**Tugas Kecil Strategi Algoritma 2**

**Perbandingan Perkalian Polinom Menggunakan Algoritma Brute Force dan Divide And Conquer**



Disusun Oleh:

Mario Gunawan(K03) 13518114

# BAB I DASAR TEORI

## Algoritma Brute Force

Algoritma Brute Force adalah algoritma sederhana, mudah dipikirkan, dan jelas untuk menyelesaikan suatu masalah tanpa memikirkan kecepatan algoritmanya. “Brute Force” berarti tenaga, dan algoritma ini sering dikaitkan dengan banyaknya tenaga yang dikeluarkan untuk menyelesaikan persoalan, tenaga yang dimaksud adalah tenaga computer melakukan kalkulasi, sehingga biasanya algoritma yang memiliki penyelesaian selain dengan brute force memiliki kecepatan komputasi yang lebih cepat daripada brute force.

Namun, ada banyak juga kelebihan brute force. Brute force biasanya digunakan sebagai acuan kecepatan komputasi. Selain itu, penyelesaian dengan cara brute force terkadang adalah satu satunya cara penyelesaian yang ada. Setelah melihat beberapa kompleksitas waktu algoritma lain, penulis juga dapat menyimpulkan algoritma brute force terkadang lebih cepat mengkomputasi daripada beberapa algoritma lain misalkan ukuran elemen yang dikomputasikan relatif rendah. Selain itu, algoritma brute force mudah dimengerti.

Kekurangan brute force adalah komputasi yang biasanya lebih lamban dibanding algoritma lain pada ukuran elemen yang lebih tinggi. Oleh karena komputasi yang lebih lamban, algoritma ini seringkali tidak efisien. Biasanya, algoritma brute force sering dikaitkan dengan “polos”nya seorang programmer, oleh karena itu lebih baik tidak digunakan saat *interview* dan semacamnya.

Contoh dari algoritma yang dapat diselesaikan dengan algoritma brute force dan merupakan algoritma termangkus untuk persoalan tersebut adalah mencari elemen N dalam sebuah senarai dan mencari nilai maksimum dalam senarai. Sementara, contoh algoritma yang bisa dengan lebih mangkus diselesaikan menggunakan algoritma lain adalah *sorting*, mencari sebuah substring dari string, dan *fractional knapsack problem*.

## Algoritma Divide and Conquer

A close up of a map

Description automatically generated

Gambar : Strategi Penyerangan Amerika di Pantai Omaha Melawan Prajurit Jerman

sumber: <https://upload.wikimedia.org/wikipedia/commons/thumb/6/65/Omaha_Beach_1944_Objectives.jpg/350px-Omaha_Beach_1944_Objectives.jpg>

Algoritma Divide and Conquer adalah algoritma yang lebih kompleks daripada algoritma brute force. Seperti strategi penyerangan, algoritma ini membagi permasalahan menjadi beberaba sub-permasalahan dan menyelesaikan sub permasalahan itu saat sudah cukup kecil. Biasanya, algoritma ini lebih cepat daripada brute force dalam penyelesaian suatu persoalan. Walaupun lebih cepat, algoritma ini menggunakan memory yang banyak karena cara penyelesaian divide and conquer merupakan cara penyelesaian rekursif.

Kelebihan divide and conquer adalah biasanya lebih cepat daripada algoritma brute force. Selain itu, algoritma ini menunjukan kekreatifan seorang programmer dalam memecahkan suatu permasalahan.

Kekurangan divide and conquer adalah algoritma ini lebih sulit untuk dibayangkan. Selain itu, memori yang digunakan divide and conquer tidaklah sedikit karena pemanggilan fungsi rekursif memerlukan memori untuk menyimpan data dari setiap pemanggilan fungsi.

Contoh Persoalan yang lebih cepat dikerjakan menggunakan algoritma divide and conquer dibandingkan algoritma brute force adalah persoalan *sorting* dan mencari dua titik terdekat dalam *cartesian plane*.

# Kode Program

Dalam tugas ini, bahasa yang saya gunakan adalah bahasa c++. Saya membagi file ke dalam 3 file, yaitu Polinom.cpp, Polinom.hpp, dan tucil2stima.cpp. File Polinom.hpp berfungsi sebagai *header file* yang memuat detail kelas Polinom yang akan diimplementasikan pada Polinom.cpp. Polinom.cpp berisi implementasi dari Polinom.hpp. tucil2stima.cpp berisi program utama yang bisa di-*compile* untuk membuat file executable yang bisa dijalankan. Selain itu, ada file tambahan run.bat yang bisa langsung dijalankan untuk meng-*compile* file tucil2stima.cpp dan menjalankannya di OS windows.

## Polinom.hpp

RANDOMIZE <- true

NOT\_RANDOMIZE <- false

MAX\_KOEF\_RANDOM <- 100

class Polinom

private:

max\_derajat : integer

koef : Array of integer

jumlah\_polinom : static integer

public:

Polinom kaliBruteForce(input Polinom other)

friend Polinom kaliDivideAndConquer(input Polinom A,B, integer n)

Ada 3 variabel private dalam kelas Polinom, yaitu koef yang merupakan array of koefisien, max\_derajat yang adalah integer, dan jumlah polinom yang merupakan variabel kelas bertipe integer. Ada banyak method kelas dalam Polinom.hpp, namun fungsi utamanya adalah kaliBruteForce, dan kaliDivideAndConquer.

