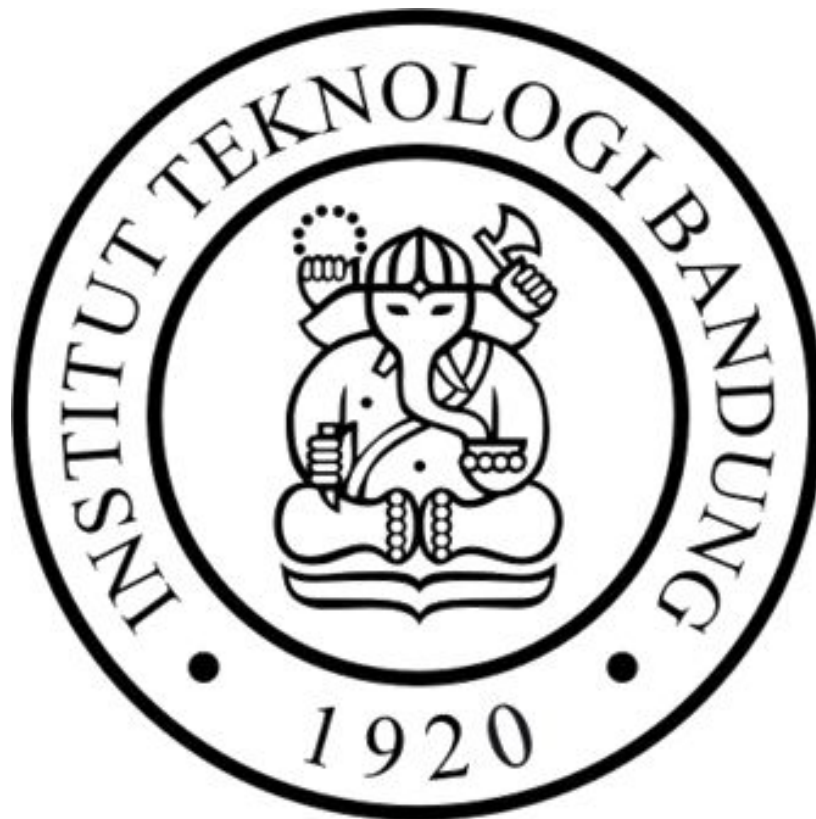


IF2211 Strategi Algoritma
LAPORAN TUGAS KECIL 2

**Membuat Pustaka untuk Perkalian Polinom
dengan
Algoritma Divide and Conquer**



Dosen : Rinaldi Munir

Oleh

Jonathan Yudi Gunawan

13518084

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2019/2020

DAFTAR ISI

DAFTAR ISI	2
BAB I	
DESKRIPSI MASALAH	3
1.1 Deskripsi Masalah	3
BAB II	
ALGORITMA PROGRAM	4
2.1 Algoritma Brute Force	4
2.2 Algoritma Divide and Conquer	5
BAB III	
KODE PROGRAM	7
3.1 Struktur Program	7
3.2 Main.java	7
3.3 Polynom.java	8
3.4 BruteforceMultiplication.java	11
3.5 DivideAndConquerMultiplication.java	11
BAB IV	
SCREENSHOT PROGRAM	13
4.1 Hasil Eksekusi Program untuk $n = 5$	13
4.2 Hasil Eksekusi Program untuk $n = 10$	13
4.3 Hasil Eksekusi Program untuk $n = 20$	13
4.4 Hasil Eksekusi Program untuk $n = 50$	14
BAB V	
KESIMPULAN, SARAN, DAN REFLEKSI	15
5.1 Kesimpulan	15
5.2 Saran	15
5.3 Refleksi	15
BAB VI	
LAIN-LAIN	16
6.1 Spesifikasi Komputer	16
6.2 Checklist Implementasi	16
DAFTAR REFERENSI	16

BAB I

DESKRIPSI MASALAH

1.1 Deskripsi Masalah

1. Buatlah dua buah pustaka (library) dalam Bahasa C/C++/Java (pilih salah satu) 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 $O(n \log 3)$.

