Analisis Kompleksitas Algoritma untuk Menghitung Rata-rata Nilai Ujian

Izzulhaq Mahardika (1305220010)
Devin Mairhan (1305220073)
Aditya Aulia Rahman(1305220005)
Valentino Fredrick Albert Mamesah (1305223104)
School of Computing
Telkom University
Bandung, Indonesia

Skema

- SKEMA 1:

Buatlah suatu algoritma yang dapat menyelesaikan suatu permasalahan sederhana. Lalu:

a. Buktikan kebenaran dari algoritma
b. Tentukan fungsi waktu dan analisis Kompleksitas Algoritmanya.
c. Buatlah plot T(n) untuk n=100, 200, 400, 800.
d. Analisis hasil yang didapatkan.

Skema yang kami pilih untuk tugas besar Analisis Kompleksitas algoritma adalah skema pertama.

I. PENDAHULUAN

Kompleksitas algoritma adalah ukuran efisiensi dari suatu algoritma. Kompleksitas algoritma dapat diukur dari waktu yang dibutuhkan bagi algoritma untuk menyelesaikan semua langkahnya atau ruang memori yang dibutuhkan algoritma. Analisis kompleksitas waktu algoritma penting dilakukan untuk memilih algoritma yang paling efisien untuk menyelesaikan suatu masalah.

Dalam laporan ini, akan dianalisis kompleksitas waktu algoritma untuk menghitung rata-rata dari sekumpulan data nilai mahasiswa. Algoritma ini akan diimplementasikan dalam bahasa pemrograman Python.

Algoritma ini bertujuan untuk membantu menghitung rata-rata nilai agar dapat memberikan indikasi umum tentang kinerja mahasiswa secara keseluruhan. Rata-rata memberikan gambaran umum tentang tingkat pusat dari sekelompok nilai, sehingga dapat memberikan informasi yang berguna untuk memahami karakteristik keseluruhan dari data nilai mahasiswa.

Analisis kompleksitas waktu algoritma untuk menghitung rata-rata memiliki tujuan untuk memahami seberapa efisien algoritma tersebut dalam menangani data seiring dengan peningkatan ukuran data. Dengan mengevaluasi kompleksitas waktu, kita dapat menentukan seberapa cepat algoritma dapat memberikan hasil, dan ini memiliki dampak langsung pada kinerja aplikasi atau sistem yang menggunakan algoritma tersebut.

II. PEMBAHASAN

A. Analisis Masalah

Data yang akan dianalisa adalah nilai dari 100, 200, 400 hingga 800 siswa. Nilai rata-rata dari sekumpulan data dapat dihitung dengan menggunakan rumus berikut:

Rata - rata nilai mahasiswa = Jumlah nilai keseluruhan mahasiswa / Jumlah mahasiswa

Algoritma untuk menghitung nilai rata-rata dari sekumpulan data dapat diimplementasikan sebagai berikut:

```
def analisis_nilai_ujian(daftar_nilai, jumlah_siswa):

    total_nilai = 0
    rata_rata = 0
    indeks = 0

    while indeks < jumlah_siswa:
        total_nilai += daftar_nilai[indeks]
        indeks += 1
        rata_rata = total_nilai / indeks

    return rata_rata
```

Gambar 1. Fungsi menghitung nilai rata-rata

Fungsi ini menghitung rata-rata nilai ujian dari suatu daftar nilai siswa dengan menggunakan perulangan while. Variabel indeks digunakan untuk mengakumulasi total nilai dan menghitung rata-rata sepanjang iterasi, dan nilai rata-rata tersebut dikembalikan sebagai hasil fungsi.

B. Pembuktian Kebenaran Algoritma

```
function analisis_nilai_ujian(A, n):
     sum := 0
     i := 0
     avg := 0

     while i < n do:
          sum := sum + A[i]
          i = i + 1
          avg = sum / i
     endwhile

     return avg
```

Algoritma analisis_hasil_ujian(A, n) akan dibuktikan kebenarannya menggunakan metode pembuktian algoritma iterative, berikut adalah langkah-langkah pembuktiannya.

