算法基础 实验二 动态规划和 FFT

PB18000334
徐家恒

一、实验内容

动态规划法

FFT

实验 2.1：求矩阵链乘最优方案

n 个矩阵链乘，求最优链乘方案，使链乘过程中乘法运算次数最少。

n 的取值 5, 10, 15, 20, 25，矩阵大小见 2_1_input.txt。

求最优链乘方案及最少乘法运算次数，记录运行时间，画出曲线分析。

仿照 P214 图 15-5，打印 n=5 时的结果并截图。

实验 2.2：FFT

多项式 A(x)= ∑_i=0^n−1▒a_ix^i，系数表示为(a_0,a_1,⋯,a_n−1)。

n 取 2^3, 2^4,⋯, 2^8，不同规模下的 A 见 2_2_input.txt。

用 FFT 求 A 在ω_n^0, ω_n^1,⋯, ω_n^n−1 处的值。

记录运行时间，画出曲线分析；打印 n=2^3 时的结果并截图。

二、实验设备

Pc

三、实验环境

visual studio 2017

四、实验内容

矩阵链乘

$$cost[i][j] = \min\ (i \le k < j)\{cost[i][k] + cost[k+1][j] + p[i-1]p[k]p[j]\}(i \ne j)$$

```cpp
template<typename T>
void ChainOrder(T *p, T **cost, int **cut, int length) {
    if (length < 2) {
        //std::cout << "array too short\n";
        //system("pause");
        exit(1);
    }
    int n = length - 1;
    for (int l = 2; l <= n; ++l) {
        for (int i = 1; i <= n - l + 1; ++i) {
            int j = i + l - 1;
            cost[i][j] = LINF;
            for (int k = i; k < j; ++k) {
                T q = cost[i][k] + cost[k + 1][j] + p[i - 1] * p[k] * p[j];
                if (q < cost[i][j]) {
                    cost[i][j] = q;
                    cut[i][j] = k;
                }
            }
        }
    }
}
```

FFT

递归版本

```cpp
complex *FFT(complex a[], int n) {
    if (n == 1)
        return a;
    complex w(1, 0);
    complex wn(1, 2 * PI / n, 1);
    //complex *a0 = new complex[n / 2];
    //complex *a1 = new complex[n / 2];
    complex *y0 = new complex[n / 2];
    complex *y1 = new complex[n / 2];
    complex *y = new complex[n];
    for (int i = 0; i < n; i += 2) {
        y0[i / 2] = a[i];
        y1[i / 2] = a[i + 1];
    }
    y0 = FFT(y0, n / 2);
    y1 = FFT(y1, n / 2);
    for (int i = 0; i < n / 2; ++i) {
        y[i] = y0[i] + w * y1[i];
        y[i + n / 2] = y0[i] - w * y1[i];
        w = w * wn;
    }
    delete[] y0;
    delete[] y1;
    //delete[] a0;
    //delete[] a1;
    return y;
}
```

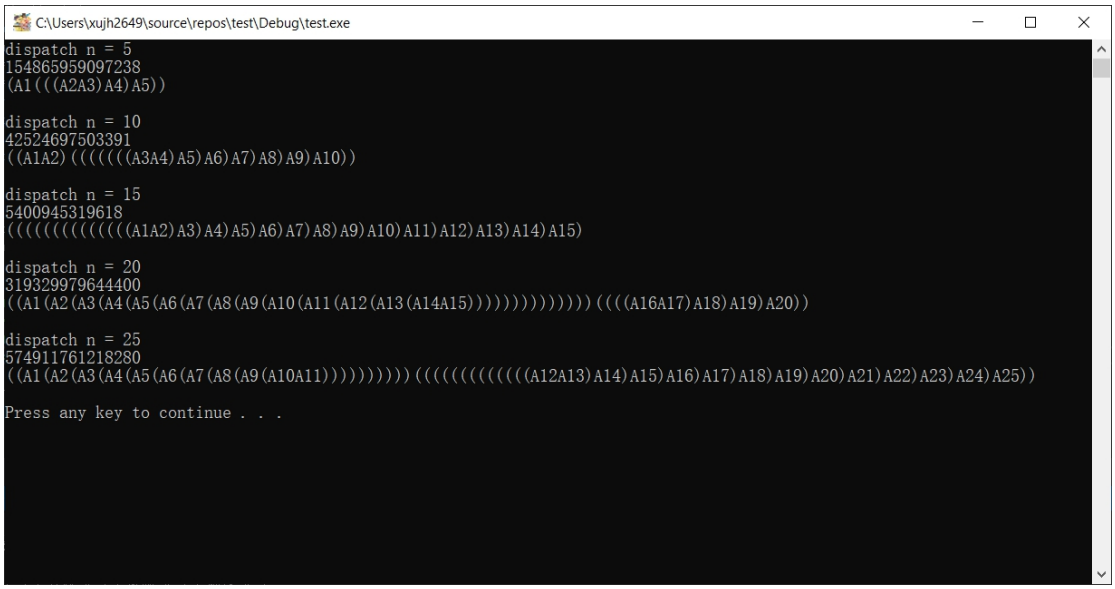
迭代版本

```
complex *FFT1(complex a[], int n) {
    complex *y = new complex[n];
    bitReverse(a, y, n);
    for (int i = 2; i <= n; i <<= 1) {
        int m = i >> 1;
        complex wn(1, 2 * PI / i, 1);
        for (int j = 0; j != n; j += i) {
            complex w(1, 0);
            for (int k = 0; k != m; ++k) {
                complex t = w * y[j + k + m];
                y[j + k + m] = y[j + k] - t;
                y[j + k] = y[j + k] + t;
                w = w * wn;
            }
        }
    }
    return y;
}
```

