Laporan Tugas Besar

Mata Kuliah Analisis Kompleksitas Algoritma

Menghitung Nilai Rata-Rata (Mean) dan Nilai Maksimum

dari Dataset Nilai Matematika Siswa

Disusun oleh :

Stephani Maria Sianturi – 103052300083

Mahira Mumtaza – 103052300107

TELKOM UNIVERSITY

BANDUNG

2024

A. Deskripsi Studi Kasus

Studi kasus ini bertujuan untuk menganalisis data nilai ujian matematika yang diperoleh oleh sekelompok siswa dari berbagai kelas. Analisis ini difokuskan pada dua informasi utama, yaitu nilai maksimum dan nilai rata-rata dari dataset. Nilai maksimum digunakan untuk mengidentifikasi siswa dengan performa terbaik, sedangkan nilai rata-rata memberikan gambaran umum mengenai performa keseluruhan kelompok siswa.

Untuk mencapai tujuan ini, dua pendekatan akan diterapkan dalam perhitungan kedua informasi tersebut: pendekatan iteratif dan pendekatan rekursif. Masing-masing pendekatan ini akan diuji dengan ukuran dataset yang bervariasi untuk menganalisis kompleksitas waktu dan performa running time.

Secara umum, dataset terdiri dari serangkaian nilai ujian yang dikelompokkan menurut kelas. Berdasarkan nilai-nilai tersebut, tugas utama adalah:

1. Menentukan Nilai Maksimum: Mencari nilai tertinggi dalam dataset untuk mengidentifikasi siswa dengan performa terbaik.
2. Menghitung Nilai Rata-rata (Mean): Menghitung rata-rata nilai ujian seluruh siswa untuk mendapatkan gambaran performa kelas secara keseluruhan.

Setelah kedua perhitungan dilakukan dengan menggunakan pendekatan iteratif dan rekursif, analisis kompleksitas waktu dan perbandingan efisiensi masing-masing algoritma akan dilakukan. Hal ini untuk mengetahui algoritma mana yang lebih optimal ketika berhadapan dengan dataset dengan jumlah siswa yang bervariasi.

B. Deskripsi Algoritma yang Dipilih

Dalam studi kasus ini, dua jenis algoritma dipilih untuk menghitung nilai maksimum dan rata-rata dalam dataset nilai ujian matematika siswa: algoritma iteratif dan algoritma rekursif.

1. Algoritma Iteratif

Algoritma iteratif menggunakan struktur kontrol loop untuk mengakses dan memproses setiap elemen dalam dataset. Dalam hal ini, dataset adalah serangkaian nilai ujian siswa yang perlu diperiksa satu per satu. Dengan menggunakan loop (misalnya, for atau while), algoritma akan:

- Menentukan Nilai Maksimum, iterasi dimulai dengan nilai pertama, yang disimpan sebagai nilai maksimum sementara. Setiap elemen berikutnya dibandingkan dengan nilai maksimum sementara, dan jika ditemukan nilai yang lebih besar, nilai maksimum diperbarui.
- Menghitung Nilai Rata-rata, selama iterasi, semua nilai ujian dijumlahkan dan dibagi dengan jumlah total siswa untuk mendapatkan rata-rata.

Langkah-langkah Algoritma Iteratif:

- Mulai dengan nilai pertama sebagai nilai maksimum.
- Iterasi melalui dataset, bandingkan setiap nilai dengan nilai maksimum saat ini, dan perbarui jika ditemukan nilai yang lebih besar.
- Hitung jumlah total nilai dan bagi dengan jumlah siswa untuk mendapatkan rata-rata.

2. Algoritma Rekursif

Algoritma rekursif memecah masalah besar menjadi sub-masalah yang lebih kecil, dengan fungsi yang memanggil dirinya sendiri untuk memproses setiap elemen dataset secara berulang. Proses ini diulang hingga seluruh dataset diproses.

