Aplikasi Sistem Informasi dan Analisis Obat dengan Benchmarking

ANALISIS KOMPLEKSITAS ALGORITMA

ANGGOTA KELOMPOK :

NUR AHMADI ADITYA NANDA (10302230014G)

ADE FATHIA NURAINI (103022300134)

DOSEN PENGAMPU :

LIDYA NINGSIH, S.T, M.T

TELKOM UNIVERSITY

PRODI S1 REKAYASA PERANGKAT LUNAK

2024

DAFTAR ISI

PENDAHULUAN

Aplikasi ini merupakan alat untuk pencarian dan analisis data obat, dilengkapi fitur benchmarking performa metode iteratif dan rekursif. Dengan antarmuka sederhana dan visualisasi data, aplikasi ini mempermudah manajemen dan evaluasi data obat untuk kebutuhan akademik atau operasional. aplikasi ini dilengkapi dengan mekanisme perbandingan performa pencarian menggunakan metode **iteratif** dan **rekursif**. Benchmark ini divisualisasikan dalam bentuk diagram garis, memungkinkan pengguna memahami efisiensi kedua metode berdasarkan ukuran data yang diuji.

DESKRIPSI STUDI KASUS PERMASALAHAN

Aplikasi ini dirancang untuk menjawab tantangan tersebut melalui implementasi metode iteratif dan rekursif untuk pencarian obat, serta menampilkan performa masing-masing metode dalam bentuk visualisasi. Dengan demikian, aplikasi ini membantu pengguna memilih pendekatan terbaik untuk pengelolaan data yang optimal.

1. Database Obat

- **Tujuan**: Menyimpan data obat berupa nama, harga, dan struktur kimia.
- **Format**: Menggunakan list of dictionaries di Python.
- **Skalabilitas**: Database ditambah dengan data dummy (2000 entri) untuk pengujian performa algoritma.

```python
# Database obat
obat_database = [
    {"nama": "Paracetamol", "harga": 5000, "struktur": "C8H9NO2"},
    {"nama": "Ibuprofen", "harga": 15000, "struktur": "C13H18O2"},
    {"nama": "Amoxicillin", "harga": 20000, "struktur": "C16H19N3O5S"},
]
```

2. Fitur Utama

A. Pencarian Obat

Metode yang Digunakan:

1. **Iteratif**
    - Melakukan pencarian dengan looping.
    - Efisien untuk ukuran data kecil hingga sedang.

```python
# Fungsi iteratif untuk pencarian obat
def cari_obat_iteratif(keyword, data):
    """
    Pencarian obat secara iteratif.

    keyword: string yang dicari.
    data: list dictionary data obat.
    """
    hasil = []
    n = len(data)
    for i in range(n):
        if keyword.lower() in data[i]["nama"].lower():
            hasil.append(data[i])
    return hasil
```

2. **Rekursif**

- Melakukan pencarian dengan pendekatan berbasis fungsi yang memanggil dirinya sendiri.
- Kurang optimal untuk data besar karena membutuhkan memori lebih.

```python
# Fungsi rekursif untuk pencarian obat
def cari_obat_rekursif(keyword, data, index=0, hasil=None):
    """
    Pencarian obat secara rekursif.

    keyword: string yang dicari.
    data: list dictionary data obat.
    index: indeks saat ini dalam pencarian.
    hasil: list untuk menyimpan hasil pencarian.
    """
    if hasil is None:
        hasil = []

    # Kondisi basis: jika indeks melebihi panjang data
    if index >= len(data):
        return hasil

    # Cek apakah keyword ditemukan
    if keyword.lower() in data[index]["nama"].lower():
        hasil.append(data[index])

    # Rekursif dengan indeks berikutnya
    return cari_obat_rekursif(keyword, data, index + 1, hasil)
```

B. Analisis Struktur Kimia

- **Tujuan**: Menampilkan struktur kimia obat yang ditemukan dalam bentuk hierarki atom.
- **Metode**: Rekursif untuk mengekstrak dan menganalisis setiap atom dalam string struktur kimia.

C. Analisis Pembelian Obat

- **Tujuan**: Mengidentifikasi pola pembelian obat dari berbagai bulan.
- **Visualisasi**: Diagram batang dan garis menggunakan **Matplotlib** untuk menunjukkan total belanja per bulan.

```
Atom: C, Level: 0
Atom: 8, Level: 1
Atom: H, Level: 2
Atom: 9, Level: 3
```

D. Benchmark Performa Algoritma

Tujuan: Membandingkan performa metode pencarian iteratif dan rekursif pada ukuran data yang berbeda.

