

# **Laporan Tugas Besar Analisis Kompleksitas Algoritma Topik Perkembangan Populasi Bakteri Menggunakan Perpangkatan Eksponensial**



Disusun oleh:

**ATHILA RAMDANI SAPUTRA**

**103012300132**

**BILL STEPHEN**

**103012330197**

**Program Studi S1 Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Bandung**

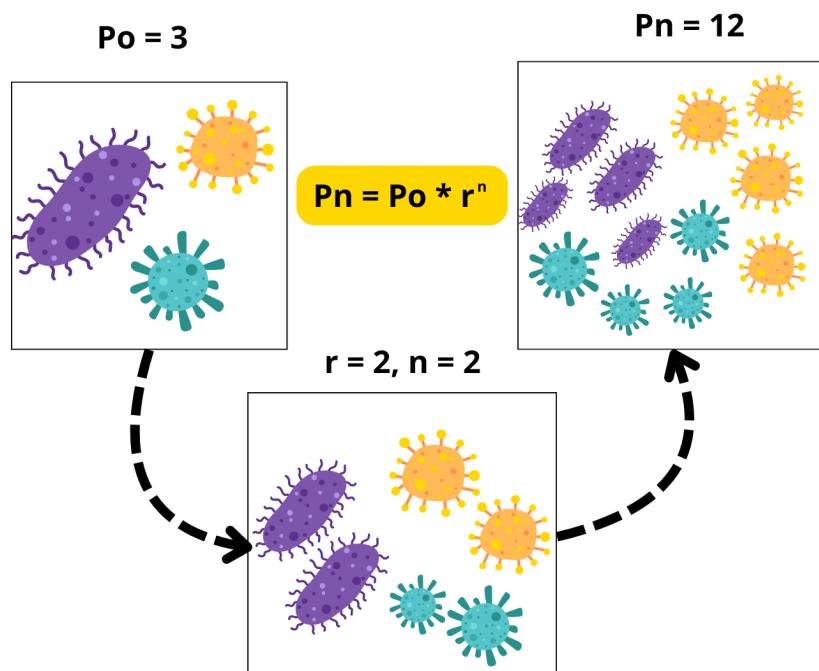
**2024-2025**

## 1. Deskripsi Studi Kasus Permasalahan

Dalam studi ini, kita menganalisis perkembangan populasi bakteri yang tumbuh secara eksponensial. Misalnya, sebuah populasi bakteri memiliki ukuran awal  $P_0$  dan berkembang dengan laju pertumbuhan  $r$  setiap satuan waktu  $n$ , maka populasi pada waktu  $n$  ( $P_n$ ) dapat dihitung menggunakan formula:

$$P_n = P_0 * r^n$$

Di sini, kita akan mengimplementasikan algoritma iteratif dan rekursif untuk menghitung populasi ini memakai algoritma dengan cara yang berbeda dari rumus serta menganalisis kompleksitas keduanya. Selain itu, grafik perbandingan running time juga akan ditampilkan



Gambar 1. Ilustrasi Perubahan Populasi

Sumber

[https://www.canva.com/design/DAGZeqZeKXE-SrmrqAvZjQWtKnk60pGtA/view?utm\\_content=DAGZeqZeKXE&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=uniquelinks&utllid=h3e41b6a](https://www.canva.com/design/DAGZeqZeKXE-SrmrqAvZjQWtKnk60pGtA/view?utm_content=DAGZeqZeKXE&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utllid=h3e41b6a)

abd

Canva edit

## 2. Deskripsi Algoritma Iteratif dan Rekursif dalam Studi Kasus Permasalahan

Pada penginisialisasi kode awal dibawah ini berfungsi untuk mengimpor modul yang diperlukan untuk membuat visualisasi data menggunakan matplotlib.pyplot. lalu, modul time dan math diimpor untuk menyediakan fungsi terkait waktu dan operasi matematika. Blok try-except digunakan untuk menangani situasi ketika matplotlib belum terinstal, dan jika itu terjadi, perintah !pip install matplotlib dijalankan untuk

menginstalnya. Terakhir, modul matplotlib.pyplot diimpor ulang untuk memastikan tersedia setelah instalasi.

```
import time
import math

try:
    import matplotlib.pyplot as plt
except ImportError:
    !pip install matplotlib

import matplotlib.pyplot as plt
```