- Nilai Maksimum: Fungsi rekursif memanggil dirinya untuk membandingkan elemen saat ini dengan nilai maksimum yang ditemukan sejauh ini. Ketika elemen terakhir tercapai, fungsi akan mengembalikan nilai maksimum yang ditemukan.
- Nilai Rata-rata: Fungsi rekursif akan menambahkan setiap nilai ujian ke total yang dihitung, dan ketika semua elemen telah diproses, fungsi akan membagi total nilai dengan jumlah siswa untuk menghitung rata-rata.

Langkah-langkah Algoritma Rekursif:

- Tentukan basis kasus (misalnya, elemen pertama dalam dataset).
- Panggil fungsi rekursif untuk elemen berikutnya, bandingkan dengan nilai maksimum yang ditemukan sebelumnya, dan perbarui jika diperlukan.
- Gunakan rekursi untuk menghitung jumlah nilai dan jumlah elemen untuk mendapatkan rata-rata.

C. Implementasi Aplikasi

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
```

```
# Fungsi iteratif untuk mencari maksimum
def iterative_maximum(data):
    max_value = data[0]
    for num in data:
        if num > max_value:
            max_value = num
    return max_value
```

```
# Fungsi iteratif untuk menghitung rata-rata
def iterative_mean(data):
    total = 0
    for num in data:
        total += num
    return total / len(data)
```

```
# Fungsi rekursif untuk mencari maksimum
def recursive_maximum(data):
    if len(data) == 1:
        return data[0]
    mid = len(data) // 2
    left_max = recursive_maximum(data[:mid])
    right_max = recursive_maximum(data[mid:])
    return max(left_max, right_max)
```

```
# Fungsi rekursif untuk menghitung rata-rata
def recursive_sum(data):
    if len(data) == 1:
        return data[0]
    mid = len(data) // 2
    return recursive_sum(data[:mid]) + recursive_sum(data[mid:])
```

```
def recursive_mean(data):
    return recursive_sum(data) / len(data)
```

```python
# Fungsi untuk mengukur waktu eksekusi
def measure_time(func, data):
    start_time = time.time()
    func(data)
    end_time = time.time()
    return end_time - start_time
```

```python
# Pengujian pada berbagai ukuran input
input_sizes = [10, 100, 1000, 5000, 10000]
iterative_times_max = []
recursive_times_max = []
iterative_times_mean = []
recursive_times_mean = []
```

```python
for size in input_sizes:
    # Generate data acak
    data = np.random.randint(1, 1000, size).tolist()

    # Waktu eksekusi untuk maksimum
    iterative_times_max.append(measure_time(iterative_maximum, data))
    recursive_times_max.append(measure_time(recursive_maximum, data))

    # Waktu eksekusi untuk rata-rata
    iterative_times_mean.append(measure_time(iterative_mean, data))
    recursive_times_mean.append(measure_time(recursive_mean, data))
```

```python
# Plot untuk maksimum
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
plt.plot(input_sizes, iterative_times_max, label="Iterative Maximum", marker='o')
plt.plot(input_sizes, recursive_times_max, label="Recursive Maximum", marker='o')
plt.xlabel("Input Size")
plt.ylabel("Execution Time (s)")
plt.title("Execution Time Comparison: Maximum")
plt.legend()
plt.tight_layout()
plt.show()
```

```python
# Plot untuk rata-rata
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 2)
plt.plot(input_sizes, iterative_times_mean, label="Iterative Mean", marker='o')
plt.plot(input_sizes, recursive_times_mean, label="Recursive Mean", marker='o')
plt.xlabel("Input Size")
plt.ylabel("Execution Time (s)")
plt.title("Execution Time Comparison: Mean")
plt.legend()
plt.tight_layout()
plt.show()
```

```
# Menampilkan hasil waktu eksekusi untuk maksimum
print("Waktu Eksekusi untuk Maximum (Iteratif):")
for i, size in enumerate(input_sizes):
    print(f"n = {size}: {iterative_times_max[i]:.6f} detik")

print("\nWaktu Eksekusi untuk Maximum (Rekursif):")
for i, size in enumerate(input_sizes):
    print(f"n = {size}: {recursive_times_max[i]:.6f} detik")

