TUGAS KECIL II
IF2211 STRATEGI ALGORITMA
MEMBUAT PROGRAM PERKALIAN POLINOM DENGAN ALGORITMA *DIVIDE AND CONQUER*

LAPORAN TUGAS KECIL

Diajukan untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma

Oleh

MUHAMMAD FAUZAN AL-GHIFARI

(13518112 - K01)

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2020

A. Algoritma *Brute Force* dan Algoritma *Divide and Conquer*

Pada tugas kecil 1 ini saya menyelesaikan permasalahan perkalian polinom menggunakan algoritma *brufe force* dan *divide and conquer* dalam bahasa C++. Berikut adalah algoritma *brute force* dan *divide and conquer* berserta kompleksitasnya.

Algoritma Brute Force

```
#include "BF.hpp"
#include <iostream>
#include <vector>
using namespace std;

vector<int> BF(vector<int> pol1, vector<int> pol2, int *cKali, int *cTambah)
{
// Output berupa array polinom hasil perkalian pol1 dan pol2 dengan metode B
rute Force
    int i,j;
    *cKali = 0;
    *cTambah = 0;
    vector<int> pol3(pol1.size()+pol2.size());

    for (i = 0; i < pol1.size(); i++){
        for (j = 0; j< pol2.size(); j++){
            pol3[i+j] += pol1[i] * pol2[j];
            *cKali += 1;        // Menghitung jumlah operasi kali
            *cTambah += 1;      // Menghitung jumlah operasi tambah
        }
    }
    return pol3;
}
```

Algoritma Brute Force merupakan jenis algoritma yang bersifat straight forward dan memiliki kompleksitas yang tinggi. Pemecahan masalah dengan brute forcedilakukan berdasarkan pada pernyataan soaldan defenisi konsep. Algoritma di atas menggunakan dua buah *for loop* sehingga algoritma di atas memiliki kompleksitas $O(n^2)$

Algoritma Divide and Conquer

```
#include "DNC.hpp"
#include <iostream>
#include <vector>
using namespace std;
```

```
// Output berupa array polinom hasil perkalian pol1 dan pol2 dengan metode D
ivide and Conquer
vector<int> DNC(vector<int> pol1, vector<int> pol2, int *cKali, int *cTambah
){
    // Basis jika polinom berderajat 0
    if(pol1.size() == 1 or pol2.size() == 1){
        vector<int> pol3(1);
        pol3[0] = pol1[0] * pol2[0];
        *cKali += 1;        // Menghitung jumlah operasi kali
        return pol3;
    }
    // Rekurens
    else{
        int i;
        int n = pol1.size()/2;
        vector<int> A0, B0, A1, B1;
        Divide(pol1, pol2, &A0, &B0, &A1, &B1, n);  // Membagi polinom menja
di dua bagian

        vector<int> Y = DNC(Sum(A0, A1), Sum(B0, B1), &*cKali, &*cTambah);
        vector<int> U = DNC(A0, B0, &*cKali, &*cTambah);
        vector<int> Z = DNC(A1, B1, &*cKali, &*cTambah);

        *cTambah += 6*A1.size();    // Mengitung jumlah operasi tambah
        return Sum(Sum(Multi(Min(Min(Y, U), Z), n), U), Multi(Z, n*2));
    }
}
```

Dalam menyelesaikan persoalan perkalian polinom, kita dapat melakukan optimisasi menggunakan algoritma *Divide and Conquer* seperti yang telah penulis lakukan di atas. Ide dasarnya adalah dengan berusaha mengurangi jumlah operasi perkalian yang dilakukan oleh program. Konsekuensi dari mengurangi jumlah operasi perkalian adalah jumlah operasi penjumlahan yang akan bertambah. Hal ini memang merupakan tujuan yang diharapkan karena operasi perkalian jauh lebih berat membebani performa program digandingkan dengan operasi penjumlahan.

Berikut ini adalah langkah-langkah yang dilakukan dalam menyelesaikan operasi perkalian polinom dengan algoritma *Divide and Conquer*.

