Penerepan Algoritma Divide and Conquer untuk Perkalian Bilangan Besar

Made Edwin Wira Putra - 13508010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If18010@students.if.itb.ac.id

***Abstraksi*— Algoritma kriptografi publik (asimetrik) membutuhkan perhitungan bilangan besar yang cukup cepat, penggunaan memori yang kecil, dan membutuhkan konsumsi daya yang cukup rendah, serta kebal terhadap side chanel attack. Untuk itu diperlukan sebuah cara untuk mempercepat perhitungan perkalian modular sebagai dasar perhitungan eksponensiasi modular. Teknik perkalian modulus biasanya membutuhkan beberapa perhitungan dasar di antaranya representasi radix, pengurangan, penambahan, perbandingan, perkalian, dan pembagian.**

***Kata Kunci*—Bilangan Besar, Divide and Conquer, Large Integer**

I. Pendahuluan

Algoritma kriptografi publik (asimetrik) membutuhkan perhitungan bilangan besar yang cukup cepat, penggunaan memori yang kecil, dan membutuhkan konsumsi daya yang cukup rendah, serta kebal terhadap side chanel attack. Untuk itu diperlukan sebuah cara untuk mempercepat perhitungan perkalian modular sebagai dasar perhitungan eksponensiasi modular. Teknik perkalian modulus biasanya membutuhkan beberapa perhitungan dasar di antaranya representasi radix, pengurangan, penambahan, perbandingan, perkalian, dan pembagian.

Berikut ini akan dibahas penerapan algoritma Divide and Conquer sebagai salah satu solusi untuk perhitungan perkalian bilangan besar. Sub-bahasan dalam paper ini antara lain kontribusi, konsep, algoritma divide and conquer, implementasi ke program, hasil pengujian, dan kesimpulan.

II. Kontribusi

Pada makalah ini akan dijelaskan implementasi algoritma Divide and Conquer secara langsung dengan menggunakan bahasa C++ untuk perhitungan perkalian bilangan besar. Melalui pengujian akan dianlisa hasil dan waktu penyelesaiannya.

III. Konsep Perhitungan dan Algoritma Divide and Conquer

A. Representasi Bilangan Besar

Jalan termudah untuk merepresentasikan sebuah bilangan besar adalah denganmenggunakan array bilangan dimana setiap slot array menyimpan satu digit. Sebagai contoh, bilangan 543127 dapat direpresentasikan ke sebuah array *S* sebagai berikut :

5	4	3	1	2	7
S[6]	*S*[5]	*S*[4]	*S*[3]	*S*[2]	*S*[1]

Untuk merepresentasikan bilangan positif dan negatif, hanya perlu menyediakan slot array urutan teratas sebagai tanda. Bisa menggunakan 0 pada slot tersebut untuk merepresentasikan bilangan positif dan 1 untuk merepresentasikan bilangan negatif. Akan diasumsikan representasi ini dan menggunakan

```
type bilangan_besar
```

sebagai array yang cukup besar untuk merepresentasikan bilangan pada aplikasi yang diinginkan.

Tidak begitu sulit untuk menulis algoritma waktu linier untuk penjumlahan dan pengurangan, dimana *n* adalah jumlah digit bilangan besar.

B. Perkalian Bilangan Besar

Sebuah algoritma waktu kuadrat sederhana untuk mengalikan bilangan besar salah satunya adalah dengan cara yang diajarkan ketika masa sekolah dasar. Algoritma yang lebih baik dari itu menggunakan divide and conquer untuk membagi sebuah *n*-digit bilangan menjadi dua buah bilangan berkisar *n*/2 digit. Berikut merupakan contoh pembagian bilangan.

$$567832 = 567 \times 10^3 + 832$$

6 digit 3 digit 3 digit

Secara umum, jika *n* adalah jumlah digit bilangan *u*, akan dibagi bilangan tersebut menjadi dua bilangan, satu $[n/2]$ dan satunya lagi $[n/2]$ sebagai berikut :

$$u = x \times 10^m + y$$

n digit $[n/2]$ digit $[n/2]$ digit

Dengan representasi ini, diberikan 10 pangkat *m* dimana

$$m = \left\lceil \frac{n}{2} \right\rceil$$

Jika memiliki dua *n*-digit bilangan

$$u = x \times 10^m + y$$

$$v = w \times 10^m + z$$

hasilnya adalah sebagai berikut

$$uv = (x \times 10^m + y)(w \times 10^m + z)$$

$$= xw \times 10^{2m} + (xz + wy) \times 10^m + yz$$

Kita dapat mengalikan *u* dan *v* dengan melakukan empat perkalian pada bilangan dengan sekitar setengah banyaknya digit dan melakukan operasi linier.

