**LAPORAN TUGAS KECIL 2**

**STRATEGI ALGORITMA 2020**

Oleh: Naufal Arfananda Ghifari/13518096 (K-3)

1. **Konsep Algoritma yang digunakan**
2. Strategi Algoritma
3. *Brute Force*

Strategi algoritma *Brute Force* yang saya lakukan untuk menyelesaikan permasalahan *Polynomial Multiplication* adalah sebagai berikut:

Terdapat dua polinom berderajat sama yakni P1(x) = dan P2(x) = . Maka P adalah polinom yang merupakan hasil perkalian P1 dan P2 yang berderajat 2n dengan P(x) = .

Dengan , untuk setiap k yang memenuhi persamaan 0 ≤ k ≤ 2n.

1. *Divide and Conquer*

Strategi algoritma *Divide and Conquer* yang saya lakukan untuk menyelesaikan permasalahan *Polinomial Multipication* adalah sebagai berikut:

Terdapat dua polinom berderajat sama yakni P1(x) = dan P2(x) = . Maka P adalah polinom yang merupakan hasil perkalian P1 dan P2 yang berderajat 2n dengan P(x) = .

Misalkan:

Sehingga

Lakukan pula pada P2(x) sehingga

Sehingga

Misalkan

Maka P(x) dapat disederhanakan menjadi

1. Kompleksitas
2. *Brute Force*

Pada algoritma *Brute Force* terdapat iterasi n kali pada setiap iterasi yang berjumlah n kali pula saat mengimplementasikan P(x) = .

Sehingga terdapat n2kali perkalian dan n2 kali penjumlahan. Maka dari itu, kompleksitasnya adalah O(n2).

1. *Divide and Conquer*

Pada algoritma *Divide and Conquer* kompleksitas dapat dihitung sebagai berikut

Misalkan n = 2h

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Maka diperoleh

dan

– 2

Sehingga

1. **Kode program**
2. Class Polinom

|  |
| --- |
| class Polinom {  public:  //Konstruktor  Polinom();  //Membuat polinom dengan derajat 0, nilai konstanta 0  Polinom(int d);  //Membuat polinom dengan derajat d, nilai koefesien dipilih acak dari bilangan bulat //pada rentang -100 hingga 100  Polinom(int d, int\* dat);  //Membuat polinom dengan derajat d dan data seluruh koefesien dat    //Copy Konstruktor  Polinom(const Polinom&);  //Destruktor  ~Polinom();    //Operator polinom  Polinom& operator=(const Polinom&);//Operator =, meng-copy polinom  Polinom operator+(const Polinom&);// Operator +, Menambahkan dua polinom  Polinom operator-(const Polinom&);// Operator -, Mengurangi polinom ini dengan //polinom masukan  Polinom operator\*(const Polinom&);// Operator \*, Mengalikan polinom secara //Brute Force    //Output  friend std::ostream& operator<<(std::ostream &os, const Polinom& P); //mencetak //polinom    protected:  int degree;// Merepresentasikan derajat polinom . >= 0  int\* data;// Merepresentasikan koefesien dan konstanta polinom dengan //koefesien x^i = data[i] |

1. Class PolinomDnC

|  |
| --- |
| class PolinomDnC : public Polinom {  public:  //Konstruktor  PolinomDnC();  //Membuat polinomdnc dengan derajat 0, nilai konstanta 0  PolinomDnC(int d);  //Membuat polinomdnc dengan derajat d, nilai koefesien dipilih acak dari bilangan //bulat pada rentang -100 hingga 100  PolinomDnC(int d, int\* dat);  //Membuat polinomdnc dengan derajat d dan data seluruh koefesien dat    //Copy Konstruktor  PolinomDnC(const PolinomDnC&);  //Operator  PolinomDnC& operator=(const PolinomDnC&);//Operator =, meng-copy //polinomdnc  PolinomDnC operator+(const PolinomDnC&);// Operator +, Menambahkan dua //polinomdnc  PolinomDnC operator-(const PolinomDnC&);// Operator -, Mengurangi polinomdnc //ini dengan polinomdnc masukan  PolinomDnC operator\*(const PolinomDnC&);// Operator \*, Mengalikan polinom //secara DIVIDE AND CONQUER    //Fungsi kawan  friend PolinomDnC Geser(const PolinomDnC& P, int n);//Mengembalikan //polinomdnc P yang sudah digeser pangkatnya sebanyak n  friend void splitP (const PolinomDnC& P, PolinomDnC\* P1, PolinomDnC\* P2); //Membagi polinomdnc menjadi dua bagian berderajat sama    }; |

