

LAPORAN
TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA
Perkalian Polinom dengan Algoritma *Brute Force* dan *Divide and Conquer*



Disusun oleh:
13518117 – Muhammad Firas

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020

BAB I TEORI DASAR

Algoritma *Brute Force* adalah pendekatan yang *straightforward* untuk memecahkan suatu masalah. Algoritma *Brute Force* memecahkan masalah dengan sangat sederhana, langsung, dan jelas. Sedangkan algoritma *Divide and Conquer* adalah algoritma yang membagi persoalan (*Divide*) menjadi beberapa upa-masalah yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Kemudian, masing-masing upa-masalah tersebut diselesaikan secara langsung atau secara rekursif (*Conquer*). Solusi masing-masing upa-masalah yang telah diselesaikan akan digabungkan (*Combine*) sehingga membentuk solusi persoalan semula.

Dalam tugas ini, masalah yang akan diselesaikan dengan algoritma *Brute Force* dan algoritma *Divide and Conquer* adalah perkalian antara dua polinom. Polinom/suku banyak adalah bentuk suku-suku dengan banyak terhingga yang disusun dari variabel dan konstanta. Contoh dari polinom adalah $A(x) = 7 - 4x + x^2$. Secara umum, sebuah polinom dapat dituliskan sebagai berikut.

$$A(x) = \sum_{i=0}^n a_i x^i$$

Dalam tugas kali ini, diharuskan membuat dua buah pustaka (library) dalam Bahasa C/C++/Java yang masing-masing dapat melakukan perkalian polinom dengan algoritma (a) *Brute Force*, dan (b) *Divide and Conquer*. Panjang suku polinom (n) tidak dibatasi, namun panjang dua buah polinom masukan sama. Algoritma *Divide and Conquer* untuk perkalian polinom yang diterapkan adalah algoritma dengan kompleksitas $O(n \log 3)$.

BAB II ANALISIS PERSOALAN

2.1 Algoritma *Brute Force*

Misal $A(x) = \sum_{i=0}^n a_i x^i$ dan $B(x) = \sum_{i=0}^n b_i x^i$.

Maka $C(x) = \sum_{k=0}^{2n} c_k x^k = A(x)B(x)$ dengan

$$c_k = \sum_{i=0}^k a_i b_{k-i}$$

Untuk semua $0 \leq k \leq 2n$.

Pendekatan langsung dengan algoritma *Brute Force* adalah menghitung semua c_k menggunakan rumus di atas. Jumlah total perkalian dan penambahan yang dibutuhkan masing-masing adalah $O(n^2)$ dan $O(n^2)$. Karenanya kompleksitasnya adalah $O(n^2)$.

2.2 Algoritma *Divide and Conquer*

Misal $A(x) = \sum_{i=0}^n a_i x^i$ dan $B(x) = \sum_{i=0}^n b_i x^i$.

Maka:

$$A_0(x) = a_0 + a_1 x + \dots + a_{\lfloor \frac{n}{2} \rfloor - 1} x^{\lfloor \frac{n}{2} \rfloor - 1}$$

$$A_1(x) = a_{\lfloor \frac{n}{2} \rfloor} + a_{\lfloor \frac{n}{2} \rfloor + 1} x + \dots + a_n x^{n - \lfloor \frac{n}{2} \rfloor}$$

$$B_0(x) = b_0 + b_1 x + \dots + b_{\lfloor \frac{n}{2} \rfloor - 1} x^{\lfloor \frac{n}{2} \rfloor - 1}$$

$$B_1(x) = b_{\lfloor \frac{n}{2} \rfloor} + b_{\lfloor \frac{n}{2} \rfloor + 1} x + \dots + b_n x^{n - \lfloor \frac{n}{2} \rfloor}$$