### a. Algoritma Iteratif

Pada pendekatan iteratif, kita menghitung populasi dengan menggunakan pengulangan (loop). Rumusnya menggunakan pertumbuhan eksponensial

$$P_n = P_0 * r^n \text{ diubah menjadi (loop)}$$

#### Pseudocode:

Function Iteratif\_bakteri( $P_0, r, n : \text{integer}$ ) -> real

Kamus

i,j : integer

result : real

Algoritma

result <-  $P_0$

for (i <- 1 to n) do

    temp <- 0

    for (j<-1 to r) do

        temp <- temp + result

    endfor

    result <- temp

  endfor

Output result

endprogram

#### Python

```
def iteratif_bakteri(P_0, r, n):
    result = P_0
    for i in range(1, n + 1): # for i <- 1 to n do
        temp = 0 # Variabel untuk menyimpan hasil penjumlahan
        for j in range(1, r + 1): # for j <- 1 to r do #
            Melakukan penjumlahan sebanyak r kali
```

```

        temp += result
        result = temp
    return result

```

## b. Algoritma Rekursif

Pada pendekatan rekursif, kita menghitung populasi dengan menggunakan

```
return r * rekursif_bakteri(P_0,r,n-1)
```

### Pseudocode:

```
Function rekursif_bakteri(P_0,r,t,n) -> real
Algoritma
    if n = 0 then
        return P_0
    else
        return r * rekursif_bakteri(P_0,r,n-1)
    endif
endfunction
```

### Python :

```
def rekursif_bakteri(P_0, r, n):
    if n == 0:
        return P_0 # Basis rekursi: populasi awal
    else:
        return r * rekursif_bakteri(P_0, r, n - 1) # Rekursi
dengan Pn = r * P(n-1)
```

## 3. Grafik Perbandingan *Running Time*

Ujicoba pada nilai n = 1 sampai 30 dengan jarak 2 angka seperti yang ada di kodingan di bawah ini dengan populasi awal 20, laju pertumbuhan 10 tiap detik

```
# Parameter studi kasus
P0 = 20 # Populasi awal
r = 10 # Laju pertumbuhan
n_values = [i for i in range(1, 30, 2)]

# Mencatat waktu eksekusi iteratif
iterative_times = []
recursive_times = []
```

```
print("Hasil waktu eksekusi dan populasi akhir (dalam detik):")
print(f"\n{'Iteratif':<10}{'Populasi Iteratif':<50}{'Rekursif':<15}{'Populasi Rekursif':<50}")

for n in n_values:
    # Iteratif
    start_time = time.time()
    iteratif_population = iteratif_bakteri(P0, r, n)
    iterative_time = time.time() - start_time
    iterative_times.append(iterative_time)

    # Rekursif
    start_time = time.time()
    rekursif_population = rekursif_bakteri(P0, r, n)
    recursive_time = time.time() - start_time
    recursive_times.append(recursive_time)

    # Cetak hasil waktu eksekusi dan populasi akhir

print(f"\n{n:<10}{iterative_time:<15.6f}{iteratif_population:<50.0f}{{recursive_time:<15.6f}{rekursif_population:<50.0f}}")
```

Gambar 2. Hasil perbandingan iterasi iteratif dan rekursif

Sumber :[https://colab.research.google.com/drive/1rr7R\\_Ll0m70GhUJxIQ2TpB9-7iEOP80c?usp=sharing](https://colab.research.google.com/drive/1rr7R_Ll0m70GhUJxIQ2TpB9-7iEOP80c?usp=sharing)  
google colab