# Menampilkan hasil waktu eksekusi untuk rata-rata
print("\nWaktu Eksekusi untuk Mean (Iteratif):")
for i, size in enumerate(input_sizes):
    print(f"n = {size}: {iterative_times_mean[i]:.6f} detik")

print("\nWaktu Eksekusi untuk Mean (Rekursif):")
for i, size in enumerate(input_sizes):
    print(f"n = {size}: {recursive_times_mean[i]:.6f} detik")
```

D. Analisis Efisiensi

Misalkan kita memiliki ukuran input n = 10000

- Waktu Eksekusi Iteratif Maksimum
  - Berdasarkan hasil pengukuran: waktu untuk n = 10000 adalah 0,000562 detik.
- Waktu Eksekusi Rekursif untuk Maksimum
  - Berdasarkan hasil pengukuran: Waktu untuk n = 10000 adalah 0.011323 detik.

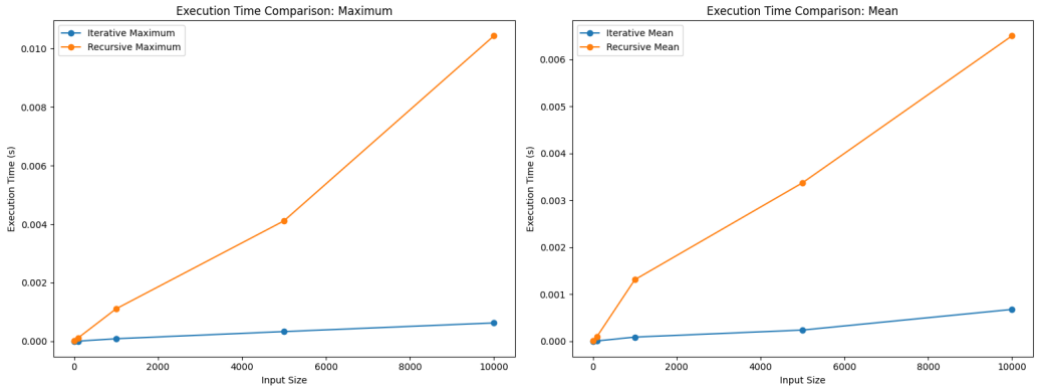
Kita menggunakan rasio waktu untuk perbandingan eksekusi antara rekursif dan iteratif:

Rasio Waktu = 0.011323/0,000562 ≈ 20.15

Kesimpulannya adalah untuk ukuran input 10000, metode rekursif lebih lambat 20 kali dibandingkan dengan metode iteratif.

Kompleksitas Waktu:

- Iteratif: T(n) = O(n)
  Iteratif lebih efisien karena tidak memerlukan overhead pemanggilan fungsi.
- Rekursif: T(n) = O(n log n)
- Rekursi memiliki overhead tinggi, sehingga lambat untuk input besar.



E. Referensi

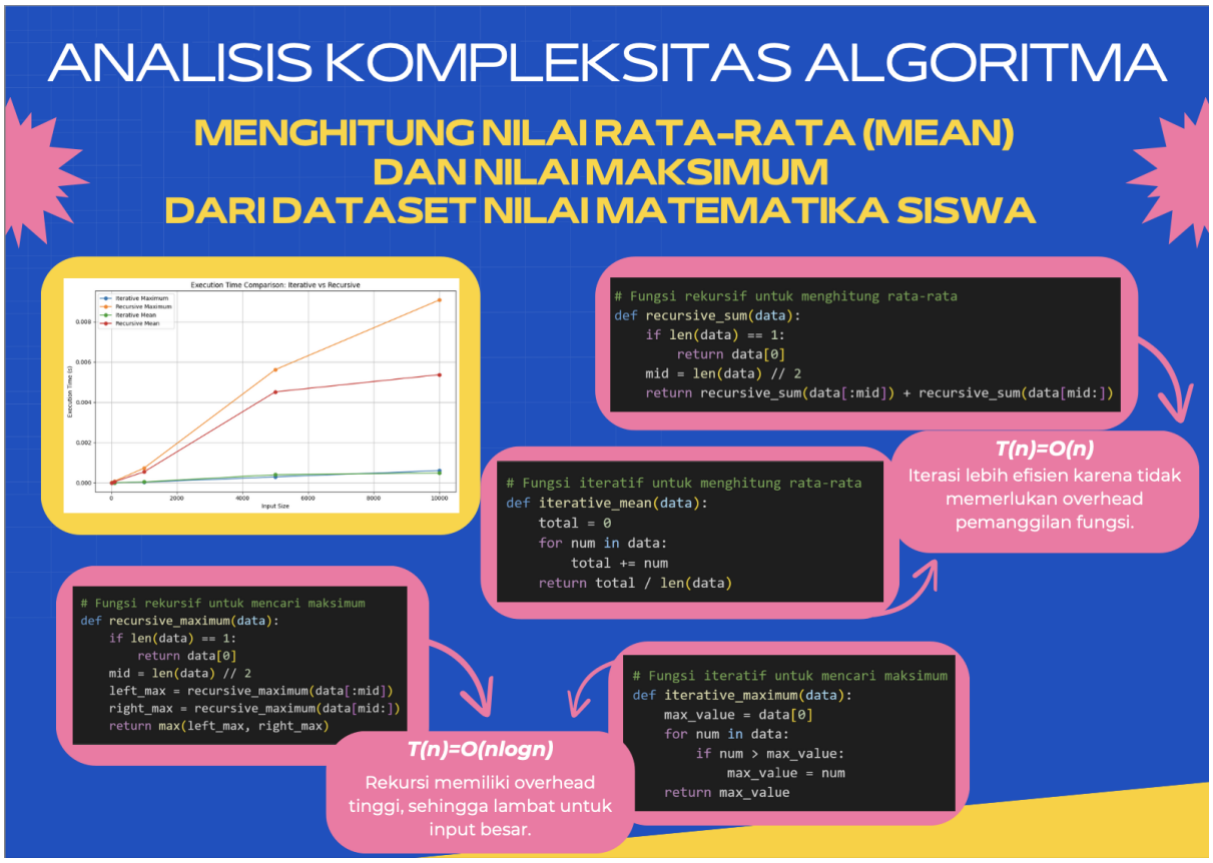
Lutfina, E., & Ramadhan, F. L. (n.d.). Perbandingan Kinerja Metode Iteratif dan Rekursif dalam Algoritma Binary Search. Universitas Nasional Karangturi, Semarang.

