IMPLEMENTASI ALGORITMA TIMSORT PADA PYTHON
LAPORAN TUGAS AKHIR MATA KULIAH

Pemrograman dan Algoritma

4420102191

Oleh:
Salsabila Rosdiana 24030214001
Elgi Agatha 24030214111
Nirmala Rosyidatul L. 24030214120
Izazil Maghfuri 240300214129
Nabilah Febri Nur A. 24030214153
MA - 2024B

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

Program Studi S1 Matematika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Negeri Surabaya

2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pengurutan data (sorting) merupakan salah satu proses fundamental dalam pemrograman dan algoritma. Hampir seluruh aplikasi komputer membutuhkan proses pengurutan data, baik dalam pencarian data, pengolahan basis data, maupun analisis data. Oleh karena itu, diperlukan algoritma sorting yang efisien dan adaptif terhadap berbagai kondisi data.

Dalam mata kuliah Pemrograman dan Algoritma, mahasiswa telah mempelajari berbagai algoritma sorting dasar seperti Bubble Sort, Selection Sort, dan Insertion Sort. Namun, algoritma tersebut belum sepenuhnya mencerminkan algoritma sorting modern yang digunakan pada bahasa pemrograman saat ini. Python sebagai bahasa pemrograman populer menggunakan algoritma Timsort sebagai algoritma sorting bawaan.

Timsort merupakan algoritma hybrid yang menggunakan Insertion Sort dan Merge Sort serta dirancang untuk bekerja optimal pada data yang sudah terurut sebagian. Oleh karena itu, proyek ini bertujuan untuk mengimplementasikan algoritma Timsort pada Python agar mahasiswa memahami cara kerja algoritma sorting modern yang digunakan di dunia nyata.

1.2. Rumusan Masalah

1. Bagaimana cara kerja algoritma Timsort?
2. Bagaimana cara mengimplementasikan algoritma Timsort dalam bahasa Python?
3. Bagaimana alur logika program Timsort dalam proses pengurutan data?

1.3. Tujuan

1. Mengimplementasikan algoritma Timsort menggunakan bahasa Python
2. Memahami alur logika dan struktur algoritma Timsort
3. Mengetahui keunggulan Timsort dibandingkan algoritma sorting dasar

