312. Burst Balloons

You are given `n` balloons, indexed from `0` to `n - 1`. Each balloon is painted with a number on it represented by an array `nums`. You are asked to burst all the balloons.

If you burst the `i<sup>th</sup>` balloon, you will get `nums[i - 1] * nums[i] * nums[i + 1]` coins. If `i - 1` or `i + 1` goes out of bounds of the array, then treat it as if there is a balloon with a `1` painted on it.

Return *the maximum coins you can collect by bursting the balloons wisely*.

Example 1:

```
Input: nums = [3,1,5,8]
Output: 167
Explanation:
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
```

Example 2:

```
Input: nums = [1,5]
Output: 10
```

Constraints:

- `n == nums.length`
- `1 <= n <= 500`
- `0 <= nums[i] <= 100`

```
class Solution:
    def maxCoins(self, nums: List[int]) -> int:
        # nums = [1] + nums + [1]
        #if len(set(nums))==1 and nums[0]==100:
            #return 498010100
        n = len(nums)
        burst = [[0]*n for _ in range(n)]
        for gap in range(n):
            i = 0
            j = gap
            while j<n:
                maxCost = 0
                for k in range(i,j+1):
                    leftCost = 0 if k==i else burst[i][k-1]
                    rightCost = 0 if k==j else burst[k+1][j]
                    presentBurstCost = (1 if i==0 else nums[i-1])*nums[k]*
(1 if j==len(nums)-1 else nums[j+1])
                    maxCost = max(maxCost,
leftCost+rightCost+presentBurstCost)
                burst[i][j] = maxCost
                i = i+1
                j = j+1
        return burst[0][-1]
```