Dynamic Programming

Video – 80

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

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

Facebook, Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

GFG POTD

5th Jan, 2024

React Hero

Bolt.up

Company :- PayU

Count possible ways to construct buildings

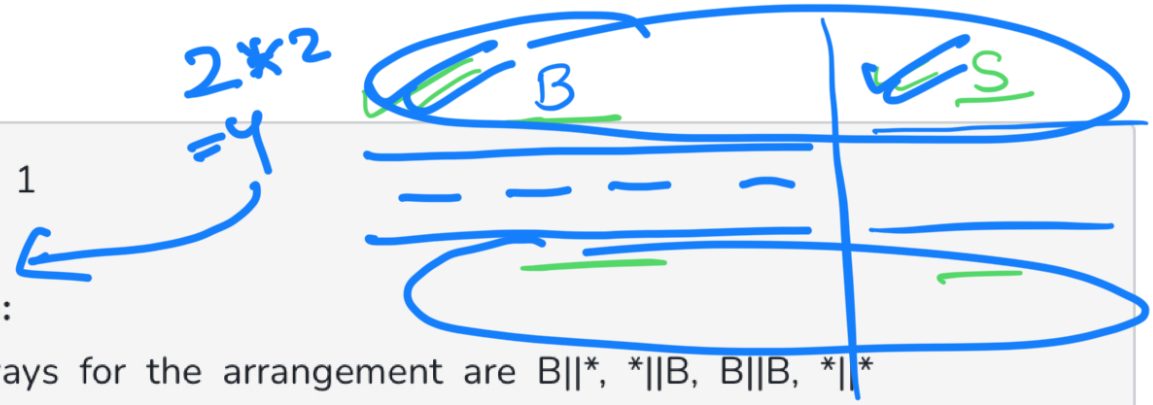
Medium Accuracy: 38.53% Submissions: 39K+ Points: 4

There is a road passing through a city with **N** plots on both sides of the road. Plots are arranged in a straight line on either side of the road. Determine the **total number** of ways to construct buildings in these plots, ensuring that no two buildings are adjacent to each other. Specifically, buildings on opposite sides of the road cannot be adjacent.

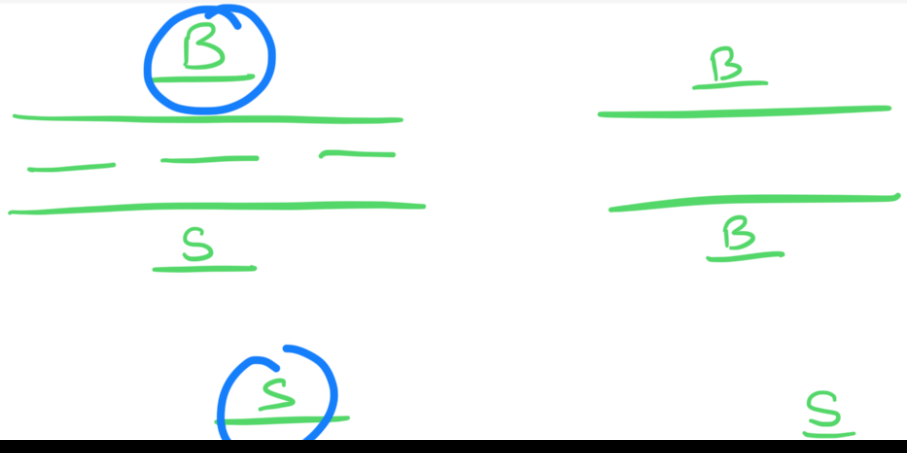
Using * to represent a plot and || for the road, the arrangement for **N = 3** can be visualized as follows: ***||***.

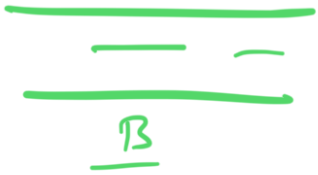
Note: As the answer can be very large, print it **mod** 10^9+7.

Example 1:



Input: N = 1
Output: 4
Explanation:
Possible ways for the arrangement are B||*, *||B, B||B, *||* where B represents a building.





