# 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。

答案需要取模1e9+7（1000000007），如计算初始结果为：1000000008，请返回1。

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 <= 1000

注意：本题与主站343题相同：

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

# 分析

## 方法一：贪心法

class Solution {

public:

//n >= 5 2\*(n-2) > n 3\*(n-3) > n 且3\*(n-3) >= 2\*(n-2)

//n = 4 2 \* 2 > 1 \* 3

//2和3不能再分了 分了就变小了 且3优于2

int cuttingRope(int n) {

if (n <= 3) return n-1;

long rs = 1;

while (n > 4) {

//3最优

rs \*= 3;

rs %= 1000000007;

n -= 3;

}

//循环结束 n只剩下1, 2 ,3,4

//1不能再分

//2，3再分会标小

//4 可以分成1 \* 3 2 \* 2,所以还是4最优

//所以 剩下的1 2 3 4 都不能再分了

return (rs \* n) % 1000000007;

}

};

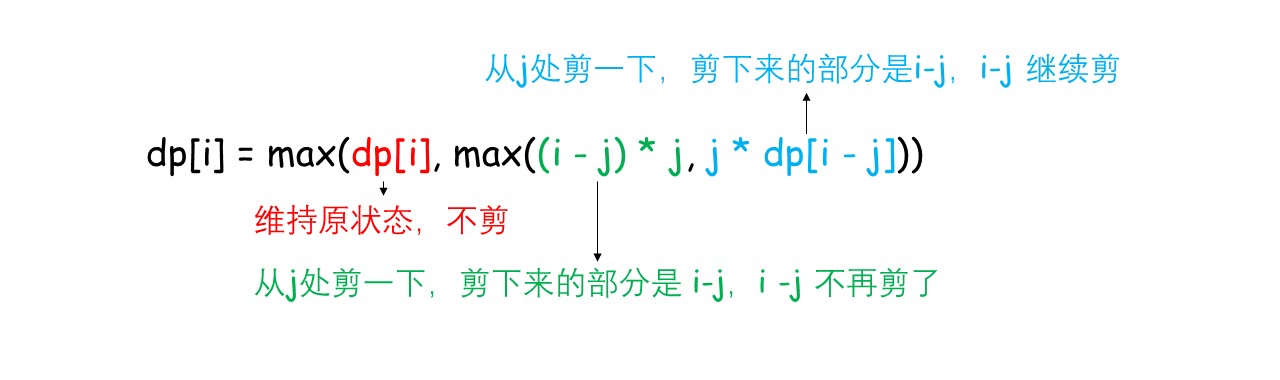
## 方法二：动态规划

**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 integerBreak(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];

}

};