Laporan Tugas Kecil

IF2211 - Strategi Algoritma

Perkalian Dua Buah Polinom Menggunakan Strategi Algoritma *Brute Force* dan *Divide and Conquer*

Disusun oleh:

Gregorius Jovan Kresnadi

13518135

SEKOLAH TINGGI ELEKTRO DAN INFORMATIKA

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2020

Bab I

Algoritma Brute Force

Strategi algoritma *Brute Force* adalah salah satu pendekatan yang lempang untuk memecahkan suatu permasalahan. Algoritma *Brute Force* memecahkan masalah dengan sangat sederhana dan didasarkan pada pernyataan pada persoalan dan definisi konsep yang dilibatkan. Dalam persoalan perkalian polinom, algoritma akan langsung mengalikan setiap koefiisien polinom pertama satu per satu dengan setiap koefisien polinom kedua.

Berikut adalah implementasi algoritma dalam bentuk *pseudocode*:

```
{ KAMUS }
constant MAX_LENGTH : integer
constant degree     : integer
type Polinom : array[1..degree] of integer

{ ALGORITMA }
function Kali (Polinom: P1, P2) -> array of integer
   { KAMUS LOKAL }
   i, j : integer
   Ph : array[1..degree*2] of integer

   {Inisiasi Polinom Hasil}
   for i<-1 to degree*2 do
      Ph[i] = 0

   { ALGORITMA }
   for i <- 1 to degree do
      for j <- 1 to degree do
         Ph[i+j] <- Ph[i+j] + P1[i] * P2[j]
   -> Ph
```

Jumlah perhitungan yang dilakukan oleh algoritma di atas adalah:

$n * (n-1) * 2 = 2n^2 - 2n$

Kompleksitas: $O(n^2)$

Bab II

Algoritma Divide and Conquer

Strategi algoritma *Divide and Conquer* membagi suatu pemecahan masalah dalam tiga bagian, yaitu membagi (*Divide),* menyelesaikan (*Conquer*), dan menggabungkan (*Combine*). Pertama, suatu permasalah akan dibagi menjadi beberapa upa-masalah yang memiliki kemiripan dengan permasalahan semulla namun berukuran lebih kecil. Kemudian, masing-masing masalah akan diselesaikan. Terakhir, solusi dari pemecahan masalah akan digabung sehingga membentuk solusi dari permasalahan semula.

Dalam persoalan perkalian dua polinom, pertama masing-masing polinom akan dibagi menjadi 2 bagian. Algoritma akan mengikuti formula berikut:

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

Define

$$\begin{aligned} Y(x) &= (A_0(x) + A_1(x)) \times (B_0(x) + B_1(x)) \\ U(x) &= A_0(x)B_0(x) \\ Z(x) &= A_1(x)B_1(x) \end{aligned}$$

Then

$$Y(x) - U(x) - Z(x) = A_0(x)B_1(x) + A_1(x)B_0(x).$$

Hence $A(x)B(x)$ is equal to

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

Sumber: http://algorithm.cs.nthu.edu.tw/~course/Extra_Info/Divide%20and%20Conquer_supplement.pdf

Berikut adalah implementasi algoritma dalam bentuk *pseudocode* yang memiliki kompleksitas $O(n^{\log 3})$:

```
{ KAMUS }
constant degree: integer
constant n     : integer

type Polinom : array[1..degree] of integer

{ ALGORITMA }
function Kali(Polinom P1, P2) -> array of integer
   if (size(P1) = 1)
      Ph : array [1..2] of integer
      Ph[0] <- P1[0] * P2[0]
      -> Ph
   else
      half1 : integer
      half2 : integer
      half1 <- floor(size(P1)/2)
      half2 <- size(P1) - half1
      Ph : array of [1..size(P1)*2] of integer
      P1A : array of [1..half1] of integer
      P1B : array of [1..half2] of integer
      P2A : array of [1..half1] of integer
```

```
    P2B : array of [1..half2] of integer
    PX  : array of [1..half2*2] of integer
    PY  : array of [1..half1] of integer
    PZ  : array of [1..half2*2] of integer
    PXYZ  : array of [1..half2*2] of integer

    i : integer
    for i <- 0 to half1 do
       P1A[i] <- P1[i]
       P2A[i] <- P2[i]
    for i to size(P1) do
       P1B[i-half1] <- P1[i]
       P2B[i-half1] <- P2[i]

    PX <- Kali(P1A+P1B, P2A+P2B)
    PY <- Kali(P1A,P2A)
    PZ <- Kali(P1B,P2B)
    PXYZ <- PX - PY - PZ
    NaikDerajat(PXYZ, half1)
    NaikDerajat(PZ, half1*2)
    Ph <- PY + PXYZ + PZ
    -> Ph
```

