Dynamic Programming

Video - 96

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

See my "DP Concepts & Qns" playlist for understanding DP from scratch...

Facebook
Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

Leetcode - 552

~~Hard~~

552. Student Attendance Record II

Hard | Topics | Companies

Google

An attendance record for a student can be represented as a string where each character signifies whether the student was absent, late, or present on that day. The record only contains the following three characters:

- `'A'`: Absent.
- `'L'`: Late.
- `'P'`: Present.

Any student is eligible for an attendance award if they meet **both** of the following criteria:

- The student was absent (`'A'`) for **strictly** fewer than 2 days **total**.
- The student was **never** late (`'L'`) for 3 or more **consecutive** days.

$0, 1 < 2$

$0, 1, 2 < 3$

Given an integer `n`, return *the **number** of possible attendance records of length* `n` *that make a student eligible for an attendance award. The answer may be very large, so return it **modulo*** `10^9 + 7`.

Example :- $n = 2$

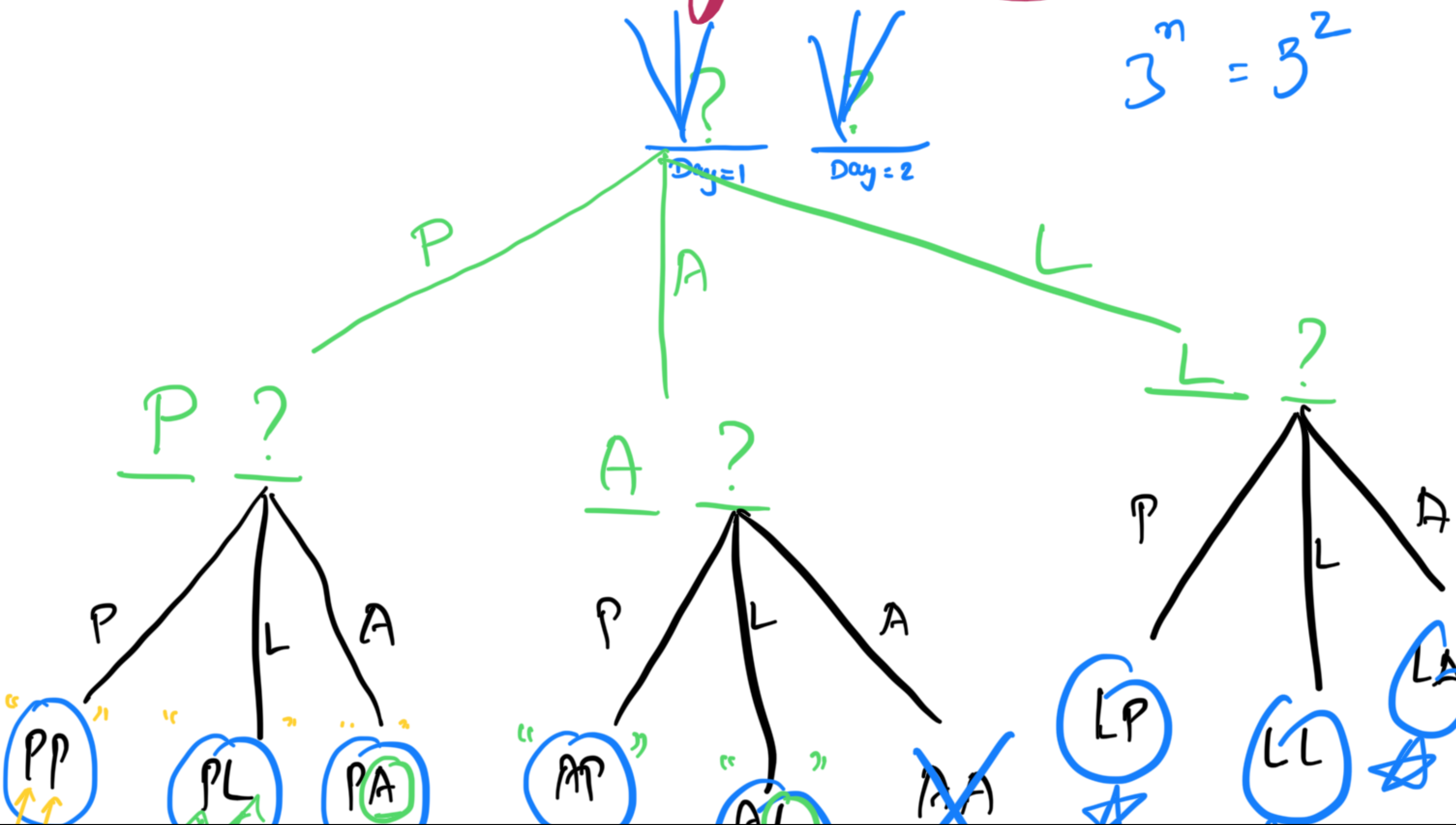
~~AA~~

Output = 8

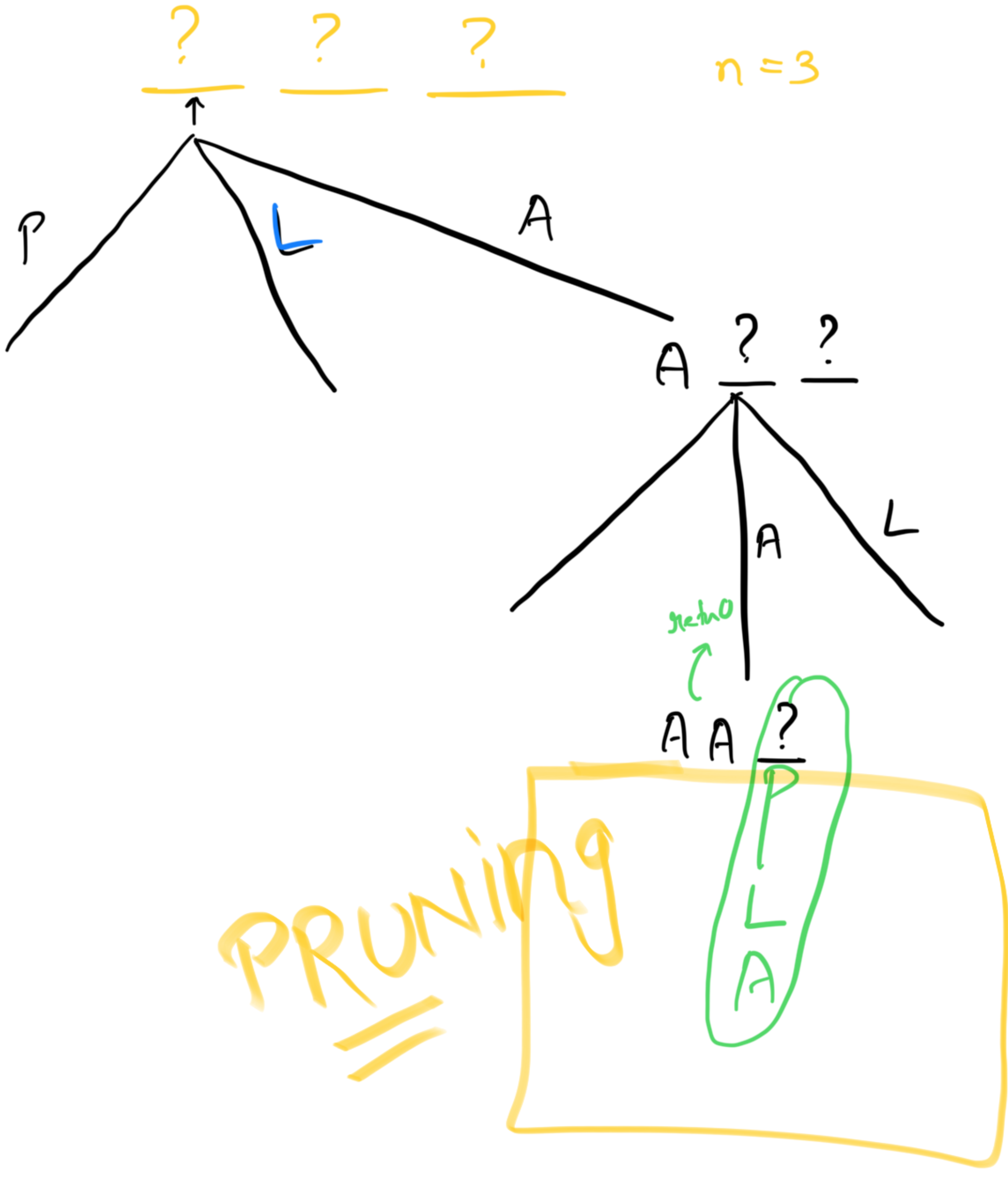
PP, PA, PL, LP, LA, LL, AP, AL

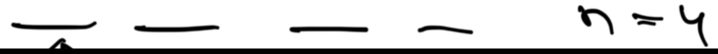
Thought Process

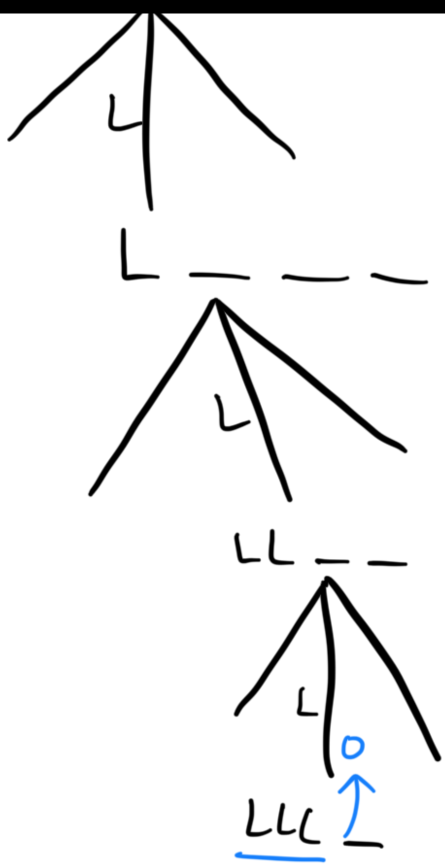
$3^n = 3^2$



Count maintain करलो
for Absence
Late







Pruning.

