LAPORAN PROYEK AKHIR

PEMOGRAMAN DAN ALGORITMA

"Perbandingan Algoritma Sorting dan Implementasi *General Tree* pada Data Berat Badan Mahasiswa M24A dan M24B"

Dosen Pengampu:

Hasanuddin Al-Habib, M.Si.

Disusun Oleh:

1.	Aditia Nugroho	24030214039
2.	Nanda Adwitiya Astadewi	24030214043
3.	Noviola Ajeng Aurelya	24030214051
4.	Fery Putra Pratama	24030214135
5.	Salma Nabelah	24030214139

PROGRAM STUDI MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2025

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan Laporan Proyek Akhir yang berjudul "Perbandingan Algoritma Sorting dan Implementasi General Tree pada Data Berat Badan Mahasiswa M24A dan Mahasiswa M24B." Laporan ini disusun untuk memenuhi salah satu tugas pada mata kuliah Pemrograman dan Algoritma, sekaligus sebagai sarana pembelajaran dalam memahami efektivitas berbagai algoritma sorting dalam pengolahan data menggunakan Microsoft Excel.

Penyusunan proposal project ini bertujuan untuk memenuhi tugas perkuliahan sekaligus sebagai sarana pembelajaran dalam memahami penerapan algoritma sorting dan struktur data General Tree. Dalam proposal ini dibahas perbandingan beberapa algoritma sorting, yaitu Bubble Sort, Quick Sort, dan Insertion Sort, yang diterapkan pada data berat badan mahasiswa kelas M24A dan M24B. Melalui perbandingan tersebut, diharapkan dapat diketahui perbedaan efisiensi serta kinerja masing-masing algoritma dalam mengolah data sederhana.

Penulis menyadari bahwa tersusunnya proposal ini tidak terlepas dari dukungan dan bimbingan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada dosen mata kuliah Pemrograman dan Algoritma serta semua pihak yang telah memberikan arahan dan bantuan selama proses penyusunan proposal ini.

Penulis menyadari bahwa proposal project ini masih jauh dari sempurna. Oleh sebab itu, kritik dan saran yang bersifat membangun sangat diharapkan agar project ini dapat disempurnakan pada tahap selanjutnya. Semoga proposal ini dapat memberikan manfaat dan menambah wawasan bagi penulis maupun pembaca.

Surabaya, 18 Desember 2025

Kelompok 5

DAFTAR ISI

Surabaya, 18 Desember 2025

Kelompok 5

BAB I
PENDAHULUAN

1.1 Latar belakang

Proses pengurutan data (sorting) merupakan salah satu kegiatan penting dalam pengolahan data karena membantu menata informasi agar lebih mudah dianalisis. Dalam dunia pemrograman, sorting dibutuhkan untuk mengolah data dalam jumlah kecil hingga besar, sehingga diperlukan algoritma yang cepat dan efisien.

Terdapat banyak algoritma sorting, namun tiga yang umum digunakan adalah Bubble Sort, Insertion Sort, dan Quick Sort. Ketiga algoritma ini memiliki cara kerja dan tingkat efisiensi yang berbeda. Oleh karena itu, penting untuk mengetahui algoritma mana yang lebih efektif untuk digunakan dalam kondisi tertentu.

Dalam penelitian ini, ketiga algoritma tersebut dibandingkan dan di implementasikan ke general tree menggunakan data berat badan mahasiswa kelas M24A dan M24B sebagai objek penelitian. Melalui perbandingan ini, diharapkan dapat terlihat algoritma atau metode mana yang paling optimal digunakan dalam pengurutan data, khususnya pada kasus pengolahan data sederhana seperti berat badan mahasiswa.

1.2 Tujuan

Tujuan dari proyek *"*Perbandingan Algoritma *Sorting* dan Implementasi *General Tree* pada Data Berat Badan Mahasiswa M24A dan M24B*"* ini adalah untuk untuk :

1. Mengimplementasikan tiga algoritma sorting pada dataset berat badan M24A dan M24B.
2. Menggunakan struktur General
   Tree sebagai metode tambahan untuk menyimpan dan membaca ulang data.
3. Menganalisis perbandingan performa antara algoritma-algoritma tersebut.
4. Menentukan algoritma mana yang
   paling optimal untuk kasus data sederhana seperti berat badan.

