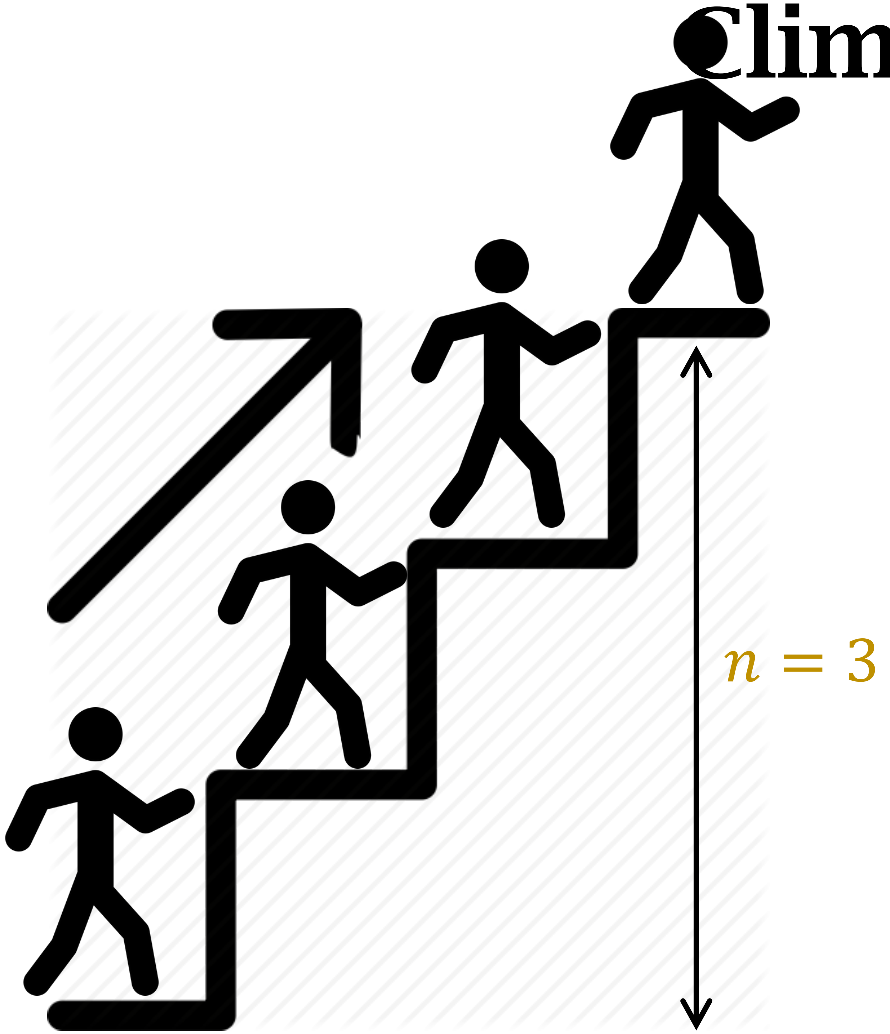
Climbing Stairs

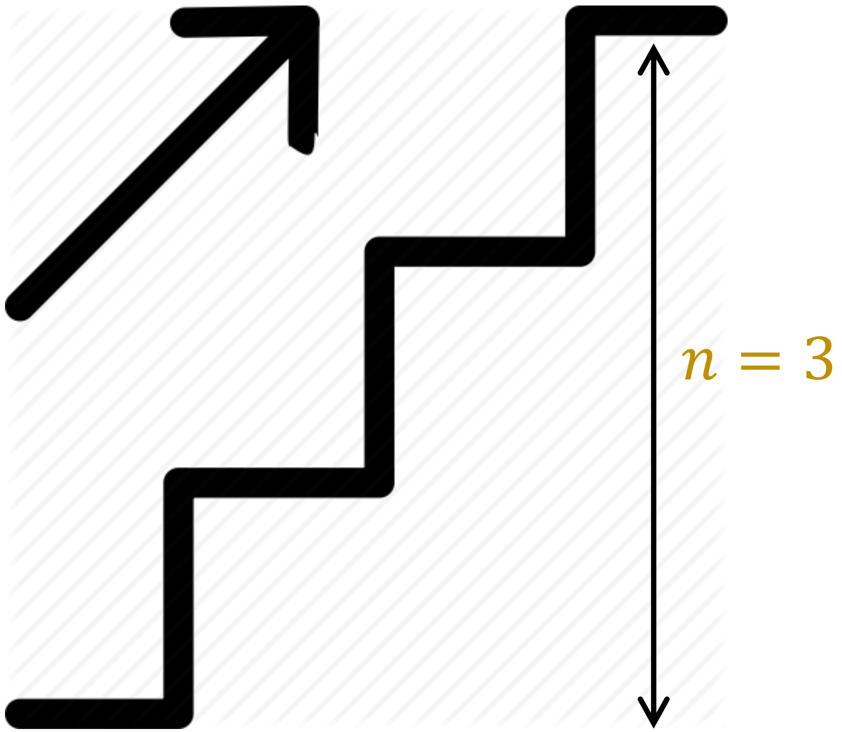
Shusen Wang

Climbing Stair Problem