1. Algoritma ini menggunakan rekursif dimana terdapat basis dan rekurens
2. Basis pada algoritma ini adalah polinom berderajat 0
3. Rekurens terjadi saat polinom berderajat lebih tinggi dari 0

4. Pada rekurens terjadi pemisahan polinom menjadi dua bagian dengan panjang polinom yang sama
5. Jika panjang polinom ganjil, maka polinom yang kedua akan memiliki satu derajat lebih tinggi. Misalnya 2 + 3x + 5x^2 + 6x^3 + 7x^4 akan di divide menjadi
   a. Polinom 1 : 2 + 3x
   b. Polinom 2 : 5 + 6x + 7x^2
6. Proses divide ini akan terus dilakukan sampai polinom mecapai basis
7. Saat sudah mencapai basis maka polinom berderajat 0 itu akan dikalikan dan di combine (pada bagian return dari fungsi Divide and Conquer)
8. Pada saat combine ini lah jumlah operasi perkalian lebih sedikit dibanding dengan algoritma *Brute Force* sehingga memiliki kompleksitas yang lebih kecil
9. Intinya dari Algoritma Divide and Conquernya adalah
   a. Divide : Terjadi saat pembagian polinom menjadi 2 buah bagian
   b. Conquer : Terjadi saat polinom mencapai basis
   c. Combiner : Terjadi saat return dari fungsi DNC

Menghitung kompleksitas Algoritma Divide and Conquer

Asumsikan $n = 2^k$. $\lg x$ menandakan $\log_2 x$.

Dengan metode substitusi

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) = 3T\left[3T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right] + cn$$

$$T(n) = 3^2T\left(\frac{n}{2^2}\right) + \left(1\frac{3}{2}\right)cn$$

$$T(n) = 3^2\left[3T\left(\frac{n}{2^3}\right) + c\frac{n}{2^2}\right] + \left(1 + \frac{3}{2}\right)cn$$

$$T(n) = 3^2T\left(\frac{n}{2^3}\right) + \left(1 + \frac{3}{2} + \left[\frac{3}{2}\right]^2\right)cn$$

....

$$T(n) = 3^hT\left(\frac{n}{h}\right) + \sum_{j=0}^{h-1}\left[\frac{3}{2}\right]^j cn$$

Jadi kita memiliki

$3^h = (2^{lg3})^h = 2^{h \lg 3} = n^{lg3} \approx n^{1.582}$

Dan

$$\sum_{j=0}^{h-1} \left[\frac{3}{2}\right]^j cn = \frac{\left(\frac{3}{2}\right)^h - 1}{\frac{3}{2} - 1} = 2\frac{3^h}{2^h} - 2 = 2n^{lg3-1} - 2$$

Sehingga Algoritma Divide and Conquer memiliki kompleksitas sbesar

$T(n) = O(n^{lg3}T(1) + 2cn^{lg3}) = \boldsymbol{O(n^{lg3})}$

B. Kode Program

Brute Force

Berisi Algoritma Brute force untuk menyelesaikan perkalian polinom

BF.hpp

```
#ifndef ___BF___HPP
#define ___BF___HPP
#include <vector>
using namespace std;

vector<int> BF(vector<int> pol1, vector<int> pol2, int *cKali, int *cTambah)
;
// Output berupa array polinom hasil perkalian pol1 dan pol2 dengan metode B
rute Force
#endif
```

BF.cpp

```
#include "BF.hpp"
#include <iostream>
#include <vector>
using namespace std;

vector<int> BF(vector<int> pol1, vector<int> pol2, int *cKali, int *cTambah)
{
// Output berupa array polinom hasil perkalian pol1 dan pol2 dengan metode B
rute Force
    int i,j;
    *cKali = 0;
    *cTambah = 0;
    vector<int> pol3(pol1.size()+pol2.size());
```

```
    for (i = 0; i < pol1.size(); i++){
        for (j = 0; j< pol2.size(); j++){
            pol3[i+j] += pol1[i] * pol2[j];
            *cKali += 1;        // Menghitung jumlah operasi kali
            *cTambah += 1;      // Menghitung jumlah operasi tambah
        }
    }
    return pol3;
}
```

Divide and Concquer

Berisi Algoritma Divide and Conquer untuk menyelesaikan perkalian polinom

DNC.hpp

```
#ifndef ___DNC___HPP
#define ___DNC___HPP

#include <vector>
#include <iostream>
#include "other.hpp"
using namespace std;

vector<int> DNC(vector<int> pol1, vector<int> pol2, int *cKali, int *cTambah
);
// Output berupa array polinom hasil perkalian pol1 dan pol2 dengan metode D
ivide and Conquer


#endif
```