Untuk menghitung perkalian $A(x)B(x)$ dengan upa-polinom di atas tentunya dapat dilakukan dengan 4 perkalian, tapi dengan total perkalian tersebut dapat diperkecil dengan sebagai berikut:

$$Y(x) = (A_0(x) + A_1(x)) \times (B_0(x) + B_1(x))$$

$$U(x) = A_0(x)B_0(x)$$

$$Z(x) = A_1(x)B_1(x)$$

Maka:

$$A(x)B(x) = U(x) + [Y(x) - U(x) - Z(x)]x^{\lfloor \frac{n}{2} \rfloor} + Z(x) \times x^{2\lfloor \frac{n}{2} \rfloor}$$

Dengan menggunakan rumus di atas, total perkalian dapat diperkecil menjadi 3 perkalian. Sehingga, didapat kompleksitas waktu dari algoritma ini adalah $O(n^{\log 3})$.

BAB III

IMPLEMENTASI

Untuk membuat program untuk menyelesaikan persoalan ini, penulis menggunakan bahasa Java dengan kode program sebagai berikut.

```
public class Polinom {
    //atribut
    private int ordeP;
    private int[] koefPolinom;
    public static int countTambah = 0;
    public static int countKali = 0;

    //constructor
    public Polinom(int orde, boolean random){
        this.ordeP = orde + 1;
        this.koefPolinom = new int[this.ordeP];

        if(random){
            Random r = new Random();
            for (int i = 0; i < this.ordeP; i++) {
                this.koefPolinom[i] = (r.nextInt(21) - 10);
            }
        }
        else{
            for (int i = 0; i < this.ordeP; i++) {
                this.koefPolinom[i] = 0;
            }
        }
    }

    //copy constructor
    public Polinom(Polinom P){
        this.ordeP = P.ordeP;
        this.koefPolinom = P.koefPolinom;
    }

    private int getLength(){
        return (this.ordeP);
    }

    private int getLastIdx(){
        return (this.ordeP - 1);
    }
    ...
}
```

Method untuk mencetak polinom ke layar

```
public void printPolinom(){
    boolean firstPass = false;
    for (int i = 0; i < this.ordeP; i++) {
        if (this.koefPolinom[i] != 0) {
            if(!firstPass){
                System.out.print(this.koefPolinom[i]);
                firstPass = true;
            }
            else {
                if(this.koefPolinom[i] != 1 && this.koefPolinom[i] != -1) {
                    System.out.print(abs(this.koefPolinom[i]));
                }
            }
            if (i != 0) {
                if (i != 1) {
                    System.out.print("x^" + i);
                } else {
                    System.out.print("x");
                }
            }
            if(i != (this.ordeP - 1)) {
                if (this.koefPolinom[i+1] >= 0) {
                    System.out.print(" + ");
                }
                else if (this.koefPolinom[i+1] < 0) {
                    System.out.print(" - ");
                }
            }
        }
    }
    System.out.println();
}
```

Perkalian dua polinom dengan algoritma *Brute Force*

```
public static Polinom BruteForceMulti(Polinom P1, Polinom P2){
    countTambah = 0;
    countKali = 0;
    Polinom bf = new Polinom(P1.ordeP + P2.ordeP - 2, false);
    for (int i = 0; i < P1.ordeP; i++) {
        for (int j = 0; j < P2.ordeP; j++) {
            bf.koefPolinom[i+j] += P1.koefPolinom[i] * P2.koefPolinom[j];
            countTambah++;
            countKali++;
        }
    }
    return bf;
}
```

Method penjumlahan dua polinom yang akan digunakan untuk algoritma *Divide and Conquer*

```
private Polinom tambahPolinom(Polinom P){
    Polinom sum;
    if (this.ordeP >= P.ordeP){
        sum = new Polinom(this.ordeP-1, false);
    }
    else{
        sum = new Polinom(P.ordeP-1, false);
    }
    for (int i = 0; i < this.ordeP; i++) {
        sum.koefPolinom[i] = this.koefPolinom[i];
    }
    for (int i = 0; i < P.ordeP; i++) {
        sum.koefPolinom[i] += P.koefPolinom[i];
        countTambah++;
    }
    return sum;
}
```

