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

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**

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

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

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

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

Dengan formula:

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

Untuk contoh di atas:

$A(x)B(x) = A_0(x)B_0(x) + (A_0(x)B_1(x)\ x^2 + A_1(x)B_0(x))x^2 + A_1(x)B_1(x)x^4$

$= 2 + 9x + 17x^2 + 23x^3 + 34x^4 + 39x^5 + 19x^6 + 3x^7 - 6x^8$

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.

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

C adalah polinom dengan derajat 2n dengan

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

dan

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

untuk semua 0 <= k <= 2n

Algoritma *brute force* yang dapat dilakukan adalah dengan meng-loop semua Ai dan Bi 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:

$$A_0(x) = a_0 + a_1 x + \cdots + a_{\lfloor \frac{n}{2} \rfloor - 1} x^{\lfloor \frac{n}{2} \rfloor - 1};$$
$$A_1(x) = a_{\lfloor \frac{n}{2} \rfloor} + a_{\lfloor \frac{n}{2} \rfloor + 1} x + \cdots + a_n x^{n - \lfloor \frac{n}{2} \rfloor};$$

   dan

$$B_0(x) = b_0 + b_1 x + \cdots + b_{\lfloor \frac{n}{2} \rfloor - 1} x^{\lfloor \frac{n}{2} \rfloor - 1};$$
$$B_1(x) = b_{\lfloor \frac{n}{2} \rfloor} + b_{\lfloor \frac{n}{2} \rfloor + 1} x + \cdots + b_n x^{n - \lfloor \frac{n}{2} \rfloor};$$

2. **Conquer**

   Selesaikan masing-masing sub-permasalahan dengan rumus:

$$U(x) = PolyMulti1(A_0(x), B_0(x));$$
$$V(x) = PolyMulti1(A_0(x), B_1(x));$$
$$W(x) = PolyMulti1(A_1(x), B_0(x));$$
$$Z(x) = PolyMulti1(A_1(x), B_1(x));$$

   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:

$$Y(x) = PolyMulti2(A_0(x) + A_1(x), B_0(x) + B_1(x))$$
$$U(x) = PolyMulti2(A_0(x), B_0(x));$$
$$Z(x) = PolyMulti2(A_1(x), B_1(x));$$

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

```
public final int MAX_ORDE = 5;
public final int MAX_COEF = 10;
TreeMap<Integer, Integer> coef;
MultBehav multBehav;
```

Constructor

```
Polynom(MultBehav multBehav) {
   this.coef = new TreeMap<Integer, Integer>();
   this.multBehav = multBehav;
}
```

Multiply

```
public Polynom multiply(Polynom other){
   return this.multBehav.multiply(this, other);
}
```

Promote

```
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;
}
```

Add dan Subtract

```
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);
   }
```

```
    // 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);
}
```

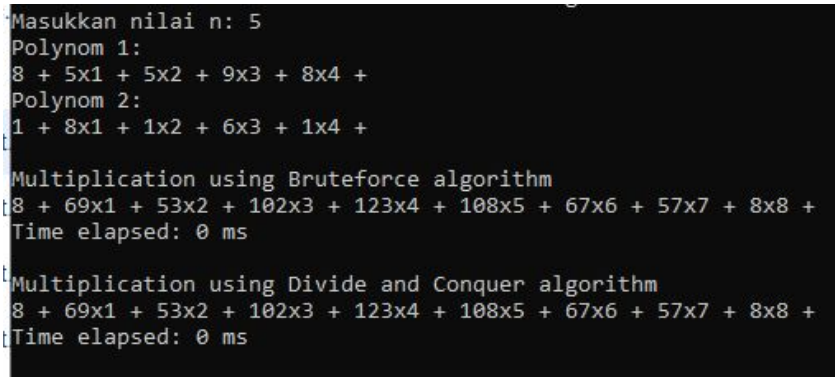
Divide

```
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());
        }
    }
}
```