Bab III

Kode Program

Berikut adalah kode program yang digunakan dalam pembuatan algoritma. Program dibuat dalam bahasa C++. File PolinomBF.hpp adalah file header untuk perhitungan polinom dengan metode *Brute Force*, dan file PolinomBF.cpp merupakan implementasinya. Sedangkan File PolinomDNC.hpp adalah file header untuk perhitungan polinom dengan metode *Divide and Conquer*, dan file PolinomDNC.cpp merupakan implementasinya. Untuk menggunakan kedua pustaka tersebut, maka dibuat file mainPolinom.cpp yang akan menerima masukan derajat tertinggi, dan menuliskan dua buah polinom yang dibentuk secara acak, hasil perkalian dengan metode *Brute Force* dan *Divide and Conquer*, beserta jumlah operasi pertambahan, perkalian, dan waktu eksekusi tiap metode.

PolinomBF.hpp

```
#ifndef POLINOMBF_HPP
#define POLINOMBF_HPP

#define RANGE 100

class PolinomBF {
  public:
    // ctor, dtor
    PolinomBF();    // ctor PolinomBF dengan orde = 0
    PolinomBF(int); // ctor PolinomBF dengan orde = n (sesuai parameter)
    ~PolinomBF();

    // Mengisi polinom dengan angka sembarang
    void FillPolinomBF();
    void FillPolinomBF(int SEED);

    // getter, setter
    int getCoefAt(int idx) const;
    int getDegree() const;
    void setCoefAt(int idx, int val);
    void setDegree(int);

    // Operator overloading
    friend PolinomBF operator+(const PolinomBF&, const PolinomBF&);
    friend PolinomBF operator*(const PolinomBF&, const PolinomBF&);

    // Mencetak polinom dengan format: A+Bx^1+Cx^2+Dx^3...dst
    void print();

  private:
```

```
    const static int MAX_LENGTH = 100000;
    int coef[MAX_LENGTH];
    int degree; // derajat tertinggi
};

#endif
```