C. Algoritma

```
Problem : Perkalian 2 buah bilangan, u dan v
Inputs : Bilangan besar u dan v
Outputs : Hasil perkalian u dan v

   function multiply(u,v : bilangan_besar):bilangan_besar
   var
      x,y,w,z,p,q : bilangan_besar;
   begin
      n:=maximum(jumlah digit u, jumlah digit v);
      if u=0 or v=0 then
          multiply := 0;
      else if n<=titik_ambang then
         multiply := u x v; /*perkalian biasa*/
      else
          m := n divide 2;
          x := u divide 10^m;
          y := u mod 10^m;
          w := u divide 10^m;
          z := v mod 10^m;
          r := multiply(x+y,w+z);
          p := multiply(x,w);
          q := multiply(y,z);
          multiply := p x 10^2m + (r-p-q) x 10^m + q;
      end if
   end
```

IV. Implementasi Divide and Conquer dalam Bahasa C++

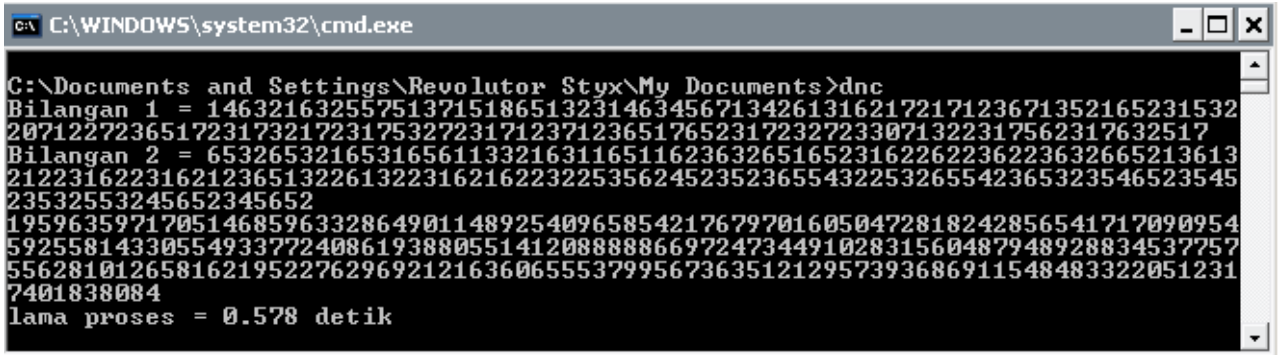
```
#include<iostream>
#include<string>
#include<time.h>
#define THRESHOLD 1

using namespace std;

struct bilangan_besar{
     int number[256];
     int nDigit;
     bool isZero;
};

void printBilangan(bilangan_besar n){
     int a ;

     for (a=n.nDigit-1;a>=0;a--){
           cout << n.number[a];
     }
     cout << endl;
}
```

```
int maximum(int nDigit1,int nDigit2){
     if (nDigit1>nDigit2) {
           return nDigit1;
     }
     return nDigit2;
}

void setInisiasi(bilangan_besar &bil){
     int a;

     for (a=0;a<=255;a++){
           bil.number[a] = 0;
     }
     bil.nDigit = 0;
     bil.isZero = true;
}

bilangan_besar kaliPangkat10(bilangan_besar n,int m){
     int a;

     for (a=n.nDigit-1;a>=0;a--){
           n.number[a+m]=n.number[a];
     }
     for (a=0;a<=m-1;a++){
           n.number[a]=0;
     }
     n.nDigit += m;
     return n;
}

bilangan_besar jumlah(bilangan_besar n1, bilangan_besar n2){
     bilangan_besar hasil;
     setInisiasi(hasil);
     int a;
     for (a=0;a<maximum(n1.nDigit,n2.nDigit);a++){
           hasil.number[a]=hasil.number[a]+n1.number[a]+n2.number[a];
           if (hasil.number[a]>9){
                 hasil.number[a+1]=1;
                 hasil.number[a]=hasil.number[a]-10;
           }
           hasil.nDigit++;
     }
     if (hasil.number[hasil.nDigit]>0){
           hasil.nDigit++;
     }
     return hasil;
}

bilangan_besar kurang(bilangan_besar n1, bilangan_besar n2){
     bilangan_besar hasil;

     if ((n1.nDigit<n2.nDigit) or (n1.nDigit==n2.nDigit && n1.number[n1.nDigit-
1]<n2.number[n2.nDigit-1])){
           hasil = kurang(n2,n1);
     }
     else {//n1 >= n2
           setInisiasi(hasil);
```

```
            int a;
            for (a=0;a<n1.nDigit;a++){
                  if (n1.number[a]<0){
                        n1.number[a] = n1.number[a] * - 1;
                        n1.number[a+1]--;
                  }
                  if (n1.number[a]>=n2.number[a]){
                        hasil.number[a]=n1.number[a]-n2.number[a];
                  }
                  else {
                        hasil.number[a]=n1.number[a]+10-n2.number[a];
                        n1.number[a+1]--;
                  }
                  hasil.nDigit++;
            }
            if (hasil.number[hasil.nDigit-1]==0){
                  hasil.nDigit--;
            }
      }
      if (hasil.nDigit==0 or (hasil.nDigit==1 && hasil.number[0]==0)){
            hasil.isZero = true;
      }
      return hasil;
  }

  bilangan_besar multiply(bilangan_besar u, bilangan_besar v){
      bilangan_besar r,x,y,w,z,p,q;
      bilangan_besar hasil;
      int n;

      n = maximum(u.nDigit,v.nDigit);
      if (((u.number[0]==0 && u.nDigit==1) or (u.nDigit==0)) or ((v.number[0]==0
&& v.nDigit==1) or (v.nDigit==0))){ // u atau v = 0
            setInisiasi(hasil);
      }
      else if (n<=THRESHOLD) {      //jumlah digit maksimum = 1
            setInisiasi(hasil);
            int temp = u.number[0] * v.number[0];
            if (temp<10){
                  hasil.number[0]=temp;
                  hasil.nDigit++;
            }
            else {
                  do {
                        hasil.number[hasil.nDigit] = temp % 10;
                        hasil.nDigit++;
                        temp = temp / 10;
                  } while (temp>=10);
                  hasil.number[hasil.nDigit] = temp;
                  hasil.nDigit++;
            }
      }
      else {
            int m = n/2;
            int a;
            setInisiasi(y);
            setInisiasi(z);
            setInisiasi(x);
```

```
            setInisiasi(w);
            //x = u divide 10^m
            for (a=m;a<u.nDigit;a++){
                  x.number[x.nDigit]=u.number[a];
                  x.nDigit++;
            }
            x.isZero = false;
            //y = u mod 10^m
            for (a=0;a<m;a++){
                  y.number[y.nDigit]=u.number[a];
                  y.nDigit++;
            }
            y.isZero = false;
            //w = v divide 10^m
            for (a=m;a<v.nDigit;a++){
                  w.number[w.nDigit]=v.number[a];
                  w.nDigit++;
            }
            w.isZero = false;
            //z = v mod 10^m
            for (a=0;a<m;a++){
                  z.number[z.nDigit]=v.number[a];
                  z.nDigit++;
            }
            z.isZero = false;
            //r = multiply(x+y,w+z)
            r = multiply(jumlah(x,y),jumlah(w,z));

            p = multiply(x,w);

            q = multiply(y,z);

            //hasil = p x 10^m + (r-p-q) x 10^m + q
            hasil                =                jumlah(jumlah(kaliPangkat10(p,
(2*m)),kaliPangkat10(kurang(kurang(r,p),q),m)),q);
      }
      return hasil;
  }

  int main(){
      string c;
      int a;
      bilangan_besar x,y;

      cout << "Bilangan 1 = ";
      setInisiasi(x);
      cin >> c;
      for (a=c.length()-1;a>=0;a--){
            x.number[x.nDigit]=(int)c[a] - 48;
            x.nDigit++;
      }
      cout << "Bilangan 2 = ";
      setInisiasi(y);
      cin >> c;
      for (a=c.length()-1;a>=0;a--){
            y.number[y.nDigit]=(int)c[a] - 48;
            y.nDigit++;
      }
      x.isZero = false;
```