L	L	A	L	L
1	2	3	4	5

Consec. late = 1+1

Absence = 1

Options → Tree.

(Recur).
↓ DP

Solve (n, 0, 0) ;

```
int Solve( n, absence, cons_late) {
    if (n == 0)
        return 1; ☆
    if (abse > 1 || cons_late > 2) return 0;
    int A = Solve(n-1, absence+1, 0); //A
    int L = solve(n-1, absence, cons_late+1); //L
    int P = solve(n-1, absence, 0); // P

    return A + L + P;
}
```

Pruning

n A L

$t[n+1][2][3]$

10^5+1

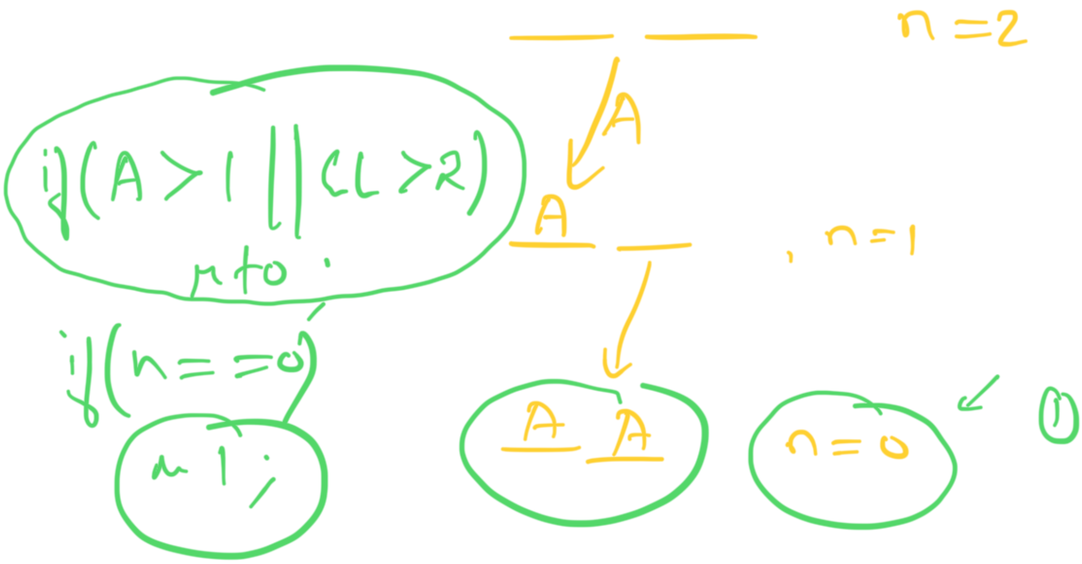
With Memo = $O(3^n)$

Memo = $t[n+1][2][3]$

$\hookrightarrow O(n)$

$S.C \simeq O(n)$

One Important thing ...



Bottom UP Derivation ...

```
int solve(int n, int absent, int consecutive_late) {

    if(absent >= 2 || consecutive_late >= 3) {
        return 0;
    }

    if(n == 0)
        return 1;

    if(t[n][absent][consecutive_late] != -1) {
        return t[n][absent][consecutive_late];
    }

    int A = solve(n-1, absent+1, 0) % M;
    int L = solve(n-1, absent, consecutive_late+1) % M;
    int P = solve(n-1, absent, 0) % M;

    return t[n][absent][consecutive_late] = ((A + L) % M + P) % M;
}

int checkRecord(int n) {
    memset(t, -1, sizeof(t));
    return solve(n, 0, 0);
}
```

t[100001][2][3]

A, CL

t[i][A][L] = x // State defin.

i A L

```
for (int i=0 ; i<=n ; i++) {

    for(int A = 0 ; A<=1 ; A++) {

        for (int L =0; L<=2 ; L++) {
            if(i==0)  t[0][A][L] = 1;
            else ans = 0;

            ans += t[i-1][A+1][0]; //A      %M
            ans+= t[i-1][A][L+1]; //L       %M
            ans+= t[i-1][A][0]; // P        %M
                                        %M
        }

    }

}

return t[n][0][0]; //solve (n,0,0);
}
```

A+1 <=1

L+1 <=2