1. Claim

   Fungsi analisis_nilai_ujian(A, n) returns $\frac{\sum_{i=0}^{n} A[i]}{n}$

2. Fact about algorithm

   $sum_0 = 0$

   $i_0 = 0$

   $avg_0 = 0$

   $sum_{j+1} = sum_j + A[i_j]$

   $i_{j+1} = i_j + 1$

   $avg_{j+1} = \frac{sum_{j+1}}{i_{j+1}}$

3. Loop invariant

   $$\forall_{j \geq 0}, \quad sum_j = \sum_{i=0}^{j} A[i]$$

   $$i_j = j$$

   $$avg_j = \frac{sum_j}{i_j} = \frac{\sum_{i=0}^{j} A[i]}{j}$$

   a. Basis

   $$j = 0 \rightarrow sum_0 = \sum_{i=0}^{0} A[i] = 0$$

   $$i_0 = 0$$

   $$avg_0 = \frac{sum_0}{i_0} = \frac{0}{0} \quad (undefined)$$

   Langkah basis terbukti benar.

   b. Asumsikan loop invariant benar untuk sembarang j = k, sehingga didapatkan

   $$\forall_{k \geq 0}, \quad sum_k = \sum_{i=0}^{k} A[i]$$

   $$i_k = k$$

   $$avg_k = \frac{sum_k}{i_k} = \frac{\sum_{i=0}^{k} A[i]}{k}$$

   c. Buktikan loop invariant untuk j = k+1, didapatkan

   $$\forall_{k \geq 0}, \quad sum_{k+1} = \sum_{i=0}^{k+1} A[i]$$

   $$i_{k+1} = k + 1$$

   $$avg_{k+1} = \frac{sum_{k+1}}{i_{k+1}} = \frac{\sum_{i=0}^{k+1} A[i]}{k+1}$$

Berikut adalah pembuktian dari algoritma analisis_hasil_ujian(A, n).

$$sum_{k+1} = sum_k + A[i_k] \quad (by\ fact)$$
$$= \sum_{i=0}^{k} A[i] + A[i_k] \quad (by\ assume)$$
$$= \sum_{i=0}^{k+1} A[i]$$

Variabel sum terbukti benar.

$$i_{k+1} = i_k + 1 \quad (by\ fact)$$
$$= k + 1 \quad (by\ assume)$$
$$= k + 1$$

Variabel i terbukti benar.

$$avg_{k+1} = \frac{sum_{k+1}}{i_{k+1}}$$
$$= \frac{\sum_{i=0}^{k+1} A[i]}{k+1}$$

Variabel avg terbukti benar.

4. Conclusion
   a. Claim

      Algoritma analisis_hasil_ujian(A,n) berakhir dengan

$$avg = \frac{\sum_{i=0}^{n} A[i]}{n}$$

   b. Termination
      - Algoritma berakhir saat i = n
      - Asumsikan algoritma berhenti saat iterasi ke *t*
      - Maka menggunakan loop invariant didapatkan:

$$i_t = t$$
$$n = t$$
$$t = n$$

   c. Result

$$avg_t = \frac{\sum_{i=0}^{t} A[i]}{t} = \frac{\sum_{i=0}^{n} A[i]}{n}$$

Dengan menggunakan metode pembuktian algoritma iteratif, dapat dinyatakan bahwa hasil yang diperoleh dari algoritma untuk menghitung rata-rata nilai mahasiswa sesuai dengan klaim yang dibuat, sehingga algoritma terbukti benar.

C. Analisis Kompleksitas Waktu

```
function analisis_nilai_ujian(A, n):
    sum := 0
    avg := 0

    for i = 0 To n-1 Do:.....((n-1)-0)+1 = n
        sum := sum + A[i] .....(bo2)
        avg = sum / i.....(bo1)
    endwhile

    return avg
```

Gambar 2. Fungsi menghitung nilai rata-rata

Bentuk algoritma diatas merupakan bentuk lain dari algoritma yang kami buat menggunakan perulangan for namun fungsionalitas algoritma tersebut masih sama.

Algoritma yang kelompok kami buat merupakan algoritma iteratif yang berfungsi untuk menghitung rata rata perolehan nilai dari jumlah siswa tertentu dengan cara menambahkan nilai tiap mahasiswa sejumlah n lalu dibagi jumlah siswanya. Terdapat beberapa metode yang dapat digunakan untuk menentukan kompleksitas waktu dari suatu algoritma, namun pada kesempatan ini kami menggunakan dua metode yaitu :

- *Direct Observation Method*

Metode ini dilakukan dengan cara mengamati langsung jumlah iterasi yang dibutuhkan untuk saikan setiap persamaan.