PolinomBF.cpp

```
#include "PolinomBF.hpp"
#include <iostream>
#include <cstdlib>
#include <chrono>
#include <ratio>

using namespace std;
using namespace std::chrono;

// ctor, cctor, dtor, op=
PolinomBF::PolinomBF(int degree) {
    this -> degree = degree;
    for (int i = 0; i < MAX_LENGTH; i++) {
        this -> coef[i] = 0;
    }
}

PolinomBF::PolinomBF() : PolinomBF(0) {};

PolinomBF::~PolinomBF() {}

// Mengisi polinom dengan angka sembarang
void PolinomBF::FillPolinomBF() {
    srand(time(NULL));
    for (int i = 0; i < degree; i++) {
        coef[i] = rand() % RANGE;
    }
}

void PolinomBF::FillPolinomBF(int SEED) {
    srand(SEED);
    for (int i = 0; i < degree; i++) {
        coef[i] = rand() % RANGE;
    }
}
```

```
// getter, setter
int PolinomBF::getCoefAt(int idx) const {
    return coef[idx];
}
int PolinomBF::getDegree() const {
    return degree;
}
void PolinomBF::setCoefAt(int idx, int val) {
    coef[idx] = val;
}
void PolinomBF::setDegree(int idx) {
    degree = idx;
}

// Operasi penjumlahan
PolinomBF operator+(const PolinomBF & P1, const PolinomBF & P2) {
    PolinomBF Ph;
    Ph.setDegree(P1.getDegree());
    for (int i = 0; i < Ph.getDegree(); i++) {
        Ph.coef[i] = P1.coef[i] + P2.coef[i];
    }
    return Ph;
}

// Operasi perkalian
PolinomBF operator*(const PolinomBF & P1, const PolinomBF & P2) {
    PolinomBF Ph;
    long long int opTambah = 0;
    long long int opKali = 0;
    cout << "\n[] Perkalian Polinom Metode Brute Force []\n";

    // Perhitungan
    auto start = high_resolution_clock::now();
    Ph.setDegree(P1.getDegree() + P2.getDegree());
    for (int i = 0; i < P1.getDegree(); i++) {
        for (int j = 0; j < P2.getDegree(); j++) {
            Ph.coef[i+j] += P1.coef[i] * P2.coef[j];
            opTambah++;
            opKali++;
        }
    }
    auto stop = high_resolution_clock::now();

    // Results
```

```
    Ph.print();
    auto dur = duration_cast<microseconds>(stop-start);
    cout << endl << "[] ============--------- - - - -  -  -   -" << endl;
    cout << "|| Jumlah operasi tambah: " << opTambah << endl;
    cout << "|| Jumlah operasi kali  : " << opKali << endl;
    cout << "|| Jumlah total operasi : " << opTambah + opKali << endl;
    cout << "|| Waktu penghitungan: ";
    cout << dur.count() << " microseconds" << endl;
    cout << "[] ============--------- - - - -  -  -   -" << endl;
    return Ph;
}

// Mencetak polinom dengan format: A+Bx^1+Cx^2+Dx^3...dst
void PolinomBF::print() {
    cout << coef[0];
    for (int i = 1; i <= degree; i++) {
        if (coef[i] < 0) cout << coef[i] << "x^" << i;
        else if (coef[i] == 0) continue;
        else cout << "+" << coef[i] << "x^" << i;
    }
    cout << endl;
}
```

PolinomDNC.hpp

```
#ifndef POLINOMDNC_HPP
#define POLINOMDNC_HPP

#define RANGE 100
#include <vector>

class PolinomDnC {
public:
  // Vars
  std::vector<int> P;
  int degree; // derajat tertinggi

  // ctor, cctor, dtor
  PolinomDnC();    // ctor Polinom dengan orde = 0
  PolinomDnC(int); // ctor Polinom dengan orde = n (sesuai parameter)
  PolinomDnC& operator=(const PolinomDnC&);
  ~PolinomDnC();

  // Mengisi polinom dengan angka sembarang
```

```
  void FillPolinomDnC();
  void FillPolinomDnC(int);

  // Operator overloading
  friend PolinomDnC operator+(const PolinomDnC&, const PolinomDnC&);
  friend PolinomDnC operator-(const PolinomDnC&, const PolinomDnC&);

  // Menaikkan derajat polinom sebanyak n
  void Mundur(int n);

  // Mencetak polinom dengan format: A+Bx^1+Cx^2+Dx^3...dst
  void print();

};

// Fungsi perkalian 2 polinom
PolinomDnC Kali(const PolinomDnC& P1, const PolinomDnC& P2, long long int &coun
tPlus, long long int &countKali );

#endif
```