B S

① option → B, S

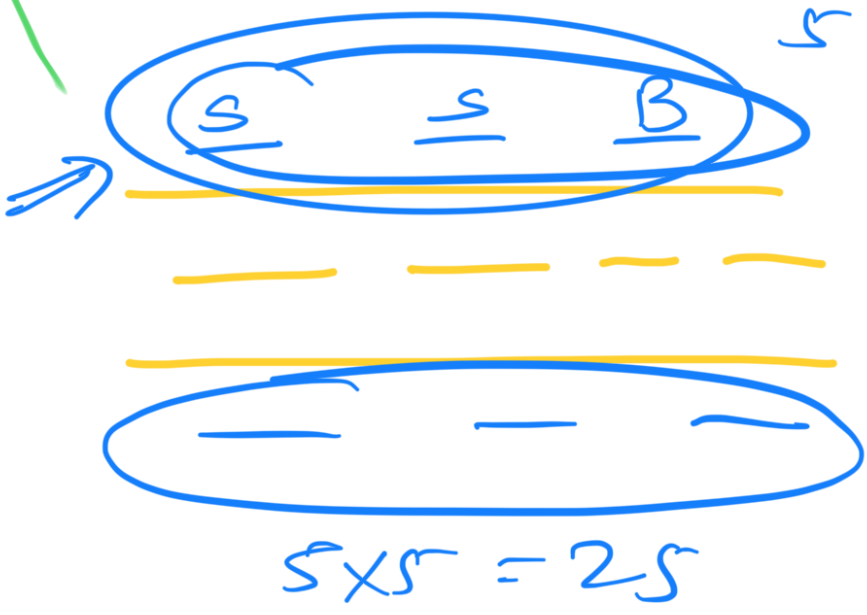
Recursion (Tree Di)

