第八次作业实验报告

实验环境

以下所有实验都处于这一环境中

操作系统

Windows 10 家庭中文版 64位 版本10.0.17134.345

硬件

CPU Intel Core i7-8750H

RAM 8GB

IDE

Microsoft Visual Studio Community 2017 VisualStudio.15.Release/15.8.5+28010.2036

Visual C++ 2017 00369-60000-00001-AA380

第一题：约瑟夫问题

实验目的

> 39个犹太人与约瑟夫躲到山洞，39个犹太人决定宁死也不要被敌人抓到，于是决定了自杀方式：40个人排成一个圆圈，由第1个人开始报数，每报到第3人该人就必须自杀，然后再由下一个重新报数，直到所有人都自杀身亡为止。然而约瑟夫并不想死，他站在某个位置上，最终逃过了这场死亡游戏。试问：这个位置是哪个编号？**（说明：完成此题时，编号规定从0开始，即40个人的编号为0–39）**
>
> 要求：
>
> 1. 分别采用递归和递推法来编程；
> 2. 采用 `<ctime>` 或 `<time.h>` 头文件中的`clock()` 来分析这两种方法的CPU耗时，指出哪种方法更快，并分析其原因。说明： `clock()`使用参见“课外资料料之三”；
> 3. 调试递归程序，截屏“调用窗口口”，写在实验报告里。

学习递归和递推/迭代的不同

实验内容

分析

对于此题而言，遍历是比较好想的。考虑一个有四十个元素的数组，初始状态全部为0，表示尚未死亡。从第一个人（编号0）开始，每数3个0，将第三个0改为1，表示死亡。不断重复这个操作直到只剩一个0，对应的编号即为约瑟夫应该站的位置。

下面给出一个六个人的时候的示例：

```
[0,0,0,0,0,0]
[0,0,1,0,0,0]
[0,0,1,0,0,1]
     ...
[0,1,1,1,1,1]
```

而对于递归，可以如下考虑：

```
一个人的话，他活下来
[0]

两个人的时候：
[0,0]
[1,0]

三个人的时候：
[0,0,0]
[0,0,1]注意到此时变为两个人的情形，可以调用两个人的时候活着的人的编号对应到三个人的时候

四个人时：
[0,0,0,0]
[0,0,1,0]
此时可以对上面活下来的三人重新编号，然后化为三个人的情形

n个人时
[0,0,0,0,...,0]
[0,0,1,0,...,0]
同样重新编号，化为n-1人的情形

对于n和n-1，可以建立一个映射将n-1时活下来的人的编号对应到n时活下来的人的编号
记n个人时活下来的人是第i_{n}个（编号i_{n}-1），n-1个人的时候活下来的人是第i_{n-1}，则有i_{n}=(3+i_{n-1})%n
```

由递归的思路，我们可以的得到一个递推关系：

对于n个人的情况，活下来的人的编号是n-1个人时 `编号+3`

由此可以得出 `编号_{k}=(编号_{k-1}+3)%k`

而对于用时分析，由于题给数字较小，可以采取多次执行来放大差异，再取平均值得出结果

代码

递推

```cpp
#include <iostream>
#include <ctime>
using namespace std;

int main() {
    clock_t start, finish;
    double time;
    start = clock();

    for (int i = 0; i < 1000; ++i) {
        int iPlace = 0;
        for (int i = 2; i < 41; ++i) {
            iPlace = (iPlace + 3) % i;
        }
        cout << iPlace << " ";
    }

    finish = clock();
    time = (double)(finish - start) / CLOCKS_PER_SEC;
    cout << "\n耗时" << 1000*time << "ms\n";
    cout << "\nclock_tick = " << start << " " << finish;
    return 0;
}
```

递归

```cpp
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;

int iKill(int);

int main() {
    clock_t start, finish;
    double time;
    start = clock();

    for (int i = 0; i < 1000; ++i) {
        cout << setw(3) << iKill(40) - 1;
    }

    finish = clock();
    time = (double)(finish - start) / CLOCKS_PER_SEC;
    cout << "\n耗时" << 1000 * time << "ms\n";
    cout << "\nclock_tick = " << start << " " << finish;

