戳气球

有 `n` 个气球，编号为 `0` 到 `n-1`，每个气球上都标有一个数字，这些数字存在数组 `nums` 中。

现在要求你戳破所有的气球。如果你戳破气球 `i` ，就可以获得 `nums[left] * nums[i] * nums[right]` 个硬币。 这里的 `left` 和 `right` 代表和 `i` 相邻的两个气球的序号。注意当你戳破了气球 `i` 后，气球 `left` 和气球 `right` 就变成了相邻的气球。

求所能获得硬币的最大数量。

说明:

- 你可以假设 `nums[-1] = nums[n] = 1`，但注意它们不是真实存在的所以并不能被戳破。
- 0 ≤ `n` ≤ 500, 0 ≤ `nums[i]` ≤ 100

示例:

```
输入: [3,1,5,8]
输出: 167
解释: nums = [3,1,5,8] --> [3,5,8] -->   [3,8]   -->  [8]  --> []
     coins =  3*1*5      +  3*5*8    +  1*3*8      + 1*8*1   = 167
```

思路：
动态规划
假设现在就剩下气球 k，这个是最后被戳爆的，没球了，只有 开区间的首尾 i 和 j 了
现在假设dp[i][j]是开区间（i，j）能够拿到的最多的金币：
dp[i][j] = dp[i][k](i到k这个区间能拿到的金币数) + val[i] * val[k] * val[j] + dp[k][j](k到j这个区间能拿到的金币数)
因为在i和j之间有很多的k，所以选择其中最大的

```
def maxCoins(self, nums: List[int]) -> int:
        n = len(nums)
        %23 n+2，是因为加了nums[0] 和 nums[n]
        dp = [[0] * (n + 2) for _ in range(n+2)]
        val = [1] + nums + [1]

        %23 这里不懂了～～～
```

```
        for i in range(n - 1, -1, -1):
            for j in range(i + 2, n + 2):
                for k in range(i + 1, j):
                    total = val[i] * val[j] * val[k]
                    total += dp[i][k] + dp[k][j]
                    dp[i][j] = max(dp[i][j], total)
        return dp[0][n + 1]
```

时间复杂度O(N^3)
空间复杂度O(N^2)