- *Sum Notation Method*

Sum Notation Method adalah metode analisis kompleksitas waktu yang menggunakan notasi sigma (Σ) untuk merepresentasikan jumlah operasi yang dilakukan oleh sebuah algoritma. Metode ini lebih formal dan matematis dibandingkan dengan Direct Observation Method, dan dapat digunakan untuk menganalisis fungsi-fungsi yang lebih kompleks. Terdapat 2 aturan dasar *Sum Notation Method* yaitu :

$$\sum_{i=k}^{n} ca_i = c\sum_{i=k}^{n} a_i$$

$$\sum_{i=k}^{n} (a_i \pm b_i) = \sum_{i=k}^{n} a_i \pm \sum_{i=k}^{n} b_i$$

Gambar 3. Aturan dasar Sum Notation Method

Mencari kompleksitas waktu menggunakan metode *Direct Observation Method*

Inisialisasi Cop dan Cn
di dapatkan,

Cop = 2 dan

Cn = n

Sehingga di dapatkan T(n) =
Cn * Cop = 2n ∈ O(n).

Gambar 4. Mencari kompleksitas waktu

Dalam *Direct Observation Method* kita perlu menentukan operasi dasar dari algoritmanya dalam kasus ini terdapat dua operasi dasar di dalam perulangan "for i = 0 to n-1" dengan menggunakan rumus saat terjadi perulangan

for a to b

b – a + 1

Gambar 5. Rumus operasi perulangan

Dalam kasus ini a merupakan i = 0 dan b = n-1.

Sehingga setelah melakukan perhitungan dihasilkan seperti pada Gambar 4, karena perulangan dan operasi dasarnya memiliki perbedaan level maka akan didapatkan hasil T(n) hasil perkalian Cn dengan Copnya yaitu 2n anggota bilangan big O(n).

Mencari kompleksitas waktu menggunakan metode *Sum Notation Method*

$$\sum_{i=k}^{n} 2 = (n - k + 1) * 2$$

$$2\sum_{i=0}^{n-1} i = (n-1) - 0 + 1 = 2 * n$$

$$T(n) = 2n \in O(n).$$

Gambar 6. Mencari kompleksitas waktu T(n)

Dikarenakan kita memiliki operasi dasar sejumlah 2 maka notasi sumnya akan berbentuk Gambar 6 dengan mengimplementasikan rumus (b-a+1) lalu dikalikan jumlah operasi dasarnya. Dengan mengimplementasikan aturan dasar Gambar 3 sehingga didapatkan untuk hasil T(n)-nya adalah 2n anggota bilangan big O(n).

Dapat kita lihat dengan menggunakan dua metode untuk menemukan kompleksitas algoritmanya di dapatkan hasil yang sama untuk kedua metode yang digunakan yaitu :

$$T(n) = 2n \in O(n).$$

Gambar 7. Hasil pencarian kompleksitas waktu T(n)

Dalam algoritma untuk mencari nilai rata rata yang telah dibuat tidak mungkin untuk memiliki kasus terbaik, kasus terburuk, ataupun kasus rata ratanya. Karena algoritma ini tidak bergantung pada nilai masukannya atau tidak bergantung pada suatu posisi seperti berhenti di tengah tengah saat prosesnya dijalankan serta kasus perulangannya pasti akan dijalankan dari awal hingga akhir maka kompleksitas waktunya selalu mendapatkan kasus atau tipe *worst case*.

D. Fungsionalitas Program

```
def cari_jumlah_siswa(daftar_nilai):
    jumlah_siswa = 0

    for _ in daftar_nilai:
      jumlah_siswa += 1

    return jumlah_siswa
```

Gambar 8. Fungsi mencari jumlah siswa

Fungsi ini memiliki satu parameter yaitu daftar_nilai, parameter ini merupakan sebuah array yang berisikan nilai siswa siswa yang berjumlah n. Fungsi ini melakukan iterasi terhadap array daftar_nilai, pada setiap iterasi akan ditambahkan nilai 1 pada variabel jumlah_siswa sehingga pada akhir perulangan atau iterasi variabel jumlah_siswa akan menyimpan hasil penjumlahan jumlah_siswa sebanyak isi array daftar_nilai. Dalam kata lain fungsi ini berguna untuk mengetahui berapa panjang array daftar_nilai yang direpresentasikan sebagai nilai nilai pelajaran siswa.

```
def generate_daftar_nilai(jumlah_siswa):
    daftar_nilai = [random.randint(0, 100) for _ in range(jumlah_siswa)]
    return daftar_nilai
```