1. Algoritma *Brute Force*

|  |
| --- |
| Polinom Polinom::operator\*(const Polinom& P){  int n\_degree = this->degree + P.degree;  int\* dat = new int[n\_degree+1];  for (int i =0; i<= n\_degree; ++i){  dat[i] = 0;  }  for(int i =0; i<=P.degree;++i){  for(int j=0; j<= this->degree;++j){  dat[i+j] += this->data[j] \* P.data[i];  }  }  return Polinom(n\_degree, dat);  } |

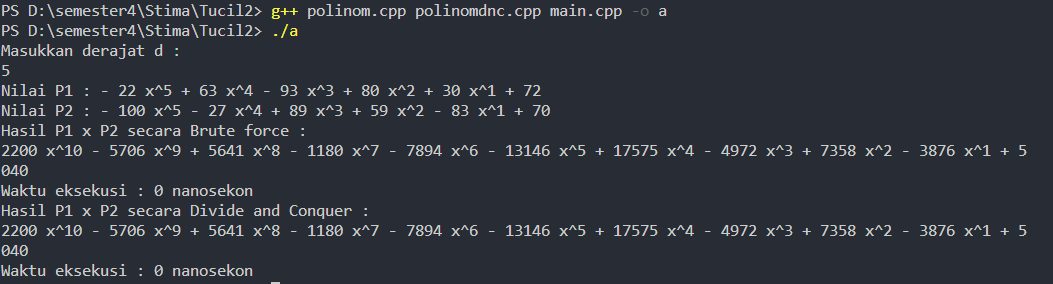
1. Algoritma *Divide and Conquer*

|  |
| --- |
| PolinomDnC PolinomDnC :: operator\*(const PolinomDnC& P){  //Exception  if (this ->degree != P.degree){  throw "Derajat polinom harus sama";  }  //basis  if (this->degree ==0){  int \* dat = new int[1];  dat[0]= this->data[0] \* P.data [0];  return PolinomDnC (0,dat);  }  else{  //rekurens  int div\_P = this->degree / 2 +1;  PolinomDnC A0,A1;  splitP(\*this, &A0, &A1);  PolinomDnC B0, B1;  splitP(P, &B0, &B1);  PolinomDnC Y = (A0+A1 ) \* (B0+B1);    PolinomDnC U = A0 \* B0;    PolinomDnC Z = A1 \* B1 ;    return (U + Geser(Y-Z-U, div\_P) + Geser(Z,div\_P+div\_P));  }  } |

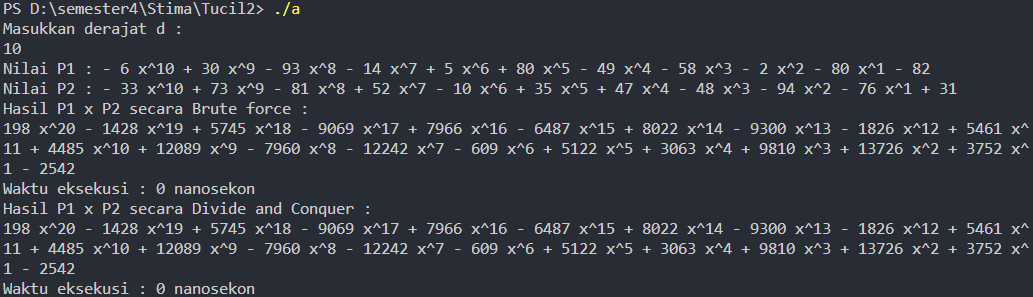
1. *Split* Polinom dan Geser Polinom

|  |
| --- |
| //Fungsi kawan  PolinomDnC Geser(const PolinomDnC& P, int x){  int n\_degree = P.degree + x;  int \*dat = new int[n\_degree+1];  for (int i=0; i< x;++i){  dat[i] = 0;  }  for(int i =0; i<=P.degree;++i){  dat[i+x] = P.data[i];  }  return PolinomDnC(n\_degree, dat);  }  void splitP (const PolinomDnC& P, PolinomDnC\* P1, PolinomDnC\* P2){  P1 -> degree = P.degree / 2;  P2 -> degree = P.degree /2;    P1 -> data = new int[P1-> degree +1];  P2 -> data = new int[P2 -> degree +1];  for (int i=0; i<= P1->degree;++i){  P1 -> data[i] = P.data[i];  }  for (int i=P1 ->degree+1; i<= P.degree;++i){  P2 -> data[i- P1 -> degree-1] = P.data[i];  }  if (P.degree%2 == 0){  P2 -> data [P.degree/2] = 0;  }  } |

