[Best Time to Buy and Sell Stock III](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/) - similar to II just add cap here f(ind, buy, cap)

Task: get the max profit by buying and selling the stocks max 2 times but it should be in a sequence first buy then sell.

**B S B S => 2 transactions**

**Profit:** selling price - buying price; so take buying price -ve and selling price +ve.

**Recursive Approach:**

Since transactions are fix here so keep a cap= 2, and after every transaction(one buy + one sell) do cap-1;

1. On any day I should have an idea, is this turn to buy or sell the stocks so for that we will pass a parameter buy: 0/1

1-> buy; 0-> not-buy

1. If (buy==1) that means we can buy here: 2 possibilities buy and not buy

-> buy: take which gives the max profit

max(-prices[ind] + f(ind+1, 0, cap), 0+f(ind+1, 1, cap))

1. Else we have chance to sell it: 2 possibilities sell and not-sell; selling means we have completed 1 transaction

-> sell: take which gives the max profit

max(prices[ind]+ f(ind+1, 1, cap-1), 0+f(ind+1, 0, cap))

**Tabulation:** using dp[n][2][3] & dp[2][3]

make dp table in reverse ind= n-1 to 0, buy = 0 to 1, cap = 1 to 2

1. Make dp[n+1][2][3]; //(n==0 || cap==0) return 0;
2. Use base case and the recurrence relation to build dp table
3. Ans will be stores at dp[0][1][2]

**Using transactions:** use ahead[4] & curr[4]

1. There are total 4 transactions **B S B S** => 0 1 2 3
2. Ahead will store [0 0 0 0 0] initially. //base: ahead[4] = 0;
3. Curr will be calculated using ahead(tabulation- space optimized)
4. Ans will be stored at ahead[0]