Method pengurangan dua polinom yang akan digunakan untuk algoritma *Divide and Conquer*

```
private Polinom kurangPolinom(Polinom P){
    Polinom sub;
    if (this.ordeP >= P.ordeP){
        sub = new Polinom(this.ordeP-1, false);
    }
    else{
        sub = new Polinom(P.ordeP-1, false);
    }
    for (int i = 0; i < this.ordeP; i++) {
        sub.koefPolinom[i] = this.koefPolinom[i];
    }
    for (int i = 0; i < P.ordeP; i++) {
        sub.koefPolinom[i] -= P.koefPolinom[i];
        countTambah++;
    }
    return sub;
}
```

Method untuk membagi suatu polinom menjadi upa-polinom

```
private static Polinom subPolinom(Polinom P, int startIdx, int endIdx){
    Polinom subP = new Polinom(endIdx-startIdx, false);
    for (int i = 0, j = startIdx; i < subP.ordeP && j <= endIdx; i++, j++) {
        subP.koefPolinom[i] = P.koefPolinom[j];
    }
    return subP;
}
```

Method perkalian suatu polinom dengan variabel orde tertentu

```
private Polinom Pengali(int pengali){
    Polinom newP = new Polinom(this.getLastIdx() + pengali, false);
    for (int i = 0, j = pengali; i < this.getLength() && j < this.getLength() + pengali; i++, j++) {
        newP.koefPolinom[j] = this.koefPolinom[i];
    }
    return newP;
}
```

Perkalian dua polinom dengan algoritma *Divide and Conquer*

```
public static Polinom DivideAndConquerMulti(Polinom A, Polinom B){
    Polinom DnC = new Polinom(A.ordeP+B.ordeP - 2, false);
    if(A.getLength() == 1 && B.getLength() == 1){
        DnC.koefPolinom[0] = A.koefPolinom[0] * B.koefPolinom[0];
        countKali++;
        return DnC;
    }
    else{
        Polinom A0 = subPolinom(A, 0, (A.ordeP/2) - 1);
        Polinom A1 = subPolinom(A, A.ordeP/2, A.getLastIdx());
        Polinom B0 = subPolinom(B, 0, (B.ordeP/2) - 1);
        Polinom B1 = subPolinom(B, B.ordeP/2, B.getLastIdx());

        Polinom A0A1sum = A0.tambahPolinom(A1);
        Polinom B0B1sum = B0.tambahPolinom(B1);

        Polinom Y = DivideAndConquerMulti(A0A1sum, B0B1sum);
        Polinom U = DivideAndConquerMulti(A0,B0);
        Polinom Z = DivideAndConquerMulti(A1,B1);

        Polinom subYUZ = Y.kurangPolinom(U).kurangPolinom(Z);

        Polinom PengaliSubYUZ = subYUZ.Pengali(A.ordeP/2);
        Polinom PengaliZ = Z.Pengali((A.ordeP/2)*2);

        DnC = U.tambahPolinom(PengaliSubYUZ).tambahPolinom(PengaliZ);
        return DnC;
    }
}
```