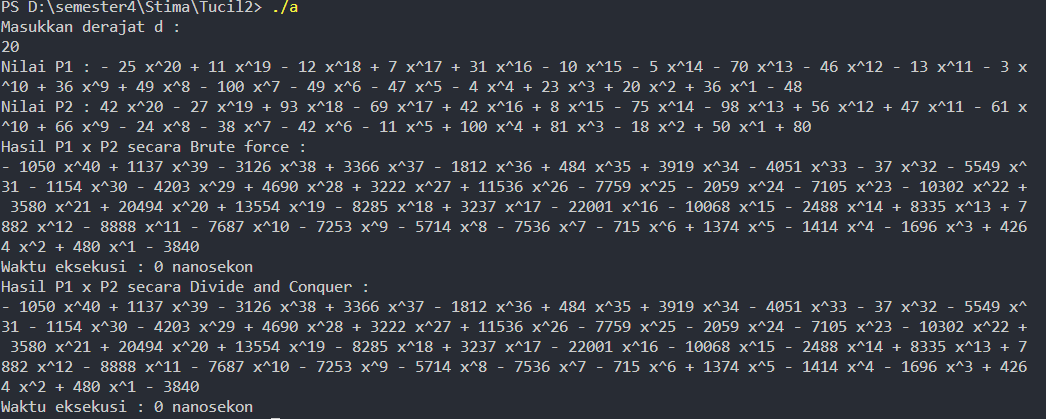
1. **Testing**
2. Derajat 5



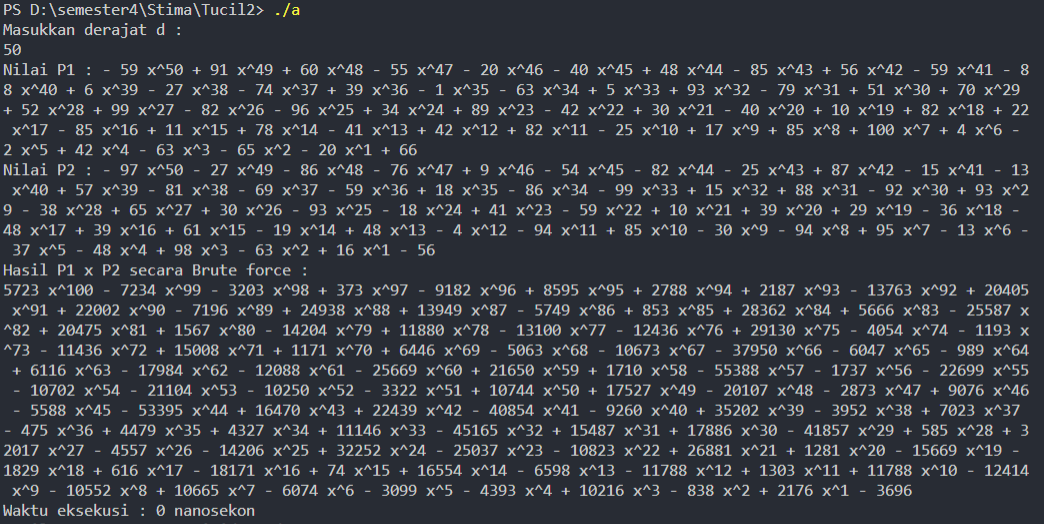
1. Derajat 10

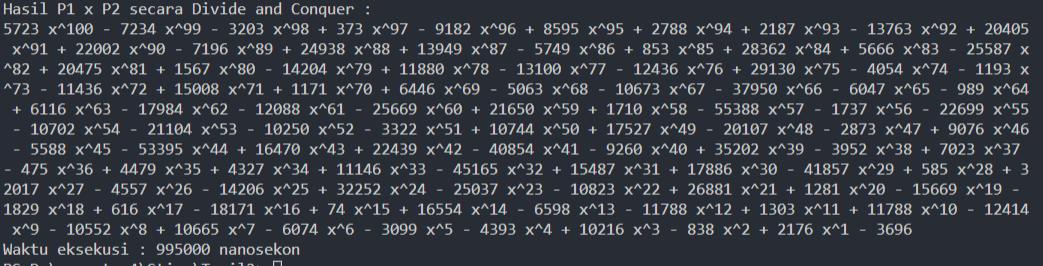


1. Derajat 20



1. Derajat 50

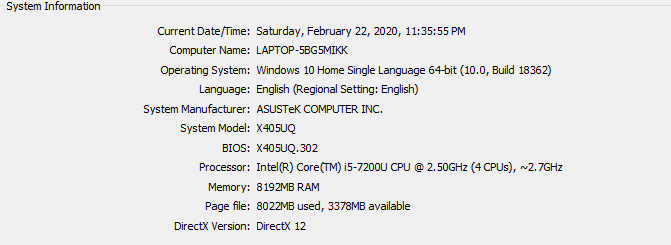




1. **List tambahan**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil running | V |  |
| 1. Program dapat menerima input dan menuliskan output | V |  |
| 1. Luaran sudah benar untuk semua n | V |  |

1. **Spesifikasi Laptop**



**Daftar pustaka**

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

<http://informatika.stei.itb.ac.id/~rinaldi.munir/>