18/10/25

Task (13):- Finding the winning strategy in a cardgate.

Aim:- To a row of n cards with integer values, two players alternately pick either the left most or rightmost card.

Algorithm:-

We compute dp[i][j] = maximum score differen-ce the current player can over the oppone-nt when playing optimally on the subarray cards[i:j];

```
d[i][j] : max(
    cards[i] - dp[i+1][j], # take left : you gail cards[i], then opponent's best diff is dp[i+1]
    cards[j] - dp[i][j-1] # take right.
)
```

If dp[0][n-1] > 0 → player 1 wins.

If dp[0][n-1] == 0 → tie.

If < 0 → player 2 wins.

Program:-

```
def optimal_cardgame(cards):
    n=len(cards)
    #dp[i][j] = max score.
difference current player can achieve on cards [i..j]
    dp= [[0]*n for _ in range(n)]
    # fill base cases
    for in range(n):
        dp[i][i] = cards[i].
```

Output:-

To output the actual sequence of player 1 moves, reconstruct choices by comparing the two options at each step using the ~~tap~~ DP table.

Test case 1: cards = [5, 3, 7, 10]

Winner: player 1

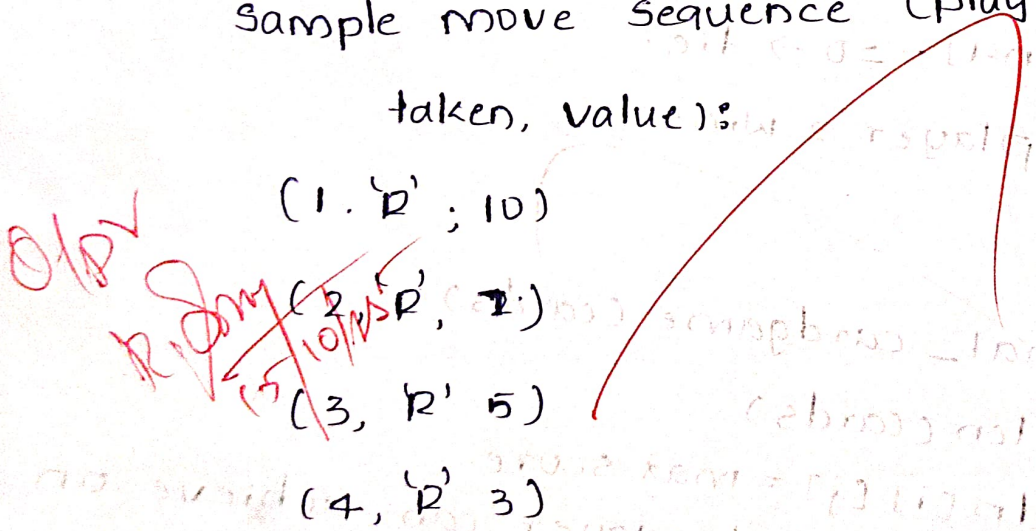
Sample move sequence (player, side taken, value):

(1, 'R', 10)

(2, 'R', 7)

(3, 'R' 5)

(4, 'R' 3)



Score from reconstructed moves:

{1:15, 2: 10}

```
# length 2..n
for length in range (2, n+1):
    for i in range (0, n-length+1):
        j = i + length - 1
        take_left = cards[i] -
d[i+1][j]
        take_right = cards[j] -
d[i][j-1]
        d[i][j] = max(take_left, take_right)
Print(f"Move {k+1}: take {side} (value
    {val})")
Print("Final score (assuming reconstruction
    above):")
Print("player1:",
    result["player1_score"])
```

VEL TECH - CSE	
EX NO.	13
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	15/10/25

Result:- Thus a row of n cards with left most and right most card integer values.