Polinom.java setelah itu akan dibuat sebuah pustaka (*library*) dengan format .jar yang nantinya akan di-*import* dengan cara memasukkannya ke dalam java classpath, contoh main program adalah seperti berikut:

```
public class MainProgram {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        System.out.print("Masukkan banyaknya n: ");
        n = in.nextInt();
        Polinom A = new Polinom(n, true);
        Polinom B = new Polinom(n, true);

        System.out.print("A(x) = ");
        A.printPolinom();
        System.out.print("B(x) = ");
        B.printPolinom();
        System.out.println();

        long StartTime = System.nanoTime();
        Polinom bruteForce = A.BruteForceMulti(A,B);
        long EndTime = System.nanoTime();
        long TimeElapsed = EndTime-StartTime;
        double Microseconds = ((double)TimeElapsed)/1000;

        System.out.println("Perkalian dengan algoritma Brute Force:");
        System.out.print("A(x)B(x) = ");
        bruteForce.printPolinom();
        System.out.println("Kompleksitas waktu : " + Microseconds + "
mikrosekon");
        System.out.println("Total operasi tambah : " + bruteForce.countTambah);
        System.out.println("Total operasi kali : " + bruteForce.countKali);
        System.out.println();

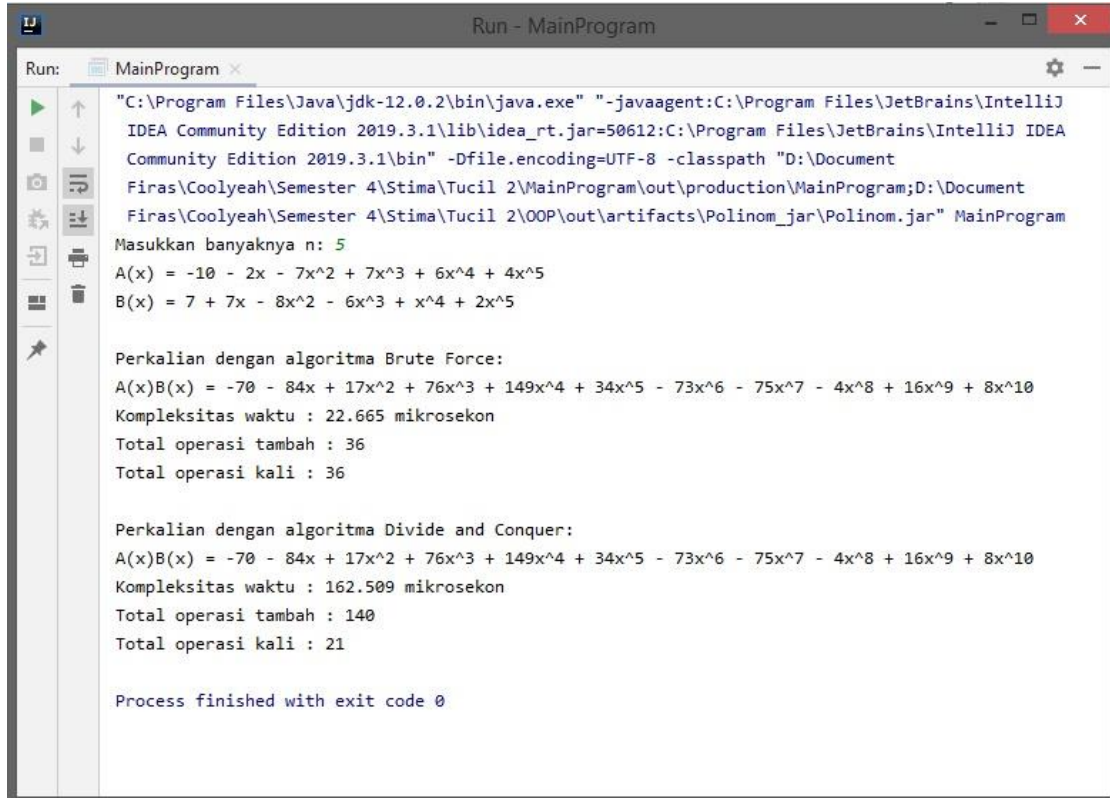
        bruteForce.countTambah = 0;
        bruteForce.countKali = 0;

        long StartTime2 = System.nanoTime();
        Polinom DnC = A.DivideAndConquerMulti(A,B);
        long EndTime2 = System.nanoTime();
        long TimeElapsed2 = EndTime2-StartTime2;
        double Microseconds2 = ((double)TimeElapsed2)/1000;

        System.out.println("Perkalian dengan algoritma Divide and Conquer:");
        System.out.print("A(x)B(x) = ");
        DnC.printPolinom();
        System.out.println("Kompleksitas waktu : " + Microseconds2 + "
mikrosekon");
        System.out.println("Total operasi tambah : " + DnC.countTambah);
        System.out.println("Total operasi kali : " + DnC.countKali);
    }
}
```


