Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com

P-ISSN: 2810-093X
E-ISSN: 2810-0166

Analisis efisiensi pencarian greatest common divisor dengan metode euclidean algorithms, middle school procedure dan CIC

[1]Fenisa Lourence Br Tobing*, [2]Alex Chandra, [3]Fenina Adline Twince Tobing, [4]Rena Nainggolan, [5]Prayogo

[12]Program Studi Manajemen Informatika, AMIK Widya Loka, Medan, Sumatera Utara, Indonesia
[3]Program Studi Informatika, Fakultas Teknik dan Informatika Universitas Multimedia Nusantara, Tangerang, Banten, Indonesia
[4]Program Studi Komputerisasi Akuntansi, Universitas Methodist Indonesia, Medan, Sumatera Utara, Indonesia
[5]Program Studi Teknik Informatika, AMIK Mapan, Tangerang, Banten, Indonesia
Email : [1]fenishatobing@gmail.com, [2]Alexsiburian03@gmail.com, [3]fenina.tobing@umn.ac.id

Received: December 28,2021, **Revised:** January 8, 2022, **Accepted:** January 10, 2022

ABSTRAK

Permasalahan yang ada saat mencari GCD sangatlah beragam, untuk itu perlu diteliti metode mana yang efisiensinya paling tinggi untuk setiap masalah yang ada saat mencarinya. Efisiensi yang kita cari dilihat dari faktor pemakaian memori dan waktu dalam menjalankan algoritma tersebut. Dalam penelitian ini, digunakam tiga metode tersebut dalam mencari *Greatest Common Divisor (GCD) yaitu Euclidean Algorithms, Consecutive Integer Checking (CIC)* dan *Middle School Procedure.* Hasil penelitian menunjukkan bahwa metode *Consecutive Integer Checking* menggunakan waktu yang paling sedikit dibandingkan dua metode lainnya, tetapi metode ini menggunakan memori yang sangat banyak daripada metode lain sehingga metode ini tidak dapat dikatakan sebagai metode yang paling efisien. Metode *Euclidean Algorithms* adalah metode yang paling efektif karena tidak memerlukan waktu yang banyak dan memori yang digunakan juga sedikit.

Kata Kunci - Greatest Common Divisor, Efisiensi, Euclidean Algorithm, Consecutive Integer Checking (CIC), Middle School Procedur

ABSTRACT

The problems that exist when searching for GCD are very diverse, so it is necessary to examine which method has the highest efficiency for each problem that occurs when searching for it. The efficiency we are looking for is seen from the memory usage factor and the time it takes to run the algorithm. In this study, these three methods were used to find the Greatest Common Divisor (GCD). namely Euclidean Algorithm, Consecutive Integer Checking (CIC), and Middle School Procedure.

The results showed that the Consecutive Integer Checking method used the least amount of time compared to the two methods other, but this method uses a lot more memory than other methods so it cannot be said to be the most efficient method. The Euclidean Algorithms method is the most effective method because it doesn't take much time Lots and memory that used too a little.

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com

P-ISSN: 2810-093X
E-ISSN: 2810-0166

Keywords - Greatest Common Divisor, Efficiency, Euclidean Algorithm, Consecutive Integer Checking (CIC), Middle School Procedure

1. Pendahuluan

Greatest Common Divisor atau yang biasa lebih kita kenal sebagai Faktor Persekutuan Terbesar (FPB) adalah bilangan bulat positif terbesar yang dapat membagi habis dua buah bilangan.

Permasalahan yang ada saat mencari sebuah GCD sangatlah beragam, untuk itu perlu dilakukannya analisa metode mana yang efisiensinya paling tinggi untuk setiap masalah yang ada saat mencari sebuah bilangan GCD.
Ada beberapa metode yang akan dianalisa untuk mencari Greatest Common Divisor, yaitu Euclidean Algorithm, Consecutive Integer Checking (CIC), dan Middle School Procedure.
Tolak ukur yang digunakan untuk menentukan metode mana yang paling efisien dalam mencari GCD dari dua bilangan adalah dari waktu yang digunakan untuk melakukan eksekusi dan memori yang digunakan untuk melakukan eksekusi tersebut. Efisiensi yang kita cari dilihat dari faktor pemakaian memori dan waktu dalam menjalankan algoritma tersebut.

