Tugas Besar
“Perbandingan Efisiensi Algoritma Iteratif dan Rekursif untuk Pencarian *Least Common Multiple* (LCM)”

Disusun Oleh:

Muthia Rezi Aisyah - 103052300114
Amalia Ananda Putri - 103052330078

ANALISIS KOMPLEKSITAS ALGORITMA

PROGRAM STUDI S1 DATA SAINS

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

2024

DAFTAR ISI

PENDAHULUAN

1.1 Latar Belakang

Least Common Multiple (LCM) atau Kelipatan Persekutuan Terkecil (KPK) merupakan salah satu konsep penting dalam matematika yang memiliki beragam aplikasi, mulai dari pengaturan jadwal hingga komputasi berbasis algoritma. Dalam ranah komputasi, pencarian LCM sering kali dikombinasikan dengan pendekatan *Greatest Common Divisor* (GCD) karena hubungan antara keduanya yang dinyatakan dengan formula matematis tertentu. Namun, pendekatan langsung untuk mencari LCM tanpa memanfaatkan GCD dapat memberikan wawasan tambahan mengenai efisiensi algoritma, terutama ketika diterapkan pada berbagai pasangan bilangan bulat positif.

Studi mengenai efisiensi algoritma menjadi semakin relevan seiring dengan meningkatnya kebutuhan akan proses komputasi yang cepat dan efisien di era digital. Dalam konteks ini, pendekatan rekursif dan iteratif merupakan dua metode yang sering digunakan dalam menyelesaikan masalah komputasi. Kedua pendekatan ini memiliki karakteristik dan keunggulan masing-masing. Pendekatan rekursif sering kali lebih elegan dan intuitif, tetapi cenderung menggunakan lebih banyak sumber daya memori karena pemanggilan fungsi secara berulang. Di sisi lain, pendekatan iteratif biasanya lebih hemat memori, namun terkadang kurang intuitif untuk diimplementasikan.

Dengan bertambahnya ukuran input, perbedaan performa antara algoritma rekursif dan iteratif menjadi semakin signifikan. Oleh karena itu, percobaan ini bertujuan untuk mengevaluasi efisiensi kedua pendekatan tersebut dalam mencari LCM. Evaluasi dilakukan dengan mengukur waktu eksekusi masing-masing algoritma pada berbagai pasangan bilangan bulat positif. Melalui percobaan ini, diharapkan dapat diperoleh pemahaman yang lebih baik mengenai pola pertumbuhan waktu eksekusi kedua algoritma, serta rekomendasi implementasi yang lebih sesuai berdasarkan ukuran input.

1.2 Tujuan

Tujuan dari percobaan ini antara lain:

1. Membandingkan efisiensi algoritma rekursif dan iteratif dalam mencari nilai LCM (*Least Common Multiple*) tanpa menggunakan pendekatan GCD.
2. Menganalisis performa kedua algoritma berdasarkan waktu eksekusi pada berbagai pasangan bilangan bulat positif dengan ukuran input yang bervariasi.
3. Mengidentifikasi pola pertumbuhan waktu eksekusi dari algoritma rekursif dan iteratif saat ukuran input meningkat.
4. Memvisualisasikan hasil analisis performa kedua algoritma dalam bentuk grafik untuk memberikan gambaran yang lebih jelas mengenai perbandingan efisiensi keduanya.
5. Memberikan rekomendasi terkait pendekatan algoritma yang lebih optimal untuk mencari LCM berdasarkan ukuran input dan kebutuhan aplikasi komputasi.

PEMBAHASAN

2.1 LCM (*Least Common Multiple*)

1. Definisi LCM

   Least Common Multiple (LCM) atau Kelipatan Persekutuan Terkecil (KPK) adalah bilangan bulat positif terkecil yang merupakan kelipatan bersama dari dua atau lebih bilangan. Dalam matematika, LCM digunakan untuk menyelesaikan berbagai masalah, seperti penyamaan pecahan, penjadwalan, dan pengaturan interval dalam sistem terdistribusi.

   Sebagai contoh, jika diberikan dua bilangan x = 4 dan y = 6, maka kelipatan dari masing-masing bilangan adalah:

   - Kelipatan 4: 4, 8, 12, 16, 20, …
   - Kelipatan 6: 6, 12, 18, 24, …

   Maka, kelipatan persekutuan terkecilnya adalah 12 sehingga LCM(4, 6) = 12.

