

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 $A(x) = \sum_{j=0}^{n-1} a_j x^j$, 系数表示为 $(a_0, a_1, \dots, a_{n-1})$ 。

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

用 FFT 求 A 在 $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ 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

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

```

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

```

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

ctrl+f5 开始运行，结果如下：

矩阵链乘

```

C:\Users\xujh2649\source\repos\test\Debug\test.exe
dispatch n = 5
154865959097238
(A1((A2A3)A4)A5))

dispatch n = 10
42524697503391
((A1A2)((((A3A4)A5)A6)A7)A8)A9)A10))

dispatch n = 15
5400945319618
((((((((((((A1A2)A3)A4)A5)A6)A7)A8)A9)A10)A11)A12)A13)A14)A15)

dispatch n = 20
319329979644400
((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10(A11(A12(A13(A14A15))))))))))))))(((A16A17)A18)A19)A20))

dispatch n = 25
574911761218280
((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10A11))))))))))((((((((((((A12A13)A14)A15)A16)A17)A18)A19)A20)A21)A22)A23)A24)A25))

Press any key to continue . . .

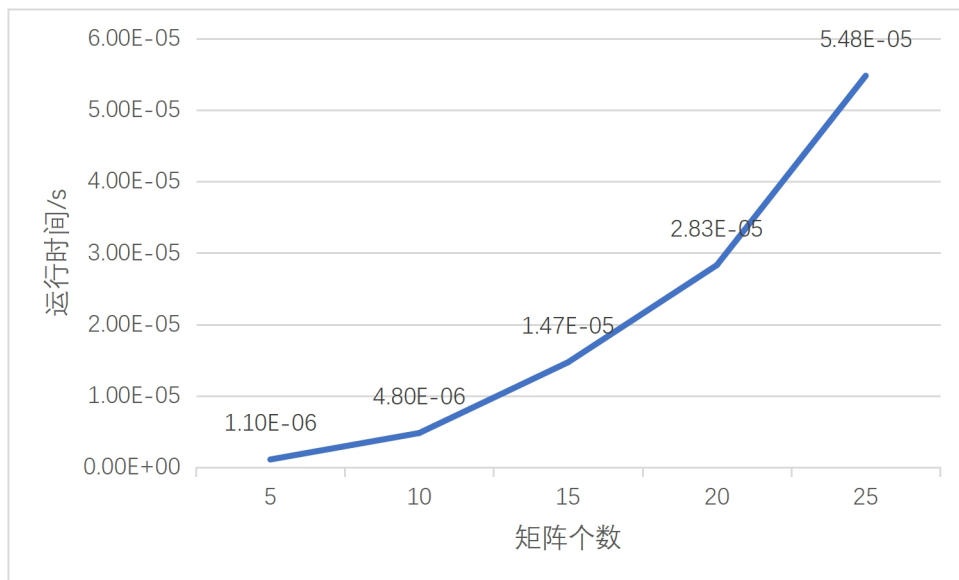
```

n=5 的截图：

```
C:\Users\xujh2649\source\repos\test\Debug\test.exe
dispatch n = 5
0      15903764653528      74062781976714      128049683226820      154865959097238
0      0      0      43981152513978      105723424955724      138766801119366
0      0      0      0      119490227350806      183439291324068
0      0      0      0      0      120958281818244
0      0      0      0      0      0

0      1      1      1      1
0      0      2      3      4
0      0      0      3      4
0      0      0      0      4
0      0      0      0      0
Press any key to continue . . .
```