Gambar 9. Fungsi membangkitkan daftar nilai

Fungsi ini berfungsi untuk menghasilkan daftar nilai siswa secara acak. terdapat satu parameter yaitu jumlah_siswa, parameter ini merupakan jumlah siswa yang akan dihasilkan nilai nilainya secara acak. Dengan menggunakan salah satu library yang terdapat pada python yaitu random.randint() yang berfungsi untuk menghasilkan nilai acak, dalam program ini nilai acak untuk nilai siswa-siswa kami batasi untuk batas bawah 0 dan batas atasnya 100. Fungsi ini akan menyimpan nilai nilai acak tersebut ke dalam sebuah array yaitu array daftar_nilai

```
def print_analisis(n_values, nilai_siswa, jumlah_siswa):

    print(f"List nilai para siswa")
    print(nilai_siswa)

    rata_siswa = analisis_nilai_ujian(nilai_siswa, jumlah_siswa)

    print(f"""
    Analisis Nilai Siswa:
    --------------------
    Jumlah Siswa = {jumlah_siswa}
    Nilai rata-rata siswa adalah {rata_siswa:.2f}
    """
    )
```

Gambar 10. Fungsi mencetak analisis nilai

Fungsi bernama print_analisis menerima tiga parameter: n_values , nilai_siswa (list nilai siswa), dan jumlah_siswa (jumlah siswa).

- Mencetak List Nilai Siswa:

Mencetak judul "List nilai para siswa" diikuti oleh mencetak list nilai siswa.

- Menghitung Rata-rata Nilai Siswa:

Memanggil fungsi analisis_nilai_ujian untuk menghitung rata-rata nilai siswa dengan menggunakan list nilai dan jumlah siswa.

- Mencetak Analisis Nilai Siswa:

Mencetak analisis nilai siswa, termasuk jumlah siswa dan rata-rata nilai, dengan menggunakan f-string.

Dengan fungsi ini, kita dapat mencetak analisis nilai siswa dengan memberikan list nilai dan jumlah siswa sebagai argumen.

```
# Analisis kompleksitas waktu
n_values = [100, 200, 400, 800]
times = []

for n in n_values:
    daftar_nilai = generate_daftar_nilai(n)
    jumlah_siswa = cari_jumlah_siswa(daftar_nilai)

    print_analisis(n, daftar_nilai, jumlah_siswa)

    # Hitung waktu komputasi untuk analisis_nilai_ujian
    start_time = time.time()
    analisis_nilai_ujian(daftar_nilai, jumlah_siswa)
    end_time = time.time()
    waktu_komputasi = end_time - start_time
    print(f"Waktu komputasi saat n={n} : {waktu_komputasi:.6f}")
    times.append(waktu_komputasi)
    print("="*65)
```

Gambar 11. Kode inisialisasi nilai dan mendapatkan waktu komputasi

Kode ini melakukan pengukuran waktu eksekusi fungsi `analisis_nilai_ujian` untuk berbagai ukuran masukan (`n_values`). Setiap iterasi mencetak analisis nilai siswa, menghitung waktu komputasi, dan menyimpannya. Akhirnya, hasil waktu eksekusi divisualisasikan dalam sebuah plot untuk menganalisis kompleksitas waktu fungsi tersebut.

Visualisasi gambar plot T(n) berbentuk linier

```
print()
# Visualisasi plot T(n)
plt.plot(n_values, times, marker='o')
plt.xlabel('Jumlah Siswa (n)')
plt.ylabel('Waktu Eksekusi (s)')
plt.title('Kompleksitas Waktu Analisis Nilai Ujian')
plt.show()
```

Gambar 12. Kode untuk plot linear

Bagian ini mencetak plot visualisasi waktu eksekusi (`T(n)`) dari fungsi `analisis_nilai_ujian` untuk berbagai ukuran masukan (`n_values`). Plot tersebut menunjukkan bagaimana waktu eksekusi berubah seiring dengan peningkatan jumlah siswa.