V. Hasil Uji



VI. Kesimpulan

Penerapan Algoritma Divide and Conquer dalam menyelesaikan perhitungan perkalian bilangan besar memang tidak mudah, namun memberi hasil yang berarti. Proses perhitungan dilakukan dengan cepat, bisa dibuktikan melalui hasil tes uji di atas. Implementasi algoritma Divide and Conquer di atas sendiri masih bukan merupakan algoritma yang optimal dari segi source codenya. Dengan meng-optimasikan source code bahasa C++ di atas, performa penyelesaian perhitungan diyakini bisa lebih cepat.

Referensi

[1] Ariadi, Wildan.*Perbandingan Metode Perkalian dalam Perhitungan Bilangan Besar*.2009.

[2] Munir, Rinaldi.*Diktat Kuliah IF3051 : Strategi Algoritma*, 2009

[3] Levitin, Introduction to the Design & Analysis of Algorithms, Addison-Weasley, 2003.

[4] Horowitz, Ellis, & Sartaj Sahni, *Fundamental of Computer Algorithm*, Pitman Publishing Limited, 1978.

[5] Neapolitan, Richard, *Foundation of Algorithm*, D.C. Health and Company, 1996

[6] http://bytes.com/topic/c/answers/497338-c-get-time-milliseconds tanggal akses : 9 Desember 2010

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2010

ttd

Made Edwin Wira Putra / 13508010