Report for assignment 1

Shashwat Gupta (14IE10028)

January 13, 2016

Multiplication of two big integers using FFT and IFFT

1. **Implementing FFT/IFFT**

   FFT of a polynomial is found by evaluating the polynomial at n^{th} roots of unity. We solve FFT recursively using divide and conquer by separating the odd and even coefficients, and obtain a time complexity of *O(nlogn)*. We use Eulers formulas to compute the n^{th} roots of unity. We can derive IFFT also in a similar way.

2. **Generation Random Big Integers**

   Big integers are stored using data structures, dynamically allocated with random numbers. Each number is represented as complex number of the form **a + ib** implemented using structures A complex data structure is made which hold two contents: real part *r* and imaginary part *im*, both of which are of *double* datatype.

   At the runtime, user is asked to input number of digits, for two numbers and the big integer array is created using *create* and it is first allocated space and then it is populated with random numbers (0-9 only), while keeping in mind that Most Significant Digit should not be zero.

3. **Multiplication algorithm**

   Let integer A and B be represented as

   $$A = \sum_{k=0}^{k=n-1} a_k x^k \tag{1}$$

   and

   $$B = \sum_{k=0}^{k=n-1} b_k x^k \tag{2}$$

   Let applying FFT for order 2n yield the respective fourier transforms as F(A) and F(B) defined by

$$F(A) = [A_0, A_1, A_2, ..., A_{2n-1}] \tag{3}$$

where

$$A_i = \sum_{k=0}^{k=2n-1} a_k w_{2n}^{ik} \tag{4}$$

$$F(B) = [B_0, B_1, B_2, ..., B_{2n-1}] \tag{5}$$

where

$$B_i = \sum_{k=0}^{k=2n-1} b_k w_{2n}^{ik} \tag{6}$$

where

$$w_{2n}^k = e^(\frac{2.pi.k}{2n}) \tag{7}$$

$$a_k = b_k = 0 for k > n - 1 \tag{8}$$

Let C be the product of A and B i.e. **C = AB**

Then fourier transform of C is given by

$$F(C) = [C_0, C_1, C_2, ..., C_{2n-1}] \tag{9}$$

where

$$C_i = A_i * B_i \tag{10}$$

The integer C is therefore obtained by taking IFFT of F(C).

4. **Time Complexity Calculation**

   In FFT, for each value of n, two recursive calls of length $\frac{n}{2}$ is performed and rest of combining operation is performed in linear time therefore

$$T(n) = \begin{cases} a & : n = 1 \\ 2T(\frac{n}{2}) + bn + c & : n > 1 \end{cases}$$

   Solving this equation by recursive method yields T(n) $\in \Theta(n.lg(n))$.