Visualisasi gambar plot T(n) berbentuk bar

```
colors = plt.cm.viridis(np.linspace(0, 1, len(n_values)))

plt.figure(figsize=(10, 6))
bars = plt.bar(n_values, times, color=colors, alpha=0.7, width=50, label=[f'Jumlah Siswa = {n}' for n in n_values])

for bar, time_val in zip(bars, times):
    plt.text(bar.get_x() + bar.get_width() / 2 - 20, bar.get_height() + 0.02, f'{time_val:.4f}', ha='center', color='black')

plt.xlabel('Jumlah Siswa (n)')
plt.ylabel('Waktu Eksekusi (s)')
plt.title('Kompleksitas Waktu Analisis Nilai Ujian')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.legend(title='Penjelasan Warna', loc='upper left', bbox_to_anchor=(1, 1))

plt.show()
```

Gambar 13. Kode untuk plot bar

Kode ini adalah modifikasi dari visualisasi plot sebelumnya. Sekarang, setiap bar pada plot memiliki warna yang berbeda berdasarkan colormap 'viridis' dan memiliki label yang menunjukkan jumlah siswa. Selain itu, ditambahkan teks di atas setiap bar yang menampilkan nilai waktu eksekusi.

E. Hasil Output Program

Hasil running time dari algoritma untuk mencari rata rata nilai mahasiswa, untuk n = 100, 200, 400, 800 adalah sebagai berikut.

Hasil output program.

```
List nilai para siswa
[15, 69, 13, 35, 65, 42, 21, 99, 0, 27, 85, 43, 6, 24, 48, 50, 4, 29, 63,

    Analisis Nilai Siswa:
    ---------------------
    Jumlah Siswa = 100
    Nilai rata-rata siswa adalah 50.27

Waktu komputasi saat n=100 : 0.000026
================================================================
List nilai para siswa
[88, 34, 47, 95, 85, 47, 30, 72, 22, 66, 75, 85, 76, 3, 20, 68, 85, 73, 80

    Analisis Nilai Siswa:
    ---------------------
    Jumlah Siswa = 200
    Nilai rata-rata siswa adalah 46.05

Waktu komputasi saat n=200 : 0.000046
================================================================
List nilai para siswa
[85, 32, 38, 49, 97, 19, 62, 18, 97, 38, 51, 47, 30, 29, 94, 16, 62, 76, 8

    Analisis Nilai Siswa:
    ---------------------
    Jumlah Siswa = 400
    Nilai rata-rata siswa adalah 51.23

Waktu komputasi saat n=400 : 0.000100
================================================================
List nilai para siswa
[70, 46, 23, 13, 21, 69, 9, 98, 41, 2, 1, 82, 26, 35, 33, 14, 30, 92, 82,

    Analisis Nilai Siswa:
    ---------------------
    Jumlah Siswa = 800
    Nilai rata-rata siswa adalah 52.16

Waktu komputasi saat n=800 : 0.000212
================================================================
```

Gambar 14. Hasil analisis nilai siswa

Menampilkan list nilai mahasiswa sejumlah n siswa dalam bentuk array dan menampilkan hasil rata rata serta waktu komputasi setiap daftar nilai.

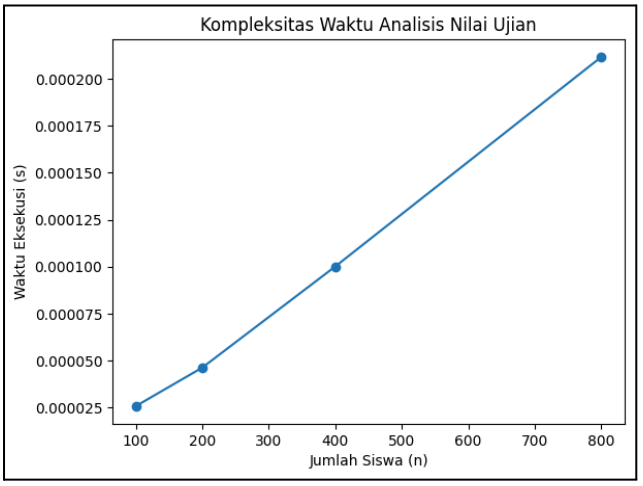
Tabel Running Time untuk hasil rata rata nilai dan jumlah siswa.

