

中国科学技术大学计算机学院
《算法基础》实验报告



实验题目：lab2_动态规划和 FFT

学生姓名：胡毅翔

学生学号：PB18000290

完成日期：2020 年 12 月 9 日

计算机实验教学中心制

2019 年 09 月

实验目的

- 1.实现 5 种排序算法:矩阵链乘和 FFT 算法。
- 2.对给定的输入得到正确结果。
- 3.对获得的实验数据进行分析, 并与理论进行比较。

实验原理

本次实验所实现的算法有:matrix chain order 和 FFT。其正确性已在《算法导论》一书中得到证明。

1.matrix chain order

```
void matrix_chain_order(int *array, int length)
{
    int n = length;
    int i, j, k, l;
    long long tmp;
    for (i = 1; i <= n; i++)
    {
        m[i][i] = 0;
    }
    for (l = 2; l <= n; l++)
    {
        for (i = 1; i <= n - l + 1; i++)
        {
            j = i + l - 1;
            m[i][j] = LLONG_MAX;
            for (k = i; k <= j - 1; k++)
            {
                tmp = m[i][k] + m[k + 1][j] + ((long long)array[i - 1] * (long long)array[k] *
                (long long)array[j]);
                if (tmp < m[i][j])
                {
                    m[i][j] = tmp;
                    s[i][j] = k;
                }
            }
        }
    }
}
```

2.FFT

在复现算法过程中, 为使代码结构更加紧凑, 在 [complex.h](#) 中实现复数类及相关操作的定义。

```

void recursive_fft(complex *array, complex *result, int length)
{
    int n = length;
    if (n == 1)
    {
        equal(array, result);
        return;
    }
    complex omiga_n, tmp;
    exp_i(2 / (float)n, &omiga_n);
    complex omiga = {1, 0};
    complex *a_0, *a_1, *y_0, *y_1;
    a_0 = (complex *)calloc(n / 2, sizeof(complex));
    a_1 = (complex *)calloc(n / 2, sizeof(complex));
    y_0 = (complex *)calloc(n / 2, sizeof(complex));
    y_1 = (complex *)calloc(n / 2, sizeof(complex));
    for (int i = 0; i < n / 2; i++)
    {
        equal(&array[2 * i], &a_0[i]);
        equal(&array[2 * i + 1], &a_1[i]);
    }
    recursive_fft(a_0, y_0, n / 2);
    recursive_fft(a_1, y_1, n / 2);
    for (int k = 0; k < n / 2; k++)
    {
        multiply(&omiga, &y_1[k], &tmp);
        plus(&y_0[k], &tmp, &result[k]);
        miuns(&y_0[k], &tmp, &result[k + n / 2]);
        multiply(&omiga_n, &omiga, &tmp);
        equal(&tmp, &omiga);
    }
    free(a_0);
    free(a_1);
    free(y_0);
    free(y_1);
    return;
}

```

实验环境

1.PC 一台

2.Windows 系统

3.gcc 编译器

实验过程

目录框架

本次实验的目录框架如下图所示:

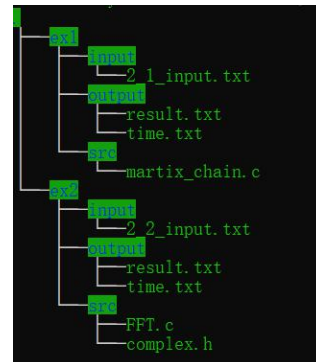


图 1

程序执行

执行 martix_chain.exe:

- 1.读入一组输入。
- 2.开始计时。
- 3.调用 `martix_chain_order()` 进行求解。
- 4.结束计时。
- 5.记录运行时间，保存运行结果。
- 6.若仍有输入转 1，否则结束程序。



图 2

运行过程截图

```
result.txt X
project2 > ex1 > output > result.txt
1 (A1((((A2A3)A4)A5))154865959097238
2 ((A1A2)((((A3A4)A5)A6)A7)A8)A9)A10))42524697503391
3 (((((((((((A1A2)A3)A4)A5)A6)A7)A8)A9)A10)A11)A12)A13)A14)A15)5400945319618
4 ((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10(A11(A12(A13(A14(A15)))))))))))))(((A16A17)A18)A19)A20))319329979644400
5 ((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10A11))))))))))((((((((A12A13)A14)A15)A16)A17)A18)A19)A20)A21)A22)A23)A24)A25))574911761218280
6

time.txt X
project2 > ex1 > output > time.txt
1 5.600000
2 5.900000
3 11.700000
4 39.100000
5 49.200000
6
```

图 3

输出文件结果截图

执行 FFT.exe:

- 1.读入一组输入。
- 2.开始计时。
- 3.调用 recursive_fft()进行求解。
- 4.结束计时。
- 5.记录运行时间，保存运行结果。
- 6.若仍有输入转 1，否则结束程序。

```
D:\USTC\algorithm2020_labs\project2\ex2\src\FFT.exe
size of input: 8
result of 2^3:
-10.000000 15.778174 5.000000 0.221826 -8.000000 0.221826 5.000000 15.778173
size of input: 16
size of input: 32
size of input: 64
size of input: 128
size of input: 256
Process returned 0 (0x0) execution time : 0.076 s
Press any key to continue.
```

