Analisis Perbandingan Algoritma Iteratif dan Rekursif Menghitung Hasil Kuadrat dari Matriks Persegi

Disusun Oleh:

Diva Danar Yuniar – 103052400007

Arendra Isa Pratama – 103052430025

Putri Ayu Nur Hakim – 103052400057

Telkom University

PENDAHULUAN

1. Deskripsi Studi Kasus

Studi kasus kali ini membahas pendekatan dengan dua algoritma berbeda untuk menghitung hasil kuadrat dari matriks persegi (A^2). Kedua algoritma tersebut akan menghasilkan hasil akhir yang sama, namun karena terdapat perbedaan cara kerja maka akan memengaruhi efektivitas komputasi. Masing-masing algoritma akan dibandingkan running time dalam berapa lama mengeksekusi kasus yang sama.

2. Deskripsi Algoritma

Dalam studi kasus ini, dua pendekatan algoritma yang digunakan untuk menghitung perkalian matriks dengan dirinya sendiri yaitu:

A) Algoritma Iteratif

Algortima ini menggunakan tiga lapis perulangan bersarang (nested loop) untuk mengalikan setiap angka pada baris matriks pertama dengan angka di kolom matriks kedua. Proses diawali dengan menginisialisasi matriks nol sebagai penampung, kemudian secara sistematis menghitung hasil perkalian titik (*dot product*) antara baris dan kolom yang bersesuaian.

B) Algoritma Rekursif,

Algoritma ini menggunakan pendekatan *divide and conquer* dengan memecah matriks berukuran *n x n* menjadi empat sub-matriks yang lebih kecil berukuran $\frac{n}{2}$ x $\frac{n}{2}$. Proses perhitungan dilakukan dengan memanggil fungsi itu sendiri secara berulang untuk menyelesaikan delapan operasi perkalian sub-matriks, yang kemudian hasilnya dijumlahkan dan digabungkan kembali.

PEMBAHASAN

1. Analisis Kompleksitas Waktu Algoritma

Untuk membandingkan algoritma rekursif dan iteratif, ada beberapa metode yang dapat digunakan. Langkah pertama yang kami ambil adalah untuk membandingkan kompleksitas waktunya. Dengan menentukan ukuran input dan operasi dasarnya, kompleksitas waktu dari suatu algoritma pemrograman dapat dihitung dan kemudian dibandingkan.

a. Algoritma Rekursif

Untuk algoritma rekursif, kami mengimplementasikan algoritma perkalian matriks menggunakan metode *divide and conquer* yang memecah matriks menjadi 4 submatriks secara rekursif sebagai berikut:

```python
# Fungsi Perkalian Matriks Rekursif (Divide and Conquer)
def multiply_recursive(A, B):
    n = len(A)
    if n == 1:
        return A * B

    mid = n // 2
    # Membagi matriks menjadi 4 kuadran
    a11, a12 = A[:mid, :mid], A[:mid, mid:]
    a21, a22 = A[mid:, :mid], A[mid:, mid:]
    b11, b12 = B[:mid, :mid], B[:mid, mid:]
    b21, b22 = B[mid:, :mid], B[mid:, mid:]

    # Rekursi untuk 8 perkalian sub-matriks
    c11 = multiply_recursive(a11, b11) + multiply_recursive(a12, b21)
    c12 = multiply_recursive(a11, b12) + multiply_recursive(a12, b22)
    c21 = multiply_recursive(a21, b11) + multiply_recursive(a22, b21)
    c22 = multiply_recursive(a21, b12) + multiply_recursive(a22, b22)

    # Menggabungkan kembali hasil perkalian
    top = np.hstack((c11, c12))
    bottom = np.hstack((c21, c22))
    return np.vstack((top, bottom))
```

Dari potongan kode python tersebut, kami dapat menentukan kompleksitas waktunya.

Diketahui :
- Ukuran input = n
- Operasi dasar = perkalian
- Pemanggilan rekursif = 8 kali

Maka, relasi rekurens nya adalah :

$$T(n) = \begin{cases} 1 & , n = 1 \\ 8T\left(\frac{n}{2}\right) & , n > 1 \end{cases}$$

Dengan bentuk relasi rekurensi tersebut, kami dapat memanfaatkan Teorema Master untuk mendapatkan kompleksitas waktunya.

Diperoleh :
$a = 8$
$b = 2$
$d = 0$
Karena $a > b^d$
$T(n) = n^{\log_b a}$
$T(n) = n^{\log_2 8}$
$T(n) = n^3$

$T(n) = n^3 \in O(n^3)$

b. Algoritma Iteratif

Implementasi algoritma perkalian matriks secara iteratif dilakukan dengan memanfaatkan tiga lapis perulangan *(nested loop)* untuk mengakses seluruh elemen dalam matriks dan kemudian menghitung hasilnya untuk matriks yang baru. Algoritma tersebut ditulis dalam bahasa python seperti berikut:

```python
# Fungsi Perkalian Iteratif
def multiply_iterative(A, B):
    n = len(A)
    C = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C
```

Dari potongan kode tersebut, dapat ditentukan kompleksitas waktunya sebagai berikut:

Diketahui :
- Ukuran input = n
- Operasi dasar = perkalian
- Misalkan C(n) = banyaknya perkalian dieksekusi

Karena C(n) hanya bergantung pada ukuran input dan tidak ada kondisi yang membuatnya selesai lebih awal, C(n) tidak memiliki best case dan worst case.

$$C(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} n \cdot 1$$

$$= \sum_{i=1}^{n} n \cdot n \cdot 1$$

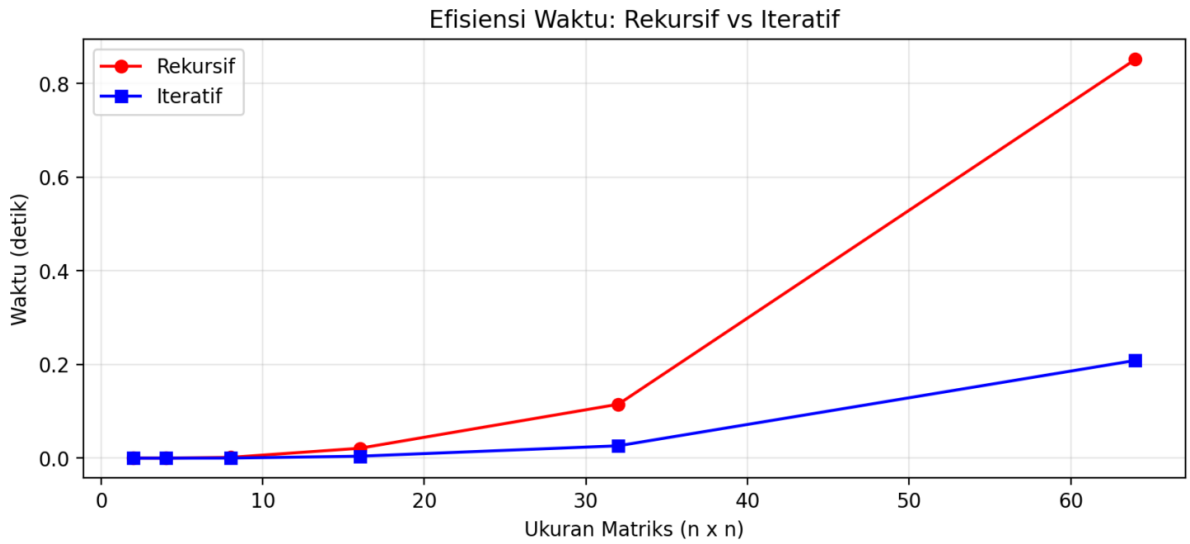
$$= n \cdot n \cdot n$$

$$C(n) = n^3 \in O(n^3)$$

Dari perhitungan kedua algoritma tersebut, dapat dilihat bahwa mereka memiliki *order od growth* yang sama, yaitu pada notasi asimtotik $O(n^3)$. Running time dari kedua algoritma akan memiliki skalabilitas yang sama.

2. Grafik Perbandingan Runtime Algoritma

Dari pengujian kedua algoritma melalui beberapa input dari ukuran matriks 2x2 hingga 32x32, dihasilkan grafik perbandingan runtime berikut:



Grafik tersebut menunjukkan bahwa hingga matriks berukuran sebesar 60x60, algoritma iteratif akan dapat menghitung hasil kuadratnya lebih cepat dibandingkan dengan algoritma rekursif. Hal ini terjadi karena algoritma rekursif adanya terdapat waktu dan memori yang perlu dialokasikan saat memanggil suatu fungsi, sehingga ketika algoritma rekursif memanggil suatu fungsi secara berulang-ulang, akan memakan waktu dan ruang memori yang lebih besar.

Grafik juga memperlihatkan bahwa untuk matriks yang lebih besar dari ukuran 60x60, ada besar kemungkinan bahwa algoritma iteratif akan tetap menjadi algoritma yang lebih cepat. Hal ini dikarenakan kedua algoritma memiliki skalabilitas runtime yang sama.

PENUTUP

Kesimpulan

Berdasarkan hasil pengujian dan analisis yang kami lakukan terhadap algoritma iteratif dan rekursif untuk menghitung hasil kuadrat dari suatu matriks persegi, dapat disimpulkan bahwa algoritma iteratif memiliki waktu eksekusi yang lebih cepat. Hal ini disebabkan oleh alokasi waktu dan memori tambahan yang diperlukan algoritma rekursif untuk terus memanggil fungsi.

Diantara kedua pendekatan tersebut, pendekatan iteratif menjadi pilihan yang lebih baik dibandingkan pendekatan secara rekursif, baik dari segi waktu maupun ruang memori. Namun kedua algoritma memiliki *order of growth* yang sama yaitu $O(n^3)$, keduanya akan kesulitan untuk memproses data yang besar seperti Big Data. Sehingga diperlukan pengujian dan analisis lanjut untuk menemukan algoritma penghitung hasil kuadrat matriks persegi yang dapat bekerja berdampingan dengan Big Data.

REFERENSI

Thudi, S. (2017, June 15). Matrix Multiplication. Shiva Thudi's Blog. https://shivathudi.github.io/jekyll/update/2017/06/15/matr-mult.html