PolinomDNC.cpp

```
#include "PolinomDNC.hpp"
#include <iostream>
#include <vector>
#include <ctime>
#include "math.h"

using namespace std;

// ctor, dtor
PolinomDnC::PolinomDnC(int degree) {
    this->degree = degree;
    P.reserve(degree+1);
}

PolinomDnC::PolinomDnC() : PolinomDnC(1) {}

PolinomDnC& PolinomDnC::operator=(const PolinomDnC& Pol) {
    degree = Pol.degree;
    for (int i=0; i<Pol.P.size(); i++) {
        P.push_back(Pol.P[i]);
    }
```

```
    return *this;
}

PolinomDnC::~PolinomDnC() {}

// Mengisi polinom dengan angka sembarang
void PolinomDnC::FillPolinomDnC() {
    srand(time(NULL));
    for (int i = 0; i < degree; i++) {
        P.push_back(rand() % RANGE);
    }
}
void PolinomDnC::FillPolinomDnC(int SEED) {
    srand(SEED);
    for (int i = 0; i < degree; i++) {
        P.push_back(rand() % RANGE);
    }
}

// Operasi penjumlahan
PolinomDnC operator+(const PolinomDnC & P1, const PolinomDnC & P2) {
    PolinomDnC Ph;
    int i;
    if (P1.P.size() > P2.P.size()) {
        for(i = 0; i<P2.P.size(); i++) {
            Ph.P.push_back(P1.P[i] + P2.P[i]);
        }
        for(i; i<P1.P.size(); i++) {
            Ph.P.push_back(P1.P[i]);
        }
    } else {
        for(i = 0; i<P1.P.size(); i++) {
            Ph.P.push_back(P1.P[i] + P2.P[i]);
        }
        for(i; i<P2.P.size(); i++) {
            Ph.P.push_back(P2.P[i]);
        }
    }
    return Ph;
}

// Operasi pengurangan
PolinomDnC operator-(const PolinomDnC & P1, const PolinomDnC & P2) {
    PolinomDnC Ph;
    int i;
```

```
    if (P1.P.size() > P2.P.size()) {
        for(i = 0; i<P2.P.size(); i++) {
            Ph.P.push_back(P1.P[i] - P2.P[i]);
        }
        for(i; i<P1.P.size(); i++) {
            Ph.P.push_back(P1.P[i]);
        }
    } else {
        for(i = 0; i<P1.P.size(); i++) {
            Ph.P.push_back(P1.P[i] - P2.P[i]);
        }
        for(i; i<P2.P.size(); i++) {
            Ph.P.push_back(P2.P[i]);
        }
    }
    return Ph;
}

// Menaikkan derajat polinom sebanyak n
void PolinomDnC::Mundur(int n) {
    for (int i=0; i < n; i++) {
        P.insert(P.begin(), 0);
    }
}

// Mencetak polinom dengan format: A+Bx^1+Cx^2+Dx^3...dst
void PolinomDnC::print() {
    cout << P[0];
    for (int j= 1; j< P.size(); j++) {
        if (P[j] < 0) cout << P[j] << "x^" << j;
        else if (P[j] == 0) continue;
        else cout << "+" << P[j] << "x^" << j;
    }
    cout << endl;
}

// Fungsi perkalian 2 polinom
PolinomDnC Kali(const PolinomDnC& P1, const PolinomDnC& P2, long long int &coun
tPlus, long long int &countKali) {
    if (P1.P.size() == 1) {
        PolinomDnC Ph(2);
        Ph.P.push_back(P1.P[0] * P2.P[0]);
        countKali+=1;
        return Ph;
    } else {
```

```
        int half1 = floor(P1.P.size()/2);
        int half2 = P1.P.size() - half1;
        PolinomDnC Ph(P1.P.size()*2);
        PolinomDnC P1A(half1), P1B(half2), P2A(half1), P2B(half2);
        PolinomDnC PX(half2*2), PY(half1), PZ(half2*2);
        PolinomDnC PXYZ(half2*2);
        int i;
        for (i=0; i<half1; i++) {
            P1A.P.push_back(P1.P[i]);
            P2A.P.push_back(P2.P[i]);
        }
        for (i; i<P1.P.size(); i++) {
            P1B.P.push_back(P1.P[i]);
            P2B.P.push_back(P2.P[i]);
        }
        PX = Kali(P1A+P1B, P2A+P2B, countPlus, countKali);
        PY = Kali(P1A,P2A,countPlus,countKali);
        PZ = Kali(P1B,P2B,countPlus,countKali);
        PXYZ = PX - PY - PZ;
        PXYZ.Mundur(floor(P1.P.size()/2));
        PZ.Mundur(2*floor(P1.P.size()/2));
        Ph = PY + PXYZ + PZ;
        countPlus += 4;
        return Ph;
    }
}
```