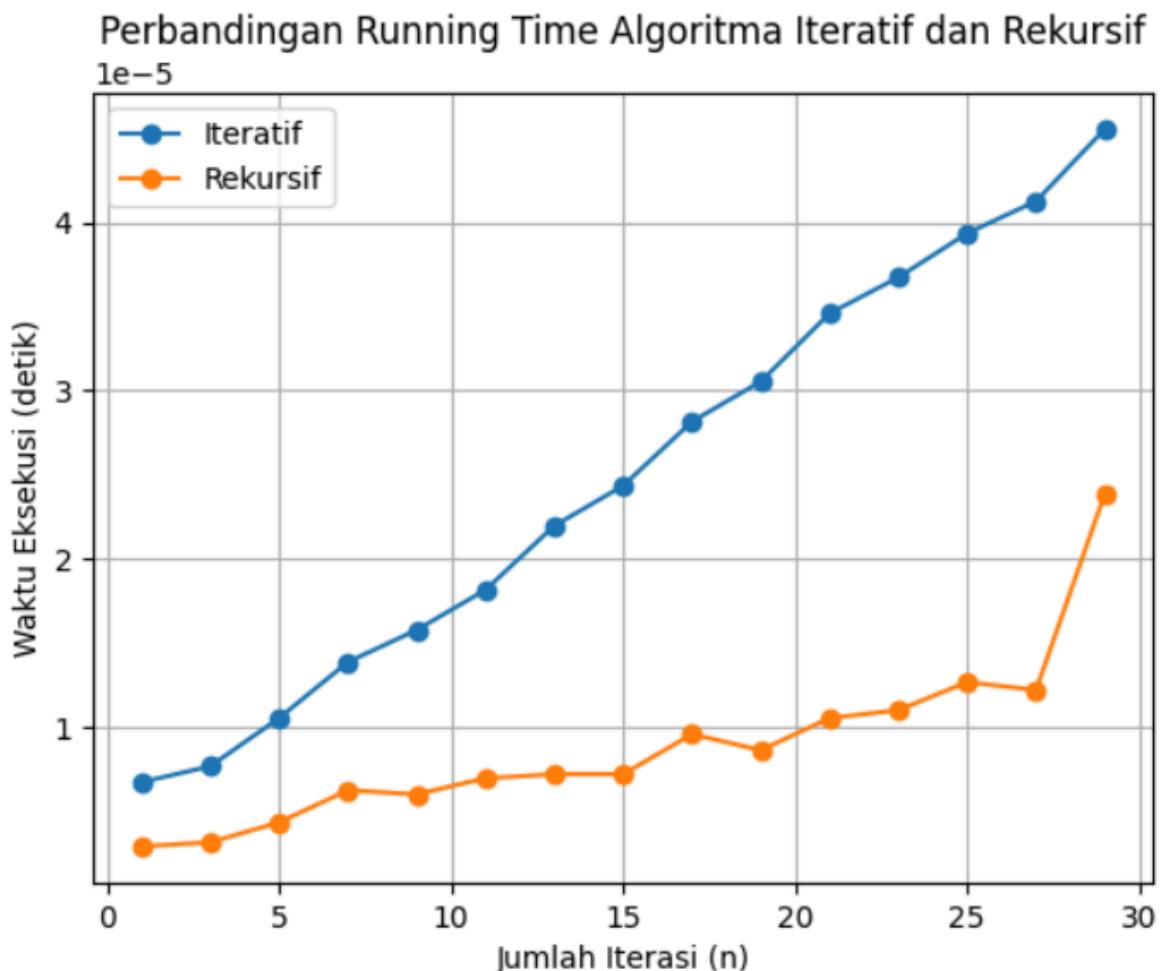
Maka hasil perbandingan running time antara iteratif dan rekursif dalam bentuk grafik adalah sebagai berikut :

```
/ # Plot hasil perbandingan
plt.plot(n_values, iterative_times, label='Iteratif', marker='o')
plt.plot(n_values, recursive_times, label='Rekursif', marker='o')
plt.xlabel('Jumlah Iterasi (n)')
plt.ylabel('Waktu Eksekusi (detik)')
```

```

plt.title('Perbandingan Running Time Algoritma Iteratif dan Rekursif')
plt.legend()
plt.grid()
plt.show()

