剑指14-I 剪绳子

动态规划或贪心算法

动态规划

我一开始认为的态转移方程：

dp[n] = max(dp[n-1]*1, dp[n-2]*2 .... ,dp[1]*(n-1));

忽视的一点，就是内部有可能不剪的情况。比如只剪1刀的结果不是dp[n-1]*1，因为dp[n-1]是必须减了一刀的，所以还要加一种情况是没剪： (n-1)*1

应该是：

dp[n] = max(dp[n], max(x * dp[n - x], x * (n - x))) // 也可以把第一个max函数的逻辑分离。

解析：

我们先把绳子剪掉第一段（长度为j），如果只剪掉长度为1，对最后的乘积无任何增益，所以从长度为2开始剪。

剪了第一段x后，剩下(n - x)长度可以剪也可以不剪。

不剪的话结果为：x * (n - x)

剪的话结果为：x * dp[n - x]

最后，由于x是可变的，所以我们要取结果最大的选择，更新dp[n]的时候每次都要和自己比一下。

总结：无论如何复杂度都是O(n^2)，动态规划也是可能有双循环的。

贪心算法

其实就是分成最多的3，需要提前知道这个推论的是有数学证明的。//不知道的话做不了

```cpp
// 标准动态规划
class Solution {
public:
    int cuttingRope(int n) {
        vector<int> dp(n + 1, 0);
        dp[2] = 1;
        for(int i = 3; i <= n; i++) {
            int tmp = 0;
            for(int j = 2; j < i; j++) {
                tmp = max(j * (i - j), j * dp[i - j]);
                if(ans > dp[i]) {
                    dp[i] = tmp;
                }
            }
        }
        return dp[n];
    }
};
```