BAB IV UJI KASUS

- *Screen-shot* input-output program untuk $n = 5$



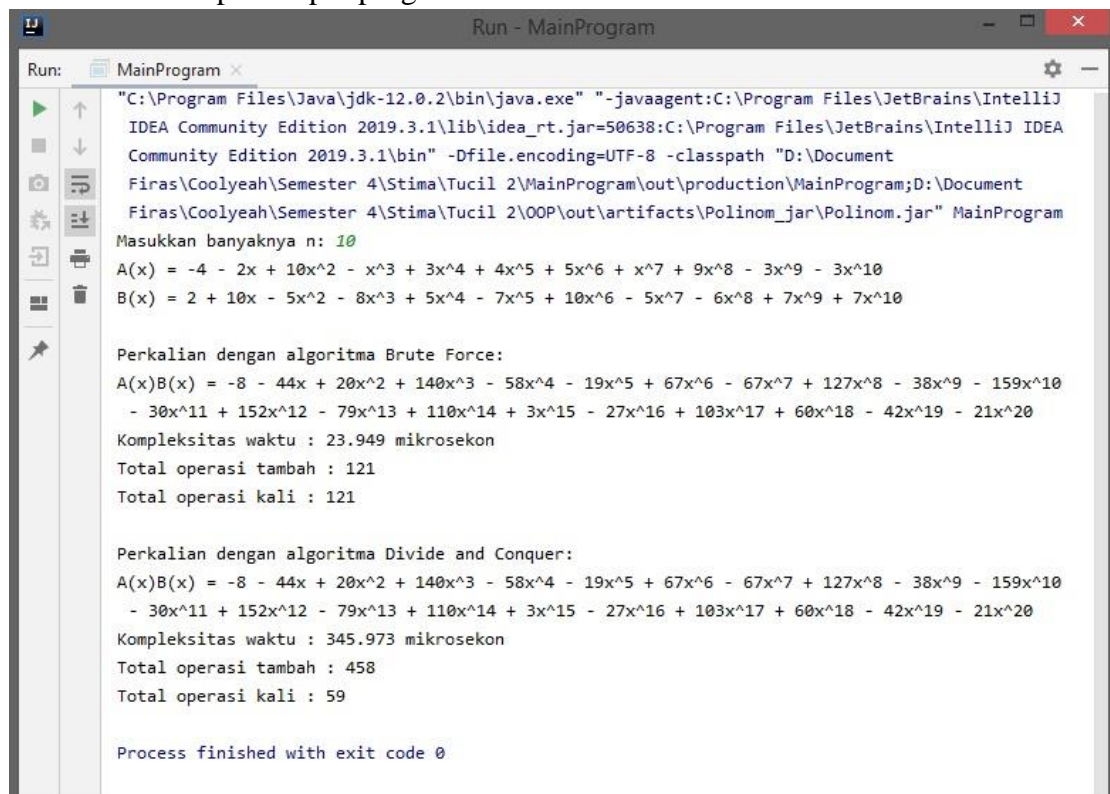
```
Run: MainProgram
"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\lib\idea_rt.jar=50612:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\bin" -Dfile.encoding=UTF-8 -classpath "D:\Document Firas\Coolyeah\Semester 4\Stima\Tucil 2\MainProgram\out\production\MainProgram;D:\Document Firas\Coolyeah\Semester 4\Stima\Tucil 2\OOP\out\artifacts\Polinom_jar\Polinom.jar" MainProgram
Masukkan banyaknya n: 5
A(x) = -10 - 2x - 7x^2 + 7x^3 + 6x^4 + 4x^5
B(x) = 7 + 7x - 8x^2 - 6x^3 + x^4 + 2x^5

Perkalian dengan algoritma Brute Force:
A(x)B(x) = -70 - 84x + 17x^2 + 76x^3 + 149x^4 + 34x^5 - 73x^6 - 75x^7 - 4x^8 + 16x^9 + 8x^10
Kompleksitas waktu : 22.665 mikrosekond
Total operasi tambah : 36
Total operasi kali : 36

Perkalian dengan algoritma Divide and Conquer:
A(x)B(x) = -70 - 84x + 17x^2 + 76x^3 + 149x^4 + 34x^5 - 73x^6 - 75x^7 - 4x^8 + 16x^9 + 8x^10
Kompleksitas waktu : 162.509 mikrosekond
Total operasi tambah : 140
Total operasi kali : 21

Process finished with exit code 0
```