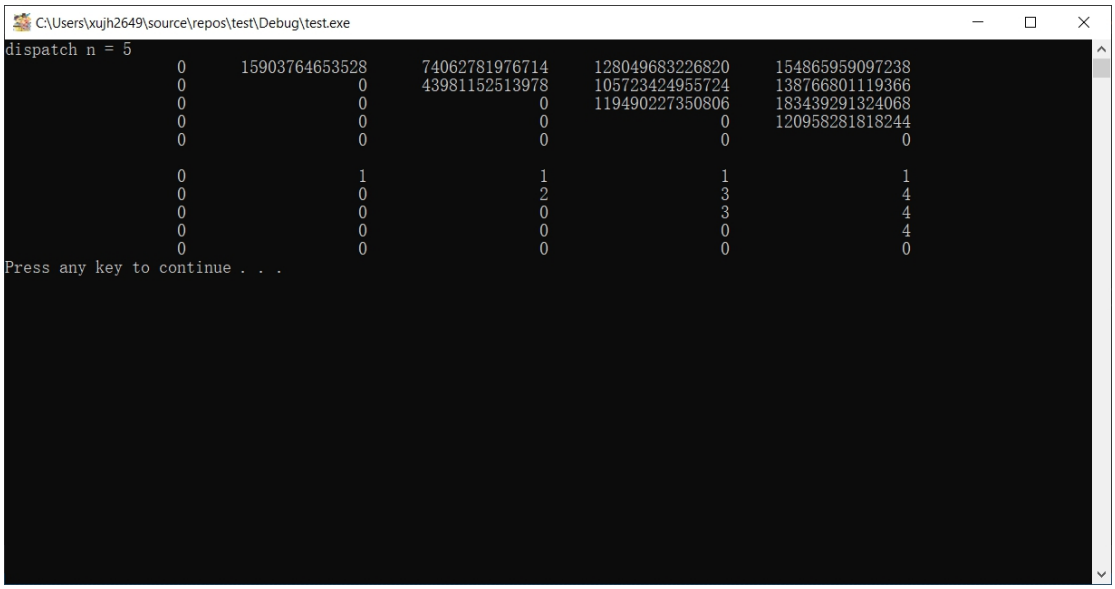
五、实验步骤

crtl+f5 开始运行，结果如下：

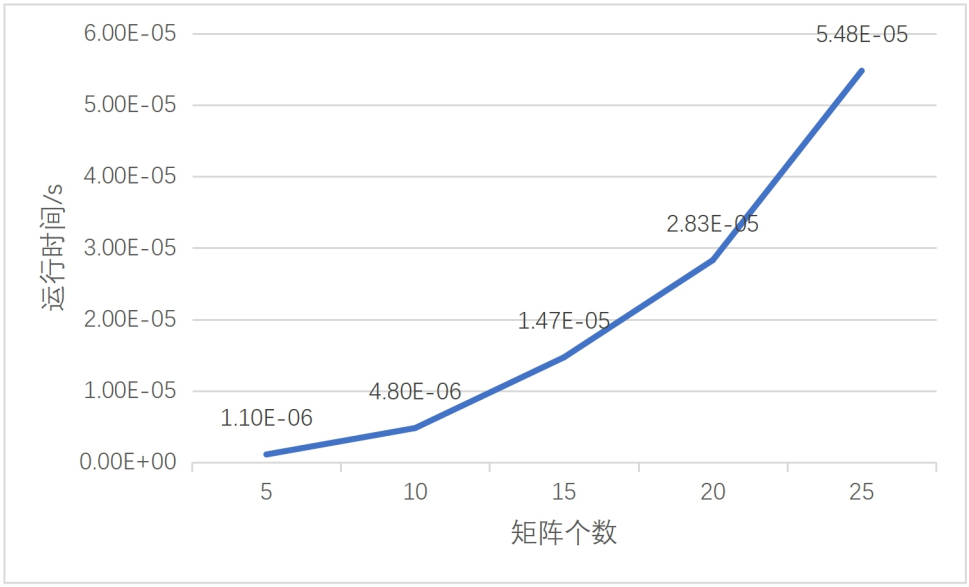
矩阵链乘



n=5 的截图：