DNC.cpp

```
#include "DNC.hpp"
#include <iostream>
#include <vector>
using namespace std;

// Output berupa array polinom hasil perkalian pol1 dan pol2 dengan metode D
ivide and Conquer
vector<int> DNC(vector<int> pol1, vector<int> pol2, int *cKali, int *cTambah
){
    // Basis jika polinom berderajat 0
    if(pol1.size() == 1 or pol2.size() == 1){
        vector<int> pol3(1);
        pol3[0] = pol1[0] * pol2[0];
        *cKali += 1;        // Menghitung jumlah operasi kali
```

```
        return pol3;
    }
    // Rekurens
    else{
        int i;
        int n = pol1.size()/2;
        vector<int> A0, B0, A1, B1;
        Divide(pol1, pol2, &A0, &B0, &A1, &B1, n);  // Membagi polinom menja
di dua bagian

        vector<int> Y = DNC(Sum(A0, A1), Sum(B0, B1), &*cKali, &*cTambah);
        vector<int> U = DNC(A0, B0, &*cKali, &*cTambah);
        vector<int> Z = DNC(A1, B1, &*cKali, &*cTambah);

        *cTambah += 6*A1.size();    // Mengitung jumlah operasi tambah
        return Sum(Sum(Multi(Min(Min(Y, U), Z), n), U), Multi(Z, n*2));
    }
}
```

Other

Berisi fungsi-fungsi tambahan yang digunakan dalam algoritma *Divide and Conquer*. Fungsi pada other ini dibutuhkan karena implementasi polinom pada pogram kali ini berupa array, sehingga untuk operasi seperti penambahan atau perkalian dibutuhkan fungsi khusus untuk menanganinya

Other.hpp

```
#ifndef ___OTHER___HPP
#define ___OTHER___HPP
#include <vector>
using namespace std;

void CetakPol (vector<int> pol);
//Prosedur untuk mencetak polinom

int max(int x, int y);
//Fungsi untuk menghasilkan integer maksimum

vector<int> Sum (vector<int> pol1, vector<int> pol2);
//Fungsi untuk menjumlahkan polinom

vector<int> Min (vector<int> pol1, vector<int> pol2);
//Fungsi untuk mengurangkan polinom

vector<int> Multi (vector<int> pol1, int n);
```

```
//Fungsi untuk menaikkan derajat polinom sebanyak n

void Divide (vector<int> pol1, vector<int> pol2, vector<int> *A0, vector<int
> *B0, vector<int> *A1, vector<int> *B1, int n);
//Membagi polinom menjadi dua bagian yang sama besar, jika ganjil dilebihkan
 pada polinom yang kedua

#endif
```

other.cpp

```
#include "other.hpp"
#include <iostream>
#include <vector>
using namespace std;

void CetakPol (vector<int> pol){
//Prosedur untuk mencetak polinom
    int i;
    for (i = 0; i < pol.size(); i++){
        if (pol[i] != 0){
            if (i == 0){
                cout << pol[i];
            }
            else if (i == 1){
                if (pol[i] < 0) cout << " - " << (-1*pol[i]) << "x";
                else if (pol[i] > 0) cout << " + " << pol[i] << "x";
            }
            else{
                if (pol[i] < 0) cout << " - " << (-1*pol[i]) << "x^" << i;
                else if (pol[i] > 0) cout << " + " << pol[i] << "x^" << i;
            }
        }
    }
    cout << endl;
}

int max(int x, int y){
//Fungsi untuk menghasilkan integer maksimum
    return (x>y)? x: y;
}

vector<int> Sum (vector<int> pol1, vector<int> pol2){
//Fungsi untuk menjumlahkan polinom
    int size = max(pol1.size(), pol2.size());
    int i;
    vector<int> sum(size);
    for (int i = 0; i < pol1.size(); i++){
```

```
        sum[i] = pol1[i];
    }
    for (int i = 0; i < pol2.size(); i++){
        sum[i] += pol2[i];
    }
    return sum;
}

vector<int> Min (vector<int> pol1, vector<int> pol2){
//Fungsi untuk mengurangkan polinom
    int size = max(pol1.size(), pol2.size());
    int i;
    vector<int> sum(size);
    for (int i = 0; i < pol1.size(); i++){
        sum[i] = pol1[i];
    }
    for (int i = 0; i < pol2.size(); i++){
        sum[i] -= pol2[i];
    }
    return sum;
}

vector<int> Multi (vector<int> pol1, int n){
//Fungsi untuk menaikkan derajat polinom sebanyak n
    int i;
    vector<int> multi(pol1.size()+n);
    for (i = 0; i < pol1.size(); i ++){
        multi[i+n] = pol1[i];
    }
    return multi;
}


void Divide (vector<int> pol1, vector<int> pol2, vector<int> *A0, vector<int
> *B0, vector<int> *A1, vector<int> *B1, int n){
//Membagi polinom menjadi dua bagian yang sama besar, jika ganjil dilebihkan
 pada polinom yang kedua
    int i;

    for(i = 0; i <= n-1; i++){
        (*A0).push_back(pol1[i]);
        (*B0).push_back(pol2[i]);
    }
    for(i = n; i < pol1.size(); i++){
        (*A1).push_back(pol1[i]);
        (*B1).push_back(pol2[i]);
    }
}
```

