第35题——剪绳子I

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

贪心算法(1)

思路:

```
对f(n)取余然后求幂次方
```

分成三种情况：

1、 余1，则取一个3补成4比较划算，与补1凑成2个2是一样的 2、 余2，则不补，因为前面说的，5不如拆成2和3划算 3、 余0

```
class Solution {
public:
    int cuttingRope(int n) {
        if(n<=3){
            return n-1;
        }
        if(n%3==2){
            return pow(3,n/3)*2;
        }
        else if(n%3==1){
            return pow(3,(n/3)-1)*4;
        }
        else{
            return pow(3,n/3);
        }
    }
};
```

动态规划(2)

有三种情况是比较特殊的:

1. 长度为1时，没法剪，最大乘积为0
2. 长度为2时，最大乘积为1 × 1 = 1
3. 长度为3时，最大乘积为1 × 2 = 2

所以我们先创建数组来存储这些值;

vector dp(n + 1, 0);

> dp[i]表示剪长度为i的绳子所能得到的最大乘积，dp[n]表示长度为n的绳子
>
> 所以数组的长度要为n+1

上面我们已经分析过了,长度<=3时,就不应该接着剪了,因为剪掉之后的乘积<长度n ;

因此,我们只循环>3的部分即可

> 外层循环i表示每一段要剪的绳子，去掉特殊情况从4开始
>
> 内层循环j表示将绳子剪成长度为j和i-j的两段
>
> dp[i]记录当前长度绳子剪过之后的最大乘积
>
> 1. j只需要遍历到i/2就可以了，两边对称的。比如4剪成 1|3 和 3|1 结果是一样的
> 2. 这样双层循环就相当于从下向上完成了剪绳子的逆过程
> 3. 剪绳子本来是将大段的绳子剪成小段，然后再在每小段上继续剪
> 4. 双层循环中外层循环从4开始一直到原始绳子长度n，每一段都到内层循环进行剪绳子
> 5. 这样就得到长度在[4, n]区间内的每段绳子剪过之后的最大乘积

> 最后我们返回绳子的最大乘积-->长度为n的最大乘积就是dp[n]

```
class Solution {
public:
    int cuttingRope(int n) {
        if(n<=3)
{
            return n-1;
        }
        vector<int> dp(n+1,0);//创建大小为n+1的数组并初始化为0
        dp[0]=0;dp[1]=1;dp[2]=2;dp[3]=3;

        for(int i=4;i<=n;i++){
            for(int j=1;j<=i/2;j++)
            {
                //循环i/2次,得到最大的乘积dp[i]
                dp[i]=max(dp[i],dp[j]*dp[i-j]);
            }
        }
        return dp[n];
    }
};
```