运行时间：



大致符合 $O(n^3)$

FFT：

```
C:\Users\ujh2649\source\repos\test\Debug\vest.exe
dispatch n = 8
-10 15.7782 5 0.221825 -8 0.221825 5 15.7782

dispatch n = 16
25 7.75777 -15.1213 -18.176 -30 18.2765 -10.8787 12.1417 -33 12.1417 -10.8787 18.2765 -30 -18.176 -15.1213 7.75777

dispatch n = 32
-11 -7.72241 -3.29501 27.1165 -51.598 30.1261 17.7266 -0.917174 11 -21.8657 19.6287 24.5882 27.598 -33.0613 -30.0603 5.73574 -19 5.73574 -30.0603 -33.0613 27.598 24.5882 19.6287 -21.8657 11
-0.917174 17.7266 30.1261 -51.598 27.1165 -3.29501 -7.72241

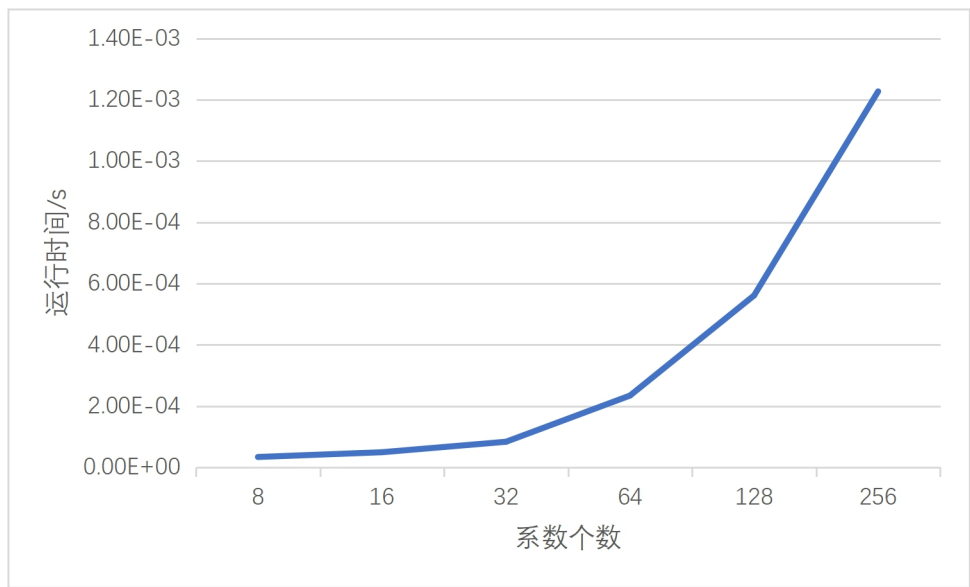
dispatch n = 64
7 -0.483583 73.3833 -42.884 -13.6508 -72.2322 23.284 -163 -14.1213 20.1566 -9.5881 -7.20311 -36.907 -16.9201 21.0256 20.0379 52 36.7471 -0.118593 -15.8745 64.7648 -8.92523 55.4967 33.4035 -9.87883 -10.5394 9.63588 13.4211 69.7929 4.10833 -21.1187 38.1878 -119.38 1878 -21.1187 4.10833 69.7929 13.4211 9.63588 -10.5394 -9.87883 33.4035 55.4967 -8.92523 64.7648 -15.8745 -0.118593
36.7471 52 -20.0379 21.0256 -16.9201 -36.907 -7.20311 -9.5881 20.1566 -14.1213 -103 23.284 -72.2322 -13.6508 -42.884 73.3833 -0.483583

dispatch n = 128
-149 -18.2285 -8.85864 -24.9128 -8.26867 62.8578 -24.6854 -2.01697 -18.6555 -3.65042 -0.555004 -43.6897 27.2215 -57.569 -13.1063 17.7726 8.65328 22.0189 -37.6563 40.0899 -19.5693 -105.04 -5
8.8081 -13.859 20.0291 -0.0899882 -79.036 -113.972 29.3441 -76.4542 32.1176 -57.4047 18 -29.2738 20.4935 53.2743 6.86665 0.941726 -16.8323 0.0719307 61.0419 -41.8934 6.82615 53.3711 48.7285
-19.6118 -1.90362 12.7493 137.347 -57.9478 -63.9264 -4.31101 -52.3808 -3.23619 -44.0831 15.8507 85.5844 -48.5812 -110.591 12.8122 -23.9421 -55.1192 16.6046 37.0509 -107 37.0509 16.6046 -55
1192 -23.9421 12.8122 -110.591 -48.5812 85.5844 15.8507 -44.0831 -3.23619 -52.3808 -4.31101 -63.9264 -57.9478 137.347 12.7493 -1.90362 -19.6118 48.7285 53.3711 6.82615 -41.8934 61.0419 0 0
719307 -16.8323 0.941726 6.86665 53.2743 20.4935 -29.2738 18 -57.4047 32.1176 -76.4542 29.3441 -113.972 -79.036 -0.0899882 20.0291 -13.859 -58.8081 -105.04 -19.5693 40.0899 -37.6563 22.0189
8.65328 17.7726 -13.1063 -57.569 27.2215 -43.6897 -0.555004 -3.65042 -18.6555 -2.01697 -24.6854 62.8578 -8.26867 -24.9128 -8.85864 -18.2285