mainPolinom.cpp

```
#include "PolinomBF.hpp"
#include "PolinomDNC.hpp"
#include <iostream>
#include <ctime>
#include <chrono>
#include <ratio>

#define SEEDX 13518135
#define SEEDY 16518017

// ! UBAH SEED DISINI
#define SEED1
#define SEED2 SEEDX+time(0)

using namespace std;
```

```
using namespace std::chrono;

int main() { // Welcome
    cout << "[]  / \\ / \\ / \\ / \\ / \\  []" << endl;
    cout << " \\   Pengali 2 Polinom   / " << endl;
    cout << " \\ - - - - - - - - - - - / " << endl;
    cout << " \\        Metode:        / " << endl;
    cout << " \\    1. BruteForce      / " << endl;
    cout << " \\  2. Divide & Conquer  / " << endl;
    cout << "[]  \\ / \\ / \\ / \\ / \\ /  []" << endl;
    cout << "||" << endl;
    cout << "[]=>>  Masukan suku tertinggi polinom: ";

    // Input
    int n;
    cin >> n;
    cout << endl;

    // Inisiasi
    PolinomBF PBF1(n+1);
    PolinomBF PBF2(n+1);
    PolinomBF PBFh;
    PolinomDnC PDnC1(n+1);
    PolinomDnC PDnC2(n+1);
    PolinomDnC PDnCh(n+n+1+1);

    // Perhitungan BF
    cout << "[] Polinom 1 []\n";
    PBF1.FillPolinomBF(SEED1);
    PBF1.print();
    cout << "\n[] Polinom 2 []\n";
    PBF2.FillPolinomBF(SEED2);
    PBF2.print();
    PBFh = PBF1 * PBF2;

    // Perhitungan DnC
    cout << "\n[] Perkalian Polinom Metode Divide & Conquer []\n";
    PDnC1.FillPolinomDnC(SEED1);
    PDnC2.FillPolinomDnC(SEED2);
    long long int countPlus = 0;
    long long int countKali = 0;
    auto start = high_resolution_clock::now();
    PDnCh = Kali(PDnC1, PDnC2, countPlus, countKali);
    auto stop = high_resolution_clock::now();
    auto dur = duration_cast<microseconds>(stop-start);
```

```
    PDnCh.print();
    cout << endl << "[] ============--------- - - - -  -  -   -" << endl;
    cout << "|| Jumlah operasi tambah: " << countPlus << endl;
    cout << "|| Jumlah operasi kali  : " << countKali << endl;
    cout << "|| Jumlah total operasi : " << countPlus + countKali << endl;
    cout << "|| Waktu penghitungan: ";
    cout << dur.count() << " microseconds" << endl;
    cout << "[] ============--------- - - - -  -  -   -" << endl;

    return 0;
}
```

Bab IV

Pengujian Program

Dalam proses pengujian program, diambil sampel data dengan n=5, n=10, n=20, dan n=50 di mana n adalah derajat tertinggi polinom. Untuk menjalankan programnya, berikut adalah spesifikasi komputer yang digunakan:

- Prosesor: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (8 CPUs) ~ 2.1GHz
- RAM: 8.00 GB
- Sistem Operasi: Windows 10 Home version, 64-bit

Berikut adalah *screenshot* dari hasil pengujian program:

1. n = 5

```
[]  / \ / \ / \ / \ / \  []
 \   Pengali 2 Polinom   /
 \ - - - - - - - - - - - /
 \        Metode:        /
 \    1. BruteForce      /
 \  2. Divide & Conquer  /
[]  \ / \ / \ / \ / \ /  []
||
[]=>>  Masukan suku tertinggi polinom: 5

[] Polinom 1 []
39+58x^1+95x^2+54x^3+13x^4+90x^5

[] Polinom 2 []
84+97x^1+63x^2+88x^3+75x^4+83x^5

[] Perkalian Polinom Metode Brute Force []
3276+8655x^1+16063x^2+20837x^3+20344x^4+28170x^5+26240x^6+18749x^7+13377x^8+7829x^9+7470x^10

[] ========------- - - - -  -  -   -
|| Jumlah operasi tambah: 36
|| Jumlah operasi kali  : 36
|| Jumlah total operasi : 72
|| Waktu penghitungan: 0 microseconds
[] ========------- - - - -  -  -   -

[] Perkalian Polinom Metode Divide & Conquer []
3276+8655x^1+16063x^2+20837x^3+20344x^4+28170x^5+26240x^6+18749x^7+13377x^8+7829x^9+7470x^10

[] ========------- - - - -  -  -   -
|| Jumlah operasi tambah: 40
|| Jumlah operasi kali  : 21
|| Jumlah total operasi : 61
|| Waktu penghitungan: 0 microseconds
[] ========------- - - - -  -  -   -
```

