```
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。
/// 答案需要取模 1e9+7(1000000007),如计算初始结果为: 1000000008,请返回 1。
/// **n的取值变为 [2, 1000]**
class Solution {
   func cuttingRope( n: Int) -> Int {
       if (n \le 3) { return n-1 }
       let threeNumsCount: Int = Int(n/3)
       let remainNumber: Int = Int(n%3)
       if remainNumber == 1 {
           return customPower(3, threeNumsCount-1) * 4 % (Int(1e9)+7)
       } else if remainNumber == 2 {
           return customPower(3, threeNumsCount) * 2 % (Int(1e9)+7)
       } else {
           return customPower(3, threeNumsCount)
       }
   }
   func customPower(_ a: Int, _ b: Int) -> Int {
       var res: Int = 1
       for _ in 0..<b {
           res = (res * a) % (Int(1e9)+7)
       }
       return res
   }
   /// 快速幂运算
   func quickPow(_ a: Int, _ n: Int) -> Int {
       var a: Int = a
       var res: Int = 1
       var _n: Int = _n
       while (n > 0) {
           /// 判断 n的奇偶性
           if (n \& 1 == 1) \{ res = (res * a) \% (Int(1e9)+7) \}
           _n >>= 1
           _a = (_a * _a) % (Int(1e9)+7)
       }
       return res
   }
}
let solution = Solution()
print(solution.cuttingRope(2))
print(solution.cuttingRope(127))
```