Contoh: $n=4$

$$A(x) = 2 + 5x + 3x^2 + x^3 - x^4 \rightarrow A_0(x) = 2 + 5x; A_1(x) = 3 + x - x^2$$

$$A(x) = A_0(x) + A_1(x)x^2$$

$$B(x) = 1 + 2x + 2x^2 + 3x^3 + 6x^4 \rightarrow B_0(x) = 1 + 2x, B_1(x) = 2 + 3x + 6x^2$$

$$B(x) = B_0(x) + B_1(x)x^2$$

Dengan formula:

$$A(x)B(x) = A_0(x)B_0(x) + A_0(x)B_1(x)x^{n/2} + A_1(x)B_0(x)x^{n/2} + A_1(x)B_1(x)x^n$$

Untuk contoh di atas:

$$\begin{aligned} A(x)B(x) &= A_0(x)B_0(x) + (A_0(x)B_1(x)x^2 + A_1(x)B_0(x))x^2 + A_1(x)B_1(x)x^4 \\ &= 2 + 9x + 17x^2 + 23x^3 + 34x^4 + 39x^5 + 19x^6 + 3x^7 - 6x^8 \end{aligned}$$

2. Gunakan kedua pustaka tersebut ke dalam sebuah program sederhana yang menerima masukan dua buah polinom $A(x)$ dan $B(x)$, koefisien-koefisien polinom dibangkitkan secara acak, lalu menghitung hasil kali keduanya. Luaran program adalah hasil perkalian dua buah polinom yang ditampilkan dalam layar, jumlah operasi tambah dan kali, dan waktu (dalam microsecond) operasi perkalian tersebut. Tuliskan spek komputer yang digunakan.

BAB II

ALGORITMA PROGRAM

2.1 Algoritma *Brute Force*

Misalkan A dan B adalah polinom dengan derajat n.

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

C adalah polinom dengan derajat 2n dengan

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

dan

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

untuk semua $0 \leq k \leq 2n$

Algoritma *brute force* yang dapat dilakukan adalah dengan meng-loop semua A_i dan B_i lalu mengalikannya dan memasukkannya ke polinom C. Sehingga diperlukan loop di dalam loop. Total perkalian dan penjumlahan yang diperlukan adalah sebanyak n^2 . Sehingga kompleksitas waktunya sebesar $O(n^2)$.

Berikut pseudocode nya:

```
Polinom A,B,C
for i from 1 to n
    for j from 1 to n
        C[i+j] = A[i]*B[j]
```

2.2 Algoritma *Divide and Conquer*

Algoritma ini dibagi menjadi 3 tahap yaitu:

1. Divide

Masing-masing polinom dibagi menjadi 2 polinom berukuran setengah dari polinom semula. setengah polinom awal (derajat 0 hingga $n/2 - 1$) dan setengah polinom akhir (derajat $n/2$ hingga n), maka akan terbentuk 4 polinom.

Misalkan polinom yang ingin dikalikan A dan B, maka hasil divide akan menjadi:

$$A_0(x) = a_0 + a_1x + \dots + a_{\lfloor \frac{n}{2} \rfloor - 1}x^{\lfloor \frac{n}{2} \rfloor - 1};$$

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

dan

$$B_0(x) = b_0 + b_1x + \dots + b_{\lfloor \frac{n}{2} \rfloor - 1}x^{\lfloor \frac{n}{2} \rfloor - 1};$$

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

2. Conquer

Selesaikan masing-masing sub-permasalahan dengan rumus:

$$U(x) = \text{PolyMulti1}(A_0(x), B_0(x));$$

$$V(x) = \text{PolyMulti1}(A_0(x), B_1(x));$$

$$W(x) = \text{PolyMulti1}(A_1(x), B_0(x));$$

$$Z(x) = \text{PolyMulti1}(A_1(x), B_1(x));$$

Bila ukuran sub-permasalahan sudah cukup kecil (dalam hal ini 1), maka cukup diselesaikan dengan algoritma *brute force*. Namun metode conquer ini tidak cukup efektif karena bila dihitung dengan teorema master, kompleksitasnya tetap sama seperti algoritma *brute force*. Maka perlu dilakukan manipulasi matematika sehingga jumlah perkalian / rekursif berkurang 1 menjadi 3, yaitu dengan rumus:

$$Y(x) = \text{PolyMulti2}(A_0(x) + A_1(x), B_0(x) + B_1(x))$$

$$U(x) = \text{PolyMulti2}(A_0(x), B_0(x));$$

$$Z(x) = \text{PolyMulti2}(A_1(x), B_1(x));$$

Bila dihitung dengan teorema master maka kompleksitasnya akan turun menjadi $O(n^{\log 3})$.