```

```c
    return 0;
}

int iKill(int n) {
    if (n == 1 || n == 2)return n;
    else if (n == 3)return 2;
    else if (n > 3) {
        int i = 3 + iKill(n - 1);
        if (i > n)i -= n;
        return i;
    }
    else return -1;
}
```

结果

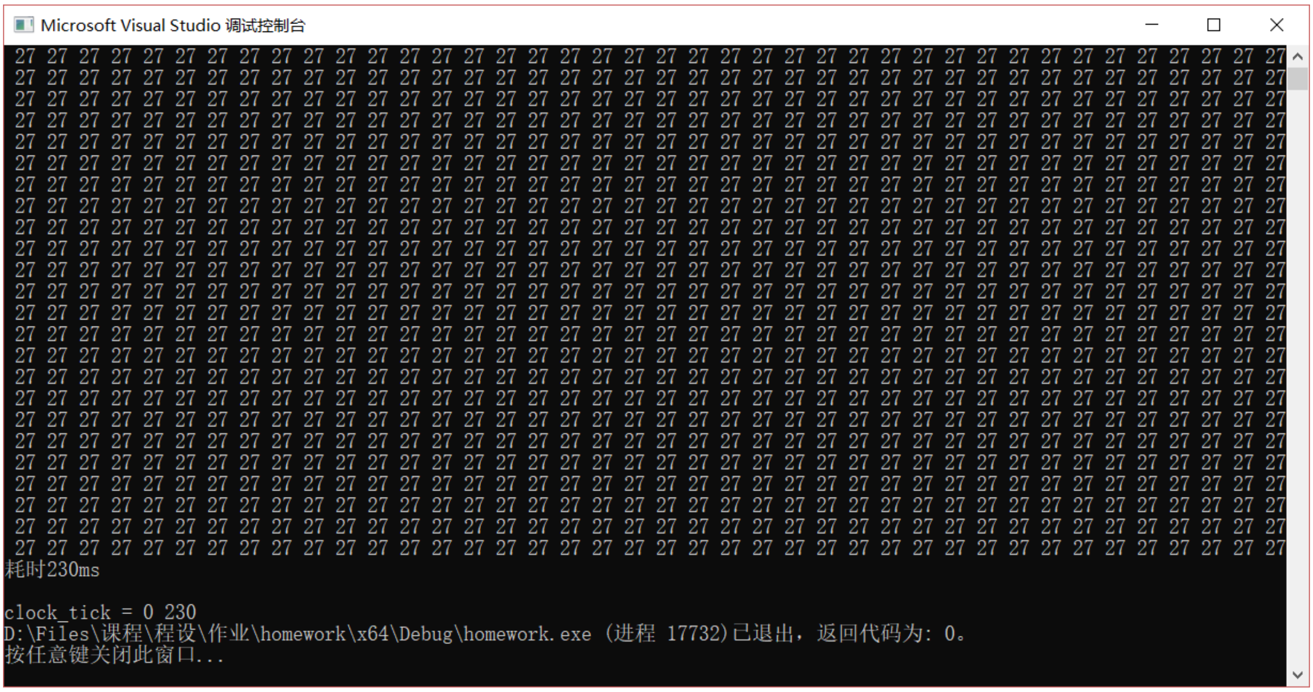
约瑟夫应该站在编号27的位置上

递推

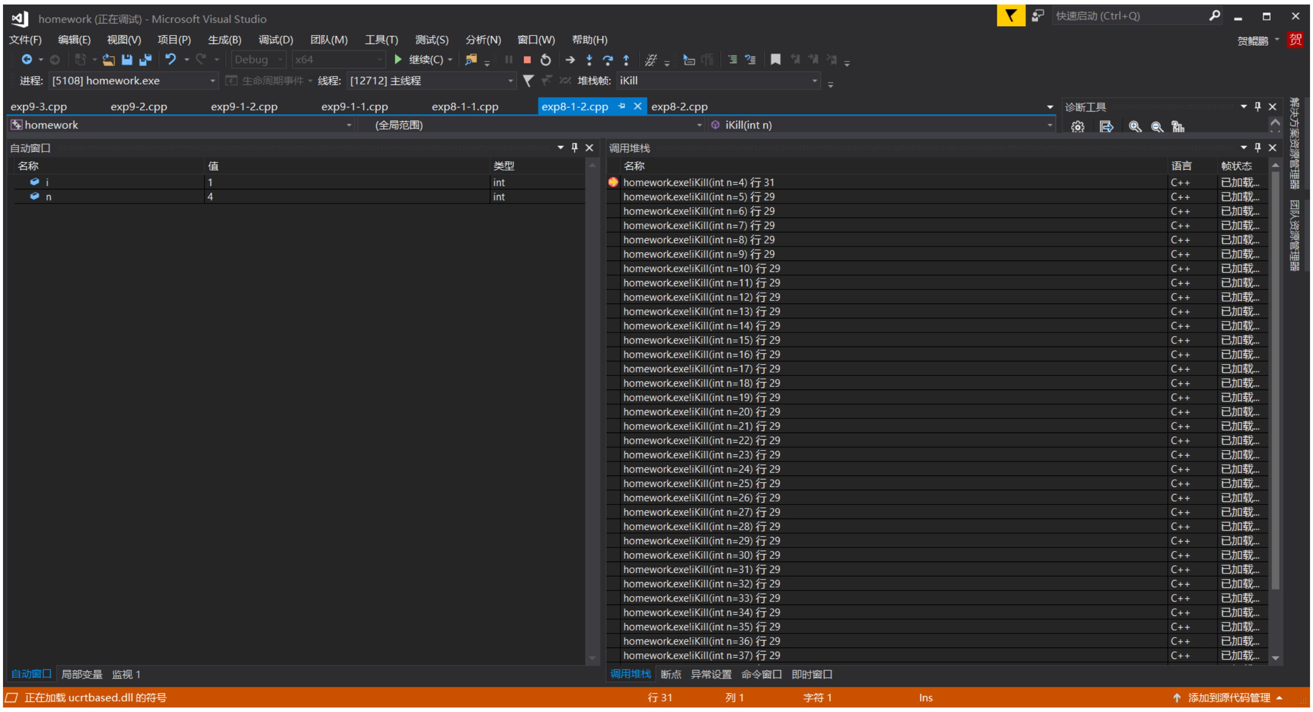
Microsoft Visual Studio 调试控制台

```
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
耗时83ms
18 101
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 12328)已退出，返回代码为: 0。
按任意键关闭此窗口...
```

递归


Microsoft Visual Studio 调试控制台
耗时230ms
clock_tick = 0 230
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 17732)已退出，返回代码为: 0。
按任意键关闭此窗口...

调试界面


homework (正在调试) - Microsoft Visual Studio
文件(F) 编辑(E) 视图(V) 项目(P) 生成(B) 调试(D) 团队(M) 工具(T) 测试(S) 分析(N) 窗口(W) 帮助(H)
进程: [5108] homework.exe
线程: [12712] 主线程
堆栈帧: iKill
exp9-3.cpp exp9-2.cpp exp9-1-2.cpp exp9-1-1.cpp exp8-1-1.cpp exp8-1-2.cpp exp8-2.cpp
自动窗口
名称 值 类型
i 1 int
n 4 int
调用堆栈
名称 语言 帧状态
homework.exe!iKill(int n=4) 行 31
homework.exe!iKill(int n=5) 行 29
homework.exe!iKill(int n=6) 行 29
homework.exe!iKill(int n=7) 行 29
homework.exe!iKill(int n=37) 行 29
正在加载 ucrtbased.dll 的符号
行 31 列 1 字符 1 Ins
添加到源代码管理

时间统计

递归	递推
294	83
318	86
386	93
270	88
316	86
362	84
297	92
335	98
平均322.25ms	平均88.75ms