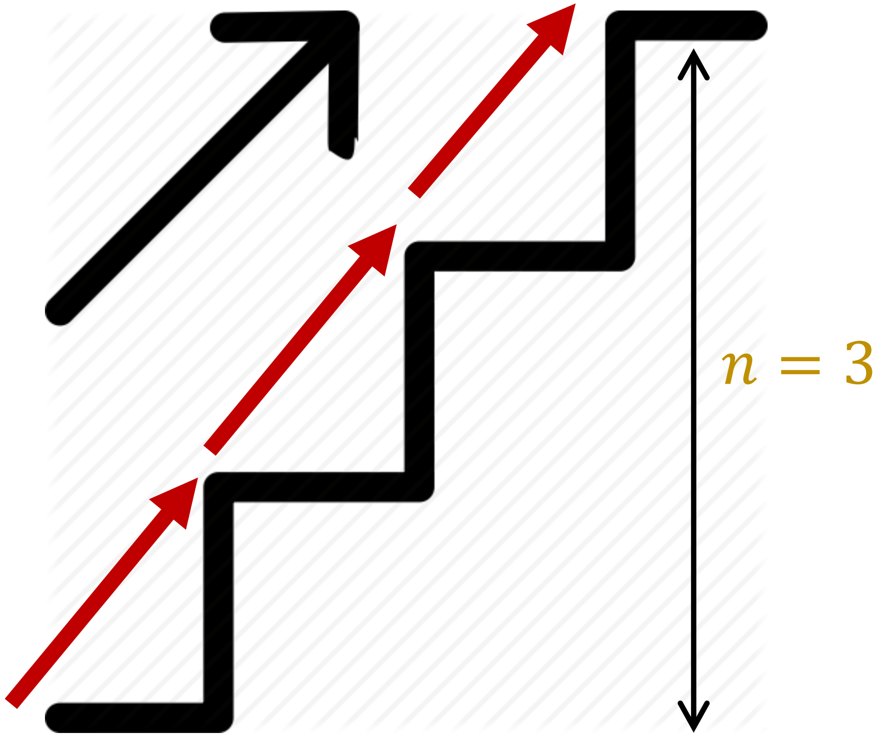
- You are climbing a stair case. It takes n steps to reach to the top.