3. Merge

Setelah sub-persoalan diselesaikan, maka dilakukan merge dengan rumus:

$$\left(U(x) + [Y(x) - U(x) - Z(x)]x^{\lfloor \frac{n}{2} \rfloor} + Z(x)x^{2\lfloor \frac{n}{2} \rfloor} \right);$$

hasil merge ini merupakan hasil perkalian polinom A dengan B.

BAB III

KODE PROGRAM

3.1 Struktur Program

Program terdiri dari 1 file Main.java, 1 file class Polinom, 1 file Interface Perkalian Polinom, dan 2 file Perkalian Polinom.

```
## File Structure
L Main.java
L polynommult
  L Polynom.java
  L MultBehav.java
  L BruteforceMultiplication.java
  L DivideAndConquerMultiplication.java
```

3.2 Main.java

Digunakan sebagai driver program dan untuk testing apakah program dapat berjalan dengan lancar.

```
public class Main{
    public static void main(String[] args){
        Polynom p1 = new Polynom(new BruteforceMultiplication());
        Polynom p2 = new Polynom(new DivideAndConquerMultiplication());

        p1.generateRandomPolynom(n);
        p2.generateRandomPolynom(n);

        System.out.println("Polynom 1:");
        p1.print();
        System.out.println("Polynom 2:");
        p2.print();

        // System.out.println("Add");
        // p1.subtract(p2).print();
        // System.out.println("Subtract");
        // p1.add(p2).print();

        long startTime, endTime;
```

```

        startTime = System.nanoTime();
        Polynom p3 = p1.multiply(p2);
        endTime = System.nanoTime();
        System.out.println("Multiplication using Bruteforce algorithm");
        p3.print();
        System.out.println("Time elapsed: " + (endTime-startTime)/1000 + "
mikrosekon");

        startTime = System.nanoTime();
        Polynom p4 = p2.multiply(p1);
        endTime = System.nanoTime();
        System.out.println("Multiplication using Divide and Conquer
algorithm");
        p4.print();
        System.out.println("Time elapsed: " + (endTime-startTime)/1000 + "
mikrosekon");
    }
}

```

3.3 Polynom.java

Berisikan class Polynom yang memiliki atribut MAX_ORDE, MAX_COEF, coef (TreeMap of Integer), dan multBehav (Interface Multiplikasi Polinom).

Coef menyimpan daftar koefisien dari masing-masing derajat polinom (selanjutnya akan disebut sebagai “term”), sedangkan multBehav untuk menyimpan algoritma apa yang akan digunakan untuk mengalikan polinom ini.

Class Polynom memiliki method multiply, add, subtract, promote, generateRandomPolynom, setTerm, dan print.

Multiply, add, dan subtract merupakan operasi dasar pada class polinom. Promote merupakan fungsi yang secara khusus dibuat untuk mengimplementasikan algoritma *Divide and Conquer*. Fungsinya untuk me”naik”kan derajat polinom sebesar n, atau dengan kata lain mengalikan polinom tersebut dengan polinom p yang memiliki koefisien 1 pada derajat n, dan 0 pada derajat lainnya.

Method `setTerm` secara spesifik meng-set koefisien polinom pada suatu derajat dengan suatu nilai, sedangkan method `print` untuk menampilkan isi polinom.

Berikut cuplikan implementasi dari class `Polynom`:

Attribute
<pre> public final int MAX_ORDE = 5; public final int MAX_COEF = 10; TreeMap<Integer, Integer> coef; MultBehav multBehav; </pre>
Constructor
<pre> Polynom(MultBehav multBehav) { this.coef = new TreeMap<Integer, Integer>(); this.multBehav = multBehav; } </pre>
Multiply
<pre> public Polynom multiply(Polynom other){ return this.multBehav.multiply(this, other); } </pre>
Promote
<pre> public Polynom promote(int degree){ Polynom p = new Polynom(this.multBehav); for (Map.Entry<Integer, Integer> term : this.coef.entrySet()) { p.coef.put((int)term.getKey() + degree, term.getValue()); } return p; } </pre>
Add dan Subtract
<pre> private Polynom doOperation(String op, Polynom other){ Polynom p = new Polynom(this.multBehav); // copy from this for (Map.Entry<Integer, Integer> term : this.coef.entrySet()){ int degree = term.getKey(); int coef = term.getValue(); p.coef.put(degree, coef); } } </pre>

```

// do op from other
for(Map.Entry<Integer, Integer> term : other.coef.entrySet()){
    int degree = term.getKey();
    int coef = 0;
    if(p.coef.containsKey(degree)){
        coef += p.coef.get(term.getKey());
    }
    switch(op){
        case "+":
            coef += term.getValue();
            break;
        case "-":
            coef -= term.getValue();
            break;
    }
    p.coef.put(degree, coef);
}
return p;
}