```



Gambar 3. Hasil perbandingan iterasi iteratif dan rekursif menggunakan plot

Sumber :[https://colab.research.google.com/drive/1rr7R\\_LI0m70GhUJxIQ2TpB9-7IEOP80c?usp=sharing](https://colab.research.google.com/drive/1rr7R_LI0m70GhUJxIQ2TpB9-7IEOP80c?usp=sharing)  
google colab

```

# Menghitung rata-rata waktu eksekusi
average_iterative_time = sum(iterative_times) /
len(iterative_times)
average_recursive_time = sum(recursive_times) /
len(recursive_times)

# Menampilkan diagram batang rata-rata
labels = ['Iteratif', 'Rekursif']
average_times = [average_iterative_time, average_recursive_time]

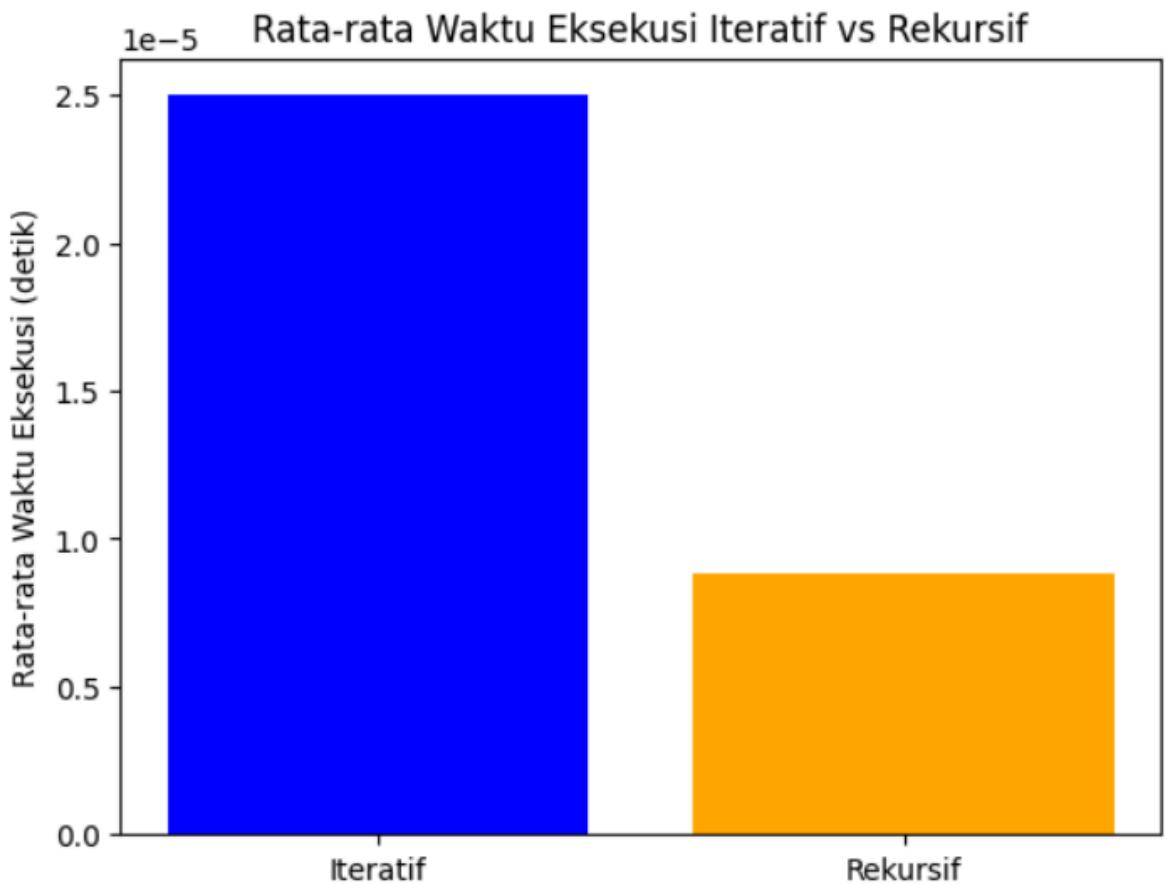
plt.bar(labels, average_times, color=['blue', 'orange'])

```

```

plt.ylabel('Rata-rata Waktu Eksekusi (detik)')
plt.title('Rata-rata Waktu Eksekusi Iteratif vs Rekursif')
plt.show()