Climbing Stair Problem



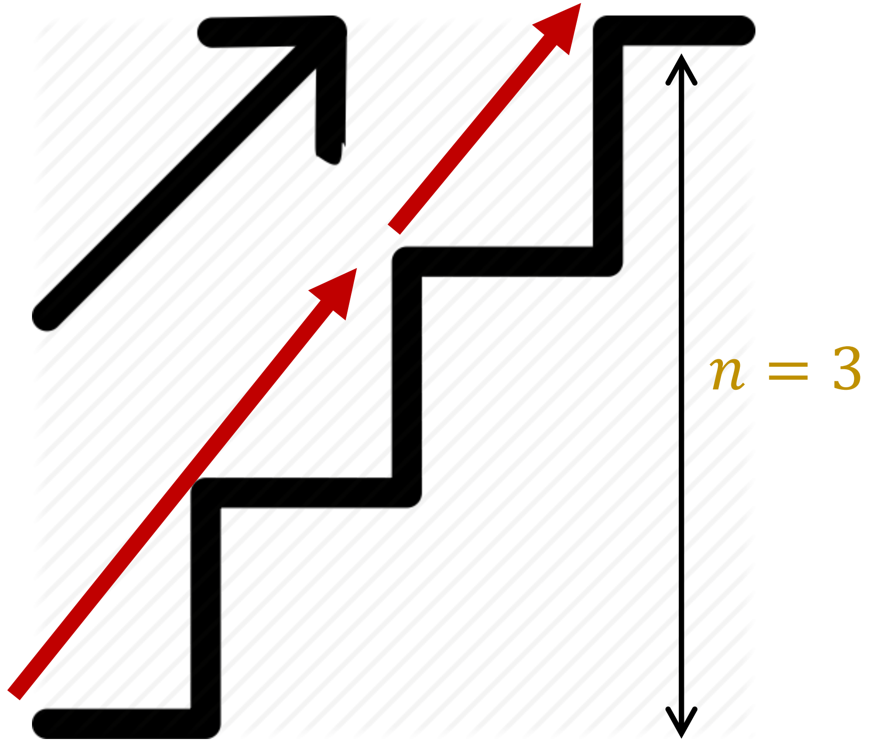
- You are climbing a stair case. It takes n steps to reach to the top.
- Each time you can either climb 1 or 2 steps.
- **Question:** In how many distinct ways can you climb to the top?

Climbing Stair Problem



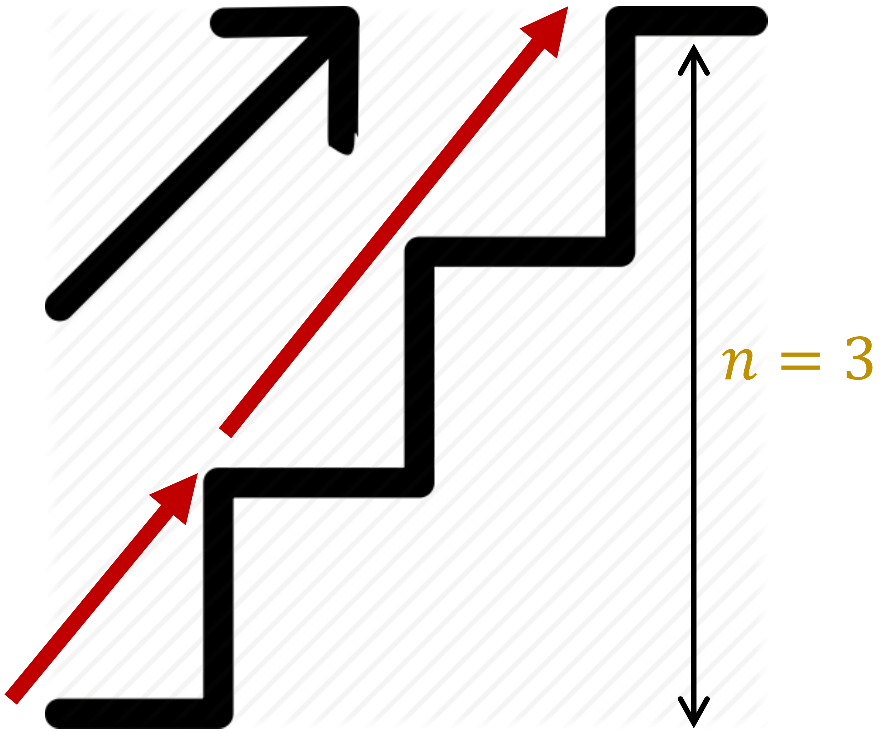
- For $n = 3$, there are 3 distinct ways:
  - 1 step + 1 step + 1 step.

