**Tugas Analisis Algoritma 6BILPIL**

**Kelompok:**
Ahmad Emir Alfatah 09021381722130
Jessica Julia Paradina Siregar 09021281722032

**Algoritma Perkalian Matriks Strassen**

Algoritma strassen merupakan sebuah algoritma yang digunakan untuk perkalian matriks yang lebih cepat dan berguna untuk matriks berukuran besar. Algoritma ini dibuat oleh Volker Strassen.
Kompleksitas waktu algoritma perkalian Strassen:

$$T(n) = O(n^{\log 7}) = O(n^{2.81})$$

Perkalian dua buah bilangan bulat yang besar dapat diselesaikan dengan beberapa cara:

1. Brute force
    a. Kompleksitas algoritma : O(n^2)

    b. Pseudo-code

```
function Kali1(input X, Y : LongInteger, n : integer) → LongInteger
{ Mengalikan X dan Y, masing-masing panjangnya n digit dengan algoritma
brute force.
  Masukan: X dan Y yang panjangnya n angka
  Keluaran: hasil perkalian
}
Deklarasi
   temp, AngkaSatuan, AngkaPuluhan : integer

Algoritma:
   for setiap angka yi dari yn, yn-1, …, y1 do
      AngkaPuluhan ← 0
      for setiap angka xj dari xn, xn-1, …, x1 do
       temp ← xj * yi
       temp ← temp + AngkaPuluhan
       AngkaSatuan ← temp mod 10
       AngkaPuluhan ← temp div 10
       tuliskan AngkaSatuan
      endfor
   endfor
   Z ← Jumlahkan semua hasil perkalian dari atas ke bawah
   return Z
```

    c. Summary
    Perkalian dua buah bilangan bulat menggunakan algoritma brute force akan mengalami 2 dimensi perulangan, perulangan dilakukan dengan mengambil digit angka Y dari digit terakhir ke awal pada perulangan dimensi pertama kemudian digit Yi dikalikan didalam perulangan dimensi kedua dengan seluruh digit angka X sehingga mendapat hasil perkalian setiap digit Y yang akan dijumlahkan hasilnya menjadi Z. Algoritma ini sangat

tidak mangkus bila jumlah digitnya besar karena akan mengalami perulangan yang besar sebanyak n^2.

2. Divide and Conquer
    a. Kompleksitas algoritma : O(n^2)

    b. Pseudo-code :

```
function Kali2(input X, Y : LongInteger, n : integer) → LongInteger
{ Mengalikan X dan Y, masing-masing panjangnya n digit dengan algoritma
Divide and Conquer.
  Masukan: X dan Y
  Keluaran: hasil perkalian X dan Y
}
Deklarasi
    a, b, c, d : LongInteger
    s : integer

Algoritma:
    if n = 1 then
        return X * Y     { perkalian biasa }
    else
        s←n div 2         { bagidua pada posisi s }
        a←X div 10^s
        b←X mod 10^s
        c← Y div 10^s
        d← Y mod 10^s
        return Kali2(a, c, s)*10^2s + Kali2(b, c, s)*10^s +
               Kali2(a, d, s)*10^s + Kali2(b, d, s)
  endif
```

    c. Summary

    Perkalian dua buah bilangan bulat menggunakan algoritma divide and conquer akan mengalami rekursif. Fungsi akan melakukan pengecekan jika rekursif telah selesai atau belum, jika belum akan dilakukan perhitungan panjang s dengan membagi 2 panjang digit angka dan melakukan proses pembagian dan pengambilan hasil bagi dengan 10^s untuk variabel a, b, c, dan d (a dan b merepresentasikan angka X, c dan d merepresentasikan angka Y). Jika variabel telah didapatkan maka akan dipanggil kembali fungsi rekursif tersebut sampai n = 1. Algoritma ini lebih baik dari pada brute force tetapi kompleksitasnya tetap O(n^2) sehingga belum memperbaiki kompleksitas algoritma perkalian brute force dengan signifikan.

