实验2

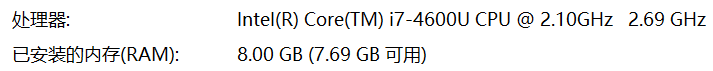
## 实验要求

实现FFT算法，对n的取值分别为4、16、32、60(注意当n取值不为2的整数幂时的处理方法)，随机生成2n个实数值（a0、a1、…、an-1）和（b0、b1、…、bn-1）分别作为多项式A(x)和B(x)的系数向量，使用FFT计算多项式A(x)与多项式B(x)的乘积，统计算法运行所需时间 ，与普通乘法进行比较，画出时间曲线。

## 实验环境

编译环境：Code::Blocks 16.01 GNU GCC Compiler

机器配置：



## 实验过程

1. 生成随机数据，存放在input/input.txt文件中。值得注意的是，由于所要求的数据是实数，故分别随机生成-100~100的整数部分和0~99的小数部分。使用fprintf函数来将生成的数字存放到文件中。

2. 按照所要求的输入数据从input.txt中读取数据。分别存在在数组a和b中。数组a、b的长度均为2\*n而不是n，将多出的部分全部填0。

3. 开始计时

4. 对a和b进行RECURSIVDE-FFT，将输出的DFT存放在ya和yb数组中。

5. 将ya和yb进行点乘，并将结果存放于数组yc中。

6. 对yc进行REVERSE-FFT，得到DFT-1(yc)，存在数组c中.

7. 停止计时。

8. 将数组c的数据存在result.txt中，将时间存在time.txt中，由于不同输入规模存放于一个文件，fopen函数选择参数”a”。

## 程序设计思路

本实验算法如课本所述，此处不再赘述。

需要注意的是两点：

1.

复数的表示：此处定义一个结构体Complex，内有两个元素r和i，分别表示复数的实数部分和小数部分；

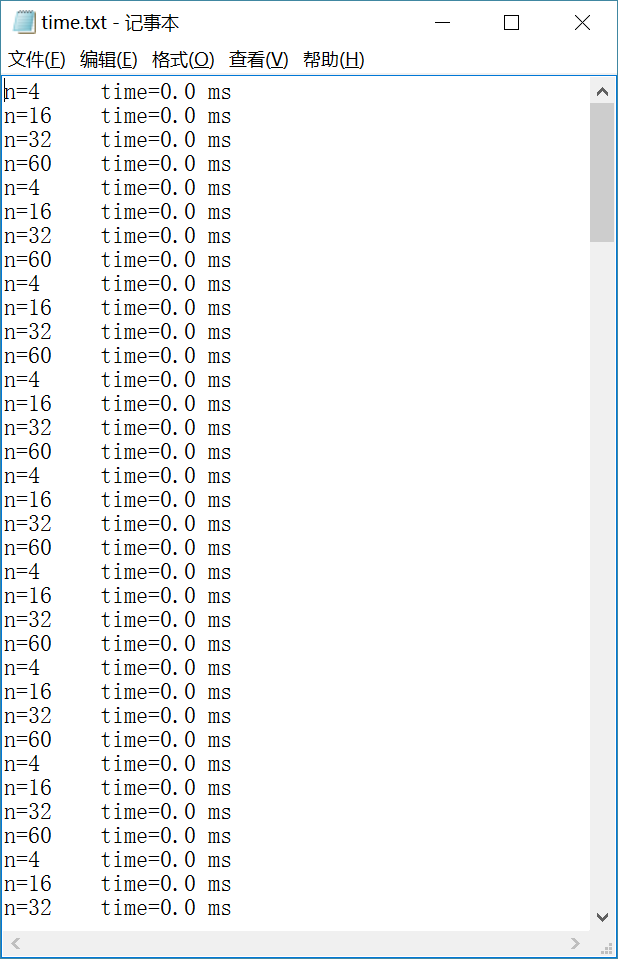
复数的乘法：result = a×b，即result.r = a.r \* b.r – a.i\*b.i; result.i = a.i\*b.r +a.r\*b.i;

2. 若给数据不是2的整数次幂，此处读取输入数据大小n后，先检查n是否是2的整数次幂，若不是，寻找比n大的，最小的2的整数次幂，作为新的输入数据长度，多出来的部分使用0来补充而不是读取更多数据。

此处使用课本定理30.8（卷积定理）来计算A(x)\*B(x)。即先计算出a和b的DFT后，将DFT(a)和DFT(B)进行点乘，然后再求结果的逆DFT。此处求DFT-1和DFT使用Recursive FFT。