Main

Merupakan program utama yang menyatukan semua file di atas. Program akan meminta input berupa n yang merupakan panjang suku polinom

main.cpp

```
#include <iostream>
#include "DNC.hpp"
#include "BF.hpp"

#include <iostream>
#include <vector>
#include <stdio.h>
#include <chrono>
#include <cstdlib>
#include <ctime>

using namespace std;
using namespace std::chrono;

int main(){
    int n, i,a ,b;
    int cKali, cTambah;
    vector<int> pol1;
    vector<int> pol2;
    vector<int> pol3;
    vector<int> pol4;

    // Meminta input panjang suku polinom
    cout << "Masukkan panjang suku polinom : ";
    cin >> n;
    while (n < 0){
        cout << "Minimal n bernilai 0" << endl;
        cout << "Masukkan panjang suku polinom : ";
        cin >> n;
    }

    // Membangun angka random
    srand(time(0));
    for (i = 0; i <= n; i++){
        a = (rand() % 200) - 100;
        b = (rand() % 200) - 100;
        pol1.push_back(a);
        pol2.push_back(b);
    }

    cout << "polinom 1  : "; CetakPol(pol1);
    cout << "polinom 2  : "; CetakPol(pol2);
```

```
    cout << endl;

/ Memulai algoritma brute force
    cout << "[ALGORITMA BRUTE FORCE]" << endl;
    // Menghitung durasi algoritma brute force
    auto startBF = high_resolution_clock::now();
    pol3 = BF(pol1, pol2, &cKali, &cTambah);
    auto stopBF = high_resolution_clock::now();
    auto durationBF = duration_cast<microseconds>(stopBF - startBF).count();


    // Mencetak hasil algoritma brute force
    cout << "hasil perklian polinom : "; CetakPol(pol3);
    cout << "jumlah operasi kali    : " << cKali << endl;
    cout << "jumlah operasi tambah  : " << cTambah << endl;
    cout << "waktu yang dibutuhkan  : " << durationBF << " microseconds" <<
endl;
    cout << endl;


    // Memulai algoritma divide and conquer
    cout << "[ALGORITMA DIVIDE AND CONQUER]" << endl;
    cKali = 0;
    cTambah = 0;
    // Menghitung durasi algoritma divide and conquer
    auto startDNC = high_resolution_clock::now();
    pol4 = DNC(pol1, pol2, &cKali, &cTambah);
    auto stopDNC = high_resolution_clock::now();
    auto durationDNC = duration_cast<microseconds>(stopDNC - startDNC).count
();

    // Mencetak hasil algoritma divide and conquer
    cout << "hasil perklian polinom : "; CetakPol(pol4);
    cout << "jumlah operasi kali    : " << cKali << endl;
    cout << "jumlah operasi tambah  : " << cTambah << endl;
    cout << "waktu yang dibutuhkan  : " << durationDNC << " microseconds" <<
 endl;
    cout << endl;


    return 0;
}
```