```



Gambar 4. Hasil perbandingan iterasi iteratif dan rekursif menggunakan diagram batang  
Sumber :<https://colab.research.google.com/drive/1rr7RLl0m70GhUxlQ2TpB9-7iEOP80c?usp=sharing>  
google colab

#### 4. Analisis Algoritma Iteratif dan Rekursif dalam Studi Kasus Permasalahan

##### Kompleksitas Waktu

Iteratif:

$$\begin{aligned}
 T(n) &= \sum_{i=1}^n \sum_{j=1}^r 1 = \sum_{i=1}^n r-1+1 \\
 &\hookrightarrow \sum_{i=1}^n r = r \sum_{i=1}^n 1 = r(n-1, H) = n \cdot r
 \end{aligned}$$

$$\boxed{T(n) = n \cdot r \in O(n \cdot r)}$$

$$T(n) = n.r$$

Waktu:  $O(n.r)$  (loop sebanyak  $n$  dikali  $r$ ).

**Rekursif:**

$$T(n) = \begin{cases} 0 & n = 0 \\ T(n-1) + 1 & n > 0 \end{cases}$$

$$\begin{aligned} T(n) &= T(n-1) + 1 \\ &= (T(n-2) + 1) + 1 = T(n-2) + 2 \\ &= ((T(n-3) + 1) + 1) + 1 = T(n-3) + 3 \\ &\vdots \\ &\vdots \\ &\vdots \\ &\geq T(n-i) + i \\ &\vdots \\ &= T(n-n) + n = T(0) + n \\ &= 0 + n = n \end{aligned}$$

$$\boxed{T(n) = n \in O(n)}$$

$$T(n) = n$$

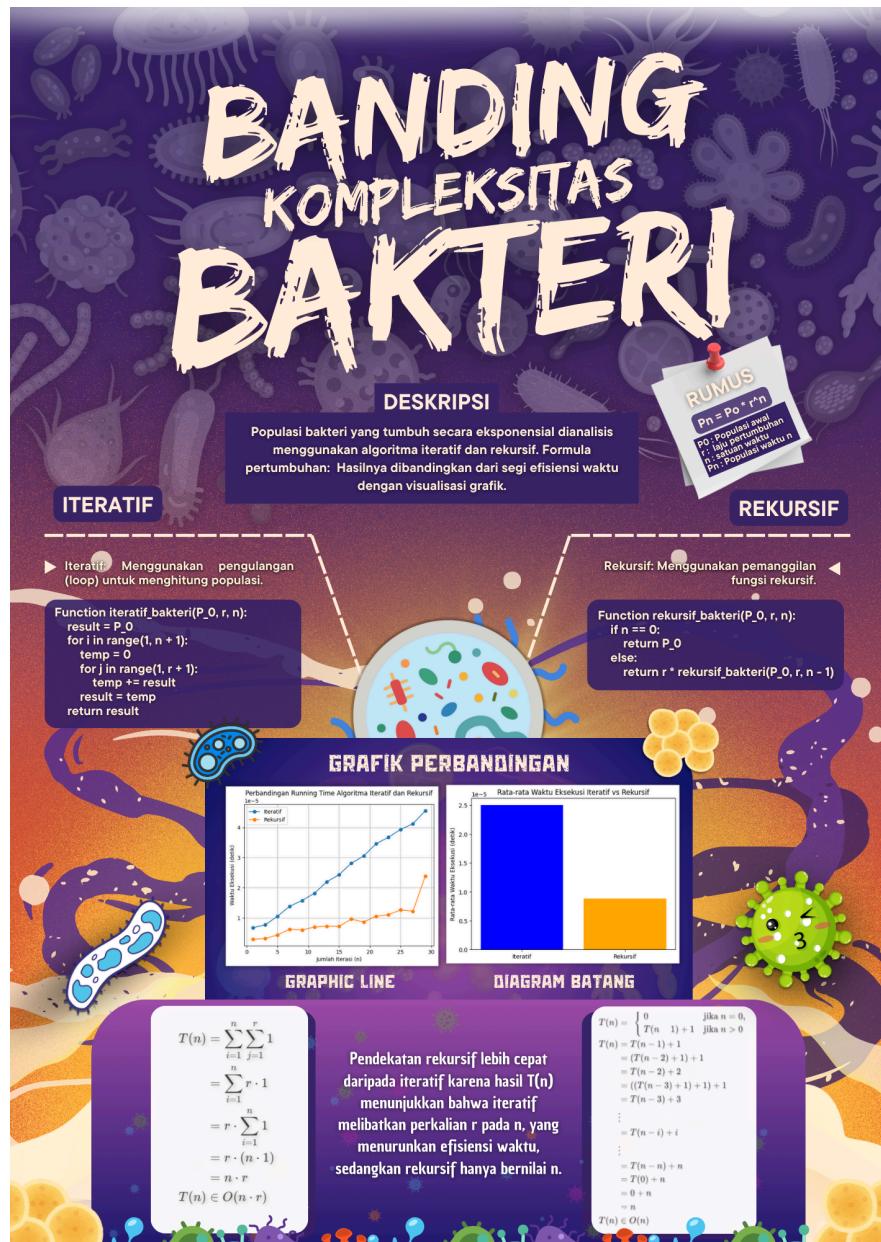
Waktu:  $O(n)$  (pemanggilan rekursif hingga  $n$  kali).