2. n = 10

```
[]  / \ / \ / \ / \ / \  []
 \   Pengali 2 Polinom   /
 \ - - - - - - - - - - - /
 \        Metode:        /
 \    1. BruteForce      /
 \  2. Divide & Conquer  /
[]  \ / \ / \ / \ / \ /  []
||
[]=>>  Masukan suku tertinggi polinom: 10

[] Polinom 1 []
13+88x^1+57x^2+57x^3+69x^4+55x^5+79x^6+70x^7+12x^8+70x^9+39x^10

[] Polinom 2 []
58+59x^1+25x^2+91x^3+63x^4+48x^5+49x^6+24x^7+62x^8+60x^9+39x^10

[] Perkalian Polinom Metode Brute Force []
754+5871x^1+8823x^2+10052x^3+17617x^4+20041x^5+23191x^6+27326x^7+24600x^8+30881x^9+34749x^10+28
082x^11+26573x^12+23634x^13+18974x^14+16815x^15+11616x^16+8726x^17+7086x^18+5070x^19+1521x^20

[] ---------- - - - -  -  -   -
|| Jumlah operasi tambah: 121
|| Jumlah operasi kali  : 121
|| Jumlah total operasi : 242
|| Waktu penghitungan: 0 microseconds
[] ---------- - - - -  -  -   -

[] Perkalian Polinom Metode Divide & Conquer []
754+5871x^1+8823x^2+10052x^3+17617x^4+20041x^5+23191x^6+27326x^7+24600x^8+30881x^9+34749x^10+28
082x^11+26573x^12+23634x^13+18974x^14+16815x^15+11616x^16+8726x^17+7086x^18+5070x^19+1521x^20

[] ---------- - - - -  -  -   -
|| Jumlah operasi tambah: 116
|| Jumlah operasi kali  : 59
|| Jumlah total operasi : 175
|| Waktu penghitungan: 7885 microseconds
[] ---------- - - - -  -  -   -
```

