Dynamic Programming

Video - 60

Note :- This playlist is only for explanation of Qns & solutions.

See my **DP Concepts & Qns** playlist for understanding DP from Scratch...

codestorywithMIK

Facebook
Instagram → codestorywithMIK

Twitter → cswithMIK

Leetcode - 403

~~HARD~~ → Easy **Simple Recursion + Memoization**

Frog Jump

Company :- Google amazon Meta

403. Frog Jump

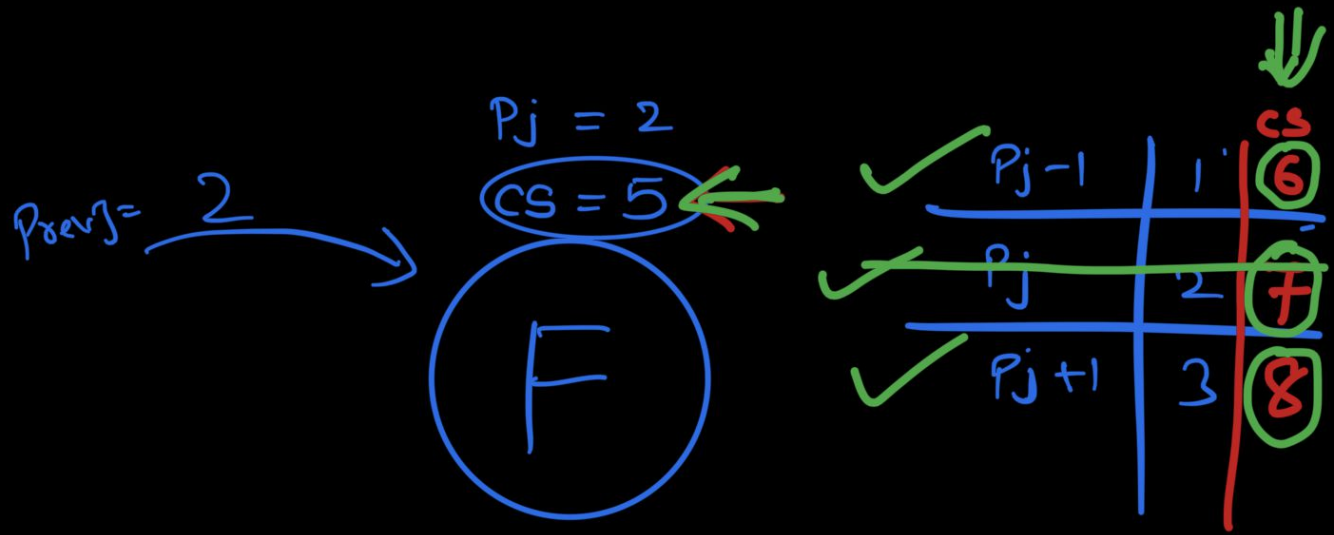
Hard 4282 206 Add to List Share

A frog is crossing a river. The river is divided into some number of units, and at each unit, there may or may not exist a stone. The frog can jump on a stone, but it must not jump into the water.

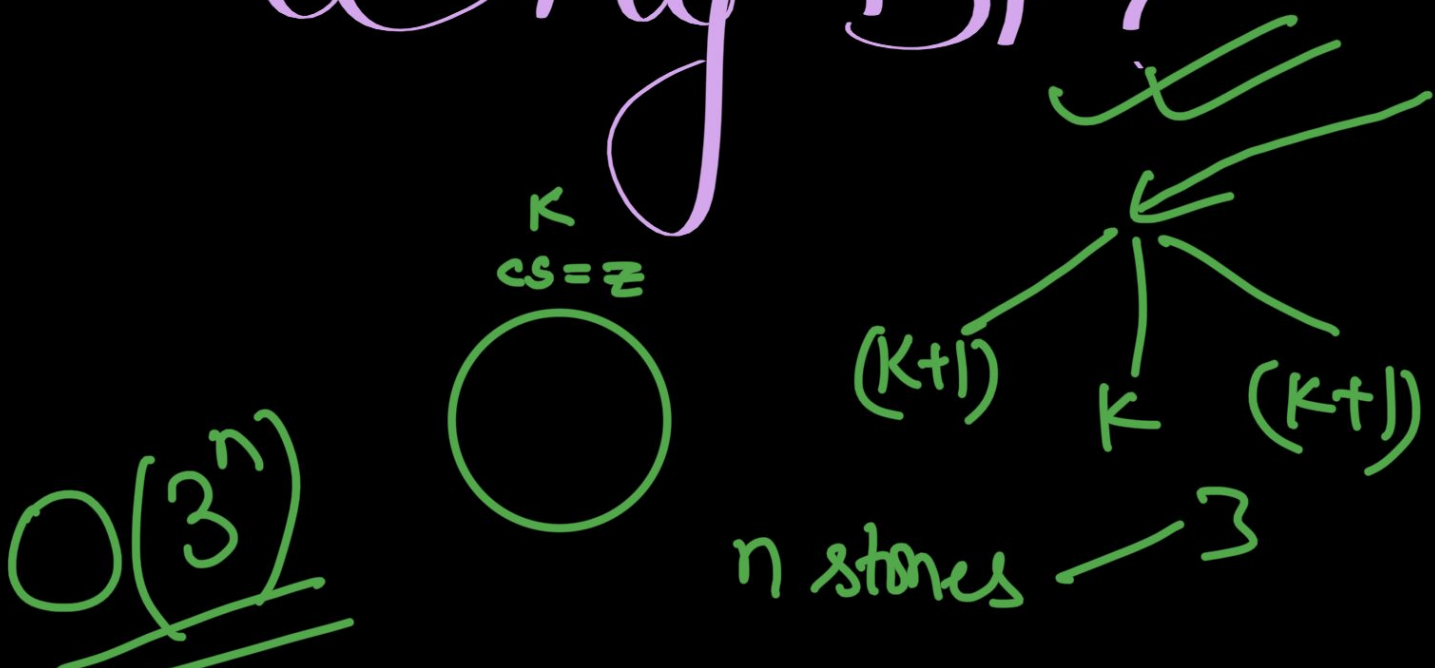
Given a list of `stones` ' positions (in units) in sorted **ascending order**, determine if the frog can cross the river by landing on the last stone. Initially, the frog is on the first stone and assumes the first jump must be `1` unit.

