yang tersimpan di excel

Ukuran Data	Jenis Dataset	Bubble Sort (s)	Insertion Sort (s)	Quick Sort (s)
100	Random Numbers	0,000708	0,000387	0,00015
100	ASCII Codes	0,000672	0,000316	0,00013
100	Datetime Objects	0,000862	0,000458	0,000167
500	Random Numbers	0,025474	0,013535	0,00152
500	ASCII Codes	0,02745	0,013705	0,001376
500	Datetime Objects	0,036819	0,013973	0,00173
1000	Random Numbers	0,112802	0,053586	0,002446
1000	ASCII Codes	0,106598	0,040199	0,003316
1000	Datetime Objects	0,126114	0,053301	0,002857

Lampiran

```
import random
import string
import datetime
import time
import os
import pandas as pd
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt

# SORTING ALGORITHMS
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
```

```
    return arr

def partition(arr, low, high):
    pivot = arr[high]
    i = low - 1
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1

def quick_sort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quick_sort(arr, low, pi - 1)
        quick_sort(arr, pi + 1, high)
    return arr

# DATA GENERATOR
def generate_random_numbers(n):
    return [random.randint(1, 10000) for _ in range(n)]

def generate_random_ascii(n):
    return [ord(random.choice(string.ascii_letters)) for _ in range(n)]

def generate_random_datetimes(n):
    base = datetime.datetime(2025, 1, 1)
    return [base + datetime.timedelta(days=random.randint(0, 365)) for _ in range(n)]
```

```
# TEST FUNCTION
def test_sorting(algorithm, data, is_quick=False):
    arr = data.copy()
    start = time.time()
    if is_quick:
        algorithm(arr, 0, len(arr) - 1)
    else:
        algorithm(arr)
    return time.time() - start

# MAIN
if _name_ == "_main_":

    sizes = [100, 500, 1000]

    results_for_plot = {
        "Random Numbers": {"Bubble": [], "Insertion": [], "Quick": []},
        "ASCII Codes": {"Bubble": [], "Insertion": [], "Quick": []},
        "Datetime Objects": {"Bubble": [], "Insertion": [], "Quick": []}
    }

    table_results = []

    print("\nHASIL PERBANDINGAN SORTING\n")
    print("No | Data Size | Dataset Type      | Bubble Sort | Insertion Sort | Quick Sort")
    print("-------------------------------------------------------------------------")

    no = 1
```

```
for n in sizes:
    datasets = {
        "Random Numbers": generate_random_numbers(n),
        "ASCII Codes": generate_random_ascii(n),
        "Datetime Objects": generate_random_datetimes(n)
    }

    for name, data in datasets.items():
        b = test_sorting(bubble_sort, data)
        i = test_sorting(insertion_sort, data)
        q = test_sorting(quick_sort, data, is_quick=True)

        results_for_plot[name]["Bubble"].append(b)
        results_for_plot[name]["Insertion"].append(i)
        results_for_plot[name]["Quick"].append(q)

        table_results.append({
            "Ukuran Data": n,
            "Jenis Dataset": name,
            "Bubble Sort (s)": round(b, 6),
            "Insertion Sort (s)": round(i, 6),
            "Quick Sort (s)": round(q, 6)
        })

        print(f"{no:<3}| {n:<9} | {name:<17} | {b:.6f} | {i:.6f} | {q:.6f}")
        no += 1

# SIMPAN KE EXCEL
df = pd.DataFrame(table_results)
```

```
df.to_excel("hasil_sorting.xlsx", index=False)

# GRAFIK (DISIMPAN KE FILE)
output_dir = "hasil_grafik"
os.makedirs(output_dir, exist_ok=True)

for dataset in results_for_plot:
    plt.figure()
    plt.plot(sizes, results_for_plot[dataset]["Bubble"], marker='o')
    plt.plot(sizes, results_for_plot[dataset]["Insertion"], marker='o')
    plt.plot(sizes, results_for_plot[dataset]["Quick"], marker='o')

    plt.xlabel("Ukuran Data")
    plt.ylabel("Waktu Eksekusi (detik)")
    plt.title(f"Perbandingan Algoritma Sorting ({dataset})")
    plt.legend(["Bubble Sort", "Insertion Sort", "Quick Sort"])
    plt.grid(True)

    filename = dataset.replace(" ", "_").lower() + ".png"
    plt.savefig(os.path.join(output_dir, filename))
    plt.show()
    plt.close()

print("\nSemua grafik disimpan di folder 'hasil_grafik'")
print("Tabel hasil disimpan sebagai 'hasil_sorting.xlsx'")
```

Daftar Pustaka

Muhammad Bima Sena, R. M. (2024). Perbandingan Kinerja Algoritma Sorting dalam Pengurutan Data Menggunakan Bahasa Python. *Prosiding Seminar Nasional Sains dan Teknologi Seri 02*.

UT, C. (2024). Perbandingan Kinerja Algoritma Sorting dalam Pengurutan Data Menggunakan Bahasa Python. *Prosiding Seminar Nasional Sains dan Teknologi Seri 02*.