逻辑航线信息学奥赛系列教程

动态规划之整数拆分

力扣343：整数拆分

题目描述

给定一个正整数 n，将其拆分为至少两个正整数的和，并使这些整数的乘积最大化。 返回你可以获得的最大乘积。

示例1：

输入：2
输出：1
解释：2 = 1 + 1，1 × 1 = 1。

示例2：

输入：10
输出：36
解释：10 = 3 + 3 + 4，3 × 3 × 4 = 36。

解析

现在开始五步分析

1、确定dp数组及其下标的含义
拆分数字i，可以得到的最大乘积。

2、推导状态转移方程
最大乘积怎么得到的呢？
从1开始遍历j，有两种方法可以得到dp[i]
1、j x (i-j)，单纯的将i拆成了两个数字相乘。
2、j x dp[i-j]，将i拆成了两个以上的数字相乘。
dp[i] = max(dp[i],j x (i-j),j x dp[i-j])

3、初始化DP数组
比较令人困惑的是dp[0]和dp[1]的值该是多少呢？很明显，这两个值的拆分没有意义。因此，我们应该从dp[2]开始推导，2拆分后的最大乘积为1。因此dp[2] = 1。

4、确定遍历顺序
遍历顺序从前向后。

5、举例推导DP数组
当n为10的时候，开始进行推导，可以得出下图：

2	3	4	5	6	7	8	9	10
1	2	4	6	9	12	18	27	36

编码

```
#include <bits/stdc++.h>

using namespace std;

class Solution {
public:
    int dp[60];

    int integerBreak(int n) {
        //dp[0],dp[1]没有意义
       dp[2] = 1;
        //从3开始，遍历到n
        for (int i = 3; i <= n; ++i) {
           //这里的j<i-1原因如下
          //例如将5拆成1和4以及最后的4和1,结果是一样的。
          //因此，我们只需要拆分到3即可，即,j<i-1
            for (int j = 1; j < i - 1; ++j) {
                dp[i] = max({dp[i], (i - j) * j, dp[i - j] * j});
            }
        }
        return dp[n];
    }

};


int main() {
    Solution s{};
    int n;
    cin >> n;
    cout << s.integerBreak(n);
    return 0;
}
```