If the frog's last jump was `k` units, its next jump must be either `k - 1` `k` or `k + 1` units. The frog can only jump in the forward direction.

Example :- Stones = {0, 1, 3, 5, 6, 8, 12, 17}

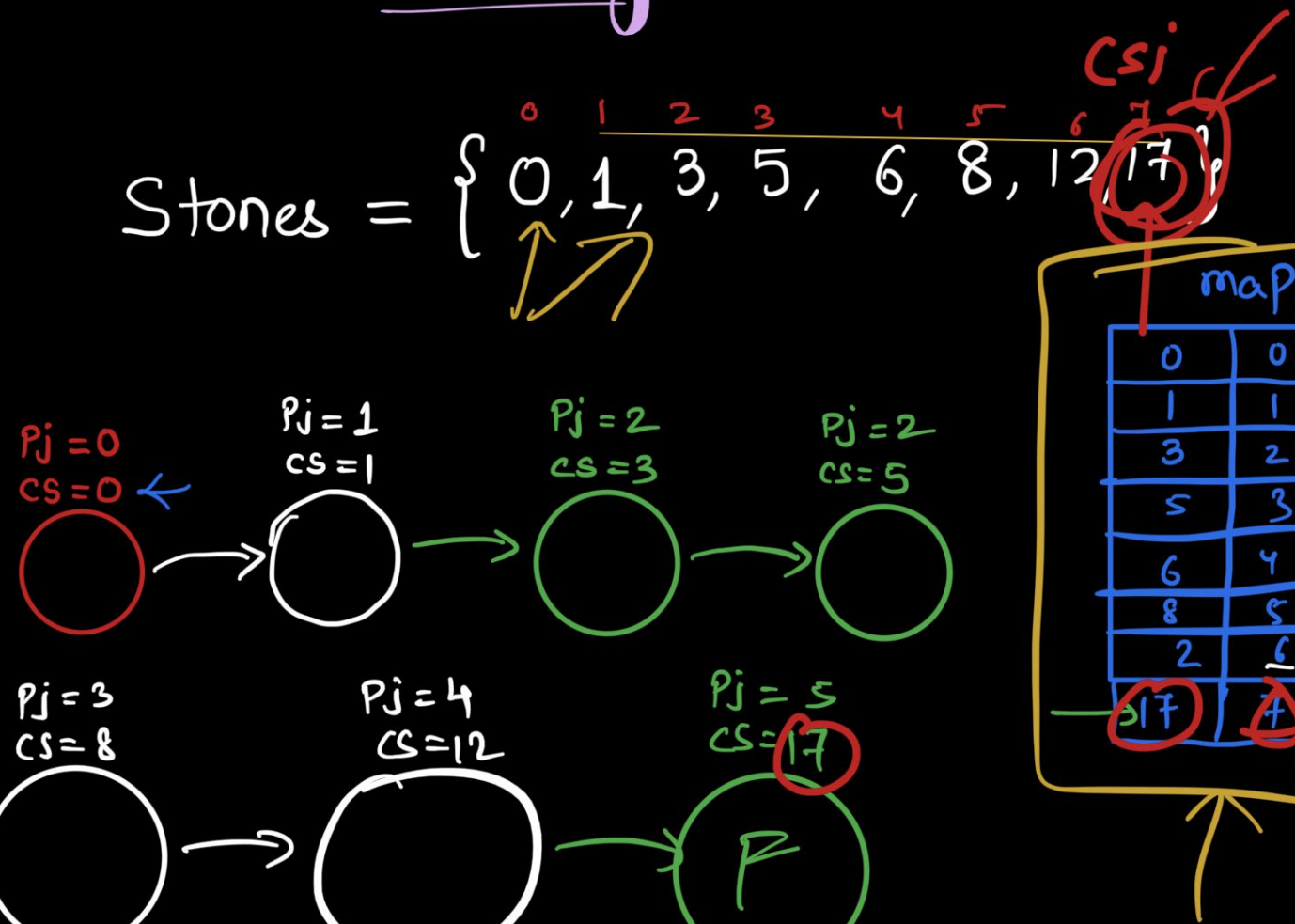


Why DP?



Dry Run :-

Dry Run :-



if (curr_stone_ind == n-1)

we Jump:

Story To Code :-

Story To Code:

Solve(mp[0], 0)

map

t[ ][ ];

solve(int curr_stone_idx, int prevJump) {

if (curr_stone_idx == n-1) {

return True;

}

bool result = False;

for (int nextJump = prev_jump-1 ; nextJump <= prevJ+1 ; nextJ++) {

if (nextJump > 0) {

int next_stone = stones[curr_stone_idx] + nextJump ;

```
if (curr_stone_idx == n-1) {
    return true;
}

bool result = false;
for (int nextJump = prevJump-1; nextJump <= prevJump+1; nextJump++) {
    if (nextJump > 0) {
        int next_stone = stones[curr_stone_idx] + nextJump;
        if (mp.find(next_stone) != mp.end())
            result = result ||
                solve(mp[next_stone],
```