Constructor dari kelas polinom ada 2, yaitu default dan dengan parameter. Constructor default menetapkan max\_derajat sebanyak 10, dan menetapkan besar senarai koef sebesasr 11(senarai dimulai dari 0) dan menambahkan jumlah polinom sebanyak 1. Constructor dengan parameter menerima 2 argumen, yaitu max\_derajat dan boolean isRandom. Bila isRandom adalah true, maka seluruh elemen di senarai diinisialisasikan dengan angka yang random, bila tidak diinisialisasikan dengan angka 0.

Selain itu ada method setElement(input index, value) dan addElement( input index, value) yang memasukan nilai value di koef[index] dan menambahkan value di koef[index] secara berurutan.

## Polinom.cpp

### Brute Force

Polinom Polinom::kaliBruteForce(input Polinom other)

Polinom res(other.getMaxDerajat(), NOT\_RANDOMIZE);

for(int i = 0 ; i <= this->max\_derajat ; i++) {

for(int j = 0 ; j <= other.max\_derajat ; j++) {

Value = this->koef[i] \* other.koef[j];

res.addElement(i+j, Value)

return res;

Method brute force dipanggil dengan cara Polinom1.kaliBruteForce(Polinom2). Fungsi ini mengalikan polinom asal dengan polinom lainnya dengan metode brute force. Algoritma ini bekerja dengan cara mengalikan satu per satu elemen di polinom pertama dengan elemen di polinom kedua, dan di setiap pengaliannya, jumlahkan pangkat elemen pertama dengan kedua, lalu tambahkan elemen di hasil[index elemen1 + index elemen2] dengan hasil perkalian.

### Divide And Conquer

Polinom kaliDivideAndConquer(Polinom &Polinom1, Polinom &Polinom2, int n) {

Polinom produk(2\*(n-1), NOT\_RANDOMIZE);

if(n == 1) {

Polinom temp(0, NOT\_RANDOMIZE);

temp.setElement(0, Polinom1.getElement(0) \* Polinom2.getElement(0));

return temp;

}

Diatas merupakan basis dari perkalian divide and conquer. Bila n atau size adalah 1, maka return Polinom yang berisi Polinom1[0] \* Polinom2[0]. Bila tidak:

Inti dari code diatas adalah, membagi Polinom 1 dan Polinom 2 ke menjadi masing masing 2 bagian, lalu menyelesaikannya secara rekursif. Sampai akhirnya didapat basis( elemen = 1), dan menambahkannya ke dalam Polinom produk lalu mengembalikannya.

else

int setengahSize = n / 2;

int highSize = n % 2 == 1 ? setengahSize : setengahSize - 1;

Polinom A1(highSize, NOT\_RANDOMIZE);

Polinom B1(highSize, NOT\_RANDOMIZE);

Polinom A0(n - (highSize + 1), NOT\_RANDOMIZE);

Polinom B0(n - (highSize + 1), NOT\_RANDOMIZE);

for(int i = 0 ; i <= highSize ; i++)

A1.setElement(i, Polinom1.getElement(i + setengahSize))

B1.setElement(i, Polinom2.getElement(i + setengahSize))

for(int i = 0; i < setengahSize ; i++)

A0.setElement(i, Polinom1.getElement(i))

B0.setElement(i, Polinom2.getElement(i))

Polinom A0plusA1(highSize, NOT\_RANDOMIZE)

Polinom B0plusB1(highSize, NOT\_RANDOMIZE)

for(int i = 0 ; i <= setengahSize ; i++)

A0plusA1.setElement(i, A0.getElement(i) + A1.getElement(i))

B0plusB1.setElement(i, B0.getElement(i) + B1.getElement(i))

Polinom A0B0 = kaliDivideAndConquer(A0, B0, (highSize + 1));

Polinom produkTengah = kaliDivideAndConquer(A0plusA1, B0plusB1, (A0plusA1.getMaxDerajat() + 1));

Polinom A1B1 = kaliDivideAndConquer(A1, B1, (highSize + 1));

for(int i = 0; i < n-1 ; i++) {

produk.addElement(i, A0B0.getElement(i));

produk.addElement((i + setengahSize), produkTengah.getElement(i) - A0B0.getElement(i) - A1B1.getElement(i))

for(int i = 0; i < (A1B1.getMaxDerajat() + 1) ; i++) {

produk.addElement(i+(2\*setengahSize), A1B1.getElement(i))

return produk

# Input Output

## Tampilan Awal Program

A screenshot of a cell phone

Description automatically generated

## Polinom derajat 5

Polinom: A close up of a screen

Description automatically generated

Brute Force:

A close up of a screen

Description automatically generated

Divide And Conquer

A close up of a logo

Description automatically generated

## Polinom derajat 10

Polinom:

Brute Force:

A close up of a screen

Description automatically generated

Divide And Conquer:

A black sign with white text

Description automatically generated

## Polinom derajat 20

Polinom:A black sign with white text

Description automatically generated

Brute Force:

A white sign with black text

Description automatically generated

Divide And Conquer:

A picture containing text

Description automatically generated

## Polinom Derajat 50

Polinom:A black and white photo

Description automatically generated

Brute Force:

A picture containing text

Description automatically generated

Divide And Conquer:

A black and silver text on a white background

Description automatically generated

Pada hasil pengujian diatas, kecepatan algoritma brute force terlihat lebih cepat daripada divide and conquer, berbeda dengan hipotesis awal. Menurut penulis, ini dikarenakan pada algoritma divide and conquer, setiap senarai harus diinisiasi dengan 0 ,sementara pada algoritma brute force setiap senarai tidak harus diinisiasi, oleh karena itu , jumlah looping yang ada pada program meningkat drastis pada algoritma divide and conquer.

Perbedaan waktu lebih jelas terllihat ketika penulis mencoba memasukkan senarai dengan derajat 1000 dan 4000, brute force masih lebih cepat sekitar 3 kali lipat.