Climbing Stair Problem



- For $n = 3$, there are 3 distinct ways:
  - 1 step + 1 step + 1 step.
  - 2 steps + 1 step.

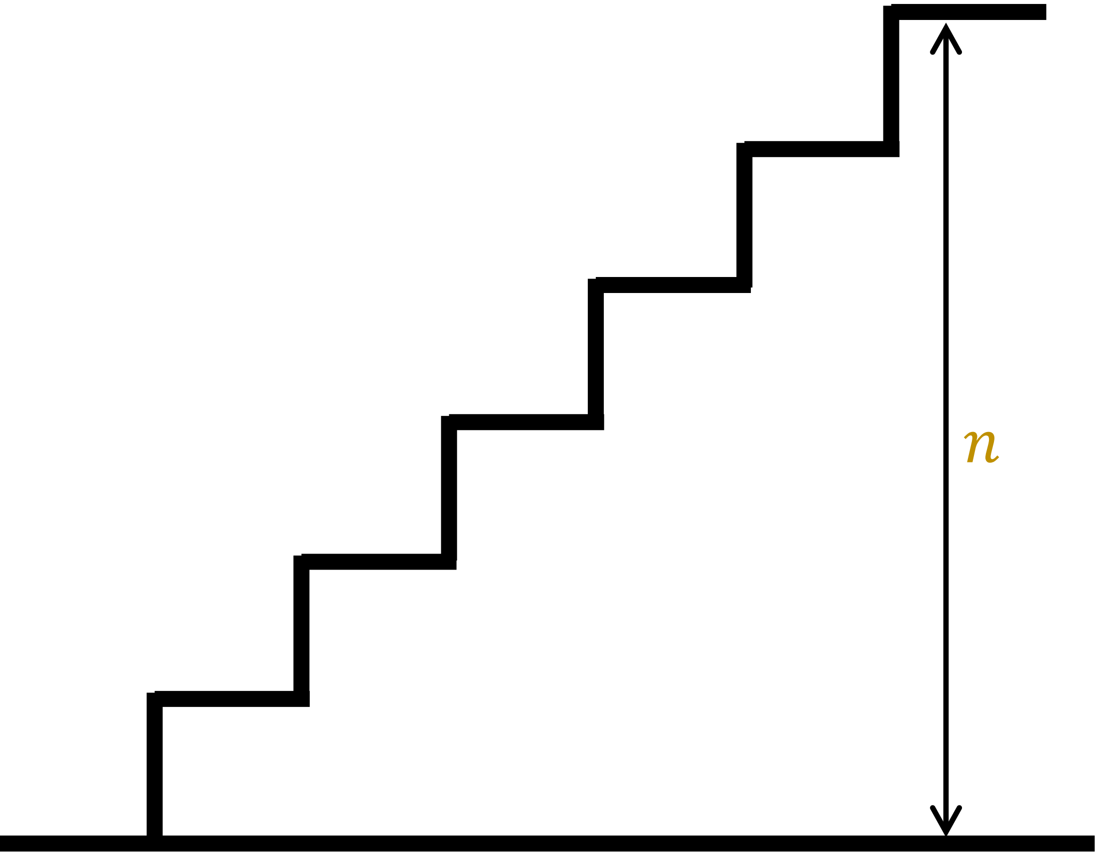
Climbing Stair Problem



- For $n = 3$, there are 3 distinct ways:
  - 1 step + 1 step + 1 step.
  - 2 steps + 1 step.
  - 1 step + 2 steps.