### Hasil Perbandingan

Pendekatan rekursif lebih cepat dibandingkan iteratif karena hasil  $T(n)$  yang sudah dieksekusi diatas membuktikan bahwa dengan adanya perkalian  $r$  pada  $n$  di dalam iteratif menimbulkan penurunan performa kecepatan dalam efisien waktu jika dibandingkan dengan rekursif yang hanya memiliki nilai  $n$ . Oleh karena itu, *Running*

*Time* pada eksekusi iteratif dengan big-O( $n \cdot r$ ) akan lebih lambat performa kecepatan pengeksekusian pada perpangkatan eksponensial jika dibanding dengan eksekusi rekursif dengan big-O( $r$ ).

Sebagai bagian dari tugas ini, kami juga telah membuat sebuah poster yang menggambarkan hasil analisis dan visualisasi dalam studi kasus ini. Poster tersebut menampilkan informasi utama, grafik hasil perbandingan algoritma iteratif dan rekursif.. Berikut adalah gambar posternya:



Gambar 5. Poster Studi Kasus Banding Kompleksitas Bakteri

Sumber :

[https://www.canva.com/design/DAGaFtGemaM/cE5JYBYz8QBAb46273PGLg/view?utm\\_content=DAGaFtGemaM&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=uniquelinks&utllid=he8a1617303](https://www.canva.com/design/DAGaFtGemaM/cE5JYBYz8QBAb46273PGLg/view?utm_content=DAGaFtGemaM&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utllid=he8a1617303)  
canva

## 5. Referensi

- 1.) Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. **Introduction to Algorithms** (3rd Edition).
- 2.) Python Software Foundation. **Python 3.11 Documentation**.
- 3.)  
[https://colab.research.google.com/drive/1rr7R\\_Ll0m70GhUJxIQ2TpB9-7iEOP80c?usp=sharing](https://colab.research.google.com/drive/1rr7R_Ll0m70GhUJxIQ2TpB9-7iEOP80c?usp=sharing)
- 4.)  
[https://www.canva.com/design/DAGZeqZeKXE/-SrmrqAvZjQWtKnk60pGtA/view?utm\\_content=DAGZeqZeKXE&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=uniquelinks&utlld=h3e41b6aab](https://www.canva.com/design/DAGZeqZeKXE/-SrmrqAvZjQWtKnk60pGtA/view?utm_content=DAGZeqZeKXE&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlld=h3e41b6aab)
- 5.)  
[https://www.canva.com/design/DAGaFtGemaM/cE5JYBYz8QBAb46273PGLg/view?utm\\_content=DAGaFtGemaM&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=uniquelinks&utlld=he8a1617303](https://www.canva.com/design/DAGaFtGemaM/cE5JYBYz8QBAb46273PGLg/view?utm_content=DAGaFtGemaM&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlld=he8a1617303)