例 2　数的划分

问题结构分析（给出问题的表示）：

用 dp[i][j]表示整数 i 分成 j 份，则原问题为求 dp[n][k]

举例：7 分成 3 份，即 n=7，k=3

数 7 分成 3 份的情况有：（1，1，5），（1，2，4），（1，3，3），（2，2，3）

1、前三种情况：（1，1，5），（1，2，4），（1，3，3）。

等价于求 dp[6][2]：(1，5)，（2，4），（3，3）。

所以前三种情况可以看成是：6 分成 2 份的情况下，再加 1 份为 1。

2、第四种情况：（2，2，3）。

可以看成是：把每一份减 1 得：（1，1，2）。这个划分刚好是 dp[4][3]的情况

因此，dp[7][3]=dp[6][2]+dp[4][3]

建立递推关系：

dp[i][j]=dp[i-1][j-1]+dp[i-j][j]

原问题 dp[n][k]=dp[n-1][k-1]+dp[n-k][k]

确定计算顺序：

计算顺序从左到右，从上到下

i, j	1	2	3	4	5	…	k
1	1	0	0	0	0	0	0
2		1	0	0	0	0	0
3			1	0	0	0	0
4				1	0	0	0
5				i-j, j	1	0	0

6						1	0
7			i-1, j-1				1
8				i, j			
9							
…							
n							

例 3 方格取数 动态规划

问题结构分析（给出问题的表示）：

向右

向下

A	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	13	0	0	6	0	0
3	0	0	0	0	7	0	0	0
4	0	0	0	14	0	0	0	0
5	0	21	0	0	0	4	0	0
6	0	0	15	0	0	0	0	0
7	0	14	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

B

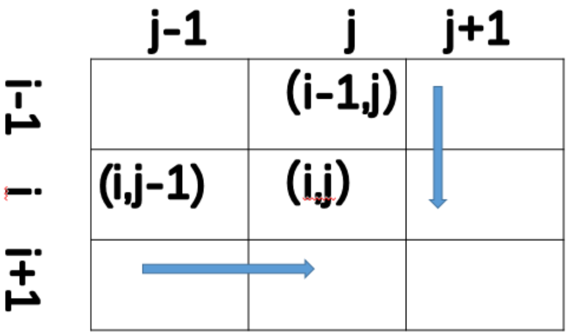
从 A 点出发到达 B 点，两个人同时走两条路径，使得取得的数之和最大。

(i,j)表示第一条路径走到的位置，(h,k)表示第二条路径走到的位置

dp[i][j][h][k]表示第一条路径在(i,j)时，第二条路径在(h,k)位置时，取得的最大数之和。

dp[0][0][0][0]=0，原问题为 dp[n][n][n][n]

建立递推关系：



当 i==h&&j==k 时，f[i][j][h][k]=

max{f[i-1][j][h-1][k],f[i][j-1][h][k-1],f[i-1][j][h][k-1],f[i][j-1][h-1][k])+map[i][j];//取上上，左左，上左，左上四个方向的最大值加上当前的值；

当 i!=h&&j!=k 时，f[i][j][h][k]=

max{f[i-1][j][h-1][k],f[i][j-1][h][k-1],f[i-1][j][h][k-1],f[i][j-1][h-1][k])+map[i][j]+map[h][k];//取上上，左左，上左，左上四个方向的最大值加上两条路径当前的值；

计算顺序：

改进策略：

i, j	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								
6								
7								
8								

每条对角线上对应单元格的坐标(x, y),对角线上单元格坐标 x+y 都相等，x+y 称为步数，用 k 表示，则问题可以表示为：dp[k][i][j]， k 表示当前所在位置步数（下标和，即 x+y），i 表示第一条路线所在位置的行，j 表示第二条路线当前所在位置的行,即 dp[k][i][j]