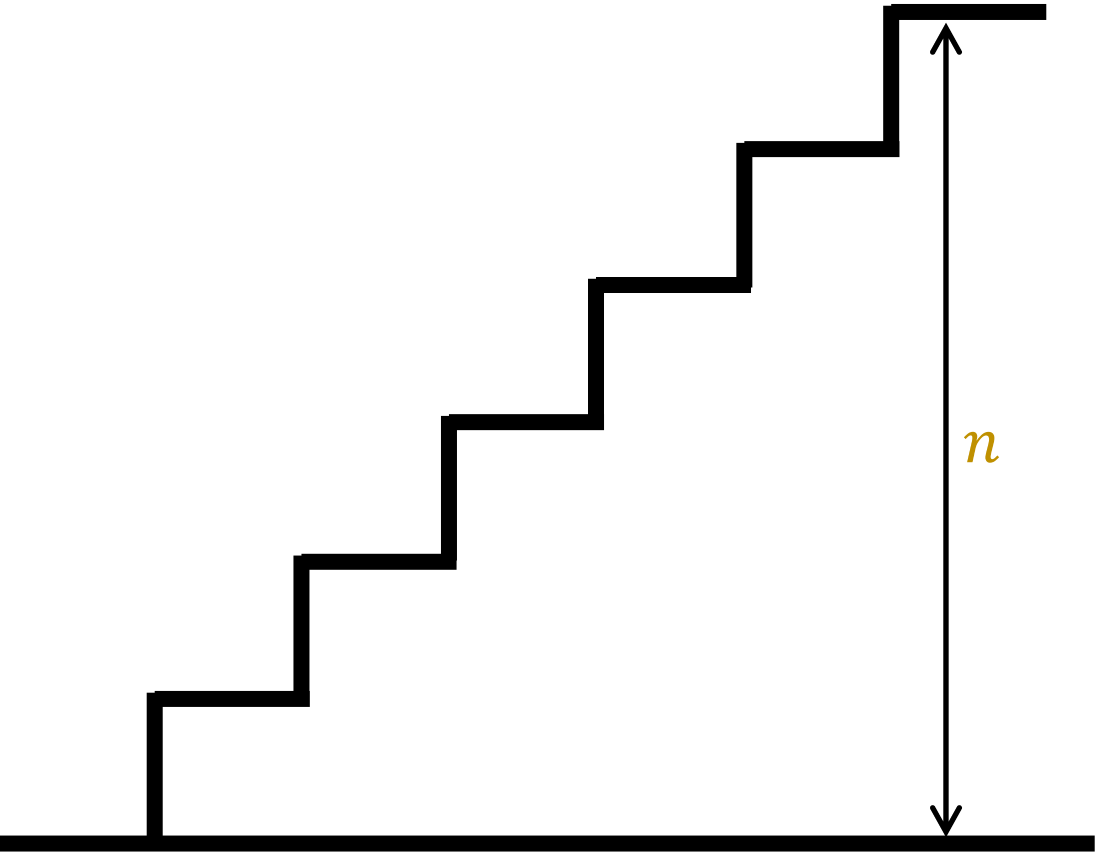
Optimal Substructure

Climbing Stair Problem



- It takes n steps to reach to the top.