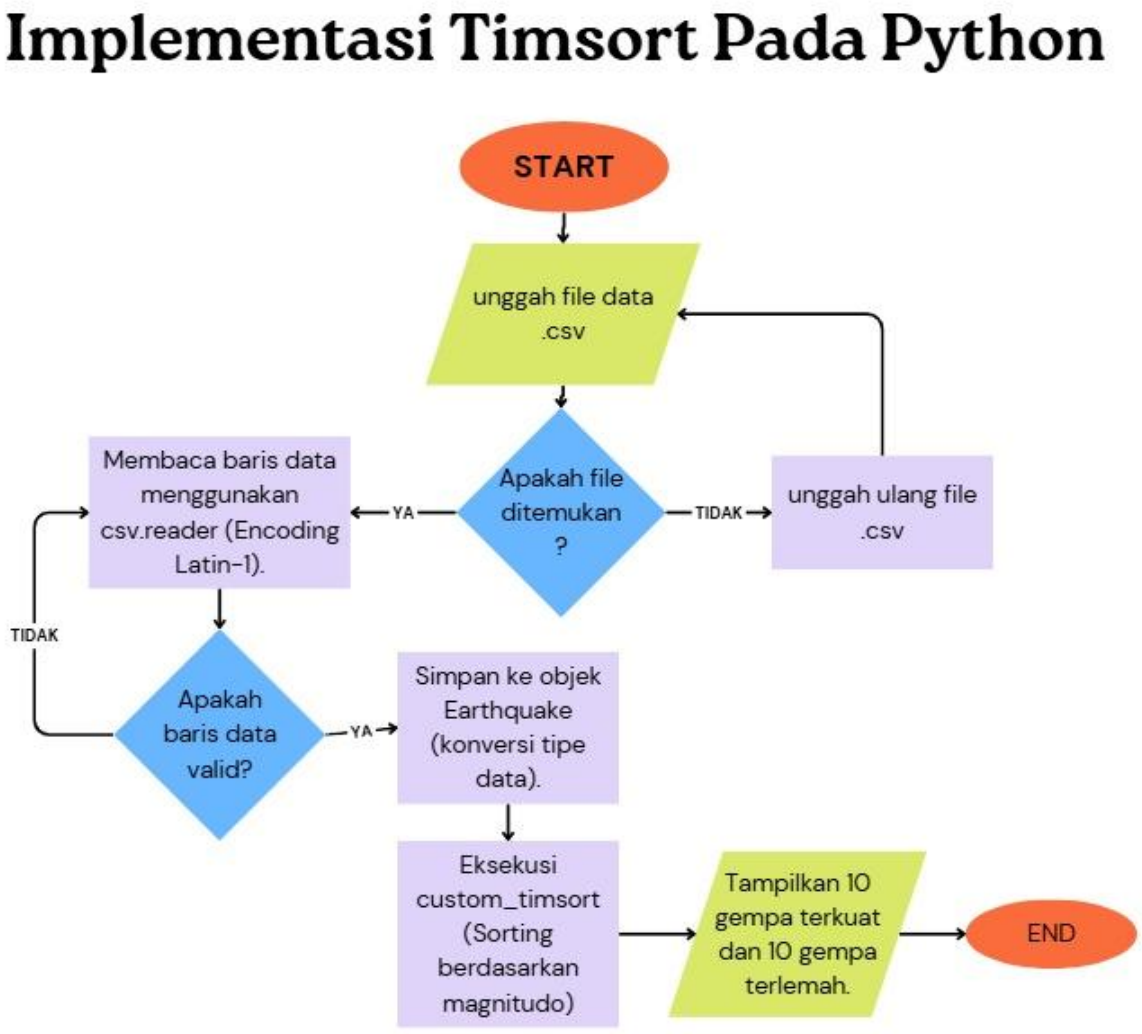
BAB II
ANALISIS DAN PENELITIAN

2.1. Analisis Algoritma Timsort

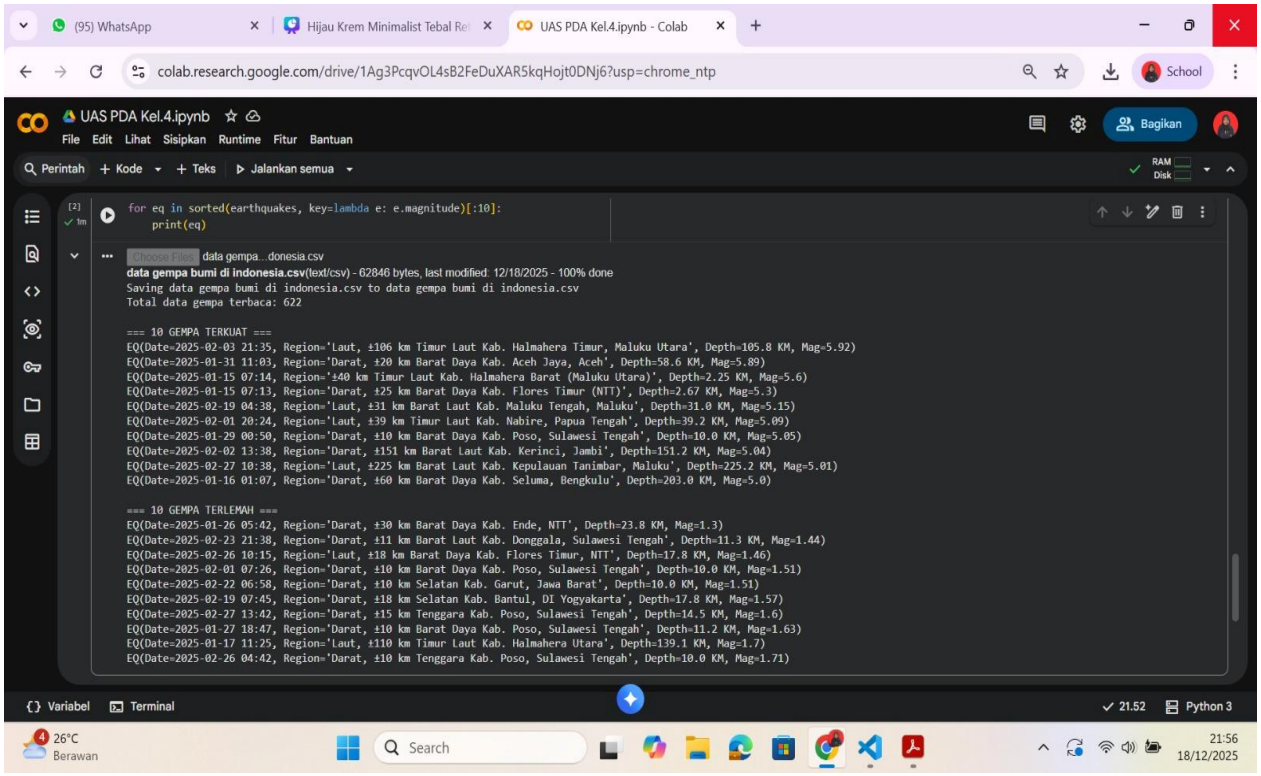
Aplikasi yang dibuat merupakan program berbasis Python dengan kebutuhan sebagai berikut:
Bahasa pemrograman: Python 3.x
Input: Dataset dalam format CSV
Output: Data yang telah terurut
Algoritma utama: Timsort