C. Screenshot Input-Output Program

Untuk n = 5

```
Masukkan panjang suku polinom : 5
polinom 1  : 22 - 92x - 49x^2 - 64x^3 + 17x^4 - 28x^5
polinom 2  : 7 + 30x - 77x^2 + 58x^3 + 36x^4 - 81x^5

[ALGORITMA BRUTE FORCE]
hasil perklian polinom  : 154 + 16x - 4797x^2 + 6442x^3 - 2572x^4 - 2694x
^5 - 173x^6 + 4807x^7 + 4172x^8 - 2385x^9 + 2268x^10
jumlah operasi kali     : 36
jumlah operasi tambah   : 36
waktu yang dibutuhkan   : 27 microseconds

[ALGORITMA DIVIDE AND CONQUER]
hasil perklian polinom  : 154 + 16x - 4797x^2 + 6442x^3 - 2572x^4 - 2694x
^5 - 173x^6 + 4807x^7 + 4172x^8 - 2385x^9 + 2268x^10
jumlah operasi kali     : 21
jumlah operasi tambah   : 90
waktu yang dibutuhkan   : 735 microseconds
```

Untuk n = 10

```
Masukkan panjang suku polinom : 10
polinom 1  : 62 + 70x + 73x^2 - 57x^3 + 10x^4 - 10x^5 + 64x^6 + 18x^7 - 3
6x^8 + 75x^9 - 16x^10
polinom 2  :  - 100x - 90x^2 - 37x^3 - 58x^4 + 52x^5 - 16x^6 + 66x^7 + 56
x^8 - 64x^9 + 39x^10

[ALGORITMA BRUTE FORCE]
hasil perklian polinom  :  - 6200x - 12580x^2 - 15894x^3 - 7056x^4 + 593x
^5 + 623x^6 + 4204x^7 - 3810x^8 + 6394x^9 - 11054x^10 - 5848x^11 + 7060x^
12 - 5117x^13 + 11206x^14 - 7886x^15 + 4534x^16 + 6150x^17 - 7100x^18 + 3
949x^19 - 624x^20
jumlah operasi kali     : 121
jumlah operasi tambah   : 121
waktu yang dibutuhkan   : 44 microseconds

[ALGORITMA DIVIDE AND CONQUER]
hasil perklian polinom  :  - 6200x - 12580x^2 - 15894x^3 - 7056x^4 + 593x
^5 + 623x^6 + 4204x^7 - 3810x^8 + 6394x^9 - 11054x^10 - 5848x^11 + 7060x^
12 - 5117x^13 + 11206x^14 - 7886x^15 + 4534x^16 + 6150x^17 - 7100x^18 + 3
949x^19 - 624x^20
jumlah operasi kali     : 59
jumlah operasi tambah   : 288
waktu yang dibutuhkan   : 2247 microseconds
```

Untuk n = 20

```
Masukkan panjang suku polinom : 20
polinom 1  : 78 + 83x - 12x^2 + 74x^3 + 1x^4 - 68x^5 + 56x^6 - 11x^7 - 7x
^8 + 49x^9 + 6x^10 - 27x^11 - 29x^12 - 4x^13 - 57x^14 + 82x^15 + 53x^16 -
 82x^17 - 96x^18 + 88x^19 + 52x^20
polinom 2  : 78 + 97x - 14x^2 + 94x^3 - 93x^4 + 86x^5 - 93x^6 + 64x^7 + 1
1x^8 - 69x^9 + 91x^10 + 43x^11 - 70x^12 + 34x^13 + 75x^14 - 79x^15 + 65x^
16 - 59x^17 + 44x^18 + 5x^19 - 80x^20

[ALGORITMA BRUTE FORCE]
hasil perklian polinom  : 6084 + 14040x + 6023x^2 + 10778x^3 + 7972x^4 -
8382x^5 + 5714x^6 - 5021x^7 + 4768x^8 + 2852x^9 - 889x^10 + 21908x^11 - 1
8292x^12 - 492x^13 + 14730x^14 - 19436x^15 + 18376x^16 + 7526x^17 - 21223
x^18 + 172x^19 + 16627x^20 - 21384x^21 + 16204x^22 - 3914x^23 - 4093x^24
- 3775x^25 + 10215x^26 + 942x^27 - 21866x^28 + 14860x^29 + 4585x^30 - 699
3x^31 - 2971x^32 + 11138x^33 + 2848x^34 - 2627x^35 - 10686x^36 + 6884x^37
 + 10408x^38 - 6780x^39 - 4160x^40
jumlah operasi kali     : 441
jumlah operasi tambah   : 441
waktu yang dibutuhkan   : 108 microseconds

[ALGORITMA DIVIDE AND CONQUER]
hasil perklian polinom  : 6084 + 14040x + 6023x^2 + 10778x^3 + 7972x^4 -
8382x^5 + 5714x^6 - 5021x^7 + 4768x^8 + 2852x^9 - 889x^10 + 21908x^11 - 1
8292x^12 - 492x^13 + 14730x^14 - 19436x^15 + 18376x^16 + 7526x^17 - 21223
x^18 + 172x^19 + 16627x^20 - 21384x^21 + 16204x^22 - 3914x^23 - 4093x^24
- 3775x^25 + 10215x^26 + 942x^27 - 21866x^28 + 14860x^29 + 4585x^30 - 699
3x^31 - 2971x^32 + 11138x^33 + 2848x^34 - 2627x^35 - 10686x^36 + 6884x^37
 + 10408x^38 - 6780x^39 - 4160x^40
jumlah operasi kali     : 169
jumlah operasi tambah   : 888
waktu yang dibutuhkan   : 6265 microseconds
```