图 4

运行过程截图

```
result.txt X
project2 > ex2 > output > result.txt
1 -10.000000 15.778174 5.000000 0.221826 -8.000000 0.221826 5.000000 15.778173
2 25.000000 7.757766 -15.121320 -18.175968 -30.000000 18.276472 -10.878680 12.141727 -33.000000 12.141727 -10.878680 18.276474 -30.000000
3 -11.000000 -7.722007 -3.299015 27.116533 -51.507973 38.126078 17.726589 -0.912177 11.000000 -21.865707 19.628744 24.588203 27.597965 -
4 7.000000 -0.482582 73.383255 -42.883993 -13.658753 -72.232185 23.283974 -103.080259 -14.121318 20.156553 -9.588091 -7.203111 -36.906952
5 -149.000000 -18.228539 -8.858638 -24.912764 -8.268666 62.857811 -24.685421 -2.016983 -18.655453 -3.650432 -0.555027 -43.689640 27.22151
6 -103.000000 -48.738427 82.814575 20.016991 -124.433594 69.157318 -16.601955 -56.892738 -77.274246 113.692886 -32.991905 -49.861568 -62.
7

time.txt X
project2 > ex2 > output > time.txt
1 19.000000
2 28.600000
3 65.000000
4 106.200000
5 218.600000
6 620.100000
7
```

图 5

输出文件结果截图

结果分析

运行时间统计分析

矩阵链乘运行时间统计：

输入规模	T/ μ s
5	5.600000
10	5.900000
15	11.700000
20	39.100000
25	49.200000

表 1

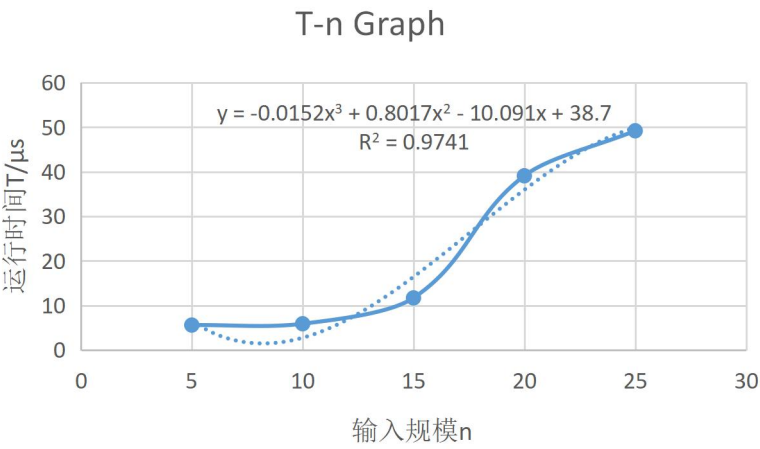


图 6

与理论 $T(n) = \Omega(n^3)$ 的拟合程度较好，可以认为满足输入规模的三次方与运行时间成正比。

FFT 运行时间统计：

n	T / μ s	$n \log n$	$\frac{T}{n \log n}$
8	19	24	1.263157895
16	28.6	64	2.237762238
32	65	160	2.461538462
64	106.2	384	3.615819209
128	218.6	896	4.098810613

表 2

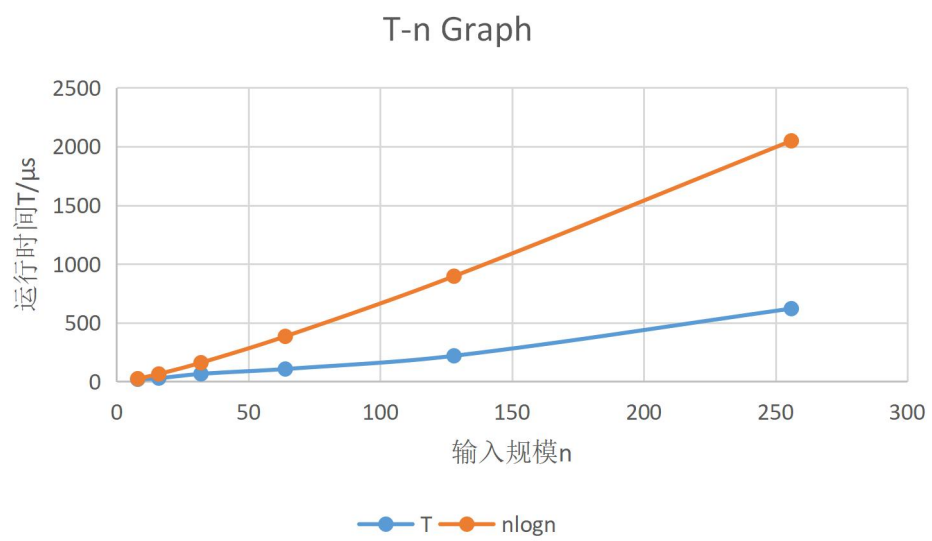


图 7

结合图象及数据可知，本次实验中 FFT 运行时间与理论 $T(n) = \Theta(n \log n)$ 一致。