1.3 Rumusan masalah

1 Bagaimana cara mengimplementasikan algoritma Bubble Sort, Insertion Sort, dan Quick Sort untuk mengurutkan data berat badan mahasiswa kelas M24A dan M24B?

2 Bagaimana penerapan struktur data General
Tree dalam menyimpan serta menampilkan kembali data hasil pengurutan berat badan ma hasiswa?

3 Bagaimana perbandingan performa ketiga algoritma sorting tersebut ditinjau dari waktu e ksekusi dan efisiensi pengolahan data?

4 Algoritma sorting manakah yang
paling optimal digunakan untuk pengolahan data sederhana seperti data berat badan maha siswa?

5 Bagaimana pengaruh penggunaan General
Tree terhadap proses pengelolaan dan pembacaan ulang data yang telah diurutkan?

BAB II
DESKRIPSI PROYEK AWAL DAN MODIFIKASI

2.1 Deskripsi Proyek Awal

Proyek ini merupakan proyek awal yang bertujuan untuk membandingkan kinerja beberapa algoritma sorting dalam pengolahan data sederhana. Data yang digunakan dalam proyek ini adalah data berat badan mahasiswa **kelas** M24A dan M2024B.
Komponen utama dalam proyek ini meliputi:

1 **Dataset Berat Badan Mahasiswa**
Data berat badan mahasiswa kelas M24B dan M2024A digunakan sebagai objek utama pengujian. Data ini bersifat numerik sehingga mudah diterapkan pada berbagai algoritma sorting dan memudahkan analisis hasil pengurutan.

2 **Algoritma Sorting**
Proyek ini menggunakan tiga algoritma sorting, yaitu Bubble Sort, Quick Sort, dan Insertion Sort. Setiap algoritma diterapkan pada dataset yang sama untuk melihat perbedaan cara kerja, kecepatan, dan efisiensi masing-masing algoritma.

3 **Struktur Data General Tree**
General Tree digunakan sebagai struktur data tambahan untuk menyimpan data berat badan mahasiswa. Data dimasukkan ke dalam node-node tree, kemudian diambil kembali melalui proses traversal sebelum dilakukan pengurutan menggunakan algoritma sorting.

4 **Proses Pengujian dan Analisis**
Setelah data diurutkan, dilakukan pengamatan terhadap waktu eksekusi dan langkah-langkah pengurutan. Hasil dari setiap algoritma kemudian dibandingkan untuk mengetahui algoritma yang paling optimal digunakan pada data sederhana.

Proyek ini dirancang sebagai media pembelajaran untuk memahami konsep algoritma sorting dan struktur data, serta membantu mahasiswa melihat perbedaan performa algoritma secara langsung melalui implementasi sederhana.

2.2 Modifikasi Proyek

Modifikasi proyek ini bertujuan untuk meningkatkan kesiapan sistem agar mendekati kondisi penggunaan nyata. Proyek yang awalnya hanya menyediakan fungsi dasar kemudian dikembangkan dengan penambahan fitur pemantauan proses, penguatan keamanan sistem, serta kemampuan untuk menangani peningkatan beban penggunaan, sehingga sistem menjadi lebih andal dan siap digunakan.

2.2.1 File dan Folder yang Dimodifikasi:

```python
import pandas as pd
import time

class Node:
    def __init__(self, value):
        self.value = value   # (nim3, bb, nama)
        self.children = []

    def add_child(self, child):
        self.children.append(child)
```

Gambar 2.2.1.1(a)

- *import pandas as pd* Kode ini digunakan untuk mengimpor library **pandas**.
- Kode *import time* berfungsi untuk mengimpor library **time** digunakan untuk menghitung waktu eksekusi program
- *class Node* mendefinisikan sebuah **class bernama Node** yang digunakan sebagai struktur dasar pembentuk **General Tree**,
- *def __init__(self, value)*Kode ini merupakan **constructor** dari class Node.
- *self.value = value # (nim3, bb, nama)* digunakan untuk menyimpan data ke dalam node.
- *self.children = []* berfungsi untuk menyimpan node-node anak yang lebih dari satu.
- Sebuah method ini *def add_child(self, child):* digunakan untuk menambahkan node anak ke dalam node induk.
- Kedalam list children *self.children.append(child)* menghubungkan antara node induk dan node anak

```python
def is_greater(a, b):
    # a dan b berbentuk (nim3, bb, nama)
    # prioritas: BB → NIM → Nama
    return (a[1], a[0], a[2]) > (b[1], b[0], b[2])
```