```

Generate Random Polynom

```

public void generateRandomPolynom(int n){
    Random rand = new Random();
    for(; n > 0; n--){
        int coef = rand.nextInt(MAX_COEF-1)+1;
        this.setTerm(n-1, coef);
    }
}

```

Set Term

```

private void setTerm(int degree, int coef){
    this.coef.put(degree, coef);
}

```

Print

```

public void print(){
    for (Map.Entry term : this.coef.entrySet()) {
        if((int) term.getValue() != 0){
            System.out.print(term.getValue());
            if((int) term.getKey() != 0){
                System.out.print("x"+term.getKey());
            }
        }
    }
}

```

```

        System.out.print(" ");
    }
}
System.out.println();
}

```

3.4 BruteForceMultiplication.java

Berisikan implementasi perkalian polinom dengan algoritma *brute force*.
Mengimplementasikan interface MultBehav.

```

public Polynom multiply(Polynom p1, Polynom p2){
    Polynom p3 = new Polynom(p1.multBehav);
    for (Map.Entry<Integer, Integer> term1 : p1.coef.entrySet()) {
        for (Map.Entry<Integer, Integer> term2 : p2.coef.entrySet()) {
            int degree = (int) term1.getKey() + (int) term2.getKey();
            int coef = (int) term1.getValue() * (int) term2.getValue();
            if(p3.coef.containsKey(degree)){
                coef += p3.coef.get(degree);
            }
            p3.coef.put(degree, coef);
        }
    }
    return p3;
}

```

3.5 DivideAndConquerMultiplication.java

Berisikan implementasi perkalian polinom dengan algoritma *divide and conquer*.
Mengimplementasikan interface MultBehav.

Pada class ini secara spesifik diimplementasikan method untuk divide, fungsinya untuk membagi polinom p menjadi 2 polinom p1 dan p2, yang kemudian akan di “promote” sebesar $-size/2$ sehingga derajat terkecilnya menjadi 0. Implementasi bagian conquer dan merge memanfaatkan fungsi yang ada pada class polynom.

Multiply

```

public Polynom multiply(Polynom p1, Polynom p2){

```

```

// Basis
if(p1.coef.size() == 1){
    p1.multBehav = new BruteforceMultiplication();
    return p1.multiply(p2);
}

// Divide
Polynom A0 = new Polynom(p1.multBehav);
Polynom A1 = new Polynom(p1.multBehav);
this.divide(p1, A0, A1);
A1 = A1.promote(-p1.coef.size()/2);

Polynom B0 = new Polynom(p1.multBehav);
Polynom B1 = new Polynom(p1.multBehav);
this.divide(p2, B0, B1);
B1 = B1.promote(-p2.coef.size()/2);

// Conquer
Polynom Y = this.multiply(A0.add(A1), B0.add(B1));
Polynom U = this.multiply(A0, B0);
Polynom Z = this.multiply(A1, B1).promote(p2.coef.size()/2*2);

// Merge
Polynom V = Y.subtract(U).subtract(Z).promote(A0.coef.size());
return U.add(V).add(Z);
}

```

Divide

```

public void divide(Polynom p, Polynom p1, Polynom p2){
    int size = 0;
    for(Map.Entry<Integer, Integer> term : p.coef.entrySet()){
        size++;
        if(size > p.coef.size()/2){
            p2.coef.put(term.getKey(), term.getValue());
        } else {
            p1.coef.put(term.getKey(), term.getValue());
        }
    }
}