Climbing Stair Problem

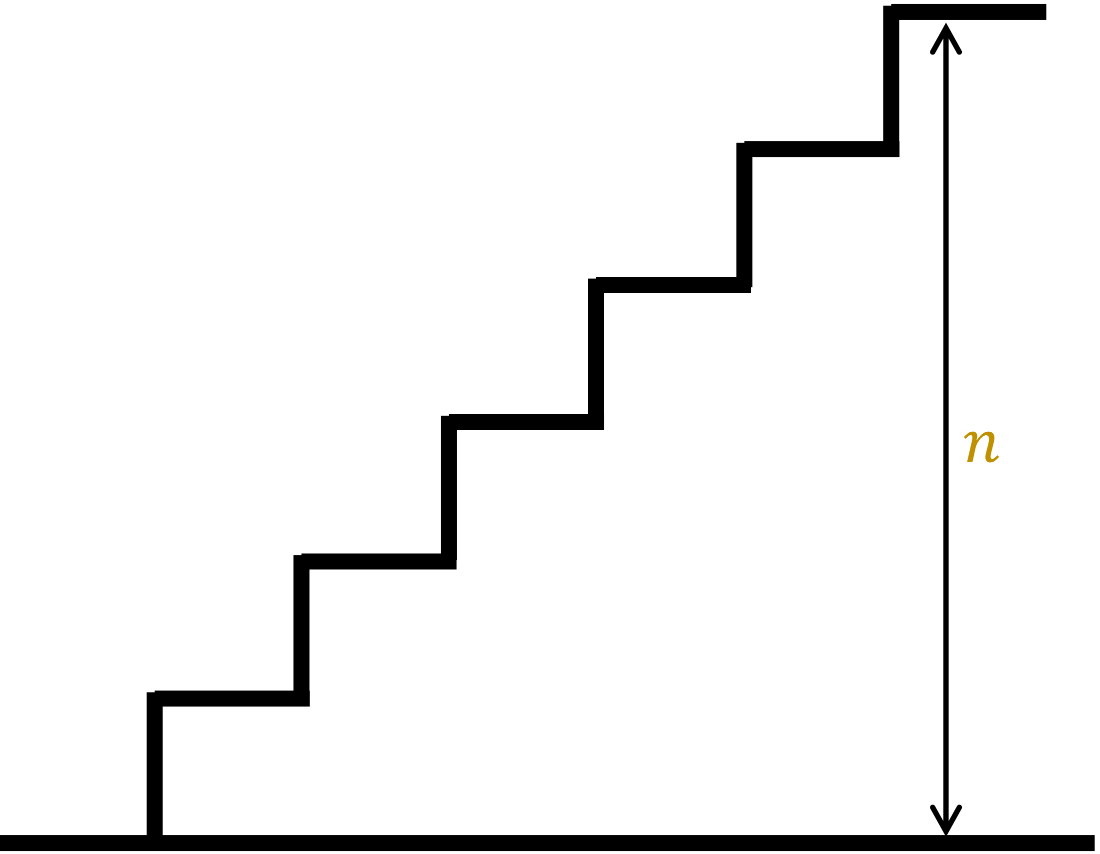


- It takes n steps to reach to the top.
- Let $W[n]$ be the number of distinct ways to reach the top, e.g.,
  - $W[1] = 1,$
  - $W[2] = 2,$
  - $W[3] = 3,$
  - $W[4] = 5,$
  - $W[5] = 8,$
  - $W[6] = 13,$

  $\vdots$

Any Pattern?

Climbing Stair Problem



- It takes n steps to reach to the top.
- Let $\mathrm{W}[n]$ be the number of distinct ways to reach the top, e.g.,
  - $\mathrm{W}[1] = 1$,
  - $\mathrm{W}[2] = 2$,
  - $\mathrm{W}[3] = 3$,
  - $\mathrm{W}[4] = 5$,
  - $\mathrm{W}[5] = 8$,
  - $\mathrm{W}[6] = 13$,

