Euler problem 18

于船长

书山有路勤为径，学海无涯苦作舟

本期内容

一. 题目讲解

二. 代码演示

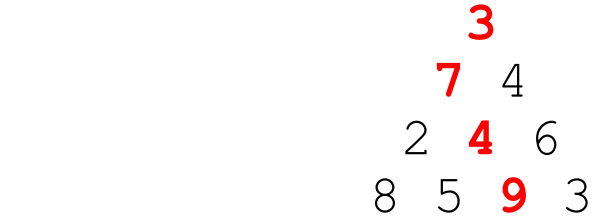
三. 动态规划基础

一. 题目讲解

题目描述

Maximum path sum I

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.



That is, 3 + 7 + 4 + 9 = 23.

题目描述

Find the maximum total from top to bottom of the triangle below:

```
                75
               95 64
              17 47 82
             18 35 87 10
            20 04 82 47 65
           19 01 23 75 03 34
          88 02 77 73 07 63 67
         99 65 04 28 06 16 70 92
        41 41 26 56 83 40 80 70 33
       41 48 72 33 47 32 37 16 94 29
      53 71 44 65 25 43 91 52 97 51 14
     70 11 33 28 77 73 17 78 39 68 17 57
    91 71 52 38 17 14 91 43 58 50 27 29 48
   63 66 04 68 89 53 67 30 73 16 69 87 40 31
  04 62 98 27 23 09 70 98 73 93 38 53 60 04 23
```

题目描述

最大路径和 I

从如下数字三角形的顶端出发，不断移动到下一行与其相邻的数直至到达底部，所能得到的最大路径和是23。

如上图，最大路径和为3 + 7 + 4 + 9 = 23。

题目描述

从如下数字三角形的顶端出发到达底部，求所能得到的最大路径和。

```
                75
               95 64
              17 47 82
             18 35 87 10
            20 04 82 47 65
           19 01 23 75 03 34
          88 02 77 73 07 63 67
         99 65 04 28 06 16 70 92
        41 41 26 56 83 40 80 70 33
       41 48 72 33 47 32 37 16 94 29
      53 71 44 65 25 43 91 52 97 51 14
     70 11 33 28 77 73 17 78 39 68 17 57
    91 71 52 38 17 14 91 43 58 50 27 29 48
   63 66 04 68 89 53 67 30 73 16 69 87 40 31
  04 62 98 27 23 09 70 98 73 93 38 53 60 04 23
```

递归+记忆化

如果没有思路的话，试着跟着下面的提示走一走：

1、试着用递归程序完成这个题目所求

2、感觉程序运行较慢的话，试一试【记忆化】

二. 代码演示

递归程序

```
1  #include <stdio.h>
2  #include <inttypes.h>
3
4  int32_t num[22][22] = {0};
5
6  int32_t GetMaxSum(int32_t i, int32_t j, int32_t maxN) {
7      if (i + 1 == maxN) return num[i][j];
8      int32_t ans1, ans2;
9      ans1 = GetMaxSum(i + 1, j, maxN) + num[i][j];
10     ans2 = GetMaxSum(i + 1, j + 1, maxN) + num[i][j];
11     return ans1 > ans2 ? ans1 : ans2;
12 }
13
14 int32_t main() {
15     for (int32_t i = 0; i < 20; ++i) {
16         for (int32_t j = 0; j <= i; ++j) {
17             scanf("%d", &num[i][j]);
18         }
19     }
20     printf("%d\n", GetMaxSum(0, 0, 20));
21     return 0;
22 }
```

代码讲解：

GetMaxSum函数计算得到从 i 层 j 列走到最底层(maxN)所能得到的最大值

ans1，为向左下走能得到的最大值

ans2，为向右下走能得到的最大值

第7行，当走到最后一层的时候，直接返回相应位置上的值

递归+记忆化程序

```
1  #include <stdio.h>
2  #include <inttypes.h>
3
4  int32_t f[22][22] = {0};
5  int32_t num[22][22] = {0};
6
7  int32_t GetMaxSum(int32_t i, int32_t j, int32_t maxN) {
8      if (i + 1 == maxN) return num[i][j];
9      if (f[i][j] != 0) return f[i][j];
10     int32_t ans1, ans2;
11     ans1 = GetMaxSum(i + 1, j, maxN) + num[i][j];
12     ans2 = GetMaxSum(i + 1, j + 1, maxN) + num[i][j];
13     f[i][j] = ans1 > ans2 ? ans1 : ans2;
14     return f[i][j];
15 }
16
17 int32_t main() {
18     for (int32_t i = 0; i < 20; ++i) {
19         for (int32_t j = 0; j <= i; ++j) {
20             scanf("%d", &num[i][j]);
21         }
22     }
23     printf("%d\n", GetMaxSum(0, 0, 20));
24     return 0;
25 }
```

代码讲解：

f 数组，用来做记忆化，f[i][j] 记录从代表从 i 层 j 列走到底层所能获得的最大值

第9行，如果之前计算过，直接使用

第13行，做记忆化

三. 动态规划基础

观察 F 数组

- 通过观察代码中 f 数组的更新过程，我们可以总结如下：

为了方便讨论，我们假设f[1]为第1层，f[20]为第20层

1. f[20][j] = num[20][j]
2. f[19][j] = max(f[20][j], f[20][j+1])+num[19][j]
3. f[i][j]=max(f[i+1][j], f[i+1][j+1])+num[i][j]

f[i]的计算只与f[i+1]的值相关，并且f[20]的值可以直接得到

综上所述，直接求得f[20]的值后，向前依次计算得到f[19]、f[18]…f[1]的值

这个过程中，不需要使用递归写法

动态规划基础--数字三角形

设 f[i][j] 代表从第 i 层第 j 列走到最底层(20层)，所能获得的最大的路径值

初始值： f[20][j] = num[20][j]

状态转移： f[i][j]=max(f[i+1][j], f[i+1][j+1])+num[i][j]

求解方向： i从19递减到1