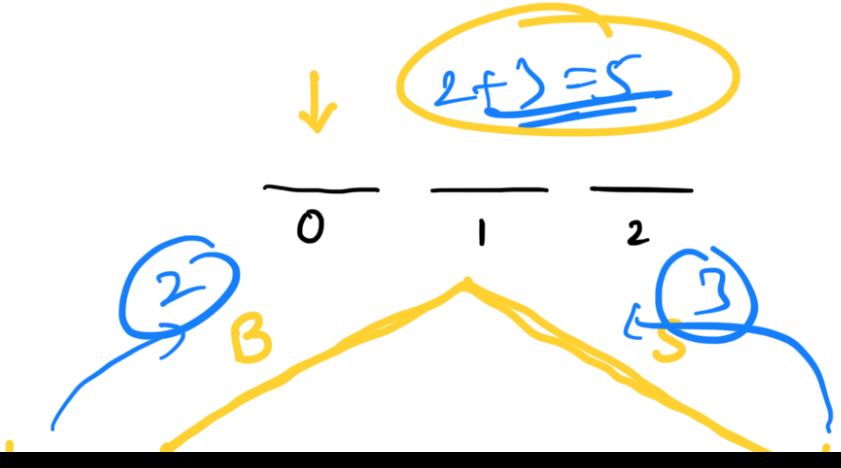
SS B

BSB SBS

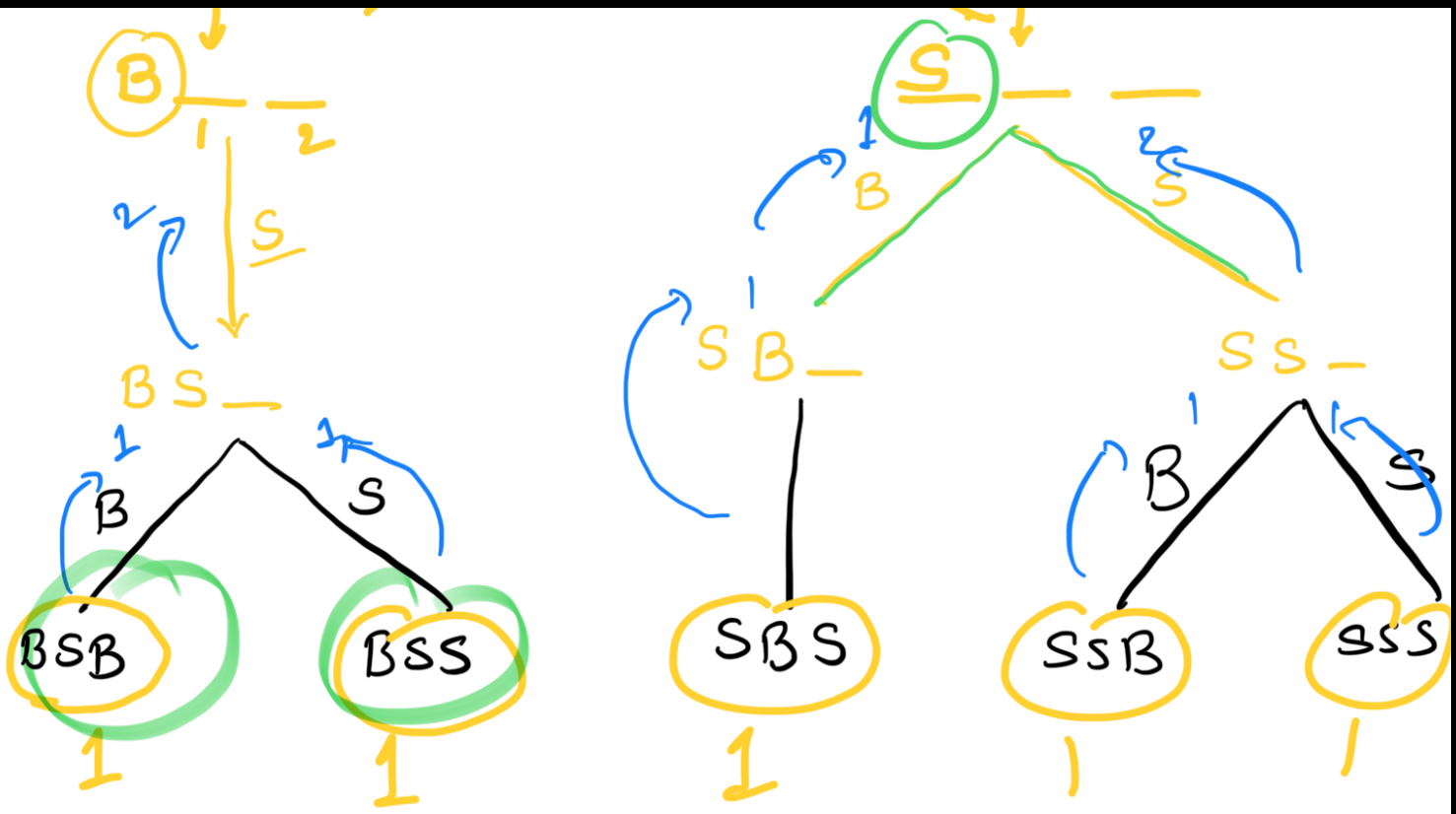
BSS SSS

N = 3





N = 3



Story to code :-

```
building = 1;
space = 0;
}
x = Solve(n-1, building) + Solve(n-1, space);
return (x*x) % M;
int Solve(plot, int states) {
    if (plot == 0) {
        return 1;
    }
```

```
if(status == building) {
    return solve(plot-1, space) ;
} else { // status == space
    return solve(plot-1, building) +
           solve(plot-1, space) ;
}
return -1;
}
```

Bottom Up

N = 3

0, 1, 2, ... i^{th} ... (n-1)

t[i][j] = no of ways of const. build till i^{th} plot with status j

j = 0 → space

j = 1 → Building

```
int solve(int plot, int status) {
    if(plot == 0) {
        return 1;
    }

    if(t[plot][status] != -1) {
        return t[plot][status];
    }

    if(status == building) {
        return t[plot][status] = solve(plot-1, space) % M;
    } else {
        return t[plot][status] = (solve(plot-1, building) % M +
                                  solve(plot-1, space) % M) % M;
    }
}
```


```
for (i = 0 ; i < N ; i++)

    for ( j=0; j<2 ; j++) {

        if (i == 0) {
            t[i][j] = 1;
        }

        if (j == building) {
            t[i][j] = t[i-1][space];
        } else {
            t[i][j] = t[i-1][building]
                    + t[i-1][space];
        }

    }

}
```

$$t[i][j]$$

$$x = t[N-1][building] + t[N-1][space]$$

$$ret \ (x * x) \% M;$$