# 题目

给你一根长度为n的绳子，请把绳子剪成整数长度的m段（m、n都是整数，n>1并且m>1），每段绳子的长度记为k[0],k[1]...k[m-1] 。请问k[0]\*k[1]\*...\*k[m-1] 可能的最大乘积是多少？例如，当绳子的长度是8时，我们把它剪成长度分别为2、3、3的三段，此时得到的最大乘积是18。

**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

**提示：**

2 <= n <= 58

**注意：**本题与主站343题相同：

https://leetcode-cn.com/problems/integer-break/

# 分析

## 方法一：暴力破解

复杂度分析

时间复杂度：O(N^2)，对于每一个i调用一次递归，递归的时间复杂度为O(N)，故时间复杂度为O(N^2)。

空间复杂度：O(N^2)。

## 方法二：记忆化技术

**思路：**

**代码：**

**复杂度分析**

时间复杂度：O(N^2)，原因同上，时间复杂度仍然为 O(N^2)，只是采用记忆化减少部分计算时间。

空间复杂度：O(N)。使用了数组f。

（记忆化技术 相关题目：70.爬楼梯，509.斐波那契数）

记忆化搜索也叫“备忘录法”，它从类似上边树形图结构中的F(n)出发，逐步递归到已知值F(2)，可以理解成为自顶向上的解决办法。

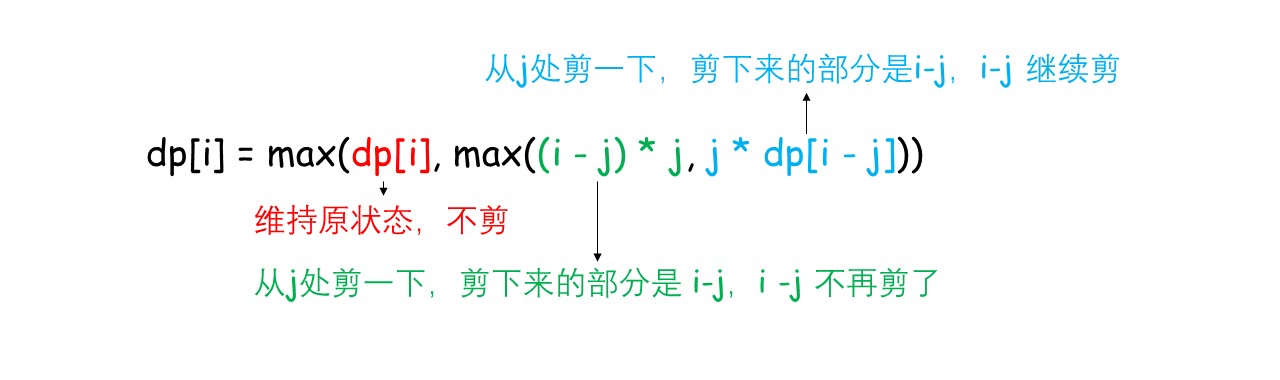
## 方法二：动态规划

**思路：**

建立一维动态数组dp：

边界条件：dp[1] = dp[2] = 1，表示长度为2的绳子最大乘积为 1；

状态转移方程：dp[i] = max(dp[i], max((i - j) \* j, j \* dp[i - j]))，可以这样理解：



**代码：**

class Solution {

public:

int cuttingRope(int n) {

int dp[60]={0};

dp[2]=1;

for(int i=3;i<=n+1;i++)

{

for(int j=1;j<i;j++)

{

dp[i]=max(dp[i],max((i-j)\*j,j\*dp[i-j]));

}

}

return dp[n];

}

};