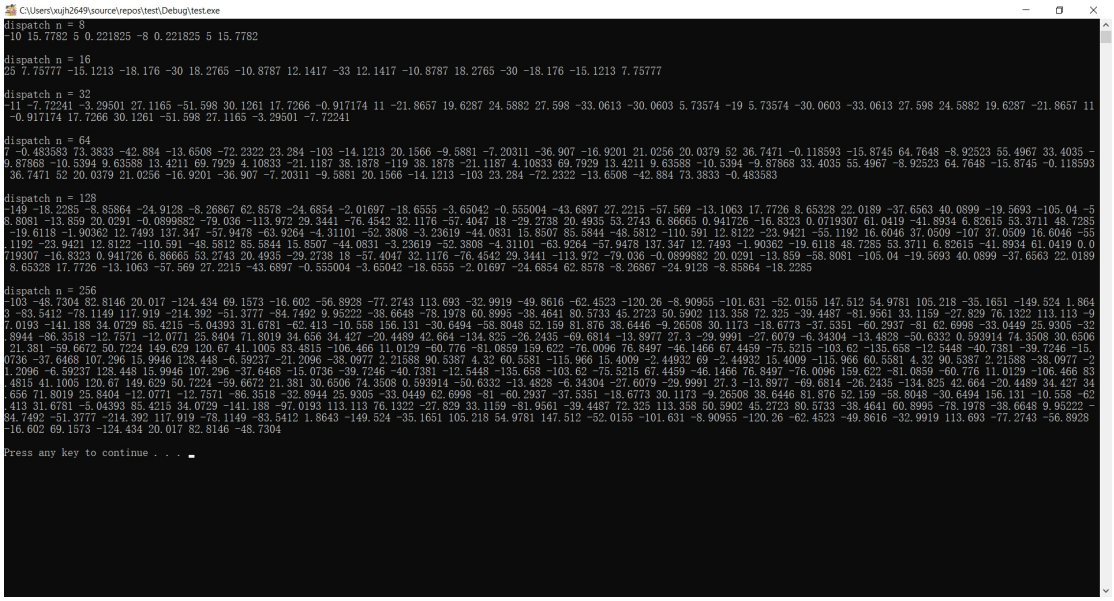


运行时间：

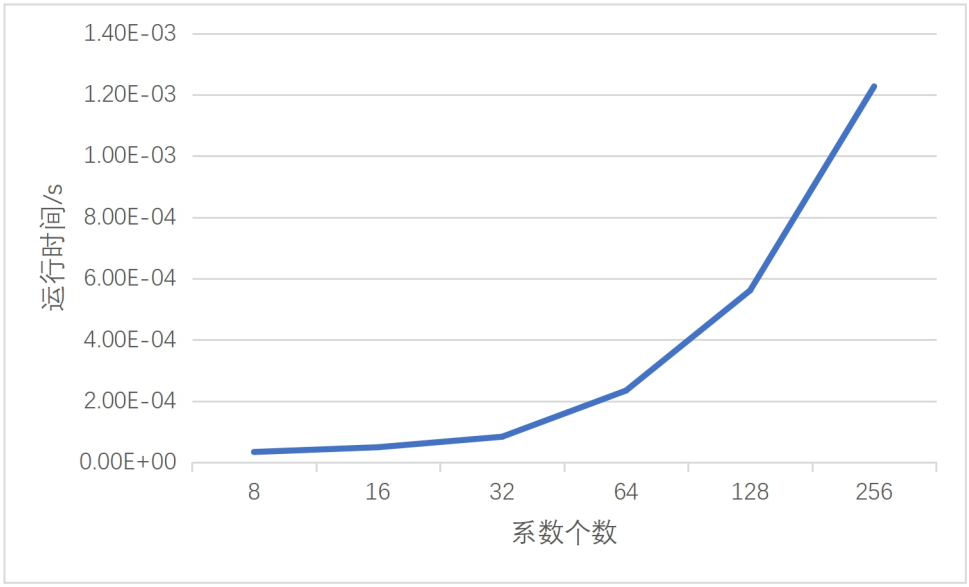


大致符合$O(n^3)$

FFT：



运行时间：



大致符合$O(n^2)$