## 结果分析

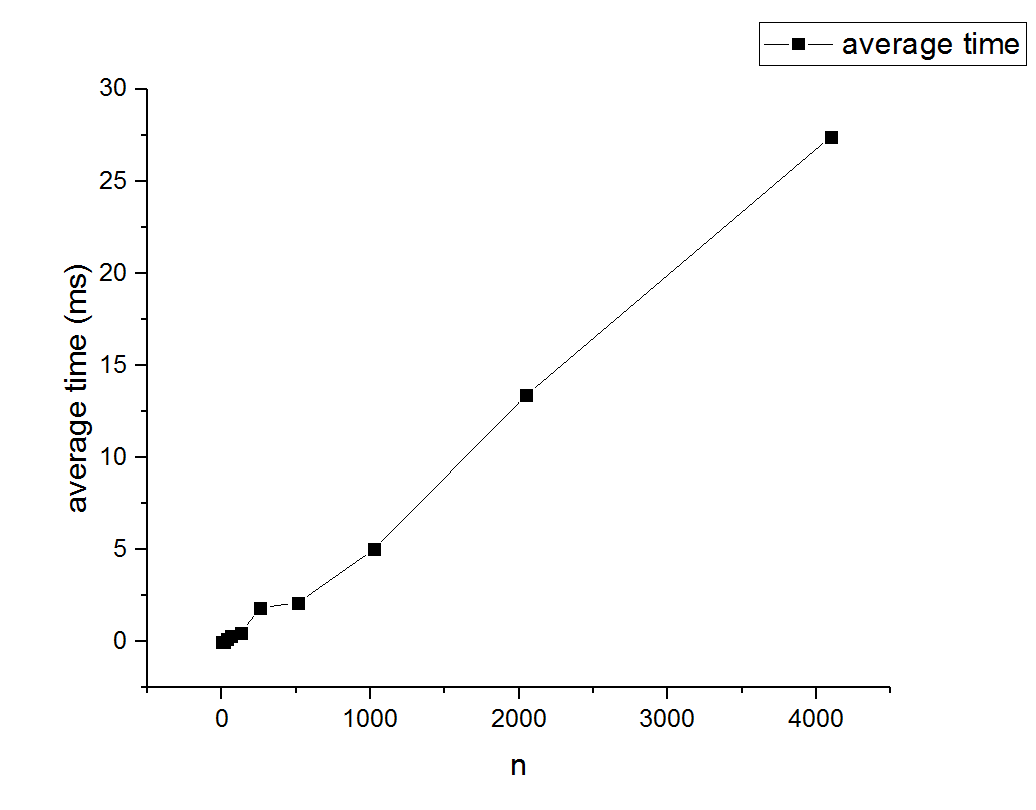
同实验1一样，同样有着因为计算规模较小，从而导致以ms为单位，输入规模为4 ,16, 32,60时，计算所需时间基本都为0.00ms。



因此，增加几组输入数据以更好地得出时间与输入规模之间的关系。在运行时栈大小允许的情况下，设置输入规模分别为：4,16,32,60,128,256,512,1024,2048,4096。其中60用来测试是否支持不是2的整数次幂的输入规模。