```

BAB IV

SCREENSHOT PROGRAM

4.1 Hasil Eksekusi Program untuk $n = 5$

```
C:\Documents\khalid\Semester 1\IF2211\Strategi Algoritma>IF2211StrategiAlgoritma.exe
Masukkan nilai n: 5
Polynom 1:
5 + 5x1 + 3x2 + 4x3 + 7x4 +
Polynom 2:
7 + 3x1 + 8x2 + 1x3 + 3x4 +
Multiplication using Bruteforce algorithm
35 + 50x1 + 76x2 + 82x3 + 105x4 + 71x5 + 69x6 + 19x7 + 21x8 +
Time elapsed: 0 ms
Total operations: 25
Multiplication using Divide and Conquer algorithm
35 + 50x1 + 76x2 + 82x3 + 105x4 + 71x5 + 69x6 + 19x7 + 21x8 +
Time elapsed: 0 ms
Total operations: 75
```

4.2 Hasil Eksekusi Program untuk $n = 10$

```
Masukkan nilai n: 20
Polynom 1:
9 + 9x1 + 8x2 + 5x3 + 9x4 + 2x5 + 8x6 + 9x7 + 9x8 + 3x9 + 3x10 + 3x11 + 9x12 + 9x13 + 2x14 + 8x15 + 4x16 + 8x17 + 8x18 + 6x19 +
Polynom 2:
8 + 6x1 + 1x2 + 9x3 + 8x4 + 5x5 + 3x6 + 8x7 + 1x8 + 6x9 + 8x10 + 4x11 + 5x12 + 3x13 + 3x14 + 6x15 + 5x16 + 5x17 + 5x18 + 7x19 +
Multiplication using Bruteforce algorithm
72 + 126x1 + 127x2 + 178x3 + 263x4 + 264x5 + 266x6 + 382x7 + 354x8 + 362x9 + 407x10 + 477x11 + 461x12 + 516x13 + 450x14 + 525x15 + 601x16 + 615x17 + 662x18 + 696x19 + 610x20 + 554x21 + 609x22 + 517x23 + 447x24 + 446x25 + 388x26 + 369x27 + 32x28 + 294x29 + 269x30 + 260x31 + 223x32 + 180x33 + 192x34 + 138x35 + 126x36 + 86x37 + 42x38 +
Time elapsed: 0 ms
Total operations: 400
Multiplication using Divide and Conquer algorithm
72 + 126x1 + 127x2 + 178x3 + 263x4 + 264x5 + 266x6 + 382x7 + 354x8 + 362x9 + 407x10 + 477x11 + 461x12 + 516x13 + 450x14 + 525x15 + 601x16 + 615x17 + 662x18 + 696x19 + 610x20 + 554x21 + 609x22 + 517x23 + 447x24 + 446x25 + 388x26 + 369x27 + 32x28 + 294x29 + 269x30 + 260x31 + 223x32 + 180x33 + 192x34 + 138x35 + 126x36 + 86x37 + 42x38 +
Time elapsed: 2 ms
Total operations: 859
```

4.3 Hasil Eksekusi Program untuk $n = 20$

```
Masukkan nilai n: 10
Polynom 1:
1 + 5x1 + 3x2 + 3x3 + 6x4 + 9x5 + 1x6 + 5x7 + 5x8 + 6x9 +
Polynom 2:
3 + 7x1 + 7x2 + 7x3 + 1x4 + 7x5 + 4x6 + 3x7 + 9x8 + 4x9 +
Multiplication using Bruteforce algorithm
3 + 22x1 + 51x2 + 72x3 + 96x4 + 123x5 + 171x6 + 174x7 + 183x8 + 216x9 + 256x10 + 182x11 + 179x12 + 169x13 + 122x14 + 88x15 + 83x16 + 74x17 + 24x18 +
Time elapsed: 0 ms
Total operations: 100
Multiplication using Divide and Conquer algorithm
3 + 22x1 + 51x2 + 72x3 + 96x4 + 123x5 + 171x6 + 174x7 + 183x8 + 216x9 + 256x10 + 182x11 + 179x12 + 169x13 + 122x14 + 88x15 + 83x16 + 74x17 + 24x18 +
Time elapsed: 1 ms
Total operations: 261
```