分析总结

递推有极大的优势

考虑到这个递归是一个线性递归，主要耗时在压栈和弹栈。如果是一个树形递归，递推的优势会更加明显，如下题。

可能的优化方向，通过指针？（不清楚）

第二题：下楼问题

实验目的

> 从楼上走到楼下共有n级台阶，每一步有三种走法，走一级、走两级或走三级台阶。问恰好走完这n级台阶共有多少种不同的方案。 要求：
>
> 1. 输入台阶的级数n，输出恰好走完n级台阶的不同方案数（不需要输出每种方案的详情）；
> 2. 请用递归思想来编程： `int GoDown(int n);`//输入参数为台阶级数n，输出不同的方案数目
> 3. 请用程序测试n=5-20时的输出结果，写在实验报告中；
> 4. 请用下列语句测试你的程序耗时： `#include <ctime>` `clock_t start = clock();`
> 5. 请测试n=15、25、35时的耗时（每个n值测三次，取平均）；
> 6. 你能将耗时降到1ms以内吗？ 优化算法试试？ （不要求必须做，本小题不占本题分数）。

进一步练习递归

实验内容

分析

对于1、2、3级台阶，可采用枚举法得出相应方案数为：1、2、4

而对于大于3级，如n级台阶，已知n-1、n-2、n-3级台阶的下楼方案数便可推知n级台阶的下楼方案数为前三者之和。

即 `GoDown(n)=GoDown(n-1)+GoDown(n-2)+GoDown(n-3)`

然而，这是一种树形递归，当n增大时操作数会急速增长（重复执行某个步骤多次），导致耗时迅速增大。一个较好的思路是将其改写为一个递推式，减少相同步骤的执行次数。

递推思路如下：

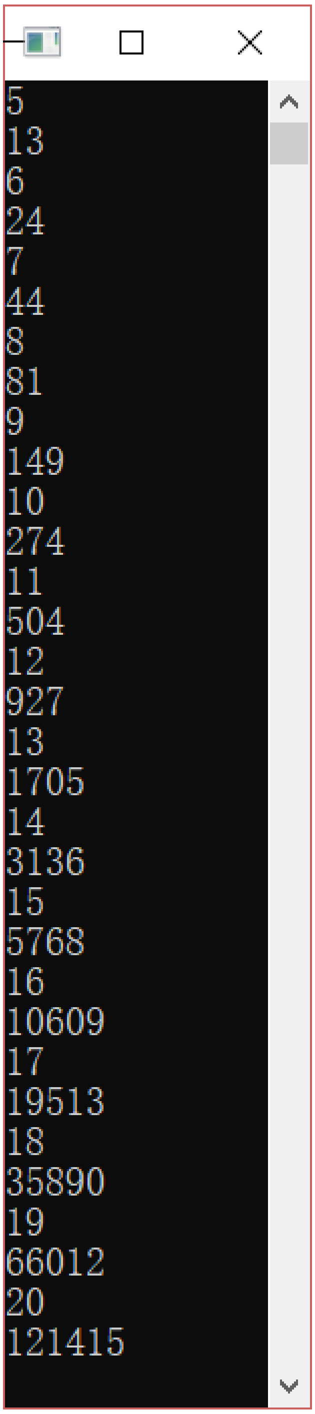
```
先用三个数记录1、2、3级台阶的方案数
a = 1, b = 2, c = 4;
然后4级台阶可由如下递推得出：
c = a + b + c; //(== 7 四级台阶方案数)
b = c - a - b; //(== 4 三级台阶方案数)
a = c - a - b; //(== 2 二级台阶方案数)

不断重复这个操作，便可得出n级台阶方案数
```