- *Screen-shot* input-output program untuk $n = 10$



```
Run: MainProgram
"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\lib\idea_rt.jar=50638:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\bin" -Dfile.encoding=UTF-8 -classpath "D:\Document Firas\Coolyeah\Semester 4\Stima\Tucil 2\MainProgram\out\production\MainProgram;D:\Document Firas\Coolyeah\Semester 4\Stima\Tucil 2\OOP\out\artifacts\Polinom_jar\Polinom.jar" MainProgram
Masukkan banyaknya n: 10
A(x) = -4 - 2x + 10x^2 - x^3 + 3x^4 + 4x^5 + 5x^6 + x^7 + 9x^8 - 3x^9 - 3x^10
B(x) = 2 + 10x - 5x^2 - 8x^3 + 5x^4 - 7x^5 + 10x^6 - 5x^7 - 6x^8 + 7x^9 + 7x^10

Perkalian dengan algoritma Brute Force:
A(x)B(x) = -8 - 44x + 20x^2 + 140x^3 - 58x^4 - 19x^5 + 67x^6 - 67x^7 + 127x^8 - 38x^9 - 159x^10 - 30x^11 + 152x^12 - 79x^13 + 110x^14 + 3x^15 - 27x^16 + 103x^17 + 60x^18 - 42x^19 - 21x^20
Kompleksitas waktu : 23.949 mikrosekond
Total operasi tambah : 121
Total operasi kali : 121

Perkalian dengan algoritma Divide and Conquer:
A(x)B(x) = -8 - 44x + 20x^2 + 140x^3 - 58x^4 - 19x^5 + 67x^6 - 67x^7 + 127x^8 - 38x^9 - 159x^10 - 30x^11 + 152x^12 - 79x^13 + 110x^14 + 3x^15 - 27x^16 + 103x^17 + 60x^18 - 42x^19 - 21x^20
Kompleksitas waktu : 345.973 mikrosekond
Total operasi tambah : 458
Total operasi kali : 59

Process finished with exit code 0
```

- *Screen-shot* input-output program untuk $n = 20$

```

Run: MainProgram x
Masukkan banyaknya n: 20
A(x) = 6 - 2x - 10x^2 - 10x^3 - 4x^4 - 10x^5 + 2x^6 - 9x^7 + 5x^8 - 2x^9 + 10x^11 - 4x^12 +
10x^13 - 7x^14 - 3x^15 - 4x^16 - 5x^17 - 10x^18 - 2x^19 - 4x^20
B(x) = -2 - x - 2x^2 + 6x^3 - 6x^4 + 8x^5 - 2x^6 + 7x^7 + 2x^8 - 4x^9 + 3x^11 - 7x^12 + 5x^14 +
4x^15 + 9x^17 - 9x^18 + 3x^19 - 7x^20

Perkalian dengan algoritma Brute Force:
A(x)B(x) = -12 - 2x + 10x^2 + 70x^3 - 10x^4 + 44x^5 - 14x^6 + 38x^7 - 103x^8 - 21x^9 - 228x^10 +
118x^11 - 174x^12 + 128x^13 - 23x^14 + 158x^15 + 161x^16 - 98x^17 + 40x^18 + 102x^19 + 169x^20
+ 116x^21 + 84x^22 + 154x^23 + 48x^24 + 21x^25 - 36x^26 + 71x^27 + 167x^28 + 208x^29 + 66x^31
+ 206x^32 - 58x^33 + 133x^34 + 128x^35 + 121x^36 + 59x^37 + 100x^38 + 2x^39 + 28x^40
Kompleksitas waktu : 63.293 mikrosekond
Total operasi tambah : 441
Total operasi kali : 441

Perkalian dengan algoritma Divide and Conquer:
A(x)B(x) = -12 - 2x + 10x^2 + 70x^3 - 10x^4 + 44x^5 - 14x^6 + 38x^7 - 103x^8 - 21x^9 - 228x^10 +
118x^11 - 174x^12 + 128x^13 - 23x^14 + 158x^15 + 161x^16 - 98x^17 + 40x^18 + 102x^19 + 169x^20
+ 116x^21 + 84x^22 + 154x^23 + 48x^24 + 21x^25 - 36x^26 + 71x^27 + 167x^28 + 208x^29 + 66x^31
+ 206x^32 - 58x^33 + 133x^34 + 128x^35 + 121x^36 + 59x^37 + 100x^38 + 2x^39 + 28x^40
Kompleksitas waktu : 1078.971 mikrosekond
Total operasi tambah : 1438
Total operasi kali : 169

Process finished with exit code 0