3. n = 20

```
[]  / \ / \ / \ / \ / \  []
 \   Pengali 2 Polinom   /
 \ - - - - - - - - - - - /
 \        Metode:        /
 \    1. BruteForce      /
 \  2. Divide & Conquer  /
[]  \ / \ / \ / \ / \ /  []
||
[]=>>  Masukan suku tertinggi polinom: 20

[] Polinom 1 []
31+15x^1+7x^2+68x^3+78x^4+45x^5+56x^6+92x^7+5x^8+16x^9+82x^10+39x^11+45x^12+92x^13+78x^14+92x^1
5+92x^16+76x^17+33x^18+69x^19+84x^20

[] Polinom 2 []
76+86x^1+43x^2+2x^3+40x^4+38x^5+25x^6+15x^7+55x^8+5x^9+15x^10+20x^11+41x^12+82x^13+18x^14+58x^1
5+74x^16+66x^17+47x^18+30x^19+16x^20

[] Perkalian Polinom Metode Brute Force []
2356+3806x^1+3155x^2+6477x^3+13347x^4+14844x^5+13241x^6+17725x^7+18599x^8+13263x^9+15852x^10+23
437x^11+22409x^12+23855x^13+29156x^14+35902x^15+38553x^16+41827x^17+42314x^18+44875x^19+51517x^
20+45867x^21+40875x^22+40140x^23+35596x^24+34150x^25+33379x^26+33306x^27+35354x^28+33282x^29+33
658x^30+32173x^31+34130x^32+29876x^33+21304x^34+19960x^35+16073x^36+10993x^37+6546x^38+3624x^39
+1344x^40

[] ========------- - - - -  -  -   -
|| Jumlah operasi tambah: 441
|| Jumlah operasi kali  : 441
|| Jumlah total operasi : 882
|| Waktu penghitungan: 0 microseconds
[] ========------- - - - -  -  -   -

[] Perkalian Polinom Metode Divide & Conquer []
2356+3806x^1+3155x^2+6477x^3+13347x^4+14844x^5+13241x^6+17725x^7+18599x^8+13263x^9+15852x^10+23
437x^11+22409x^12+23855x^13+29156x^14+35902x^15+38553x^16+41827x^17+42314x^18+44875x^19+51517x^
20+45867x^21+40875x^22+40140x^23+35596x^24+34150x^25+33379x^26+33306x^27+35354x^28+33282x^29+33
658x^30+32173x^31+34130x^32+29876x^33+21304x^34+19960x^35+16073x^36+10993x^37+6546x^38+3624x^39
+1344x^40

[] ========------- - - - -  -  -   -
|| Jumlah operasi tambah: 336
|| Jumlah operasi kali  : 169
|| Jumlah total operasi : 505
|| Waktu penghitungan: 0 microseconds
[] ========------- - - - -  -  -   -
```