Untuk n = 50

```
Masukkan panjang suku polinom : 50
polinom 1  : 23 + 10x + 78x^2 + 48x^3 - 52x^4 + 54x^5 - 23x^6 + 21x^7 + 53x^8 - 67x^9 + 14x^10 + 93x^11 + 98x^12 + 94x^13 + 95x^14 - 55x^15 +
45x^16 + 28x^17 - 40x^18 + 68x^19 - 8x^20 - 4x^21 - 58x^22 - 4x^23 - 76x^24 + 65x^25 - 30x^26 - 98x^27 + 64x^28 - 91x^29 + 50x^30 + 81x^31 + 1
3x^32 + 23x^33 + 38x^34 + 86x^35 + 18x^36 - 100x^37 - 96x^38 + 7x^39 + 77x^40 + 18x^41 + 84x^42 - 80x^43 - 32x^44 - 41x^45 + 88x^46 - 81x^47 +
 2x^48 - 72x^49 + 38x^50
polinom 2  : 43 - 43x + 53x^2 - 8x^3 - 36x^4 - 65x^5 + 67x^6 + 74x^7 + 2x^8 - 82x^9 + 41x^10 + 11x^11 - 47x^12 + 89x^13 - 98x^14 - 30x^15 + 55
x^16 - 25x^17 - 24x^18 - 40x^19 - 26x^20 + 70x^21 + 69x^22 + 95x^23 + 29x^24 - 10x^25 + 59x^26 - 80x^27 + 96x^28 + 11x^29 - 94x^30 - 52x^31 +
9x^32 + 74x^33 - 57x^34 - 70x^35 - 18x^36 + 60x^37 - 52x^38 - 72x^39 - 27x^40 - 52x^41 + 20x^42 + 48x^43 - 70x^44 + 71x^45 - 60x^46 + 2x^47 -
6x^48 - 60x^49 - 86x^50

[ALGORITMA BRUTE FORCE]
hasil perklian polinom  : 989 - 559x + 4143x^2 - 944x^3 - 1074x^4 + 4623x^5 - 8368x^6 + 744x^7 + 4489x^8 + 4695x^9 + 3789x^10 - 5706x^11 - 113
8x^12 + 12092x^13 + 2341x^14 - 491x^15 - 6938x^16 - 25000x^17 + 24776x^18 + 4761x^19 - 4847x^20 + 7473x^21 - 22675x^22 + 16016x^23 + 15453x^24
 - 355x^25 - 1664x^26 + 8375x^27 - 9328x^28 - 819x^29 - 1328x^30 - 7234x^31 - 5504x^32 + 2765x^33 + 15760x^34 + 19326x^35 + 24918x^36 - 14103x
^37 + 19656x^38 - 1438x^39 - 6715x^40 + 21472x^41 + 9201x^42 - 29094x^43 - 17067x^44 - 16291x^45 - 12996x^46 - 7666x^47 + 6651x^48 - 32441x^49
 - 32647x^50 - 34073x^51 - 9437x^52 + 12107x^53 + 785x^54 - 35731x^55 + 41779x^56 + 13538x^57 + 4978x^58 + 12818x^59 - 59627x^60 - 5737x^61 -
2382x^62 + 9371x^63 + 8333x^64 - 14524x^65 - 30324x^66 + 3776x^67 + 48636x^68 - 36150x^69 - 14072x^70 - 20668x^71 - 4530x^72 + 20371x^73 + 139
x^74 - 11928x^75 + 6774x^76 + 4890x^77 + 17413x^78 + 20773x^79 - 33627x^80 - 20737x^81 - 1392x^82 + 13754x^83 - 3947x^84 - 2446x^85 - 8180x^86
 + 26091x^87 + 6088x^88 + 1944x^89 - 16647x^90 + 4886x^91 - 16181x^92 + 21088x^93 - 3370x^94 + 5754x^95 - 5144x^96 + 7354x^97 + 3920x^98 + 391
2x^99 - 3268x^100
jumlah operasi kali     : 2601
jumlah operasi tambah   : 2601
waktu yang dibutuhkan   : 289 microseconds

[ALGORITMA DIVIDE AND CONQUER]