```

- *Screen-shot* input-output program untuk $n = 50$

```

Run: MainProgram x
Masukkan banyaknya n: 50
A(x) = 4 - 2x - 2x^2 - 10x^3 + 5x^4 - 3x^5 - 6x^6 - 4x^7 + 6x^8 + 5x^9 - 2x^10 + x^11 + 9x^12 - 9x^13 - x^14 + 2x^15 - x^16 + 4x^17 +
6x^18 + 5x^19 - 8x^20 + 6x^21 - 3x^22 + 6x^23 + 6x^24 - 10x^25 - 6x^26 - 3x^27 - 7x^28 + 8x^29 + x^30 - 10x^31 + 3x^32 - 10x^33 -
2x^34 + 6x^35 - 9x^36 + 4x^37 - 3x^38 + 5x^40 - 5x^41 + 2x^42 + 6x^43 - x^44 + 9x^45 + 9x^46 - 8x^47 - 2x^48 - 8x^49 - x^50
B(x) = 8 - 4x - 4x^2 + 9x^3 - 10x^4 + 7x^5 - 6x^6 - 10x^7 - 3x^8 - 2x^9 + 8x^10 - 10x^11 + 5x^12 - 4x^13 - 10x^14 - 7x^15 - 5x^16 +
6x^17 + 6x^18 - 3x^19 - x^20 + 9x^21 + 10x^22 - 7x^23 - 10x^24 + 7x^25 - 9x^26 + 8x^27 + 3x^28 - 5x^29 + 4x^30 - x^31 + 4x^32 - 7x^33
+ 6x^34 + 4x^36 - x^37 - 3x^38 + x^39 - 6x^40 - 3x^41 + 7x^42 + 3x^43 - 10x^44 - 6x^45 + 10x^46 - 4x^47 + 7x^48 - 4x^49 + 10x^50

Perkalian dengan algoritma Brute Force:
A(x)B(x) = 32 - 32x - 24x^2 - 28x^3 + 30x^4 + 26x^5 - 164x^6 + 107x^7 - 39x^8 + 121x^9 + 55x^10 - 6x^11 + 128x^12 - 141x^13 + 182x^14 -
77x^15 - 163x^16 + 300x^17 + 45x^18 - 78x^19 - 15x^20 + 214x^21 + 100x^22 - 408x^23 + 14x^24 - 407x^25 + 129x^26 + 129x^27 - 341x^28
+ 450x^29 - 165x^30 - 182x^31 + 55x^32 - 53x^33 + 83x^34 - 13x^35 - 39x^36 + 42x^37 - 181x^38 + 341x^39 + 429x^40 - 245x^41 + 275x^42
- 45x^43 + 3x^44 + 116x^45 + 268x^46 - 161x^47 - 9x^48 - 200x^49 + 66x^50 + 312x^51 - 339x^52 - 386x^53 + 135x^54 + 59x^55 - 190x^56 +
176x^57 - 42x^58 - 271x^59 + 211x^60 - 276x^61 + 200x^62 - 60x^63 + 202x^64 + 194x^65 - 52x^66 + 208x^67 - 282x^68 - 25x^69 - 122x^70
+ 214x^71 + 36x^72 + 127x^73 - 55x^74 - 7x^75 - 92x^76 + 59x^77 + 18x^78 - 168x^79 + 191x^80 - 138x^81 + 62x^82 - 128x^83 - 240x^84 +
156x^85 - 135x^86 + 14x^87 + 140x^88 - 63x^89 - 102x^90 + 45x^91 + 80x^92 + 100x^93 + 87x^94 - 68x^95 + 130x^96 - 124x^97 + 5x^98 -
76x^99 - 10x^100
Kompleksitas waktu : 257.875 mikrosekond
Total operasi tambah : 2601
Total operasi kali : 2601

Perkalian dengan algoritma Divide and Conquer:
A(x)B(x) = 32 - 32x - 24x^2 - 28x^3 + 30x^4 + 26x^5 - 164x^6 + 107x^7 - 39x^8 + 121x^9 + 55x^10 - 6x^11 + 128x^12 - 141x^13 + 182x^14 -
77x^15 - 163x^16 + 300x^17 + 45x^18 - 78x^19 - 15x^20 + 214x^21 + 100x^22 - 408x^23 + 14x^24 - 407x^25 + 129x^26 + 129x^27 - 341x^28
+ 450x^29 - 165x^30 - 182x^31 + 55x^32 - 53x^33 + 83x^34 - 13x^35 - 39x^36 + 42x^37 - 181x^38 + 341x^39 + 429x^40 - 245x^41 + 275x^42
- 45x^43 + 3x^44 + 116x^45 + 268x^46 - 161x^47 - 9x^48 - 200x^49 + 66x^50 + 312x^51 - 339x^52 - 386x^53 + 135x^54 + 59x^55 - 190x^56 +
176x^57 - 42x^58 - 271x^59 + 211x^60 - 276x^61 + 200x^62 - 60x^63 + 202x^64 + 194x^65 - 52x^66 + 208x^67 - 282x^68 - 25x^69 - 122x^70
+ 214x^71 + 36x^72 + 127x^73 - 55x^74 - 7x^75 - 92x^76 + 59x^77 + 18x^78 - 168x^79 + 191x^80 - 138x^81 + 62x^82 - 128x^83 - 240x^84 +
156x^85 - 135x^86 + 14x^87 + 140x^88 - 63x^89 - 102x^90 + 45x^91 + 80x^92 + 100x^93 + 87x^94 - 68x^95 + 130x^96 - 124x^97 + 5x^98 -
76x^99 - 10x^100
Kompleksitas waktu : 2560.793 mikrosekond
Total operasi tambah : 5905
Total operasi kali : 631