Pemrograman ini mengimplementasikan algoritma Timsort secara manual untuk keperluan pembelajaran, serta memanfaatkan fungsi sorting bawaan Python untuk menampilkan hasil pengurutan data secara efisien.

2.2. Diagram Alir



2.3. Sketsa Desain Antarmuka



Sketsa diatas merupakan output dari antarmuka aplikasi menggunakan Google Colab dengan output berupa:
Informasi jumlah data gempa bumi
Daftar 10 gempa bumi terkuat
Daftar 10 gempa bumi terlemah

BAB III
IMPLEMENTASI

3.1 Penjelasan Kode Program

3.1.1 Upload File dan Import Library

```
from google.colab import files
uploaded = files.upload()
```

Bagian ini digunakan untuk mengunggah file dataset berupa file CSV yang berisi data gempa bumi di Indonesia ke Google Colab.

```
from datetime import datetime
from typing import List
import csv
```

Library yang digunakan yaitu datetime untuk mengelola dan membandingkan waktu kejadian gempa, typing.List untuk memberikan anotasi tipe data, dan csv untuk membaca data dari file CSV

3.1.2 Model Data Gempa Bumi (Earthquake)

```
class Earthquake:
    def __init__(self, date_time, region, depth, magnitude):
        self.date_time = date_time
        self.region = region
        self.depth = depth
        self.magnitude = magnitude
```

Kelas Earthquake digunakan sebagai model data untuk merepresentasikan satu kejadian gempa bumi. Setiap objek memiliki tipe data date_time yang berupa waktu kejadian gempa, region yaitu wilayah gempa, depth untuk kedalaman gempa (KM), dan magnitude untuk merepresentasikan kekuatan gempa (SR)

Representasi Objek:

```
def __repr__(self):
```

Metode ini digunakan agar objek Earthquake dapat ditampilkan dalam format yang mudah dibaca saat dicetak ke layar.

Operator Perbandingan:

```
def __lt__(self, other):
    return self.date_time < other.date_time
def __le__(self, other):
    return self.date_time <= other.date_time
```

Metode ini memungkinkan objek Earthquake dibandingkan berdasarkan waktu kejadian, sehingga bisa diurutkan menggunakan algoritma sorting (Timsort)

3.1.3 Implementasi Timsort Kustom

Konstanta Minimum Run:

```
MIN_RUN = 32
```

Nilai ini menentukan panjang minimum run yang digunakan dalam algoritma Timsort. Nilai 32 dipilih karena umum digunakan dalam implementasi Timsort.

a. Insertion Sort

```
def insertion_sort(arr, left, right):
```

Insertion Sort digunakan untuk mengurutkan bagian kecil data (small runs). Algoritma ini efisien pada data berukuran kecil dan data yang sudah hampir terurut.

b. Merge

```
def merge(arr, left, mid, right):
```

Fungsi ini menggabungkan dua run yang telah terurut yaitu left_part dan right_part. Proses merge dilakukan dengan membandingkan elemen satu per satu dan menyusunnya kembali secara terurut.

c. Fungsi Utama Timsort

def custom_timsort(arr):
Fungsi ini merupakan implementasi utama algoritma Timsort, dengan tahapan:
Membagi array menjadi beberapa run dengan ukuran MIN_RUN
Mengurutkan setiap run mendeggunakan Insertion Sort
Menggabungkan run secara bertahap menggunakan proses merge
Menghasilkan data yang telah terurut

3.1.4 Membaca File CSV

def read_csv(file_name):

a. Membuka file dengan encoding latin-1
b. Membaca data menggunakan delimiter ;
c. Mengonversi kolom waktu (datetime), dan kolom kedalaman dan magnitudo (float)
d. Membersihkan karakter simbol yang tidak valid
e. Menyimpan data ke dalam objek Earthquake