4.4 Hasil Eksekusi Program untuk n = 50

```
Masukkan nilai n: 50
Polynom 1:
6 + 7x1 + 2x2 + 7x3 + 2x4 + 7x5 + 5x6 + 7x7 + 2x8 + 2x9 + 7x10 + 8x11 + 5x12 + 4x13 + 3x14 + 2x15 + 3x16 + 1x17 + 5x18 +
6x19 + 3x20 + 6x21 + 9x22 + 1x23 + 1x24 + 7x25 + 5x26 + 7x27 + 2x28 + 1x29 + 3x30 + 2x31 + 2x32 + 1x33 + 3x34 + 5x35 +
4x36 + 5x37 + 1x38 + 1x39 + 9x40 + 3x41 + 8x42 + 8x43 + 8x44 + 8x45 + 9x46 + 4x47 + 6x48 + 2x49 +
Polynom 2:
5 + 6x1 + 6x2 + 3x3 + 2x4 + 6x5 + 1x6 + 3x7 + 6x8 + 7x9 + 4x10 + 4x11 + 5x12 + 1x13 + 4x14 + 8x15 + 8x16 + 5x17 + 9x18 +
8x19 + 6x20 + 6x21 + 3x22 + 1x23 + 3x24 + 7x25 + 6x26 + 9x27 + 6x28 + 3x29 + 1x30 + 6x31 + 7x32 + 9x33 + 2x34 + 3x35 +
2x36 + 4x37 + 5x38 + 4x39 + 6x40 + 5x41 + 5x42 + 7x43 + 4x44 + 6x45 + 8x46 + 7x47 + 9x48 + 5x49 +

Multiplication using Bruteforce algorithm
30 + 71x1 + 88x2 + 107x3 + 97x4 + 145x5 + 152x6 + 164x7 + 208x8 + 202x9 + 240x10 + 265x11 + 320x12 + 290x13 + 298x14 + 3
59x15 + 366x16 + 352x17 + 407x18 + 490x19 + 493x20 + 552x21 + 562x22 + 537x23 + 507x24 + 554x25 + 627x26 + 696x27 + 684x
28 + 653x29 + 694x30 + 674x31 + 698x32 + 687x33 + 719x34 + 686x35 + 767x36 + 821x37 + 839x38 + 786x39 + 827x40 + 810x41
+ 870x42 + 963x43 + 972x44 + 1031x45 + 1069x46 + 1066x47 + 1077x48 + 1093x49 + 1049x50 + 1075x51 + 1085x52 + 1089x53 + 1
063x54 + 999x55 + 932x56 + 913x57 + 988x58 + 963x59 + 935x60 + 901x61 + 896x62 + 851x63 + 841x64 + 801x65 + 746x66 + 807
x67 + 769x68 + 703x69 + 682x70 + 630x71 + 605x72 + 673x73 + 596x74 + 581x75 + 498x76 + 475x77 + 454x78 + 443x79 + 436x80
+ 422x81 + 412x82 + 442x83 + 406x84 + 414x85 + 407x86 + 380x87 + 416x88 + 397x89 + 359x90 + 354x91 + 302x92 + 251x93 +
209x94 + 139x95 + 88x96 + 48x97 + 10x98 +
Time elapsed: 7 ms
Total operations: 2500

Multiplication using Divide and Conquer algorithm
30 + 71x1 + 88x2 + 107x3 + 97x4 + 145x5 + 152x6 + 164x7 + 208x8 + 202x9 + 240x10 + 265x11 + 320x12 + 290x13 + 298x14 + 3
59x15 + 366x16 + 352x17 + 407x18 + 490x19 + 493x20 + 552x21 + 562x22 + 537x23 + 507x24 + 554x25 + 627x26 + 696x27 + 684x
28 + 653x29 + 694x30 + 674x31 + 698x32 + 687x33 + 719x34 + 686x35 + 767x36 + 821x37 + 839x38 + 786x39 + 827x40 + 810x41
+ 870x42 + 963x43 + 972x44 + 1031x45 + 1069x46 + 1066x47 + 1077x48 + 1093x49 + 1049x50 + 1075x51 + 1085x52 + 1089x53 + 1
063x54 + 999x55 + 932x56 + 913x57 + 988x58 + 963x59 + 935x60 + 901x61 + 896x62 + 851x63 + 841x64 + 801x65 + 746x66 + 807
x67 + 769x68 + 703x69 + 682x70 + 630x71 + 605x72 + 673x73 + 596x74 + 581x75 + 498x76 + 475x77 + 454x78 + 443x79 + 436x80
+ 422x81 + 412x82 + 442x83 + 406x84 + 414x85 + 407x86 + 380x87 + 416x88 + 397x89 + 359x90 + 354x91 + 302x92 + 251x93 +
209x94 + 139x95 + 88x96 + 48x97 + 10x98 +
Time elapsed: 8 ms
Total operations: 3751
```