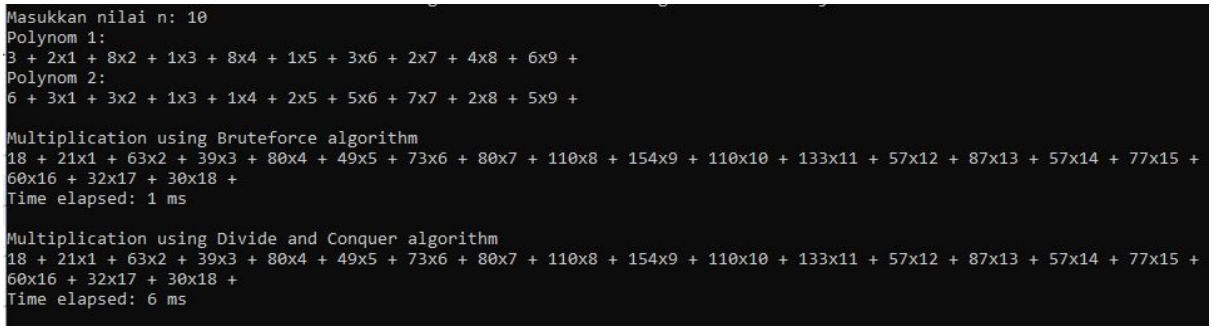
BAB IV

SCREENSHOT PROGRAM

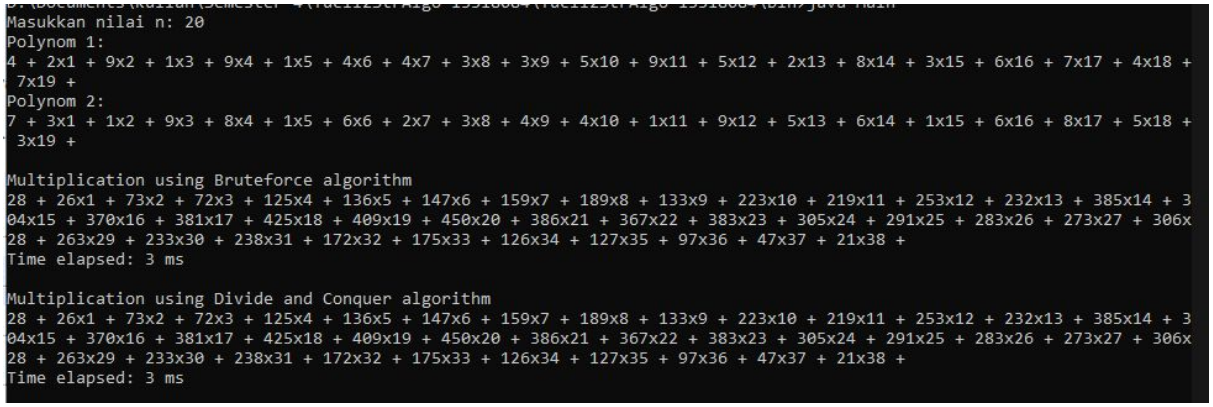
4.1 Hasil Eksekusi Program untuk n = 5



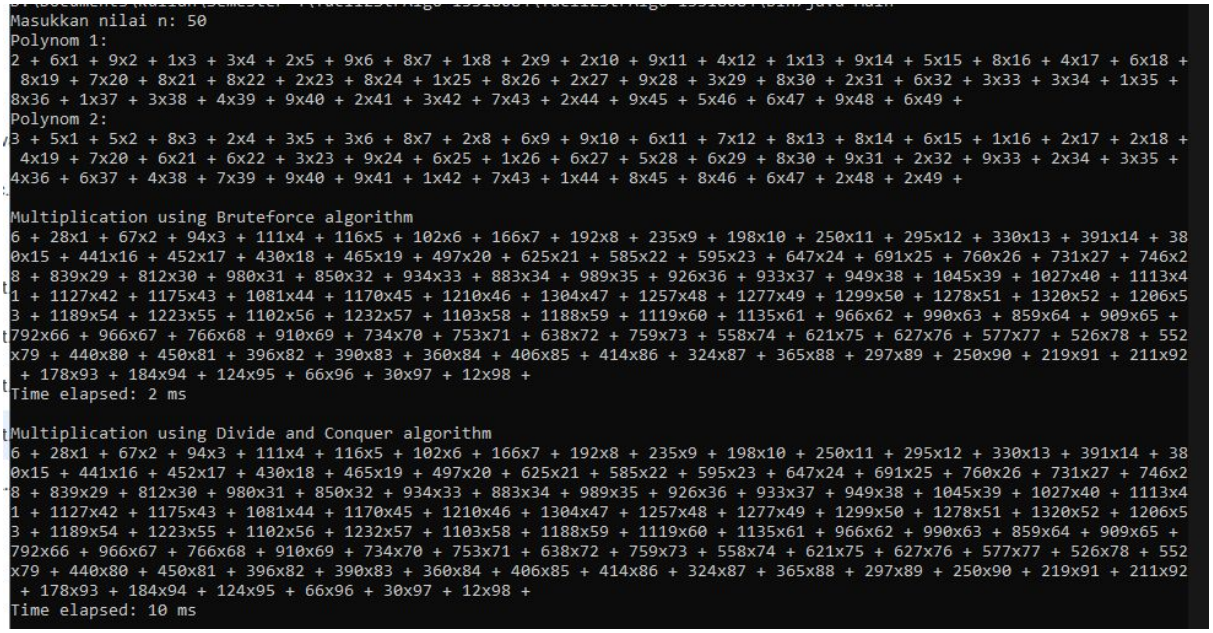
4.2 Hasil Eksekusi Program untuk n = 10



4.3 Hasil Eksekusi Program untuk n = 20



4.4 Hasil Eksekusi Program untuk n = 50



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

Multiplication using Bruteforce algorithm
6 + 28x1 + 67x2 + 94x3 + 111x4 + 116x5 + 102x6 + 166x7 + 192x8 + 235x9 + 198x10 + 250x11 + 295x12 + 330x13 + 391x14 + 38
0x15 + 441x16 + 452x17 + 430x18 + 465x19 + 497x20 + 625x21 + 585x22 + 595x23 + 647x24 + 691x25 + 760x26 + 731x27 + 746x2
8 + 839x29 + 812x30 + 980x31 + 850x32 + 934x33 + 883x34 + 989x35 + 926x36 + 933x37 + 949x38 + 1045x39 + 1027x40 + 1113x4
1 + 1127x42 + 1175x43 + 1081x44 + 1170x45 + 1210x46 + 1304x47 + 1257x48 + 1277x49 + 1299x50 + 1278x51 + 1320x52 + 1206x5
3 + 1189x54 + 1223x55 + 1102x56 + 1232x57 + 1103x58 + 1188x59 + 1119x60 + 1135x61 + 966x62 + 990x63 + 859x64 + 909x65 +
792x66 + 966x67 + 766x68 + 910x69 + 734x70 + 753x71 + 638x72 + 759x73 + 558x74 + 621x75 + 627x76 + 577x77 + 526x78 + 552
x79 + 440x80 + 450x81 + 396x82 + 390x83 + 360x84 + 406x85 + 414x86 + 324x87 + 365x88 + 297x89 + 250x90 + 219x91 + 211x92
 + 178x93 + 184x94 + 124x95 + 66x96 + 30x97 + 12x98 +
Time elapsed: 2 ms

