面试题 14：剪绳子

题目：给你一根长度为 n 的绳子，请把绳子剪成 m 段（m、n 都是整数，$n>1$ 并且 $m>1$），每段绳子的长度记为 $k[0],k[1],\cdots,k[m]$。请问 $k[0]\times k[1]\times\cdots\times k[m]$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。

- 动态规划
  - 递推公式

    ```
    f(n) = 0                    n = 1
    f(n) = 1                    n = 2
    f(n) = 2                    n = 3
    f(n) = max{dp(i) * dp(n-i)}  n > 3, 1<=i<=n-1
    ```

  - 注意：当 `n <= 3` 时因为必须剪至少一次的缘故，导致 `f(1)=0, f(2)=1*1=1, f(3)=1*2=2`；但是当 `n>=4` 时，将 `n<=3` 的部分单独作为一段能提供更大的乘积

    因此，初始化时应该 `dp[1]=1≠f(1), dp[2]=2≠f(2), dp[3]=3≠f(3)`，同时将 `f(1), f(2), f(3)` 单独返回
  - 时间复杂度：`O(N^2)`，空间复杂度：`O(N)`
- 贪心
  - 当 `n>=5` 时，尽可能多剪长度为 3 的绳子；当 `n=4` 时，剪成两段长度为 2 的绳子
  - 证明

    ```
    当 n >= 5 时，可以证明：  3(n-3) > 2(n-2) > n
    当 n == 4 时，2*2 > 3*1
    ```

  - 时间复杂度：`O(1)`，空间复杂度：`O(1)`

解法一 动态规划

$f(n)=max(f(i)f(n-i))=max(f(1)(n-1),f(2)f(n-2),,,,,,)$

```
int maxProduct(int n)
{
    if(n<2)
        return 0;
    if(n==2)
        return 1;
    if(n==3)
        return 2;
    vector<int> dp(n+1,0);
    dp[0]=0;
    dp[1]=1;
    dp[2]=2;
    dp[3]=3;
    for(int i=4;i<=n;i++)
    {
        int t=0;
        for(int j=1;j<=i/2;j++)
        {
            t=max(t,dp[j]*dp[i-j]);
            dp[i]=t;
        }
    }
```

```
    return dp[n];
}
```

解法二 贪心

n=0　　0

n=1　　0

n=2　　1

n=3　　2

n=4　　4

n=5　　2x3>1x4

n=6　　3x3>2x4>1x5

规律：尽可能先剪出长度为3的绳子，当最后绳子长度为4时，不剪出3，剪成2x2

```
int maxProduct(int n)
{
    if(n<2)
        return 0;
    if(n==2)
        return 1;
    if(n==3)
        return 2;
    int count3=n/3;
    if(n-count3*3 == 1)//说明剩下有一个4
        count3--;
    int count2=(n-count3*3)/2;
    return (pow(3,count3)*pow(2,count2));
}
```