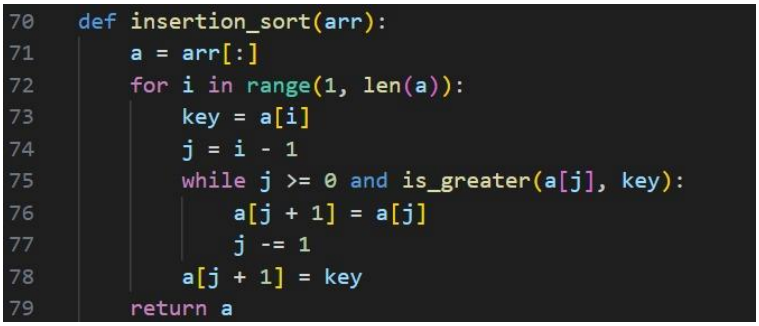
Gambar 2.2.1.1(b)

- Kode *def is_greater(a, b)*: mendefinisikan sebuah fungsi digunakan untuk **membandingkan dua data**, yaitu a dan b, untuk menentukan mana yang memiliki nilai lebih besar berdasarkan aturan tertentu.

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

Gambar 2.2.1.1(c)

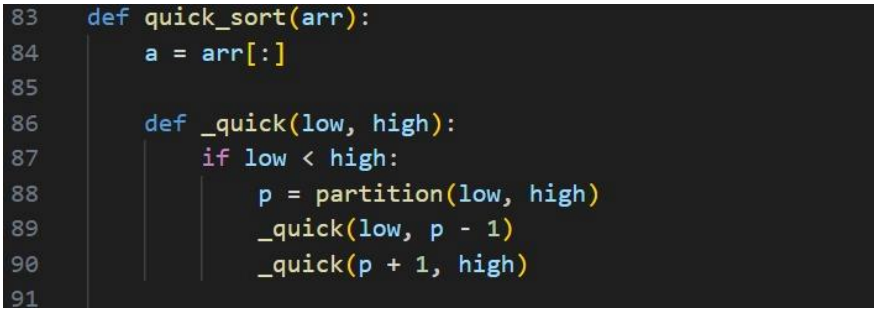
- Kode *def bubble_sort(arr):* berfungsi ini digunakan untuk **mengurutkan data** menggunakan algoritma **Bubble Sort**.
- Perulangan luar *for i in range(n):* ini digunakan untuk **mengatur jumlah proses pengurutan**. Setiap iterasi memastikan bahwa satu elemen terbesar akan berpindah ke posisi akhir.
- *if is_greater(a[j], a[j + 1]):* memanggil fungsi **is_greater** untuk membandingkan dua elemen yang bersebelahan Jika elemen di indeks j lebih besar dari elemen di indeks j + 1, maka kondisi ini bernilai benar.



```
70  def insertion_sort(arr):
71      a = arr[:]
72      for i in range(1, len(a)):
73          key = a[i]
74          j = i - 1
75          while j >= 0 and is_greater(a[j], key):
76              a[j + 1] = a[j]
77              j -= 1
78          a[j + 1] = key
79      return a
```

Gambar 2.2.1.1(d)

- Kode ini *def insertion_sort(arr):* mendefinisikan sebuah fungsi bernama **insertion_sort**. Fungsi ini digunakan untuk **mengurutkan data** menggunakan algoritma **Insertion Sort**.



```
83  def quick_sort(arr):
84      a = arr[:]
85
86      def _quick(low, high):
87          if low < high:
88              p = partition(low, high)
89              _quick(low, p - 1)
90              _quick(p + 1, high)
91
```

Gambar 2.2.1.1(e)