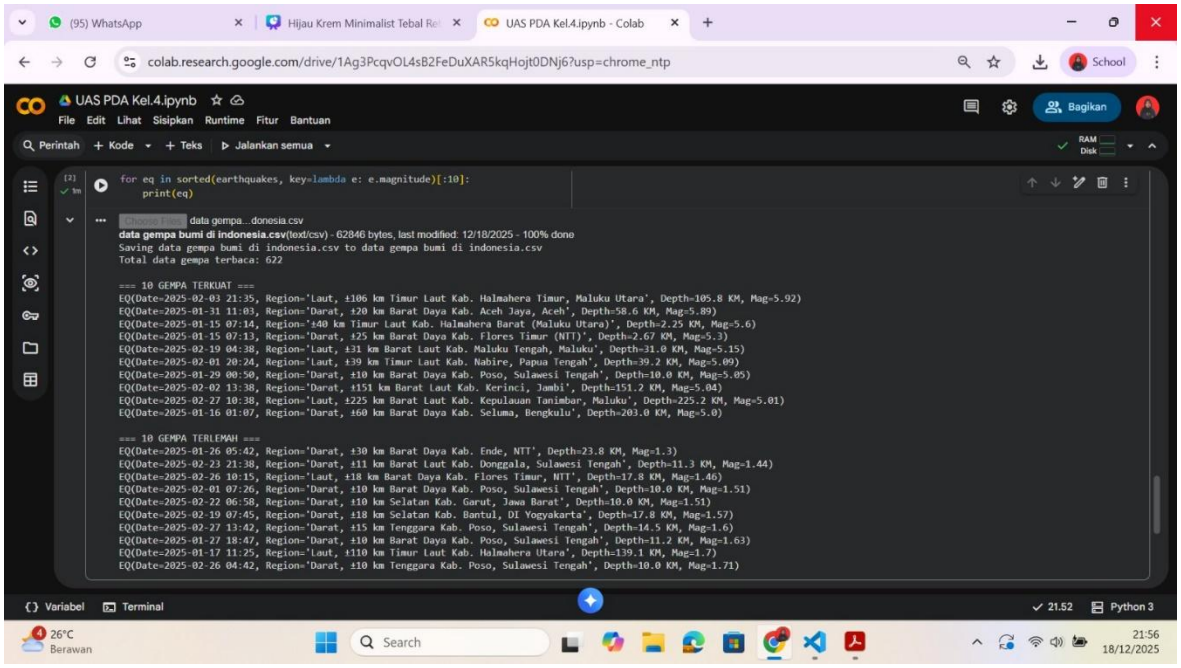
3.1.5 Program Utama

earthquake = read_csv(FILE _NAME)
print(f"total data gempa terbaca: {len(earthquakes)}")
Bagian ini membaca file CSV dan menampilkan jumlah total data gempa yang berhasil diproses.
Untuk menampilkan 10 gempa terkuat:
for eq in sorted(earthquakes, key=lambda e: e.magnitude, reverse=True)[:10]:
Untuk menampilkan 10 gempa terlemah:
for eq in sorted(earthquakes, key=lambda e: e.magnitude)[:10]:

3.2 Manual Penggunaan

1. Jalankan program pada lingkungan Python atau Google Colab.
2. Unggah file dataset gempa bumi dalam format CSV.
3. Program akan membaca dan memproses data gempa bumi.
4. Data akan diurutkan berdasarkan magnitude gempa.
5. Program menampilkan:
   - Jumlah total data gempa yang terbaca.
   - 10 gempa bumi dengan magnitudo terbesar.
   - 10 gempa bumi dengan magnitudo terkecil.

3.3 Screenshot Aplikasi



Gambar diatas menampilkan hasil pengurutan data gempa bumi berdasarkan magnitudo terbesar dan terlemah. Data ditampilkan dalam bentuk daftar yang memuat informasi waktu kejadian, wilayah gempa, kedalaman, dan magnitudo. Hasil ini menunjukkan bahwa program mampu melakukan pengurutan data dengan benar sesuai kriteria yang ditentukan.