RUMUSAN MASALAH

- Apa metode yang paling efisien digunakan untuk mencari GCD dari dua bilangan ?
- Mengapa perlu menggunakan metode berbeda – beda untuk mencari GCD ?
- Bagaimana cara mengetahui metode yang tepat untuk mencari suatu GCD ?

2. Metode Penelitian

Metode-metode yang akan kami gunakan dalam penelitian kali ini adalah Euclidean Algorithms, Middle School Procedure, dan Consecutive Integer Checking (CIC).

2.1 Euclidean Algorithms

Dalam metode Euclidean Algorithms, pencarian FPB dilakukan dengan operasi modulo, penjumlahan, dan pengurangan.

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com
P-ISSN: 2810-093X
E-ISSN: 2810-0166

```cpp
// C++ program to find GCD of two numbers
#include <iostream>
using namespace std;
// Recursive function to return gcd of a and b
int gcd(int a, int b)
{
    // Everything divides 0
    if (a == 0)
       return b;
    if (b == 0)
       return a;

    // base case
    if (a == b)
        return a;

    // a is greater
    if (a > b)
        return gcd(a-b, b);
    return gcd(a, b-a);
}

// Driver program to test above function
int main()
{
    int a = 98, b = 56;
    cout<<"GCD of "<<a<<" and "<<b<<" is "<<gcd(a, b);
    return 0;
}
```

Gambar 1 - Contoh *Koding Euclidean Algorithms*

2.2 . Middle School Procedure

Metode Middle School Procedure bisa digunakan untuk mencari GCD yang lebih dari dua bilangan. *Middle School Procedure* lebih kompleks dan lebih lambat dibandingkan *Euclidean Algorithm.* Metode ini mencari GCD dengan membagi bilangan yang kita inginkan terus menerus.

```cpp
// C++ implementation of above algorithm
#include <bits/stdc++.h>
#define MAXFACTORS 1024
using namespace std;

// struct to store factorization of m and n
typedef struct
{

    int size;
    int factor[MAXFACTORS + 1];
    int exponent[MAXFACTORS + 1];

} FACTORIZATION;

// Function to find the factorization of M and N
void FindFactorization(int x, FACTORIZATION* factorization)
{
    int i, j = 1;
    int n = x, c = 0;
    int k = 1;
    factorization->factor[0] = 1;
    factorization->exponent[0] = 1;
```

Gambar 2 - Contoh Koding *Middle School Procedure 1*

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com
P-ISSN: 2810-093X
E-ISSN: 2810-0166

```
        for (i = 2; i <= n; i++) {
            c = 0;

            while (n % i == 0) {
                c++;

                // factorization->factor[j]=i;
                n = n / i;
                // j++;
            }

            if (c > 0) {
                factorization->exponent[k] = c;
                factorization->factor[k] = i;
                k++;
            }
        }

        factorization->size = k - 1;
    }
```

Gambar 3 - Contoh Koding *Middle School Procedure 2*

```
// Function to print the factors
void DisplayFactorization(int x, FACTORIZATION factorization)
{

    int i;
    cout << "Prime factor of << x << = ";

    for (i = 0; i <= factorization.size; i++) {

        cout << factorization.factor[i];

        if (factorization.exponent[i] > 1)
            cout << "^" << factorization.exponent[i];

        if (i < factorization.size)
            cout << "*";

        else
            cout << "\n";
    }
}
```

Gambar 4 - Contoh Koding *Middle School Procedure 3*

```
// Function to find the gcd using Middle School procedure
int gcdMiddleSchoolProcedure(int m, int n)
{

    FACTORIZATION mFactorization, nFactorization;

    int r, mi, ni, i, k, x = 1, j;

    // Step 1.
    FindFactorization(m, &mFactorization);
    DisplayFactorization(m, mFactorization);

    // Step 2.
    FindFactorization(n, &nFactorization);
    DisplayFactorization(n, nFactorization);

    // Steps 3 and 4.
    // Procedure algorithm for computing the
    // greatest common divisor.
    int min;
    i = 1;
    j = 1;
    while (i <= mFactorization.size && j <= nFactorization.size) {
        if (mFactorization.factor[i] < nFactorization.factor[j])
            i++;
```