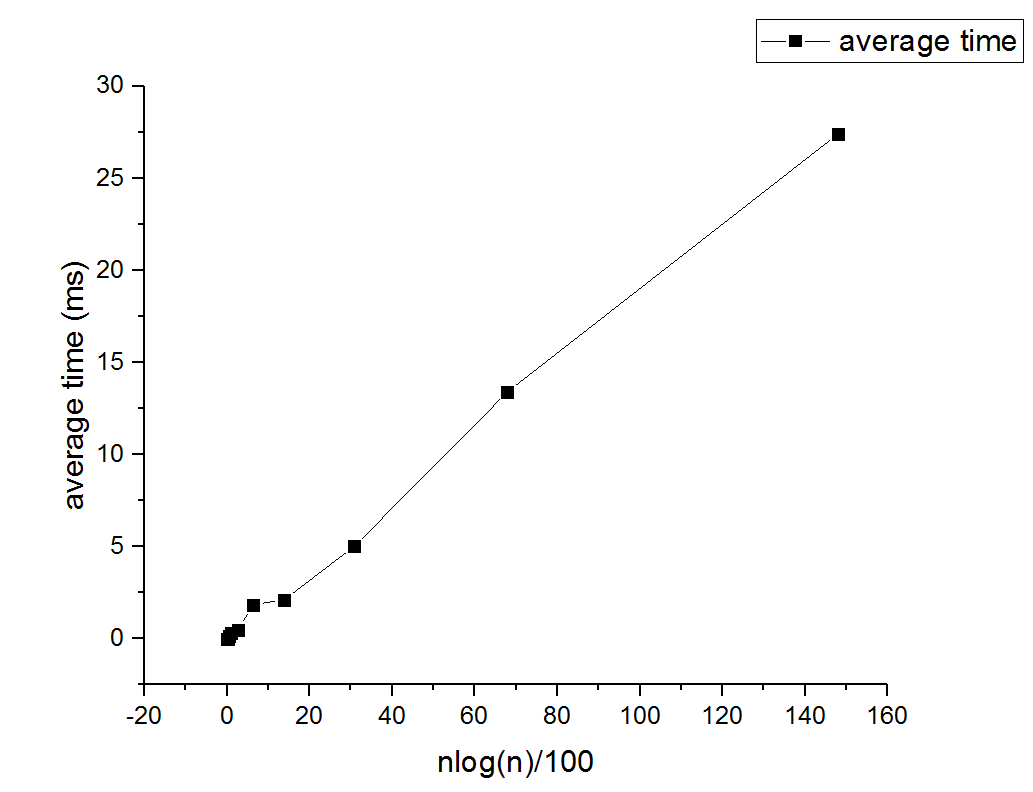
得到了相同运行环境下102组时间数据：

|  |  |
| --- | --- |
| n | average time/ms |
| 4096 | 27.41667 |
| 2048 | 13.39216 |
| 1024 | 5.019608 |
| 512 | 2.127451 |
| 256 | 1.843137 |
| 128 | 0.460784 |
| 60 | 0.294118 |
| 32 | 0.156863 |
| 16 | 0 |
| 4 | 0 |

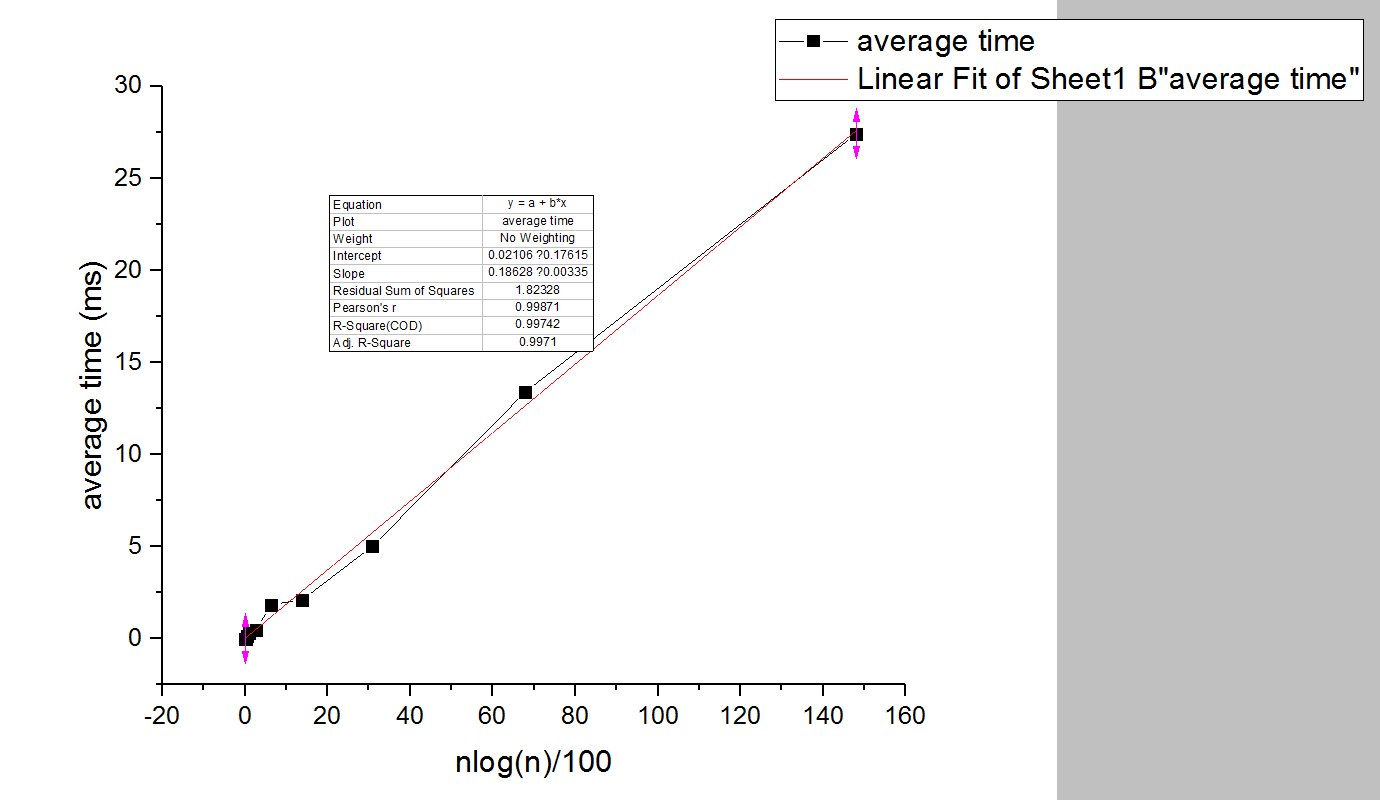


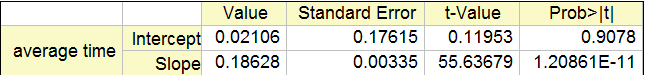
由于所选取的输入规模有限，且所用时间过短而带来了误差，所以结果的准确度有限，

作average time – nlog(n)/100的图，如下：



对它进行直线拟合：





可看出所用时间与nlog(n)大致成线性关系。

FFT算法的时间复杂度为O(nlg(n))