```swift
import Foundation
/// **剑指 Offer 14- I. 剪绳子**
/// 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。
class Solution {
    /// **将绳子以相等的长度等分为多段 ，得到的乘积最大。**
    /// **尽可能将绳子以长度 3 等分为多段时，乘积最大。**
    /// 切分规则：
    /// 最优：3 把绳子尽可能切为多个长度为 3 的片段，留下的最后一段绳子的长度可能为 0,1,2 三种情况。
    /// 次优：2 若最后一段绳子长度为 2 则保留，不再拆为 1+1 。
    /// 最差：1 若最后一段绳子长度为 1 则应把一份 3+1 替换为 2+2，因为 2×2>3×1。
    func cuttingRope(_ n: Int) -> Int {
        /// 当原始长度小于等于 3 时候口算得最大乘积为 n-1
        if (n <= 3) { return n-1 }
        let threeNumsCount: Int = Int(n/3)
        let remainNumber: Int = Int(n%3)
        if remainNumber == 1 {
            /// 最差的情况需要将 3+1 替换为 2+2
            return Int(pow(3.0, Double(threeNumsCount-1)) * 4)
        } else if remainNumber == 2 {
            /// 次优：2 若最后一段绳子长度为 2 则保留，不再拆为 1+1 。
            return Int(pow(3, Double(threeNumsCount)) * 2)
        } else {
            return Int(pow(3, Double(threeNumsCount)))
        }
    }
}

let solution = Solution()
print(solution.cuttingRope(2))
print(solution.cuttingRope(10))
```