Multiplication using Divide and Conquer algorithm
6 + 28x1 + 67x2 + 94x3 + 111x4 + 116x5 + 102x6 + 166x7 + 192x8 + 235x9 + 198x10 + 250x11 + 295x12 + 330x13 + 391x14 + 38
0x15 + 441x16 + 452x17 + 430x18 + 465x19 + 497x20 + 625x21 + 585x22 + 595x23 + 647x24 + 691x25 + 760x26 + 731x27 + 746x2
8 + 839x29 + 812x30 + 980x31 + 850x32 + 934x33 + 883x34 + 989x35 + 926x36 + 933x37 + 949x38 + 1045x39 + 1027x40 + 1113x4
1 + 1127x42 + 1175x43 + 1081x44 + 1170x45 + 1210x46 + 1304x47 + 1257x48 + 1277x49 + 1299x50 + 1278x51 + 1320x52 + 1206x5
3 + 1189x54 + 1223x55 + 1102x56 + 1232x57 + 1103x58 + 1188x59 + 1119x60 + 1135x61 + 966x62 + 990x63 + 859x64 + 909x65 +
792x66 + 966x67 + 766x68 + 910x69 + 734x70 + 753x71 + 638x72 + 759x73 + 558x74 + 621x75 + 627x76 + 577x77 + 526x78 + 552
x79 + 440x80 + 450x81 + 396x82 + 390x83 + 360x84 + 406x85 + 414x86 + 324x87 + 365x88 + 297x89 + 250x90 + 219x91 + 211x92
 + 178x93 + 184x94 + 124x95 + 66x96 + 30x97 + 12x98 +
Time elapsed: 10 ms
```

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.

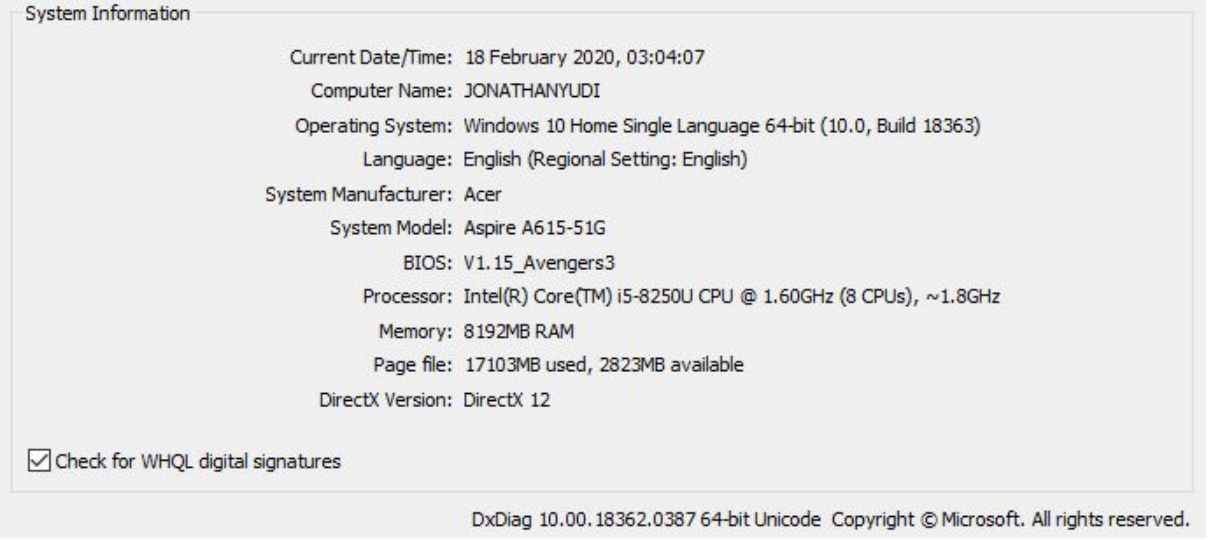
5.3 Refleksi

Setelah melakukan tugas besar ini, penulis merefleksikan bahwa tucil ini cukup menarik karena perkalian polinom ternyata dapat dilakukan dengan cukup cepat melalui teknik 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/Algoritma-Divide-and-Conquer-(2020).pdf