BAB IV
LAMPIRAN

```
# GOOGLE COLAB: UPLOAD FILE
from google.colab import files
uploaded = files.upload()   # upload: data gempa bumi di indonesia.csv

# IMPORT LIBRARY
from datetime import datetime
from typing import List
import csv

# 1. MODEL DATA GEMPA BUMI
class Earthquake:
    def __init__(self, date_time: datetime, region: str,
                 depth: float, magnitude: float):
        self.date_time = date_time
        self.region = region
        self.depth = depth
        self.magnitude = magnitude

    def __repr__(self):
        return (
            f"EQ(Date={self.date_time.strftime('%Y-%m-%d %H:%M')}, "
            f"Region='{self.region}', Depth={self.depth} KM, "
            f"Mag={self.magnitude})"
        )

    def __lt__(self, other):
        return self.date_time < other.date_time

    def __le__(self, other):
        return self.date_time <= other.date_time

# 2. IMPLEMENTASI TIMSORT KUSTOM
MIN_RUN = 32

def insertion_sort(arr: List[Earthquake], left: int, right: int) -> None:
    for i in range(left + 1, right + 1):
        key = arr[i]
        j = i - 1
        while j >= left and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key


def merge(arr: List[Earthquake], left: int, mid: int, right: int) -> None:
    left_part = arr[left:mid + 1]
    right_part = arr[mid + 1:right + 1]

    i = j = 0
    k = left
```

```
    while i < len(left_part) and j < len(right_part):
        if left_part[i] <= right_part[j]:
            arr[k] = left_part[i]
            i += 1
        else:
            arr[k] = right_part[j]
            j += 1
        k += 1

    while i < len(left_part):
        arr[k] = left_part[i]
        i += 1
        k += 1

    while j < len(right_part):
        arr[k] = right_part[j]
        j += 1
        k += 1

def custom_timsort(arr: List[Earthquake]) -> List[Earthquake]:
    n = len(arr)

    for start in range(0, n, MIN_RUN):
        end = min(start + MIN_RUN - 1, n - 1)
        insertion_sort(arr, start, end)

    size = MIN_RUN
    while size < n:
        for left in range(0, n, 2 * size):
            mid = min(left + size - 1, n - 1)
            right = min(left + 2 * size - 1, n - 1)
            if mid < right:
                merge(arr, left, mid, right)
        size *= 2
    return arr

# 3. PEMBACA FILE CSV (ENCODING & SIMBOL SUDAH AMAN)
def read_csv(file_name: str) -> List[Earthquakes]:
    data = []

    with open(file_name, encoding="latin-1") as file:
        reader = csv.reader(file, delimiter=';')

        for row in reader:
            # Skip baris tidak valid
            if len(row) < 5:
                continue

            try:
                # Kolom 0 → Waktu
                date_time = datetime.strptime(
                    row[0].strip(),
                    "%d/%m/%Y %H:%M"
```

```
            )

            # Kolom 2 → Wilayah (perbaiki simbol ±)
            region = (
                row[2]
                .strip()
                .replace('ñ', '±')
                .replace('  ', ' ')
            )

            # Kolom 3 → Kedalaman (KM)
            depth = float(row[3].replace(',', '.'))

            # Kolom 4 → Magnitudo
            magnitude = float(row[4].replace(',', '.'))

            data.append(
                Earthquake(
                    date_time=date_time,
                    region=region,
                    depth=depth,
                    magnitude=magnitude
                )
            )

        except ValueError:
            # Header / keterangan → skip
            continue
    return data

# 4. PROGRAM UTAMA
FILE_NAME = "data gempa bumi di indonesia.csv"

earthquakes = read_csv(FILE_NAME)
print(f"Total data gempa terbaca: {len(earthquakes)}")

# HASIL OUTPUT
# 10 GEMPA TERKUAT
print("\n=== 10 GEMPA TERKUAT ===")
for eq in sorted(earthquakes, key=lambda e: e.magnitude, reverse=True)[:10]:
    print(eq)

# 10 GEMPA TERLEMAH
print("\n=== 10 GEMPA TERLEMAH ===")
for eq in sorted(earthquakes, key=lambda e: e.magnitude)[:10]:
    print(eq)
```

DAFTAR PUSTAKA

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009).*Introduction to Algorithms.* MIT Press.

Peters, T. (2002). *Timsort: A Hybrid Stable Sorting Algorithm*. Python Enhancement Proposal.

Oracle. (2023). *Jawa Platform Documentation.*

Python Documentation. (2023). *Sorting HOWTO.*

GeeksforGeeks. (2023). Understanding Timsort Algorithm.