# 14. 剪绳子

## 题目链接

[牛客网](https://www.nowcoder.com/practice/57d85990ba5b440ab888fc72b0751bf8?tpId=13&tqId=33257&tab=answerKey&from=cyc_github)

## 题目描述

把一根绳子剪成多段，并且使得每段的长度乘积最大。

n = 2
return 1 (2 = 1 + 1)

n = 10
return 36 (10 = 3 + 3 + 4)

## 解题思路

### 贪心

尽可能得多剪长度为 3 的绳子，并且不允许有长度为 1 的绳子出现。如果出现了，就从已经切好长度为 3 的绳子中拿出一段与长度为 1 的绳子重新组合，把它们切成两段长度为 2 的绳子。以下为证明过程。

将绳子拆成 1 和 n-1，则 1(n-1)-n=-1<0，即拆开后的乘积一定更小，所以不能出现长度为 1 的绳子。

将绳子拆成 2 和 n-2，则 2(n-2)-n = n-4，在 n>=4 时这样拆开能得到的乘积会比不拆更大。

将绳子拆成 3 和 n-3，则 3(n-3)-n = 2n-9，在 n>=5 时效果更好。

将绳子拆成 4 和 n-4，因为 4=2\*2，因此效果和拆成 2 一样。

将绳子拆成 5 和 n-5，因为 5=2+3，而 5<2\*3，所以不能出现 5 的绳子，而是尽可能拆成 2 和 3。

将绳子拆成 6 和 n-6，因为 6=3+3，而 6<3\*3，所以不能出现 6 的绳子，而是拆成 3 和 3。这里 6 同样可以拆成 6=2+2+2，但是 3(n - 3) - 2(n - 2) = n - 5 >= 0，在 n>=5 的情况下将绳子拆成 3 比拆成 2 效果更好。

继续拆成更大的绳子可以发现都比拆成 2 和 3 的效果更差，因此我们只考虑将绳子拆成 2 和 3，并且优先拆成 3，当拆到绳子长度 n 等于 4 时，也就是出现 3+1，此时只能拆成 2+2。

public int cutRope(int n) {
 if (n < 2)
 return 0;
 if (n == 2)
 return 1;
 if (n == 3)
 return 2;
 int timesOf3 = n / 3;
 if (n - timesOf3 \* 3 == 1)
 timesOf3--;
 int timesOf2 = (n - timesOf3 \* 3) / 2;
 return (int) (Math.pow(3, timesOf3)) \* (int) (Math.pow(2, timesOf2));
}

### 动态规划

public int cutRope(int n) {
 int[] dp = new int[n + 1];
 dp[1] = 1;
 for (int i = 2; i <= n; i++)
 for (int j = 1; j < i; j++)
 dp[i] = Math.max(dp[i], Math.max(j \* (i - j), dp[j] \* (i - j)));
 return dp[n];
}