1770: Maximum Score for Performing Multiplication Operations

Ex nums = [1, 2, 3] multiplier = [3, 2, 1]

choose from start/end of "nums" and multiply with start of multiplier.
Return the maximum multiplication possible.

1st iteration: nums [1, 2, 3] mul - [3, 2, 1]

end

$3 \times 3 = 9$

previous iteration numbers are omitted

2nd iteration: nums [1, 2] mul = [2, 1]

end

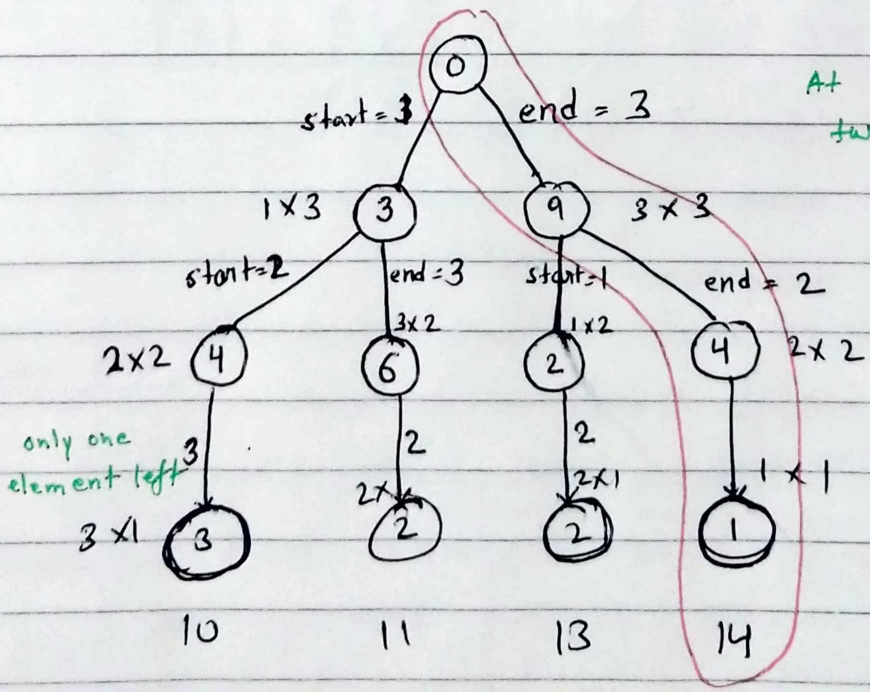
$= 2 \times 2 = 4$

3rd iteration: nums [1] mul [1]

$= 1 \times 1 = 1$

$\Rightarrow 9 + 4 + 1 = 13$

Brute-force **Decision Tree**



At each stage we have two choice, start/end

If we look number of state variables required -

start , end , pos (current position in multiplier)

① 3 - state variables

② Recurrence relation .

$$F(start, end, pos) = \max \begin{cases} \boxed{nums[start] \times mul[pos] + F(start+1, end, pos+1)} \\ \boxed{nums[end] \times mul[pos] + F(start, end+1, pos+1)} \end{cases}$$

when start is choosen

when end is choosen

③ Base case

$i ==$ length (multiplier) : return 0

It works with memorization, Top-down

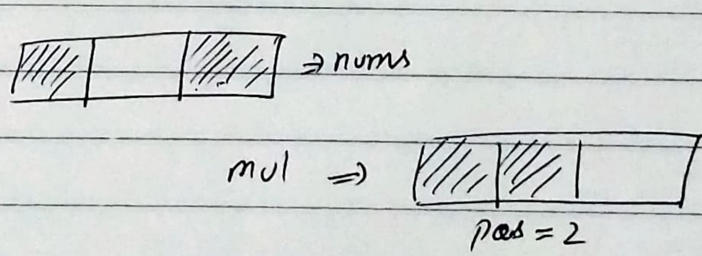
How to convert this to bottom- up approach

1) Reduce state variables

⇒ 'end' can be represented through 'pos' & 'start'

start and end signify the position in 'nums'.

⇒ The number of operations can never be more than pos (



$\boxed{end = nums.length - 1 \ (pos - start)}$

2) ∴ we have 2 state variables now.

it can be represented in 2D array.

The base case is pos == mul.length:

∴ DP should go in reverse.

DP[0][0] ⇒ should be the answer.

```
DP = [[0] x (m+1) for _ in range(m+1)]

for pos in range(m-1, -1, -1):
    for start in range(i, -1, -1):
        end = n - 1 - (pos - start)
        DP[pos][start] =
            max( nums[start] x mul[pos] + DP[pos+1][start+1]
                 nums[end] x mul[pos] + DP[pos+1][start]

return DP[0][0]
```