hasil perklian polinom  : 989 - 559x + 4143x^2 - 944x^3 - 1074x^4 + 4623x^5 - 8368x^6 + 744x^7 + 4489x^8 + 4695x^9 + 3789x^10 - 5706x^11 - 113
8x^12 + 12092x^13 + 2341x^14 - 491x^15 - 6938x^16 - 25000x^17 + 24776x^18 + 4761x^19 - 4847x^20 + 7473x^21 - 22675x^22 + 16016x^23 + 15453x^24
 - 355x^25 - 1664x^26 + 8375x^27 - 9328x^28 - 819x^29 - 1328x^30 - 7234x^31 - 5504x^32 + 2765x^33 + 15760x^34 + 19326x^35 + 24918x^36 - 14103x
^37 + 19656x^38 - 1438x^39 - 6715x^40 + 21472x^41 + 9201x^42 - 29094x^43 - 17067x^44 - 16291x^45 - 12996x^46 - 7666x^47 + 6651x^48 - 32441x^49
 - 32647x^50 - 34073x^51 - 9437x^52 + 12107x^53 + 785x^54 - 35731x^55 + 41779x^56 + 13538x^57 + 4978x^58 + 12818x^59 - 59627x^60 - 5737x^61 -
2382x^62 + 9371x^63 + 8333x^64 - 14524x^65 - 30324x^66 + 3776x^67 + 48636x^68 - 36150x^69 - 14072x^70 - 20668x^71 - 4530x^72 + 20371x^73 + 139
x^74 - 11928x^75 + 6774x^76 + 4890x^77 + 17413x^78 + 20773x^79 - 33627x^80 - 20737x^81 - 1392x^82 + 13754x^83 - 3947x^84 - 2446x^85 - 8180x^86
 + 26091x^87 + 6088x^88 + 1944x^89 - 16647x^90 + 4886x^91 - 16181x^92 + 21088x^93 - 3370x^94 + 5754x^95 - 5144x^96 + 7354x^97 + 3920x^98 + 391
2x^99 - 3268x^100
jumlah operasi kali     : 631
jumlah operasi tambah   : 3480
waktu yang dibutuhkan   : 9912 microseconds
```

Analisis Program

Dari hasil Screenshot di atas terlihat bahwa waktu yang dibutuhkan oleh algoritma *Divide and Conquer* lebih lama daripada algoritma *Brute Force*. Padahal secara teori seharusnya algoritma *Divide and Conquer* dengan kompleksitas $O(n^{log3})$ dan dengan umlah perkalian yang lebih sedikit akan menyelesaikan permasalah lebih cepat dibandung algoritma *Brute Force* dengan kompleksitas $O(n^2)$

Hal ini terjadi karena implementasi program polinom menggunakan array, sehingga untuk menambahkan, mengurangi, membagi dan memangkatkan sebuah array program harus memangil fungsi lain lagi. Di mana fungsi-fungsi yang dipanggil ini juga memanfaatkan loop di dalamnya sehingga kompleksitas algoritmanya tidak murni $O(n^{log3})$ tetapi juga terbebani oleh kompleksitas dari fungsi-fungsi lain yang ikut dipanggil.

Spesifikasi komputer yang digunakan

Nama Laptop : Asus E202SA

OS : Ubuntu 18.04

RAM : 2048 MB / 2GB

Processor : Intel(R) Celeron(R) CPU N3060 @1.60Ghz (2 CPUs), ~1.6Ghz

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