- Kode *def quick_sort(arr):* mendefinisikan sebuah fungsi bernama **quick_sort**. Fungsi ini digunakan untuk **mengurutkan data** menggunakan algoritma **Quick Sort**.
- *def _quick(low, high):* Fungsi ini digunakan untuk menjalankan proses Quick Sort secara **rekursif** pada sebagian array.
- Memastikan proses pengurutan dilakukan jikan indeks low lebih kecul dari high menggunakan code *if low < high:*
- Kode ini memanggil fungsi **partition** untuk membagi array yang berfungsi untuk, memilih sebuah **pivot,** menempatkan pivot pada posisi yang benar, memisahkan elemen yang lebih kecil dan lebih besar dari pivot

```
92      def partition(low, high):
93          pivot = a[high]
94          i = low - 1
95          for j in range(low, high):
96              if not is_greater(a[j], pivot):
97                  i += 1
98                  a[i], a[j] = a[j], a[i]
99          a[i + 1], a[high] = a[high], a[i + 1]
00          return i + 1
01
02      _quick(0, len(a) - 1)
03      return a
```

Gambar 2.2.1.1(f)

- *pivot = a[high]* menentukan **pivot**, yaitu elemen pembanding utama. Pivot diambil dari elemen terakhir pada bagian array yang sedang diproses.
- Perulangan *for j in range(low, high):* digunakan untuk **menelusuri elemen array** dari indeks low sampai high - 1. Setiap elemen akan dibandingkan dengan pivot.
- Kondisi *if not is_greater(a[j], pivot*):mengecek apakah elemen a[j] **tidak lebih besar dari pivot**. Artinya, elemen tersebut lebih kecil atau sama dengan pivot berdasarkan aturan perbandingan dari fungsi is_greater.

```
107 def benchmark(func, data):
108     start = time.time()
109     result = func(data)
110     end = time.time()
111     return result, end - start
112
113
114
115 filename = "Copy of Kelas_70 revisi II.xlsx"
116
117 root = Node.build_general_tree_from_excel(filename, max_children=3)
118 data_flat = flatten_tree(root)
119
120 bubble_res, t1 = benchmark(bubble_sort, data_flat)
121 insertion_res, t2 = benchmark(insertion_sort, data_flat)
122 quick_res, t3 = benchmark(quick_sort, data_flat)
123
```

Gambar 2.2.1.2 (a)

- *def benchmark(funt, data)* berfungsi untuk mengukur waktu eksekusi suatu fungsi terhdap data
- Baris *start = time.time()* mencatat waktu awal sebelum fungsi dijalankan.
- Setelah fungsi selesai dijalankan, baris *end = time.time()* mencatat waktu akhir eksekusi.
- Fungsi *return result, end - start* mengembalikan hasil proses dan lama waktu eksekusi dari fungsi tersebut.
- Baris *filename = "Copy of Kelas_70 revisi II.xlsx"* menyimpan nama file Excel yang akan digunakan sebagai sumber data.
- Kode *root = Node.build_general_tree_from_excel(filename, max_children=3*) membaca data dari file Excel dan membangun struktur general tree

```
126  def print_data(title, data):
127      print(f"\n=== {title} ===")
128      for nim, bb, nama in data:
129          print(f"NIM: {nim} | BB: {bb} | Nama: {nama}")
130
131  print_data("Hasil Bubble Sort (BB Terkecil)", bubble_res)
132  print_data("Hasil Insertion Sort (BB Terkecil)", insertion_res)
133  print_data("Hasil Quick Sort (BB Terkecil)", quick_res)
134
135  print("\n=== WAKTU EKSEKUSI ===")
136  print(f"Bubble Sort    : {t1:.6f} detik")
137  print(f"Insertion Sort : {t2:.6f} detik")
138  print(f"Quick Sort     : {t3:.6f} detik")
139
```

Gambar 2.2.1.2 (b)

- Fungsi *def print_data(title, data):* digunakan untuk menampilkan data hasil sorting ke layar dengan judul tertentu.
- Baris Print *(f"\n=== {title} ===")* mencetak judul tampilan sesuai dengan parameter title, serta menambahkan baris baru agar output lebih rapi.
- Baris *print(f"NIM: {nim} | BB: {bb} | Nama: {nama}")* menampilkan informasi NIM, BB, dan Nama untuk setiap data dalam format yang mudah dibaca.
- Baris *print("\n=== WAKTU EKSEKUSI ===")* mencetak judul bagian untuk menampilkan waktu eksekusi dari masing-masing algoritma sorting.