```

Spesifikasi komputer/laptop yang digunakan untuk mengeksekusi program ini adalah sebagai berikut : Intel Core i7-5500U 2.4GHz processor, Nvidia Geforce 940M 2GB graphics card, 8GB DDR3 memory (RAM), 1 TB HDD, 64-bit Operating System, x64 based processor.

Pada *screen-shot* input-output program pada keempat kasus di atas, algoritma *divide and conquer* memiliki waktu eksekusi yang lebih lama dibandingkan algoritma *brute force*. Hal tersebut dikarenakan pada algoritma *divide and conquer* setiap menjalankan proses pembagian menjadi upa-polinom (*divide*), array baru dialokasikan dengan isi setiap elemennya 0, lalu di-*assign* isi dari upa-polinom tersebut dari polinom awal. Sehingga, secara waktu eksekusi program algoritma ini membutuhkan waktu yang lebih lama dibandingkan algoritma *brute force* meskipun secara teori kompleksitas waktu algoritma *divide and conquer* lebih kecil daripada algoritma *brute force*.

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat menerima input dan menuliskan output	✓	
4. Luaran sudah benar untuk semua n	✓	

DAFTAR PUSTAKA

[http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Divide-and-Conquer-\(2020\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Divide-and-Conquer-(2020).pdf)