Jumlah siswa	Rata-rata nilai	Waktu komputasi nilai
100	50,27	0,000026
200	46,05	0,000046
400	51,23	0,000100
800	52,16	0,000212

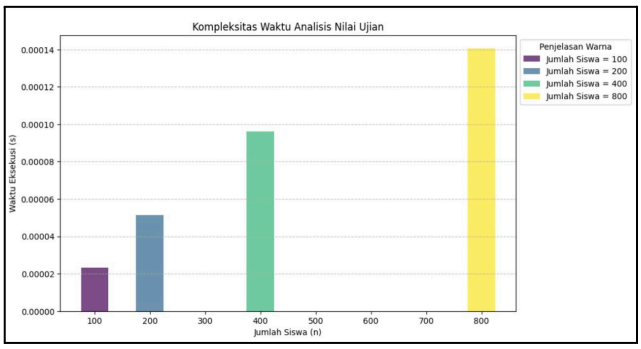
Tabel 1. hasil visualisasi algoritma

Setelah menjalankan program algoritma yang berfungsi untuk menghitung nilai rata rata siswa kami mendapatkan output yang dapat dilihat pada tabel diatas untuk masing masing masukan ketika variabel n = 100,200,400,800 adalah masing masing saat n = 100 maka waktu komputasinya 0,000026, n = 200 maka waktu komputasinya 0,000046 n = 400 maka waktu komputasinya 0,000100, dan terakhir saat n = 800 maka waktu komputasinya 0,000212.

Hasil visualisasi plot fungsi waktu



Gambar 15. Plot hasil visualisasi linier



Gambar 16. Plot hasil visualisasi bar

Kedua grafik ini menunjukkan hubungan antara jumlah siswa(n) dengan waktu eksekusi (s) algoritma analisis nilai ujian.

Grafik pertama memiliki bentuk linier dan setiap dotnya memiliki arti untuk setiap jumlah siswa tertentu memiliki waktu eksekusi sebesar y. Sedangkan pada grafik kedua hubungan antara jumlah mahasiswa dan waktu eksekusinya digambarkan dengan bentuk bar dan setiap warna memiliki artian berbeda untuk warna kuning adalah saat jumlah siswa 800 siswa, hijau 400 siswa, biru 200 siswa dan ungu 100 siswa. Kedua grafik tersebut sama sama menunjukan bahwa waktu eksekusi algoritma sebanding dengan jumlah siswa semakin banyak jumlah siswanya maka akan semakin tinggi juga waktu eksekusinya.

III. PENUTUP

Kesimpulan

Kesimpulan dari metode pembuktian algoritma iteratif untuk menghitung rata-rata nilai mahasiswa adalah bahwa algoritma tersebut telah terbukti benar sesuai dengan klaim awal. Langkah-langkah iteratif yang terstruktur memastikan bahwa algoritma memberikan hasil yang konsisten dengan harapan dan berhasil mencapai tujuan perhitungan nilai mahasiswa.

Selain itu, analisis fungsi kompleksitas algoritma iterative menunjukkan bahwa waktu eksekusi algoritma (T(n)) sebanding dengan jumlah anggota bilangan (n) dengan notasi big O(n). Dalam visualisasi fungsi waktu menggunakan dua diagram, terlihat bahwa semakin besar jumlah siswa (n), waktu eksekusi algoritma juga meningkat, mengindikasikan bahwa algoritma memiliki kompleksitas linier.

Dengan demikian, keseluruhan hasil pembuktian, analisis kompleksitas, dan visualisasi waktu memberikan keyakinan bahwa algoritma iterative untuk menghitung rata-rata nilai mahasiswa dapat diandalkan, sesuai dengan klaim, dan efisien dalam menangani jumlah siswa yang beragam.

Daftar Pustaka

Gunawan, P. H. (2022). *Logika matematika untuk analisis algoritma*.

Lampiran

Link Google Colab:

https://colab.research.google.com/drive/1iSG6_aLg3O2LdWlqyhvWT117Zv6aPfiI?usp=sharing

Link video presentasi You Tube:

https://youtu.be/Dr4cJxj4qJc

Link video presentasi Drive (cadangan):

https://drive.google.com/drive/folders/12_SO9qqnLotwuatLbT71HFm6yEShqJJj?usp=sharing