Gambar 5 - Contoh Koding *Middle School Procedure 4*

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com
P-ISSN: 2810-093X
E-ISSN: 2810-0166

```
        else /* if arr1[i] == arr2[j] */
        {
            min = nFactorization.exponent[i] > nFactorization.exponent[j]
                    ? nFactorization.exponent[j]
                    : nFactorization.exponent[i];

            x = x * nFactorization.factor[i] * min;
            i++;
            j++;
        }
    }

    return x;
}

// Driver code
int main()

{

    int m = 10, n = 15;
    cout << "GCD(" << m << ", " << n << ") = "
         << gcdMiddleSchoolProcedure(m, n);

    return (0);
}
```

Gambar 6 - Contoh Koding *Middle School Procedure 5*

3. CONSECUTIVE INTEGER CHECKING

Sedangkan *Consecutive Integer Checking* mencari GCD dengan cara mencari sebuah bilangan yang bisa me-modulo semua bilangan yang disediakan menggunakan bilangan yang sama.

Contoh Koding *Consecutive Integer Checking :*

```
var a = 98;
var b = 56;

for(i=b; i>0; i--){
    if(a%i == 0){
        if(b%i == 0){
            console.log("GCD of 98 and 56 = " + i);
        break;
        }
    }
}
```

Gambar 7 - Contoh Koding *Consecutive Integer Checking*

3. Hasil dan Pembahasan

Berikut adalah dataset yang akan digunakan untuk test-case :

1. GCD (98, 56)
2. GCD (100, 30)
3. GCD (198, 110)

1. Dataset Nomor 1 :

Metode Euclid :

```
cpu time: 0.02 sec, memory peak: 3 Mb, absolute service time: 0,71 sec
Compilation time: 0.63 sec, absolute running time: 0.08 sec,

GCD of 98 and 56 is 14
```

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com

P-ISSN: 2810-093X
E-ISSN: 2810-0166

Metode Middle School :

```
cpu time: 0.01 sec, memory peak: 3 Mb, absolute service time: 1,5 sec
Compilation time: 1.43 sec, absolute running time: 0.06 sec,

Prime factor of << x << = 1*2*7^2
Prime factor of << x << = 1*2^3*7
GCD(98, 56) = 14
```

Metode Consecutive Integer Checking :

```
memory peak: 15 Mb, absolute service time: 0,16 sec
Absolute running time: 0.15 sec, cpu time: 0.13 sec,

GCD of 98 and 56 = 14
```

2. Dataset Nomor 2 :

Metode Euclid :

```
cpu time: 0.01 sec, memory peak: 3 Mb, absolute service time: 0,71 sec
Compilation time: 0.63 sec, absolute running time: 0.07 sec,

GCD of 100 and 30 is 10
```

Metode Middle School :

```
cpu time: 0.01 sec, memory peak: 3 Mb, absolute service time: 1,52 sec
Compilation time: 1.43 sec, absolute running time: 0.08 sec,

Prime factor of << x << = 1*2^2*5^2
Prime factor of << x << = 1*2*3*5
GCD(100, 30) = 10
```

Metode Consecutive Integer Checking :

```
memory peak: 15 Mb, absolute service time: 0,18 sec
Absolute running time: 0.18 sec, cpu time: 0.15 sec,

GCD of 100 and 30 = 10
```

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com

P-ISSN: 2810-093X
E-ISSN: 2810-0166

3. Dataset Nomor 3 :

Metode Euclid :

```
cpu time: 0.02 sec, memory peak: 3 Mb, absolute service time: 0,41 sec
Compilation time: 0.33 sec, absolute running time: 0.08 sec,

GCD of 198 and 110 is 22
```