dispatch n = 256
-103 -48.7304 82.8146 20.017 -124.434 69.1573 -16.602 -56.8928 -77.2743 113.693 -32.9919 -49.8616 -62.4523 -120.26 -8.90955 -101.631 -52.0155 147.512 54.9781 105.218 -35.1651 -149.524 1.864
9 -83.5412 -78.1149 117.919 -214.392 -51.3777 -84.7492 9.95222 -38.6648 -78.1978 60.8995 -38.4641 80.5733 45.2723 50.5902 113.358 72.325 -39.4487 -81.9561 33.1159 -27.829 76.1322 113.113 -9
7.0192 -141.188 24.0729 85.4215 5.04393 31.6781 -62.413 10.558 156.131 -30.6494 -58.8048 52.159 81.876 38.6446 -9.26508 30.1173 -18.6772 -37.5351 -60.2387 -81.62 69.98 -33.0449 25.9305 32
8944 -86.3518 -12.7571 -12.0771 25.8404 71.8019 34.656 34.427 -20.4489 42.664 -134.825 -26.2435 -69.6814 -13.8977 27.3 -29.9991 -27.6079 -6.34304 -13.4828 -50.6332 0.593914 74.3508 30.6506
21.381 -59.6672 50.7224 149.629 120.67 41.1005 83.4815 -106.466 11.0129 -60.776 -81.0859 159.622 -76.0096 76.8497 -46.1466 67.4459 -75.5215 -103.62 -135.658 -12.5448 -40.7381 -39.7246 -15
0736 -37.6468 107.296 15.9946 128.448 -6.39237 -21.2096 -38.0977 2.21888 90.5387 4.32 60.5581 -115.966 15.4009 -2.44932 69 -2.44932 15.4009 -115.966 60.5581 4.32 90.5387 2.21888 -38.0977 -2
1.2096 -6.39237 128.448 15.9946 107.296 -37.6468 -15.0736 -39.7246 -40.7381 -12.5448 -135.658 -103.62 -75.5215 67.4459 -46.1466 76.8497 -76.0096 159.622 -81.0859 -60.776 11.0129 -106.466 83
4815 41.1005 120.67 149.629 50.7224 -59.6672 21.381 30.6506 74.3508 0.593914 -50.6332 -13.4828 -6.34304 -27.6079 -29.9991 27.3 -13.8977 -69.6814 -26.2435 -134.825 42.664 -20.4489 34.427 34
656 71.8019 25.8404 -12.0771 -12.7571 -86.3518 -32.8944 25.9305 -33.0449 62.6998 -81 -60.2937 -37.5351 -18.6773 30.1173 -9.26508 38.6446 81.876 32.159 -58.8048 -30.6494 156.131 -10.558 -62
413 81.6781 -5.04393 85.4215 34.0729 -141.188 -97.0193 113.113 76.1322 -27.829 33.1159 -81.9561 -39.4487 72.325 113.358 50.5902 45.2723 80.5733 -38.4641 60.8995 -78.1978 -38.6648 9.95222
84.7492 -51.3777 -214.392 117.919 -78.1149 -83.5412 1.8643 -149.524 -35.1651 105.218 54.9781 147.512 -52.0155 -101.631 -8.90955 -120.26 -62.4523 -49.8616 -32.9919 113.693 -77.2743 -56.8928
-16.602 69.1573 -124.434 20.017 82.8146 -48.7304

Press any key to continue . . .
```

运行时间：



大致符合 $O(n^2)$