Tuesday, 2 November 2021 8:39 PM

Dynamic Programming

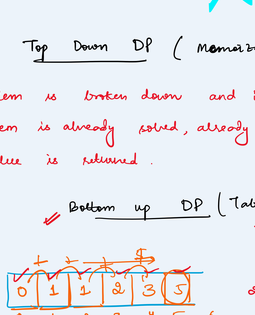
Break-down the problem into sub-problems and save the result for the future so that we don't have to compute the same problem again & again.

* Dynamic programming can be applied to any such problem that requires the re-calculation of certain values to reach the final solution.

Q Fibonacci

0 1 1 2 3 5 8 13 ...
0 1 2 3 4 5 6 7

n = 4 → 3
n = 7 → 13

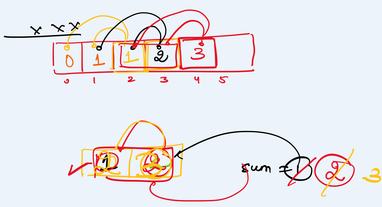


Top Down DP (Memorization)

problem is broken down and if the problem is already solved, already then calculated value is returned.

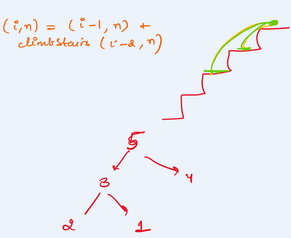
Bottom up DP (Tabulation)

Solution to the smaller problems are calculated which builds up the solution to overall problem.

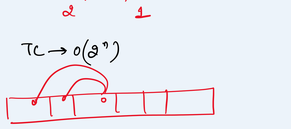


Q Climbing stairs

climbStairs(i,n) = climbStairs(i-1, n) + climbStairs(i-2, n)



TC → O(2^n)

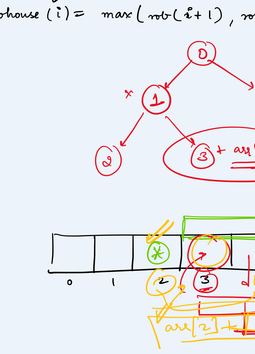


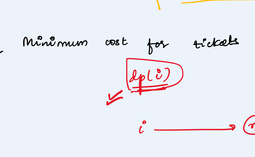
Q House robber

① arr → [1, 2, 3, 1]
ans → 4

② arr → [2, 7, 9, 3, 1]
ans → 12

robHouse(i) = max(rob(i+1), rob(i+2) + nums[i])





Q Minimum cost for tickets

dp(i)

i → n

dp(i) = min(dp(i+1) + cost[0], dp(i+7) + cost[1], dp(i+30) + cost[2])