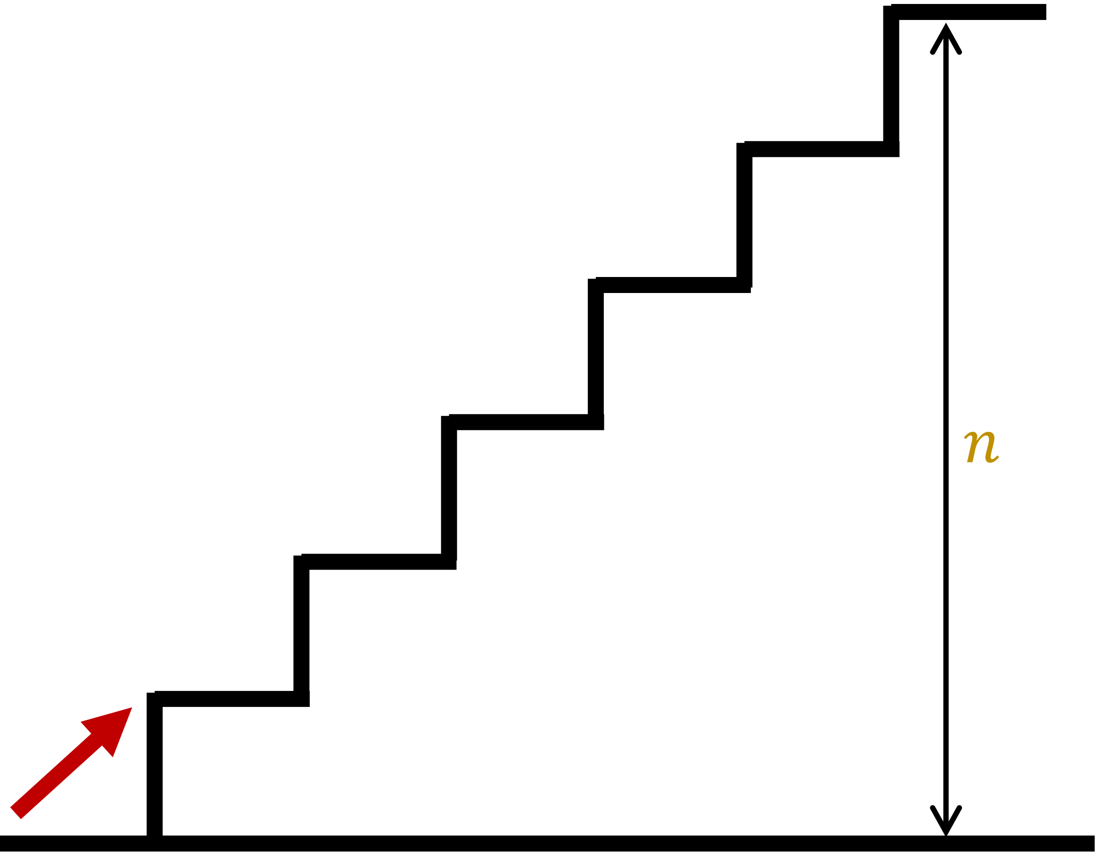
$$\vdots$$

Any Pattern?

$$\mathrm{W}[n] = \mathrm{W}[n-1] + \mathrm{W}[n-2]$$

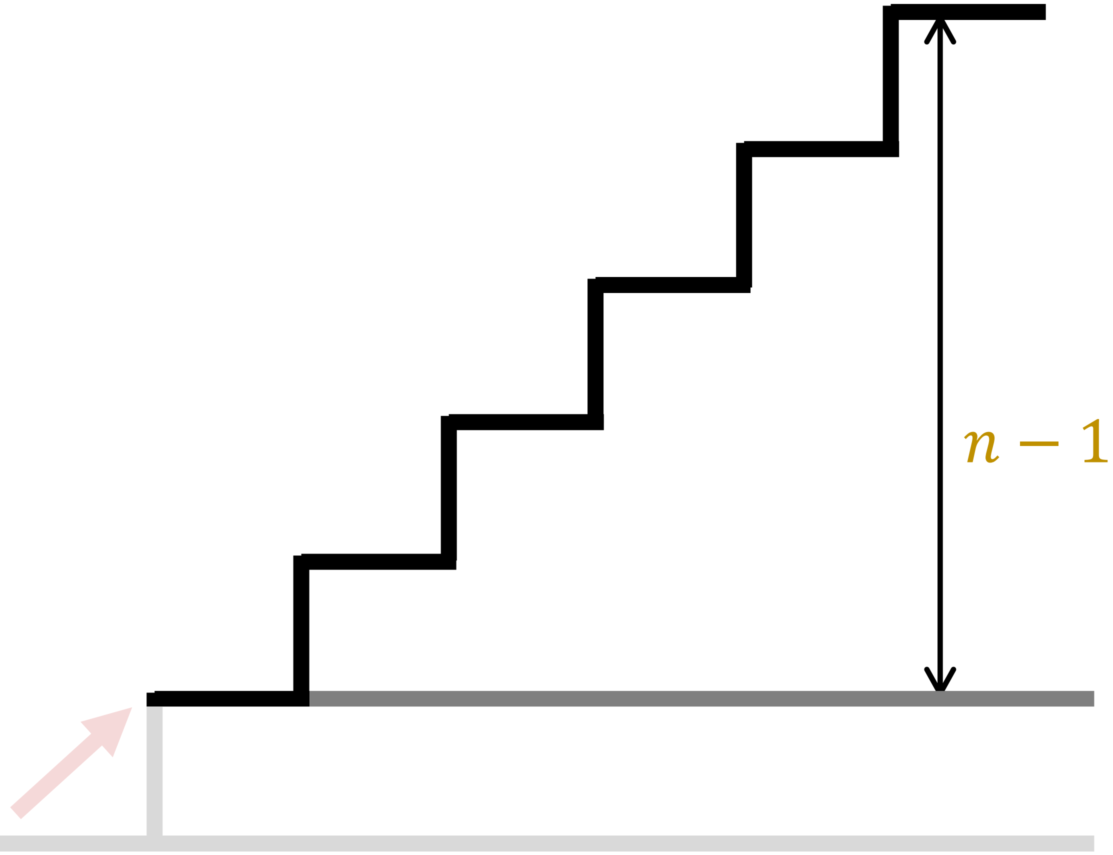
Why?