2. Rumus Dasar LCM

   Hubungan antara LCM dan *Greatest Common Divisor* (GCD) dirumuskan sebagai:

$$LCM(x, y) = \frac{|x \, . \, y|}{GCD(x, y)}$$

   Namun, penelitian ini berfokus pada pendekatan langsung untuk mencari LCM tanpa dengan menggunakan pendekatan GCD.

3. Metode Pencarian LCM

   Ada beberapa metode untuk menghitung LCM, yaitu:

   a. Pendekatan Brute Force

      Dengan mencari kelipatan setiap bilangan hingga ditemukan kelipatan bersama terkecil. Metode ini tidak efisien untuk input yang besar karena memerlukan iterasi yang sangat banyak.

   b. Pendekatan Faktorisasi Prima

      Menguraikan bilangan ke dalam faktor primanya dan menggabungkan faktor dengan pangkat tertinggi.

      Contoh:

      - $x = 12 = 2^2 . 3^1$
      - $y = 18 = 2^1 . 3^2$
      - LCM adalah $2^{max(2,1)} . 3^{max(1,2)} = 36.$

      Metode ini memerlukan algoritma faktorisasi yang kompleks untuk bilangan besar.

   c. Pendekatan Rekursif

      Menggunakan fungsi rekursif untuk menemukan kelipatan terkecil bersama. Biasanya dengan menginkrementasi kelipatan salah satu bilangan hingga ditemukan hasil yang sesuai.

   d. Pendekatan Iteratif

Menggunakan perulangan untuk mencari kelipatan terkecil bersama dengan cara serupa seperti metode rekursif, tetapi tanpa pemanggilan fungsi berulang.

2.2 Deskripsi Masalah

Pencarian LCM merupakan salah satu operasi dasar dalam matematika dan komputasi. Terdapat berbagai pendekatan untuk menemukan LCM, salah satunya menggunakan algoritma rekursif dan iteratif. Namun, seiring dengan meningkatnya ukuran input, efisiensi, kedua pendekatan ini menjadi topik yang perlu dianalisis. Setiap pendekatan memiliki karakteristik unik, baik dalam hal waktu eksekusi maupun pengguna sumber daya komputasi.

Masalah yang dihadapi adalah menentukan algoritma mana yang lebih dioptimalkan untuk mencari LCM pada berbagai pasangan bilangan bulat positif. Hal ini penting untuk memastikan bahwa implementasi algoritma yang dipilih dapat memberikan hasil yang cepat dan efisien, terutama dalam konteks aplikasi nyata yang membutuhkan pengolahan data besar.

2.3 Metode Penyelesaian Masalah

Dalam percobaan mencari LCM dari kedua bilangan, misalnya bilangan x dan y, akan digunakan dua metode, yaitu dengan algoritma iteratif dan rekursif. Berikut adalah penjelasan metode penyelesaian masalah dengan kedua pendekatan tersebut.

1. Algoritma Iteratif

   Pendekatan iteratif melibatkan penggunaan perulangan untuk mencari kelipatan persekutuan terkecil dari x dan y.

   Langkah-langkahnya sebagai berikut:

   - Inisialisasi variabel dengan nilai maksimum antara x dan y sebagai kandidat awal untuk LCM.
   - Gunakan perulangan untuk memeriksa apakah kandidat tersebut merupakan kelipatan dari x dan y.
   - Jika kandidat memenuhi syarat, kandidat tersebut adalah LCM.
   - Jika tidak, tingkatkan nilai kandidat dan ulangi proses hingga ditemukan LCM.

2. Algoritma Rekursif

   Pendekatan rekursif menggunakan fungsi yang memanggil dirinya sendiri untuk mencari LCM. Proses ini melibatkan perhitungan kelipatan hingga ditemukan nilai yang memenuhi syarat sebagai LCM.

