# APLIKASI SISTEM INFORMASI DAN ANALISIS OBAT DENGAN BENCHMARKING

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

# Menambahkan 2000 data obat untuk pengujian
for i in range(1, 2001):
    obat_database.append({
        "nama": f"Obat-{i}",
        "harga": random.randint(1000, 100000),
        "struktur": f"Struktur-{i}"
    })
```

**FUNGSI DARI DATABASE**
DATABASE INI DIGUNAKAN SEBAGAI SUMBER UNTUK APLIKASI YANG AKAN:

1. MENCARI OBAT BERDASARKAN KRITERIA TERTENTU (MISALNYA NAMA).
2. MENGAKSES INFORMASI HARGA OBAT UNTUK BERBAGAI KEPERLUAN SEPERTI TRANSAKSI ATAU ANALISIS HARGA.
3. MENAMPILKAN STRUKTUR KIMIA OBAT JIKA DIPERLUKAN

**KESIMPULAN**
OBAT_DATABASE ADALAH REPRESENTASI SEDERHANA DARI DATA OBAT DALAM BENTUK LIST OF DICTIONARIES. DATABASE INI BERGUNA UNTUK APLIKASI KECIL YANG TIDAK MEMERLUKAN DATABASE KOMPLEKS (SEPERTI SQL ATAU NOSQL). INFORMASI YANG DISIMPAN MENCAKUP NAMA, HARGA, DAN STRUKTUR KIMIA DARI SETIAP OBAT.

---

```python
# Fungsi iteratif untuk pencarian obat
def cari_obat_iteratif(keyword):
    return [obat for obat in obat_database
    if keyword.lower() in obat["nama"].lower()]

# Fungsi rekursif untuk pencarian obat
def cari_obat_rekursif(keyword, index=0, hasil=None):
    if hasil is None:
        hasil = []
    if index >= len(obat_database):
        return hasil
    if keyword.lower() in obat_database[index]["nama"].lower():
        hasil.append(obat_database[index])
    return cari_obat_rekursif(keyword, index + 1, hasil)
```

**PENJELASAN DASAR PENJUMLAHAN:**

PROGRAM INI MENJALANKAN ITERASI SEBANYAK NNN, YAITU JUMLAH ELEMEN DALAM OBAT_DATABASE.
SETIAP ITERASI MELAKUKAN PENCOCOKAN SUBSTRING (MMM), YANG DIHITUNG SEBAGAI OPERASI KONSTAN UNTUK SETIAP KARAKTER NAMA OBAT.

**FORMULA YANG COCOK:**

ITERASI ADALAH JUMLAH TOTAL ELEMEN, SESUAI DENGAN FORMULA S1:

$$\Sigma_{i=l}^{u} 1 = u - l + 1$$

DALAM KASUS INI:

$l = 1$ dan $u = n$, sehingga $\Sigma_{i=1}^{n} 1 = n$.

OPERASI PER ITERASI (SUBSTRING COMPARISON) MENGHASILKAN WAKTU TAMBAHAN M, SEHINGGA:

$$\Sigma_{i=1}^{n} m = m \cdot \Sigma_{i=1}^{n} 1 = m \cdot n$$

**RELASI REKURENSI** MENGGAMBARKAN JUMLAH OPERASI YANG DILAKUKAN OLEH FUNGSI SECARA MATEMATIS.
DALAM HAL INI:

1. **PADA SETIAP LANGKAH REKURSIF:**
   - ADA OPERASI PERBANDINGAN SUBSTRING (M).
   - FUNGSI DIPANGGIL LAGI UNTUK ELEMEN BERIKUTNYA HINGGA MENCAPAI AKHIR DATABASE.
2. **BASIS REKURSI:**
   - KETIKA INDEX >= N, FUNGSI BERHENTI DAN TIDAK ADA OPERASI TAMBAHAN.

RELASI REKURENSI UNTUK WAKTU EKSEKUSI T(N): $T(n) = T(n-1) + c$

- T(N-1)UNTUK MEMPROSES SISA N−1 ELEMEN.
- C : OPERASI KONSTAN PADA ELEMEN SAAT INI (SUBSTRING COMPARISON

BASIS: $T(0) = c$

**JENIS REKURENSI PROGRAM INI MENGGUNAKAN REKURENSI HOMOGEN, KARENA:**

$T(n) = T(n-1) + c$ HANYA BERGANTUNG PADA T(N−1)DENGAN TAMBAHAN KONSTANTA C.

**PENYELESAIAN DENGAN SUBSTITUSI**

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

SUBSTITUSI:

$T(n-1) = T(n-2) + c$

$T(n) = T(n-2) + c + c$

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

KETIKA K=NK (BASIS T(0)=C):

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

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

SEDERHANAKAN:

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

**KOMPLEKSITAS AKHIR**

KARENA C ADALAH KONSTANTA DAN M ADALAH PANJANG SUBSTRING YANG DIBANDINGKAN :

$T(n) \in O(n \cdot m)$

INI MENUNJUKKAN KOMPLEKSITAS LINEAR DENGAN JUMLAH ELEMEN (N).

**KESIMPULAN**

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.