3. Karatsuba
    a. Kompleksitas algoritma : O(n^log 3)

    b. Pseudo-code :

```
function Kali3(input X, Y : LongInteger, n : integer) → LongInteger
{ Mengalikan X dan Y, masing-masing panjangnya n digit dengan algoritma
Divide and Conquer.
  Masukan: X dan Y
  Keluaran: hasil perkalian X dan Y
}
Deklarasi
   a, b, c, d : LongInteger
   s : integer

Algoritma:
   if n = 1 then
      return X * Y     { perkalian biasa }
   else
      s←n div 2        { bagidua pada posisi s }
      a←X div 10^s
      b←X mod 10^s
      c← Y div 10^s
      d← Y mod 10^s
      p←Kali3(a, c, s)
      q←Kali3(b, d, s)
      r←Kali3(a + b, c + d, s)
      return p*10^2s + (r - p - q)*10^s + q

   endif
```

    c. Summary

    Perkalian dua buah bilangan bulat menggunakan algoritma divide and conquer dengan metode karatsuba akan mengalami rekursif, proses nya hampir sama dengan algoritma divide and conquer tetapi ada sedikit modifikasi pada proses return. Fungsi akan melakukan pengecekan jika rekursif telah selesai atau belum, jika belum akan dilakukan perhitungan panjang s dengan membagi 2 panjang digit angka dan melakukan proses pembagian dan pengambilan hasil bagi dengan 10^s untuk variabel a, b, c, dan d (a dan b merepresentasikan angka X, c dan d merepresentasikan angka Y). Jika variabel telah didapatkan maka akan dipanggil kembali fungsi rekursif tersebut pada variabel p, q, dan r dan akan di return dengan $p*10^{2s} + (r - p - q)*10^{s} + q$ sampai n = 1. Algoritma ini lebih baik daripada 2 algoritma sebelumnya dengan kompleksitas O(n^log 3) yang dapat memberikan proses yang lebih signifikan daripada algoritma sebelumnya.

**Polynomial Multiplication Problem**

1. Direct Brute Force

   Jika

   $$A(x) = \sum_{i=0}^{n} a_i x^i \qquad B(x) = \sum_{i=0}^{n} b_i x^i$$

   Maka

   $$\text{Set } C(x) = \sum_{k=0}^{2n} c_i x^i = A(x)B(x) \text{ with}$$

   $$c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

   $$\text{for all } 0 \leq k \leq 2n.$$

   Pada direct approach ini jumlah total perkalian dan penjumlahan yang dibutuhkan ⊖(n^2) sehingga kompleksitasnya adalah ⊖(n^2).

2. Divide and Conquer

   Dengan adanya improved divide and conquer

   PolyMulti2(A(x), B(x))

   **begin**
   $A_0(x) = a_0 + a_1 x + \cdots + a_{\frac{n}{2}-1} x^{\frac{n}{2}-1}$;
   $A_1(x) = a_{\frac{n}{2}} + a_{\frac{n}{2}+1} x + \cdots + a_n x^{n-\frac{n}{2}}$;
   $B_0(x) = b_0 + b_1 x + \cdots + b_{\frac{n}{2}-1} x^{\frac{n}{2}-1}$;
   $B_1(x) = b_{\frac{n}{2}} + b_{\frac{n}{2}+1} x + \cdots + b_n x^{n-\frac{n}{2}}$;
   $Y(x) = PolyMulti2(A_0(x) + A_1(x), B_0(x) + B_1(x))$;
   $U(x) = PolyMulti2(A_0(x), B_0(x))$;
   $Z(x) = PolyMulti2(A_1(x), B_1(x))$;
   **return** $(U(x) + [Y(x) - U(x) - Z(x)]x^{\frac{n}{2}} + Z(x)x^{2\frac{n}{2}})$
   **end**

   Maka kompleksitas yang didapatkan adalah ⊖(n^log2 3).

   Improved divide and conquer lebih baik daripada brute force dan divide and conquer yang memiliki kompleksitas ⊖(n^2).

3. Conclusion
   Pendekatan algoritma divide and conquer tidak selalu memberikan solusi terbaik, bahkan bisa jadi sama buruknya seperti algoritma brute force.