- **Proses**:
  - Ukuran dataset yang diuji: 100, 500, 1000, 1500, dan 2000 entri.
  - Waktu eksekusi diukur menggunakan modul time.perf_counter.
- **Hasil**: Ditampilkan dalam diagram garis yang menunjukkan waktu eksekusi (dalam detik) terhadap ukuran data.

```
# Fungsi untuk visualisasi hasil benchmarking
def visualisasi_benchmark():
    keyword = entry_keyword.get()

    # Ukuran data untuk benchmark
    sizes = [100, 500, 1000, 1500, 2000]

    waktu_iteratif, waktu_rekursif = benchmark_cari_obat(keyword, sizes)

    # Visualisasi dengan diagram garis
    plt.figure(figsize=(10, 6))
    plt.plot(sizes, waktu_iteratif, marker='o', label='Iteratif', color='blue')
    plt.plot(sizes, waktu_rekursif, marker='o', label='Rekursif', color='orange')
    plt.title("Perbandingan Waktu Eksekusi")
    plt.xlabel("Ukuran Data")
    plt.ylabel("Waktu (detik)")
    plt.legend()
    plt.grid(True)

    for i, size in enumerate(sizes):
        plt.text(size, waktu_iteratif[i], f"{waktu_iteratif[i]:.4f}", ha='right', color='blue')
        plt.text(size, waktu_rekursif[i], f"{waktu_rekursif[i]:.4f}", ha='right', color='orange')

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

3. Antarmuka Pengguna (GUI)

Dibangun menggunakan modul **Tkinter** untuk memudahkan interaksi pengguna.

Komponen GUI:

1. **Input Kata Kunci**:
    - Entry untuk memasukkan nama obat yang dicari.
2. **Tombol Fungsi**:
    - **Cari Obat (Iteratif)**: Memulai pencarian menggunakan metode iteratif.
    - **Cari Obat (Rekursif)**: Memulai pencarian menggunakan metode rekursif.
    - **Benchmark dan Visualisasi**: Menampilkan hasil benchmarking algoritma dalam bentuk diagram garis.
    - **Analisis Struktur Kimia**: Menganalisis struktur kimia dari obat yang ditemukan.
    - **Analisis Pembelian Obat**: Menampilkan diagram pola pembelian obat.

Tampilan GUI:

Antarmuka sederhana dengan tata letak yang terorganisir menggunakan frame dan grid layout.

4.Studi Kasus dan Implementasi

Studi Kasus:

- Pencarian obat berdasarkan nama, seperti Paracetamol.
- Analisis struktur kimia obat tertentu.
- Mengukur performa algoritma pencarian pada dataset besar.

Hasil Implementasi:

- Memberikan hasil pencarian secara cepat dengan visualisasi waktu eksekusi.
- Membantu pengguna memilih algoritma terbaik untuk dataset besar.

5. Analisis Kompleksitas Algoritma

A. Iteratif

```python
# Fungsi iteratif untuk pencarian obat
def cari_obat_iteratif(keyword, data):
    """
    Pencarian obat secara iteratif.

    keyword: string yang dicari.
    data: list dictionary data obat.
    """
    hasil = []
    n = len(data)
    for i in range(n):
        if keyword.lower() in data[i]["nama"].lower():
            hasil.append(data[i])
    return hasil
```

Rumus S1

1. Rumus Formula S_1:

$$S_1 = \sum_{i=l}^{u} 1 = u - l + 1 \quad \text{untuk } l \leq u$$

2. Penyesuaian Rumus pada Program:

- Loop dilakukan sebanyak n kali, di mana:
  - $l = 0$: Indeks awal iterasi.
  - $u = n - 1$: Indeks akhir iterasi.
- Jumlah iterasi total:

$$S_1 = u - l + 1 = (n - 1) - 0 + 1 = n$$

B. Rekursif

```python
# Fungsi rekursif untuk pencarian obat
def cari_obat_rekursif(keyword, data, index=0, hasil=None):
    """
    Pencarian obat secara rekursif.

    keyword: string yang dicari.
    data: list dictionary data obat.
    index: indeks saat ini dalam pencarian.
    hasil: list untuk menyimpan hasil pencarian.
    """
    if hasil is None:
        hasil = []