代码

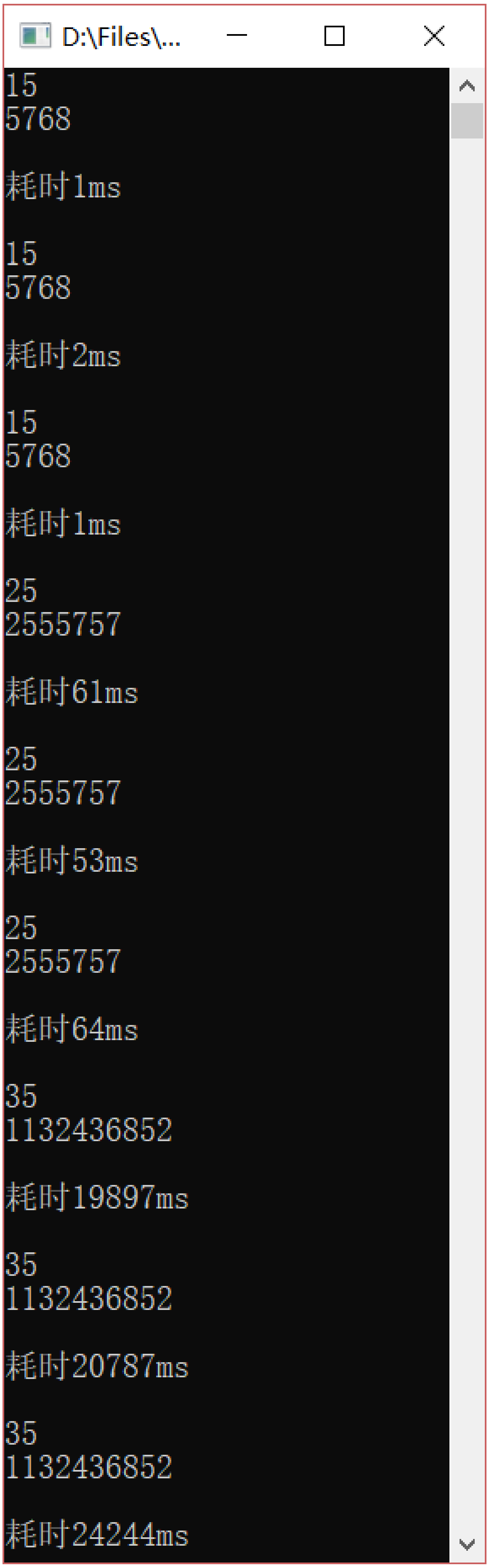
```cpp
#include <iostream>
#include <ctime>
using namespace std;

int GoDown(int);

int main() {
    clock_t start, finish;
    double time;

    int n;
    for (;;) {

        cin >> n;

        start = clock();

        cout << GoDown(n) << endl;

        finish = clock();
        time = (double)(finish - start) / CLOCKS_PER_SEC;
        cout << "\n耗时" << 1000 * time << "ms\n";
    }
    return 0;
}

int GoDown(int n) {
    /*//递推实现
    int a = 1, b = 2, c = 4;
    if (n == 1) return 1;
    else if (n == 2) return 2;
    else if (n == 3) return 4;
```

```
    else {
        for (int i = 4; i <= n; i++) {
            c = a + b + c;
            b = c - a - b;
            a = c - a - b;
        }
        return c;
    }
    */

    if (n == 1) return 1;
    else if (n == 2) return 2;
    else if (n == 3) return 4;
    else {
        int i = GoDown(n - 1) + GoDown(n - 2) + GoDown(n - 3);
        return i;
    }

}
```

结果


5
13
6
24
7
44
8
81
9
149
10
274
11
504
12
927
13
1705
14
3136
15
5768
16
10609
17
19513
18
35890
19
66012
20
121415

15、25、35时间测试


D:\Files\...
15
5768
耗时1ms
15
5768
耗时2ms
15
5768
耗时1ms
25
2555757
耗时61ms
25
2555757
耗时53ms
25
2555757
耗时64ms
35
1132436852
耗时19897ms
35
1132436852
耗时20787ms
35
1132436852
耗时24244ms

平均时间为：

15级 ≈1.33ms

25级 ≈59.33ms

35级 ≈21642.67ms

改为递推后，35级台阶的方案数耗时情况：

一定程度上耗时大幅减少，但是和1ms仍有差距。

分析总结

采用递推方式可以显著减少树形递归带来的复杂运算，实现耗时的减少。