Tugas Besar AKA.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```
#Dokumentasi Tugas Besar

import time #akses waktu
import matplotlib.pyplot as plt #akses visualisasi data
from prettytable import PrettyTable #untuk membuat tabel dengan rapih
import requests #untuk ambil data dari github
import cProfile #untuk mengukur waktu fungsi berjalan
import tracemalloc #untuk mengukur memori

nilai_N = []
waktu_iteratif = []
waktu_rekursif = []
memori_iteratif = []
memori_rekursif = []
```

```
#fungsi Mengambil dan membersihkan data dari sebuah tautan GitHub.
def baca_data(link_github, jumlah_data):
    try:
        #http request
        response = requests.get(link_github)
        response.raise_for_status()

        #memisahkan data dari separator ";"
        data = response.text.strip().split(';')

        #array menyimpan data yang sudah dipisahkan
        data_bersih = []

        #iterasi untuk memeriksa
        for item in data:
            try:
                item_bersih = item.strip() #memastikan item bersih dari spasi agar tiidak error saat casting
                if item_bersih:
                    data_bersih.append(float(item_bersih)) #casting
            except ValueError:
                continue #kalau gagal casting langsung skip

        return data_bersih[:jumlah_data] #data yang berhasil di casting disimpan ke array

    except requests.RequestException as e: #jika http request gagal
        print(f"Error mengambil data dari GitHub: {e}")
        return []
```

```
[95] def merge_sort_Iteratif(arr):
        def merge(arr, left, mid, right):
            left_half = arr[left:mid+1]
            right_half = arr[mid+1:right+1]

            i = j = 0
            k = left

            while i < len(left_half) and j < len(right_half):
                if left_half[i] <= right_half[j]:
                    arr[k] = left_half[i]
                    i += 1
                else:
                    arr[k] = right_half[j]
                    j += 1
                k += 1

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

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

        n = len(arr)
        curr_size = 1

        while curr_size < n:
            for start in range(0, n, 2*curr_size):
                mid = start + curr_size - 1
                end = min(start + 2*curr_size - 1, n - 1)

                merge(arr, start, mid, end)

            curr_size *= 2

        return arr
```

```
def merge_sort_rekursif(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = merge_sort_rekursif(arr[:mid])
    right = merge_sort_rekursif(arr[mid:])

    return merge_rekursif(left, right)

def merge_rekursif(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])

    return result
```

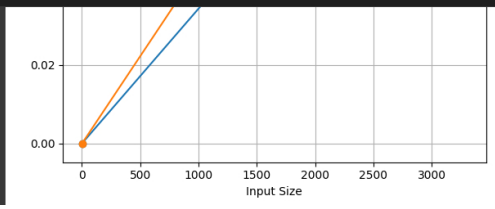
```
[97] def profile_sorting_algorithm(func, data): #fungsi untuk menghitung waktu fungsi rekursif dan iteratif
        tracemalloc.start()
        profiler = cProfile.Profile()
        profiler.enable()

        #menghitung waktu proses
        waktu_mulai = time.time()
        func(data.copy())
```

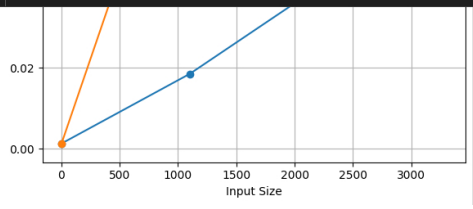
```
    waktu_selesai = time.time()

    profiler.disable()

    _, peak = tracemalloc.get_traced_memory()
    tracemalloc.stop()

    return waktu_selesai - waktu_mulai, peak/1024/1024
```

```
[98] def update_graph(nilai_N, waktu_iteratif, waktu_rekursif): # Fungsi untuk memperbarui grafik
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.plot(nilai_N, waktu_iteratif, label='Iteratif', marker='o')
    plt.plot(nilai_N, waktu_rekursif, label='Rekursif', marker='o')
    plt.title('Merge Sort: Waktu Eksekusi')
    plt.xlabel('Input Size')
    plt.ylabel('Waktu (detik)')
    plt.legend()
    plt.grid(True)

    plt.subplot(1, 2, 2)
    plt.plot(nilai_N, memori_iteratif, label='Memori Iteratif', marker='o')
    plt.plot(nilai_N, memori_rekursif, label='Memori Rekursif', marker='o')
    plt.title('Merge Sort: Penggunaan Memori')
    plt.xlabel('Input Size')
    plt.ylabel('Memori (MB)')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

```
def print_performance_table(nilai_N, waktu_iteratif, waktu_rekursif):
    table = PrettyTable()
    table.field_names = [
        "Input Size",
        "Iteratif Time (detik)",
        "Rekursif Time (detik)",
        "Selisih Waktu (detik)",
        "memori iteratif (MB)",
        "memori rekursif (MB)",
        "Selisih Memori (MB)"
    ]

    for n, waktu_iter, waktu_rek, iter_mem, rek_mem in zip(
        nilai_N, waktu_iteratif, waktu_rekursif,memori_iteratif, memori_rekursif
    ):
        table.add_row([
            n,
            f"{waktu_iter:.6f}",
            f"{waktu_rek:.6f}",
            f"{abs(waktu_iter - waktu_rek):.6f}",
            f"{iter_mem:.2f}",
            f"{rek_mem:.2f}",
            f"{abs(iter_mem - rek_mem):.2f}"
        ])

    print(table)
```

```
[104] def main():
    # URL RAW file dari GitHub
    link_github = 'https://raw.githubusercontent.com/AdithanaDharma/Iterative-vs-Recursive-Merge-Sort-for-Rating-on-MyAnimeList-Website/refs/heads/main/Data%20Rating%20MAL.txt'

    n = 1
    while n <= 4440:
        try:
            # Baca data
            data = baca_data(link_github, n)

            if not data:
                print("Gagal membaca data.")
                break

            # Profiling dan ukur waktu
            iterative_time, iter_mem = profile_sorting_algorithm(
                merge_sort_Iteratif,
                data
            )
            recursive_time, rek_mem = profile_sorting_algorithm(
                merge_sort_rekursif,
                data
            )

            # Tambahkan ke daftar
            nilai_N.append(n)
            waktu_iteratif.append(iterative_time)
            waktu_rekursif.append(recursive_time)
            memori_iteratif.append(iter_mem)
            memori_rekursif.append(rek_mem)

            # Cetak tabel performa
            print_performance_table(
                nilai_N,
                waktu_iteratif,
                waktu_rekursif
            )

            # Update grafik
            update_graph(
                nilai_N,
                waktu_iteratif,
                waktu_rekursif
            )

            # Naikkan ukuran input
            n += 1100

        except Exception as e:
            print(f"Terjadi kesalahan: {e}")
            break

if __name__ == "__main__":
    main()
```





Input Size	Iteratif Time (detik)	Rekursif Time (detik)	Selisih Waktu (detik)	memori iteratif (MB)	memori rekursif (MB)	Selisih Memori (MB)
1	0.000042	0.000020	0.000022	0.00	0.00	0.00
1101	0.037833	0.049019	0.011186	0.02	0.09	0.08
2201	0.056335	0.060324	0.003989	0.04	0.06	0.02
3301	0.083208	0.095565	0.012357	0.05	0.09	0.03
4401	0.126963	0.137280	0.010316	0.07	0.11	0.03