    # Kondisi basis: jika indeks melebihi panjang data
    if index >= len(data):
        return hasil

    # Cek apakah keyword ditemukan
    if keyword.lower() in data[index]["nama"].lower():
        hasil.append(data[index])

    # Rekursif dengan indeks berikutnya
    return cari_obat_rekursif(keyword, data, index + 1, hasil)
```

PENCARIAN METODE REKURSIF

RUMUS REKURSIF O(n)

$T(n) = T(n-1) + 1$

$T(n) = (T(n-2)+1) + 1 = T(n-2) + 2 \cdot 1$

$T(n) = T(n-k) + k \cdot 1$

Basis rekursi tercapai ketika $n - k = 0$, sehingga $k = n$ Substitusi $k = n$ ke dalam persamaan :

$T(n) = 0 + n \cdot 1$

Masukkan Nilai $T(0)$ Diketahui $T(0) = O(1)$, maka:

$T(n) = 1 + n \cdot 1$

$T(n) = O(n)$

6.Kesimpulan

Aplikasi ini adalah alat yang efektif untuk pencarian dan analisis data obat dengan evaluasi performa algoritma yang disertai visualisasi. Program ini cocok digunakan untuk mendukung pengelolaan data obat secara praktis dan efisien.

REFERENSI

Dokumentasi Tkinter

Referensi untuk membangun antarmuka GUI.

https://docs.python.org/3/library/tkinter.html

Dokumentasi Matplotlib

Referensi untuk membuat visualisasi data.

https://matplotlib.org/stable/contents.html

Penggunaan List dan Dictionary di Python

Penjelasan tentang struktur data yang digunakan untuk database.

https://docs.python.org/3/tutorial/datastructures.html

Rekursi dalam Python

Panduan untuk memahami cara kerja fungsi rekursif.

https://realpython.com/python-recursion/

Time Complexity of Algorithms

Untuk memahami analisis efisiensi algoritma iteratif dan rekursif.

https://www.bigocheatsheet.com/

Python Performance Benchmarking

Mengukur performa kode menggunakan modul time.

https://docs.python.org/3/library/time.html

POSTER

SISTEM INFORMASI DAN ANALISIS OBAT DENGAN BENCHMARKING

Pengelolaan informasi obat yang akurat dan terintegrasi sangat penting dalam layanan kesehatan untuk mendukung pengambilan keputusan medis, memastikan keamanan pasien, dan meningkatkan efisiensi operasional. Namun, banyak sistem informasi obat saat ini menghadapi masalah seperti kurangnya integrasi data, informasi yang tidak konsisten, dan kesulitan menilai kualitas obat, yang dapat menyebabkan kesalahan pemberian obat dan pemborosan biaya.

```python
# Fungsi iteratif untuk pencarian obat
def cari_obat_iteratif(keyword, data):
    """
    Pencarian obat secara iteratif.
    keyword : string yang dicari.
    data : list dictionary data obat.
    """
    hasil = []
    n = len(data)

    for i in range(n):
        if keyword.lower() in data[i]["nama"].lower():
            hasil.append(data[i])

    return hasil
```

ITERATIF

Penjelasan Program Iteratif
Fungsi: cari_obat_iteratif(keyword, data)

1. Tujuan: Fungsi ini mencari data obat berdasarkan keyword secara iteratif.
2. Parameter:
   - keyword: String yang ingin dicari.
   - data: List berisi dictionary data obat.
3. Proses:
   - Inisialisasi: Membuat list kosong hasil untuk menyimpan data obat yang sesuai.
   - Iterasi:
     - Menggunakan loop for untuk memeriksa setiap item dalam data.
     - Jika keyword ditemukan dalam nama obat (setelah diubah ke huruf kecil), data tersebut ditambahkan ke list hasil.
4. Pengembalian: Mengembalikan list hasil berisi semua obat yang sesuai.

```python
# Fungsi rekursif untuk pencarian obat
def cari_obat_rekursif(keyword, data, index=0, hasil=None):
    """
    Pencarian obat secara rekursif.
    keyword : string yang dicari.
    data : list dictionary data obat.
    index : indeks saat ini dalam pencarian.
    hasil : list untuk menyimpan hasil pencarian.
    """
    if hasil is None:
        hasil = []

    # Kondisi basis: jika indeks melebihi panjang data
    if index >= len(data):
        return hasil

    # Cek apakah keyword ditemukan
    if keyword.lower() in data[index]["nama"].lower():
        hasil.append(data[index])

    # Rekursif dengan indeks berikutnya
    return cari_obat_rekursif(keyword, data, index + 1, hasil)
```

REKURSIF

Penjelasan Program Rekursif
Fungsi: cari_obat_rekursif(keyword, data, index=0, hasil=None)

1. Tujuan: Fungsi ini mencari data obat berdasarkan keyword secara rekursif.
2. Parameter:
   - keyword: String yang ingin dicari.
   - data: List berisi dictionary data obat.
   - index: Indeks saat ini dalam pencarian (default 0).
   - hasil: List untuk menyimpan hasil pencarian (default None).
3. Proses:
   - Inisialisasi: Jika hasil belum diinisialisasi (None), maka diubah menjadi list kosong.
   - Kondisi Basis: Fungsi akan berhenti jika indeks sudah melebihi panjang list data.
   - Pencocokan:
     - Nama obat pada indeks saat ini (data[index]["nama"]) diubah ke huruf kecil.
     - Jika keyword ada di nama obat, maka dictionary obat tersebut ditambahkan ke hasil.
   - Panggilan Rekursif: Fungsi memanggil dirinya sendiri dengan index bertambah 1 untuk memproses data berikutnya.
4. Pengembalian: Mengembalikan list hasil berisi semua obat yang sesuai.

ANALYSIS

Diagram menunjukkan bahwa algoritma iteratif lebih efisien dibandingkan algoritma rekursif dalam hal waktu eksekusi, terutama untuk data berukuran besar. Pada ukuran data kecil, perbedaan waktu eksekusi antara keduanya tidak terlalu signifikan.

Namun, saat ukuran data meningkat, waktu eksekusi algoritma rekursif bertambah jauh lebih cepat dibandingkan iteratif, menunjukkan bahwa algoritma rekursif memiliki kompleksitas waktu yang lebih tinggi. Oleh karena itu, algoritma iteratif lebih cocok digunakan untuk kasus yang melibatkan data berukuran besar karena lebih stabil dan efisien.