Erzed, N. (n.d.). Modul VI: Struktur Data - Rekursivitas dan Pengurutan Lanjut. Teknik Informatika, Universitas Esa Unggul.

Link GitHub :

https://github.com/stephanimss/Tugas-Besar-AKA.git

POSTER


ANALISIS KOMPLEKSITAS ALGORITMA
MENGHITUNG NILAI RATA-RATA (MEAN) DAN NILAI MAKSIMUM DARI DATASET NILAI MATEMATIKA SISWA
Execution Time Comparison: Iterative vs Recursive
Iterative Maximum
Recursive Maximum
Iterative Mean
Recursive Mean
Execution Time (s)
Input Size
# Fungsi rekursif untuk menghitung rata-rata
def recursive_sum(data):
if len(data) == 1:
return data[0]
mid = len(data) // 2
return recursive_sum(data[:mid]) + recursive_sum(data[mid:])
T(n)=O(n)
Iterasi lebih efisien karena tidak memerlukan overhead pemanggilan fungsi.
# Fungsi iteratif untuk menghitung rata-rata
def iterative_mean(data):
total = 0
for num in data:
total += num
return total / len(data)
# Fungsi rekursif untuk mencari maksimum
def recursive_maximum(data):
if len(data) == 1:
return data[0]
mid = len(data) // 2
left_max = recursive_maximum(data[:mid])
right_max = recursive_maximum(data[mid:])
return max(left_max, right_max)
T(n)=O(nlogn)
Rekursi memiliki overhead tinggi, sehingga lambat untuk input besar.
# Fungsi iteratif untuk mencari maksimum
def iterative_maximum(data):
max_value = data[0]
for num in data:
if num > max_value:
max_value = num
return max_value