Pada tangkapan layar terlihat bahwa algoritma divide and conquer justru lebih lambat ketika dijalankan. Hal ini terjadi karena ketika proses rekursif, dibentuk banyak objek baru dari class Polynom lalu pada masing-masing operasi (seperti add dan subtract) juga dibentuk objek baru untuk menampung hasil operasi sementara, sehingga total pada satu blok fungsi multiply tersebut terdapat 21 pembentukan (dan penghancuran) objek baru, kontras dengan algoritma brute force yang hanya membentuk 1 objek baru. Namun bagaimanapun, secara asimtotik tetap algoritma divide and conquer lebih cepat. Hal ini dapat dibuktikan dengan memasukkan nilai n yang cukup besar, seperti 5000.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

Dalam bab ini akan disajikan kesimpulan dari tugas besar yang telah dikerjakan oleh penulis. Setelah itu, penulis akan memberikan saran dan refleksi bagi para pembaca laporan.

5.1 Kesimpulan

Pada tugas besar kali ini, saya berhasil membuat Pustaka untuk Perkalian Polinom. Fitur-fitur dalam program saya antara lain, mengalikan dua buah polinom dengan menggunakan dua algoritma berbeda..

5.2 Saran

Penulis menyarankan untuk ke depannya program dapat dikembangkan dengan cara menggunakan algoritma FFT yang memiliki waktu asimtotik yang lebih cepat.

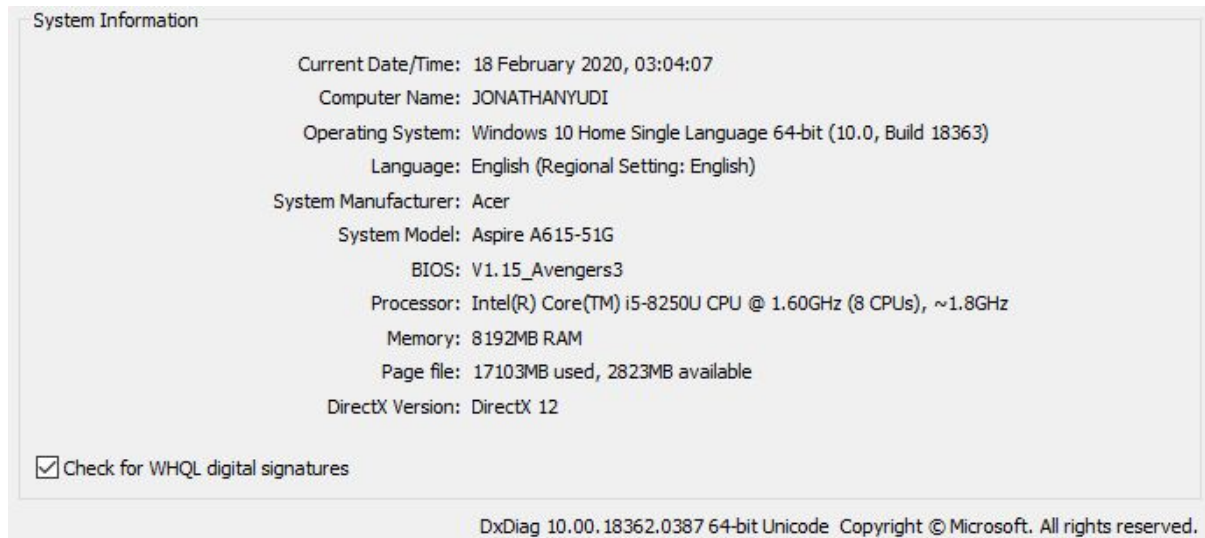
5.3 Refleksi

Setelah melakukan tugas besar ini, penulis merefleksikan bahwa tucil ini cukup menarik karena perkalian polinom ternyata dapat dilakukan dengan cukup cepat melalui teknik manipulasi matematik.

BAB VI

LAIN-LAIN

6.1 Spesifikasi Komputer



6.2 Checklist Implementasi

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi	v	
2	Program berhasil running	v	
3	Program dapat menerima input dan menuliskan output	v	
4	Luaran sudah benar untuk semua n	v	

DAFTAR REFERENSI

[http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Tugas-Kecil-2-\(2020\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Tugas-Kecil-2-(2020).pdf)

[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)