   Langkah-langkahnya sebagai berikut:

   - Inisialisasi nilai awal sebagai kelipatan pertama dari bilangan terbesar ($max\ (x,\ y)$).
   - Periksa apakah nilai tersebut adalah kelipatan bersama dari x dan y.
   - Jika ya, nilai tersebut adalah LCM.
   - Jika tidak, panggil kembali fungsi dengan meningkatkan nilai kelipatan dan ulangi proses hingga ditemukan nya LCM.

Pendekatan ini memungkinkan analisis performa kedua algoritma secara objektif sehingga dapat memberikan rekomendasi yang jelas mengenai algoritma yang lebih efisien untuk mencari LCM pada berbagai ukuran input.

2.4 Analisis Perbandingan Kedua Algoritma

Berikut adalah algoritma untuk mencari LCM dengan menggunakan pendekatan algoritma iteratif dan rekursif.

1. Algoritma Iteratif

```python
def find_lcm_iterative(x, y):
    lcm = max(x, y)
    while True:
        if lcm % x == 0 and lcm % y == 0:
            return lcm
        lcm += 1

```

- Input size (n) = max(x, y)
- Operasi dasar = evaluasi kondisi *while True*
  Hal ini karena ketika kondisi di dalam *while* terpenuhi dan algoritma memutuskan untuk keluar dari *loop*, evaluasi *while True* tetap dilakukan satu kali terakhir untuk memutuskan keluar. Maka, jumlah operasi dasar adalah jumlah iterasi *loop* ditambah satu evaluasi terakhir. Jika jumlah iterasi adalah *k*, maka operasi evaluasi *while* dilakukan *k* + 1 kali.
- *Best Case*
  Terjadi ketika salah satu bilangan adalah kelipatan dari yang lain (misalnya, x = 4, y = 8). Oleh karena itu, *loop* hanya berjalan sekali karena lcm = max(x, y). Maka, kompleksitas waktunya:
  $$T(n) = 1\,(iterasi\ pertama) + 1\,(evaluasi\ terakhir) = 2$$
  $$T(n) = O(1)$$
- *Worst Case*
  Terjadi ketika x dan y adalah bilangan prima relatif (misalnya, x = 7, y = 11). Atau jika x dan y adalah *coprime* sehingga LCM(x, y) = $x \cdot y$.
  Jumlah iterasi yang diperlukan adalah:
  $$Jumlah\ iterasi = LCM(x, y) - max(x, y) + 1$$
  Jumlah iterasi akan sangat bergantung pada produk $x \cdot y$. Jika ukuran input adalah $n = max(x, y)$, maka $x \cdot y$ akan proporsional dengan n^2. Oleh karena itu, kompleksitas waktunya menjadi:
  $$T(n) = O(n^2)$$

2. Algoritma Rekursif

```python
def find_lcm_recursive(x, y, candidate=None):
    if candidate is None:
        candidate = max(x, y)
    if candidate % x == 0 and candidate % y == 0:
        return candidate
    return find_lcm_recursive(x, y, candidate + 1)

```

- Relasi Rekurens
  1. $T(0) = 1, n = 0$
  2. $T(n) = T(n-1) + 1, n > 0$
- $T(n) = T(n-1) + 1$
  $= [T(n-2) + 1] + 1 = T(n-2) + 2$
  $= [T(n-3) + 1] + 2 = T(n-3) + 3$
  .
  .
  .
  $= T(n-i) + i$

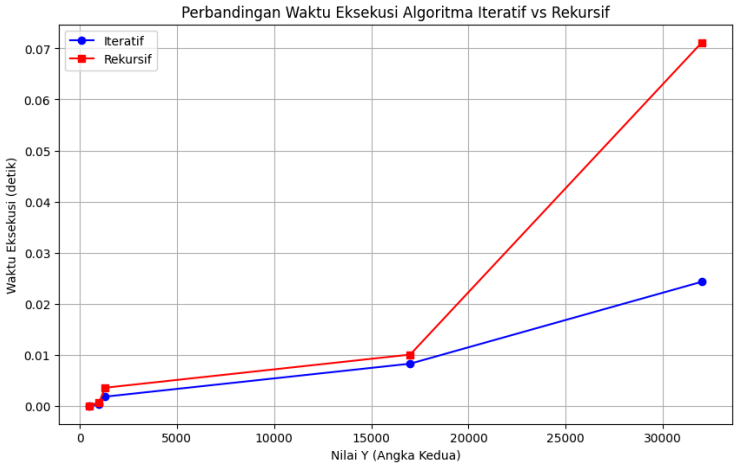
  Saat basis:
  $T(n-i) = T(0)$
  $n - i = 0$
  $i = n$
  Subtitusi:
  $T(n) = T(n-i) + i$
  $= T(n-n) + n$
  $= T(0) + n$
  $= 1 + n$
  $= n \in O(n)$
  Namun, dalam *worst case* di mana $LCM(x, y) = x \cdot y$, maka n $\approx x \cdot y$ sehingga:

$$T(n) = O(n^2)$$

3. Perbandingan *Running Time*

x dan y	Iteratif	Rekursif
x = 250 y = 500	0.000011 sekon	0.000012 sekon
x = 300 y = 1000	0.000336 sekon	0.000693 sekon
x = 1000	0.001792 sekon	0.003542 sekon

y = 1300		
x = 2500 y = 17000	0.008254 sekon	0.010036 sekon
x = 15000 y = 32000	0.024290 sekon	0.071129 sekon



Dari hasil diatas, terlihat bahwa algoritma rekursif dan iteratif sama-sama memiliki kompleksitas waktu $O(n^2)$, tetapi algoritma rekursif terbukti lebih lambat. Penyebab utama perbedaan ini adalah *overhead* pemanggilan fungsi pada algoritma rekursif, dimana setiap pemanggilan memerlukan pengelolaan *call stack* dan alokasi memori tambahan. Sebagai contoh data menunjukkan bahwa saat ukuran input meningkat, perbedaan waktu eksekusi menjadi semakin signifikan, seperti untuk x = 15000 dan y = 32000, rekursif membutuhkan waktu sekitar 0.071129 detik, hampir tiga kali lipat lebih lambat dari iteratif.

PENUTUP

3.1 Kesimpulan

Dari hasil percobaan dan analisis yang telah dilakukan, diperoleh kesimpulan sebagai berikut.

1. Kompleksitas Teoretis Sama

   Algoritma iteratif dan rekursif memiliki kompleksitas waktu yang sama, yaitu $O(n^2)$ karena keduanya bergantung pada pencarian kelipatan persekutuan terkecil (LCM) dalam ruang pencarian yang besar.
2. Performa Praktis Berbeda

   Algoritma iteratif terbukti lebih efisien dibandingkan algoritma rekursif. Penyebab utama perbedaan ini adalah *overhead* pemanggilan fungsi pada algoritma rekursif yang membutuhkan pengelolaan *call stack* dan alokasi memori tambahan.
3. Efisiensi pada Input Besar

   Perbedaan waktu eksekusi antara kedua algoritma semakin signifikan saat ukuran input meningkat. Sebagai contoh, untuk pasangan x = 15000 dan y = 32000, algoritma iteratif membutuhkan waktu 0.024290 sekon, sedangkan rekursif membutuhkan waktu 0.071129 sekon, hampir tiga kali lebih lambat.
4. Rekomendasi

   Berdasarkan hasil percobaan, algoritma iteratif lebih disarankan untuk digunakan dalam pencarian LCM, terutama untuk input yang besar karena lebih hemat waktu dan memori.

Hasil penelitian ini memberikan wawasan tambahan terkait efisiensi algoritma dalam pencarian LCM serta membantu memilih metode yang lebih optimal untuk aplikasi komputasi yang membutuhkan pengolahan data besar.

DAFTAR PUSTAKA

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.).

GeeksforGeeks. (2024, November 25). *How to teach LCM (Least Common Multiple)*. Diambil kembali dari https://www.geeksforgeeks.org/how-to-teach-lcm/

Katz, V. J. (2014). *History of Mathematics* (3rd ed.).

Parewa Labs Pvt. Ltd. . (t.thn.). *Python Program to Find LCM*. Diambil kembali dari Programiz: https://www.programiz.com/python-programming/examples/lcm

Rosen, K. H. (2012). *Discrete Mathematics and Its Applications* (7th ed.).

Weisstein, E. W. (2024, Desember 18). *Least Common Multiple*. Diambil kembali dari MathWorld--A Wolfram Web Resource: https://mathworld.wolfram.com/LeastCommonMultiple.html