Metode Middle School :

```
cpu time: 0.02 sec, memory peak: 3 Mb, absolute service time: 1,52 sec
Compilation time: 1.43 sec, absolute running time: 0.08 sec,

Prime factor of << x << = 1*2*3^2*11
Prime factor of << x << = 1*2*5*11
GCD(198, 110) = 22
```

Metode Consecutive Integer Checking :

```
memory peak: 15 Mb, absolute service time: 0,14 sec
Absolute running time: 0.14 sec, cpu time: 0.12 sec,

GCD of 198 and 110 = 22
```

4. Kesimpulan

Penelitian yang dilakukan berdasarkan waktu dan memori yang digunakan untuk mencari sebuah GCD menggunakan 3 metode yaitu *Euclidean Algorithm*, *Consecutive Integer Checking (CIC)*, dan *Middle School Procedure*

Berdasarkan hasil penelitian yang telah dilakukan dan melakukan perbandingan dengan penelitian-penelitian yang ada sebelumnya, dapat kami simpulkan bahwa metode dengan *run-time* paling cepat (waktu paling singkat) adalah metode *Consecutive Integer Checking (CIC)*, tetapi metode ini menggunakan sistem memori yang sangat besar.

Sehingga, metode yang paling efektif adalah metode *Euclidean Algorithm*. Karena memiliki *run-time* yang lebih cepat dari *metode Middle School Procedure*, dan tidak menggunakan sistem memori yang besar (menggunakan sistem memori yang sangat kecil) seperti *metode Consecutive Integer Checking (CIC).*

DOI : 10.54593/jstekwid.v1i1.62

Jurnal Sains dan Teknologi Widyaloka
Volume 1, Nomor 1, Januari 2022: halaman 81-88
https://jurnal.amikwidyaloka.ac.id/index.php/jstekwid
jurnal@amikwidyaloka.ac.id / editor.jstekwid@gmail.com

P-ISSN: 2810-093X
E-ISSN: 2810-0166

Referensi

[1] Program to find GCD or HCF of two numbers. (2018, November22) . Retrieved *from* *https://www.geeksforgeeks.org/c-program-find- gcd-hcf-two-numbers/.*

[2] GargCheck, K. D., & Garg, K. D. (2019, June 24). Program to find GCD or HCF of two numbers using Middle School Procedure. Retrieved from *https://www.geeksforgeeks.org/program-to-find- gcd-or-hcf-of-two-numbers-using-middle-school-procedure/.*

[3] Greatest Common Divisor. (n.d.). Retrieved from *http://mathworld.wolfram.com/GreatestCommon Divisor.html.*

[4] Euclidean Algorithm. (n.d.). Retrieved from *http://mathworld.wolfram.com/EuclideanAlgorithm.html.*

[5] Berkaay. (n.d.). Middle school procedure for computing gcd m n Steps 1 Find prime factors of m2. Retrieved from *https://www.coursehero.com/file/p7alqjp/Middle-school-procedure-for-computing-gcd-m-n- Steps-1-Find-prime-factors-of-m-2/.*

[6] Kid_Crown_Finch16. (n.d.). Methods for gcd mn Middle school procedure Step 1 Find the prime factorization : Course Hero. Retrieved from *https://www.coursehero.com/file/p62g3oh/Meth ods-for-gcd-mn-Middle-school-procedure-Step-1-Find-the-prime-factorization/.*

[7] Fitri Dwi Lestari - AMIK BSI Pontianak. (n.d.). Analisa Algoritma Faktor Persekutuan Terbesar (FPB) Menggunakan Bahasa Pemrograman C . etrieved from *https://ejournal.bsi.ac.id/ejurnal/index.php/evolusi/article/view/1728/0.*

[8] Al-Haija, Q. A. (n.d.). Reviewing and Analyzing Efficient GCD/LCM Algorithms for Cryptographic Design. Retrieved from *https://www.academia.edu/35267009/Reviewing_and_Analyzing_Efficient_GCD_LCM_Algorith ms_for_Cryptographic_Design.*

DOI : 10.54593/jstekwid.v1i1.62