BAB III

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Proyek akhir "Perbandingan Algoritma Sorting dan Implementasi General Tree pada Data Berat Badan Mahasiswa" telah berhasil diimplementasikan dan memberikan pemahaman praktis mengenai penerapan algoritma sorting serta struktur data dalam pemrograman. Kesimpulan utama dari proyek ini adalah sebagai berikut:

1. **Implementasi Algoritma Sorting Berhasil Dilakukan**
   Algoritma Bubble Sort, Insertion Sort, dan Quick Sort berhasil diimplementasikan menggunakan bahasa pemrograman Python. Ketiga algoritma tersebut mampu mengurutkan data berat badan mahasiswa kelas M24A dan M2024B berdasarkan prioritas berat badan, NIM, dan nama sesuai dengan aturan perbandingan yang telah ditentukan.
2. **Perbedaan Performa Algoritma Dapat Diamati**
   Hasil pengujian menunjukkan adanya perbedaan performa antar algoritma. Quick Sort memiliki waktu eksekusi yang paling efisien terutama untuk data berjumlah lebih besar, sedangkan Bubble Sort dan Insertion Sort lebih sesuai untuk dataset kecil atau sebagai sarana pembelajaran konsep dasar pengurutan data.
3. **Penerapan Struktur Data General Tree Berjalan dengan Baik**
   General Tree berhasil digunakan sebagai media penyimpanan dan pengelompokan data mahasiswa sebelum dilakukan proses pengurutan. Meskipun struktur ini tidak menghasilkan urutan secara otomatis, General Tree berperan penting dalam membantu pemahaman konsep hierarki data dan alur pengolahan data sebelum disortir.
4. **Implementasi Menggunakan *Visual Code* Mendukung Proses Pengembangan**
   Penggunaan *Visual Code* sebagai lingkungan pemrograman mempermudah proses implementasi, pengujian, dan analisis kode. Dengan bantuan pengukuran waktu eksekusi, proses evaluasi performa algoritma dapat dilakukan secara efektif dan terstruktur.

Secara keseluruhan, proyek ini berhasil menunjukkan bagaimana algoritma sorting dan struktur data dapat diterapkan secara nyata untuk mengolah data sederhana. Selain menghasilkan data yang terurut, proyek ini juga memberikan pemahaman yang lebih mendalam mengenai karakteristik algoritma, efisiensi pemrograman, dan penerapan konsep struktur data dalam pengembangan program.

B. Saran

Berdasarkan implementasi proyek ini, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan dan peningkatan di masa mendatang:

1. **Penambahan Variasi Dataset dan Skala Data**
   Proyek ini menggunakan data berat badan mahasiswa kelas M24A dan M2024B dengan jumlah data terbatas. Untuk pengembangan selanjutnya, disarankan menambahkan jumlah data yang lebih besar atau menggunakan dataset dengan karakteristik berbeda agar analisis performa algoritma sorting menjadi lebih komprehensif dan mendekati kondisi nyata.
2. **Pengembangan Visualisasi Hasil Pengurutan**
   Saat ini, hasil pengurutan ditampilkan dalam bentuk output teks. Disarankan untuk menambahkan visualisasi, seperti grafik atau tabel interaktif (misalnya menggunakan matplotlib atau pandas), agar perbedaan hasil dan performa algoritma sorting dapat dipahami dengan lebih mudah dan menarik.
3. **Penambahan Algoritma Sorting Lainnya**
   Untuk memperkaya analisis, proyek ini dapat dikembangkan dengan menambahkan algoritma sorting lain seperti Merge Sort atau Heap Sort. Dengan demikian, perbandingan performa dan efisiensi algoritma dapat dilakukan secara lebih luas dan mendalam.
4. **Optimasi dan Analisis Waktu Eksekusi yang Lebih Detail**
   Pengukuran waktu eksekusi saat ini masih bersifat sederhana. Disarankan untuk melakukan pengujian berulang dan menghitung rata-rata waktu eksekusi agar hasil analisis performa algoritma lebih akurat dan reliabel.
5. **Pengembangan Struktur Data Tambahan**
   Selain General Tree, proyek ini dapat dikembangkan dengan menerapkan struktur data lain seperti Binary Search Tree atau Graph untuk membandingkan efektivitas masing-masing struktur dalam pengolahan dan pengurutan data.