4. n = 50

```
[]  / \ / \ / \ / \ / \  []
 \   Pengali 2 Polinom   /
 \ - - - - - - - - - - - /
 \        Metode:        /
 \    1. BruteForce      /
 \  2. Divide & Conquer  /
[]  \ / \ / \ / \ / \ /  []
||
[]=>>  Masukan suku tertinggi polinom: 50

[] Polinom 1 []
39+64x^1+1x^2+25x^3+79x^4+56x^5+63x^6+96x^7+99x^8+37x^9+1x^10+93x^11+64x^12+65x^13+55x^14+93x^15+50x^
16+84x^17+11x^18+51x^19+80x^20+35x^21+6x^22+38x^23+57x^24+39x^25+46x^26+84x^27+62x^28+89x^29+69x^30+8
0x^31+26x^32+59x^33+52x^34+73x^35+23x^36+98x^37+25x^38+66x^39+47x^40+58x^41+21x^42+93x^43+55x^44+87x^
45+54x^46+11x^47+47x^48+22x^49+41x^50

[] Polinom 2 []
83+3x^1+37x^2+91x^3+41x^4+48x^5+32x^6+18x^7+82x^8+27x^9+34x^10+73x^11+60x^12+23x^13+95x^14+59x^15+32x
^16+6x^17+25x^18+12x^19+12x^20+19x^21+27x^22+56x^23+13x^24+72x^25+96x^26+41x^27+76x^28+47x^29+45x^30+
15x^31+44x^32+15x^33+22x^35+88x^36+60x^37+34x^38+36x^39+49x^40+58x^41+5x^42+94x^43+36x^44+70x^45+49x^
46+24x^47+65x^48+37x^49+59x^50

[] Perkalian Polinom Metode Brute Force []
3237+5429x^1+1718x^2+7995x^3+14092x^4+10397x^5+14956x^6+21241x^7+24753x^8+25860x^9+23978x^10+35374x^1
1+34885x^12+31979x^13+38730x^14+50088x^15+47539x^16+47762x^17+47469x^18+57811x^19+56158x^20+47882x^21
+54079x^22+58935x^23+48859x^24+53624x^25+60526x^26+67447x^27+63709x^28+75163x^29+74836x^30+81447x^31+
77621x^32+82710x^33+85167x^34+82125x^35+82013x^36+95257x^37+88492x^38+94279x^39+100980x^40+103842x^41
+101866x^42+115710x^43+108564x^44+115918x^45+105188x^46+109691x^47+115310x^48+119471x^49+110571x^50+1
21397x^51+118545x^52+118284x^53+114862x^54+112244x^55+117690x^56+111063x^57+100550x^58+96551x^59+9965
8x^60+88841x^61+84962x^62+88267x^63+80608x^64+83156x^65+76481x^66+76549x^67+70897x^68+75914x^69+73582
x^70+74759x^71+69761x^72+72033x^73+69063x^74+64368x^75+63299x^76+56994x^77+55977x^78+49357x^79+53635x
^80+46078x^81+46134x^82+40908x^83+41047x^84+38608x^85+38942x^86+33961x^87+31458x^88+31857x^89+23799x^
90+26909x^91+17724x^92+22948x^93+15557x^94+12922x^95+9185x^96+4802x^97+6252x^98+2815x^99+2419x^100

[] ---------- - - - -  -  -   -
|| Jumlah operasi tambah: 2601
|| Jumlah operasi kali  : 2601
|| Jumlah total operasi : 5202
|| Waktu penghitungan: 0 microseconds
[] ---------- - - - -  -  -   -

[] Perkalian Polinom Metode Divide & Conquer []
3237+5429x^1+1718x^2+7995x^3+14092x^4+10397x^5+14956x^6+21241x^7+24753x^8+25860x^9+23978x^10+35374x^1
1+34885x^12+31979x^13+38730x^14+50088x^15+47539x^16+47762x^17+47469x^18+57811x^19+56158x^20+47882x^21
+54079x^22+58935x^23+48859x^24+53624x^25+60526x^26+67447x^27+63709x^28+75163x^29+74836x^30+81447x^31+
77621x^32+82710x^33+85167x^34+82125x^35+82013x^36+95257x^37+88492x^38+94279x^39+100980x^40+103842x^41
+101866x^42+115710x^43+108564x^44+115918x^45+105188x^46+109691x^47+115310x^48+119471x^49+110571x^50+1
21397x^51+118545x^52+118284x^53+114862x^54+112244x^55+117690x^56+111063x^57+100550x^58+96551x^59+9965
8x^60+88841x^61+84962x^62+88267x^63+80608x^64+83156x^65+76481x^66+76549x^67+70897x^68+75914x^69+73582
x^70+74759x^71+69761x^72+72033x^73+69063x^74+64368x^75+63299x^76+56994x^77+55977x^78+49357x^79+53635x
^80+46078x^81+46134x^82+40908x^83+41047x^84+38608x^85+38942x^86+33961x^87+31458x^88+31857x^89+23799x^
90+26909x^91+17724x^92+22948x^93+15557x^94+12922x^95+9185x^96+4802x^97+6252x^98+2815x^99+2419x^100

[] ---------- - - - -  -  -   -
|| Jumlah operasi tambah: 1260
|| Jumlah operasi kali  : 631
|| Jumlah total operasi : 1891
|| Waktu penghitungan: 0 microseconds
[] ---------- - - - -  -  -   -
```

Selain menjalankan program untuk test case yang diwajibkan (N =5, 10, 20, 50), program juga diuji dengan masukan N yang lebih besar. Berikut adalah hasil dari pengujiannya:

No	N	Brute Force			Divide and Conquer		
		Waktu	Tambah	Kali	Waktu	Tambah	Kali
1	100	0 ms	10201	10201	11230 ms	3752	1877
2	500	0 ms	251001	251001	90210 ms	39216	19609
3	1000	2071 ms	1002001	1002001	279188 ms	117620	58811
4	5000	151082 ms	25010001	25010001	4749878 ms	1980608	990305
5	10000	609982 ms	100020001	100020001	14239746 ms	5940804	2970403
6	50000	15549113 ms	2500100001	2500100001	180870060 ms	70552988	35276495

Berdasarkan dari hasil pengujian yang diperoleh, terlihat kalkulasi dengan algoritma *Brute Force* lebih cepat daripada kalkulasi dengan algoritma *Divide and Conquer*. Hal demikian dapat disebabkan oleh sifat algoritma *Divide and Conquer* yang rekursif, sehingga setiap iterasinya harus menginisiasi beberapa array dinamis yang baru, dan memakan waktu yang cukup lama.

Tabel Poin

Poin	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	√	