NO.	PROBLEMS	Date	Subimission Link	DESCRIPTION	SOLUTION	TYPE	History
1	C- Bouncing Ball		https://codeforces.com/contest/1457/problem/C	in a game.. There is a row of cells. We throw a ball so that it lands on a cell p which is a platform and bounced to p+k cell and then p+2*k and so on. To add a platform in some empty cells cost is x seconds and to remove the first cell cost is y second. initially we are given the info of which cell contains platform. What is the minimum second needed so that the ball passes the whole row.	we initialize for all cells dp[i] = 1- p[i]; p[i] contains whether there is a platform in the cell or not. Recurrence relation is dp[i] = (dp[i] + dp[i+k]) if (i+k < n); dp[i] means how many platform do we need if the ball first lands at cell i. Hence the answer is min((i-p)*y + dp[i] * x) over all 1<= i <= n		
2	AC E- Queen on Grid		https://atcoder.jp/contests/abc183/tasks/abc183_e	Given a grid of H*W. A Queen is in cell (1, 1) . In one move it can go any number of cells towards right, downwards and diagonally (lower-right). The grid contain walls. The queen cannot jump over walls. What is the number of ways the queen can go from (1, 1) to (H,W) using it moves. Count it modulo 1e9 + 7	we have to use DP and suffix sum method to solve it. Let the state dp(i, j) = number of ways it can go from (i, j) cell to (H, W). We have to take three additional suffix array called suff_hor[], suff_ver[], suff_dia[] to calculate the suffix number of moves of (i, j) to (h, w). From (i, j) the queen can go to right , down and diagonally . **suff_hor[i+1][j] means number of ways if the queen go to the right cell and then go anywhere**. so the recurrence relation is dp[i][j] = (suff_hor[i+1][j] + suff_ver[i][j+1] + suff_ver[i+1][j+1]) % MOD; **the base case is dp[h][w] = 1** because there is 1 way to go from (h,w) to (h, w). we have to also take care of the suffix arrays when the base case occurs. we have to also update suffix array. Suff_hor[i][j] = dp[i][j] + suff_hor[i+1][j]... If the the grid contain walls than we just continue.	DP, Prefix_sum	A lot of things I learned. First, my bruteforce solution was giving TLE because it was o(n^3) . Then from demoralizer I came to know about the usage of suffix / prefix(depends on imple) array to count the ways and store and reuse them. Then when performing modulas i performed dp[i][j](op1 + op2 + op3) % MOD which was wrong.
3			https://atcoder.jp/contests/abc188/tasks/abc188_e				
4	C. Longest Simple Cycle		https://codeforces.com/contest/1476/submission/110012810	Given n chains. Each chain consists of c[i] veritces . Every ith vertex is connected to i+1 th vertex with an edge. You have two array a[n] and b[n]. For 2 <= i <= n .. 1st vertex of ith chain is connected to a[i] the vertex of the i-1 th chain. Last vertex is of ith chain is connected to b[i] th vertex of the i-1 th chain. Now calculate the length of longest simple cycle.	**dp[i] = length of the longest simple cycle ending at ith chain .** **Base Case : Dp[1] = 0 ;** because we cannot have any cycle ending at 1st chain. To calculate the longest simple cylce until we will always take all the edges lies in the ith chain that is (c[i] -1) . Transition : **if : a[i] == b[i]** that means the cycle is closed at i-1th chain and **dp[i] = c[i] - 1 + 2;** else : **dp[i] = c[i] - 1 + 2 + max(dp[i - 1] - abs(b[i] - a[i]), abs(b[i] - a[i]))** **ANSWER : max(dp[i]) ;; 1 <= i <= n**	dp	Moja paisi .. 1st e vabi e nai dp diye hobe. After watching CWD's YT vdo I was baffled. Transition ta xoss chilo.
5	**B - Array Walk**		https://codeforces.com/contest/1389/submission/113045161	Initially you are at 1st index . Your score is a[1] .. You have two operations : 1) You go right and add a[i+1] 2) you go left and add a[i-1] but you can make atmost 1 left move at a time. Besides in total you cannot make more than 'z' left moves. What is the max score you can get after k moves..	dp[i][left][prv] --> maximum score until i using left numbers "left" moves with prveious move as 0(right) .. 1 (left).. **Transition : if(prv == 0) dp[i][left][prv] = max(a[i] + dp[i+1][left][0] , a[i] + dp[i-1][left + 1][1]) else dp[i][left][prv] = dp[i+1][left][0] ...** ***Base Case : if(move == k) return a[i];***	greedy**(Online Counting**), dp, bruteforce	Here constraints are very important such as 1 <= k <= (n-1) .. Z <= min(5, k) (I didn't notice that first)
6	D. Explorer Space	27.04.21	https://codeforces.com/contest/1517/submission/114345055		**dp[i][j][k] ---> minimum number of boredness after k moves if we start from (i, j) Transiiton : dp[i][j][move] =min (dp[i+1][j][move + 1] + row[i][j] , dp[i-1][j][move + 1] + row[i-1][j], dp[i][j+1][move + 1] + col[i][j], dp[i][j-1][move +1] + col[i][j-1]); Base Case : if (move == (k/2)) return 0; Ans :- dp[i][j][k];**		In this contest Asad Became Expert.
7	D. Armchairs	24.05.21		There is an array of 0 and 1. An operarion is to move a 1 to the position that is 0. and cost is abs(pos(1) - pos(0)). There are atmost n/2 1s in the array. What is minimum cost to shift all 1 to 0. You should free the positions that contained 1 initially.	we can grab the positions of	dp	
8							
9							