Climbing Stair Problem



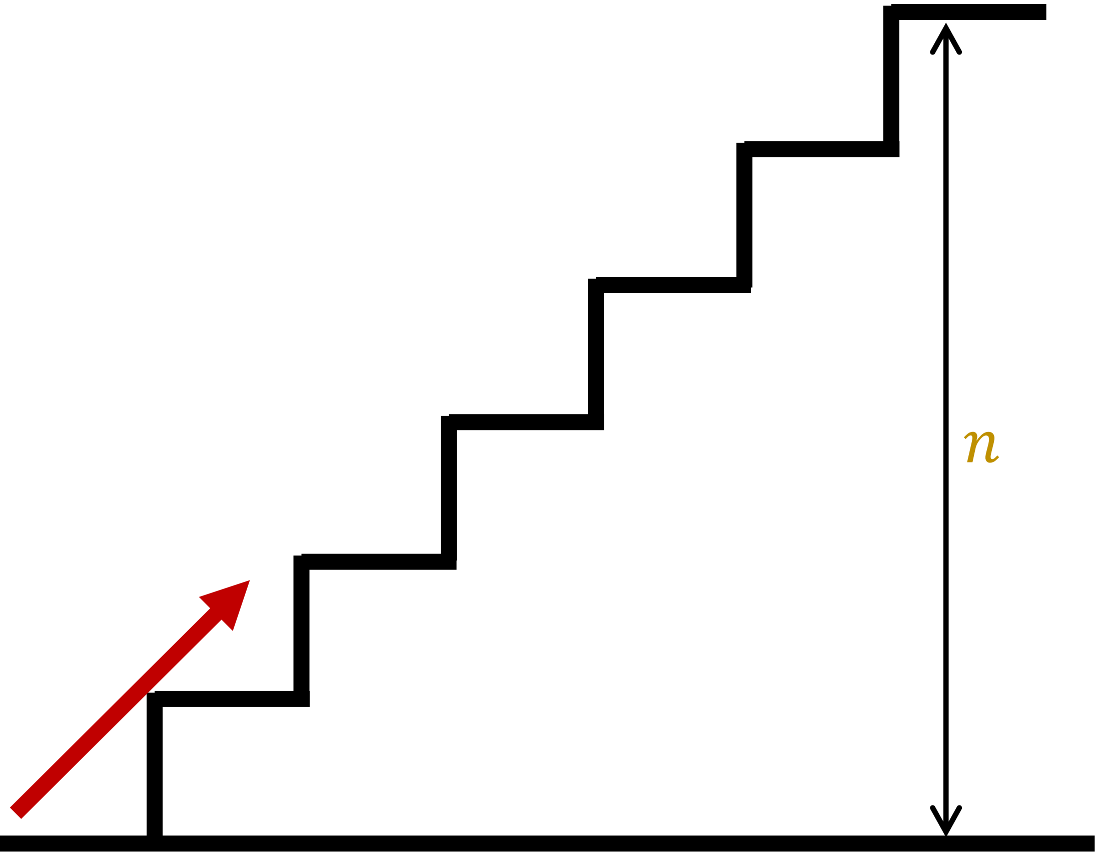
- **Case 1:** Climb one step at first.

Climbing Stair Problem



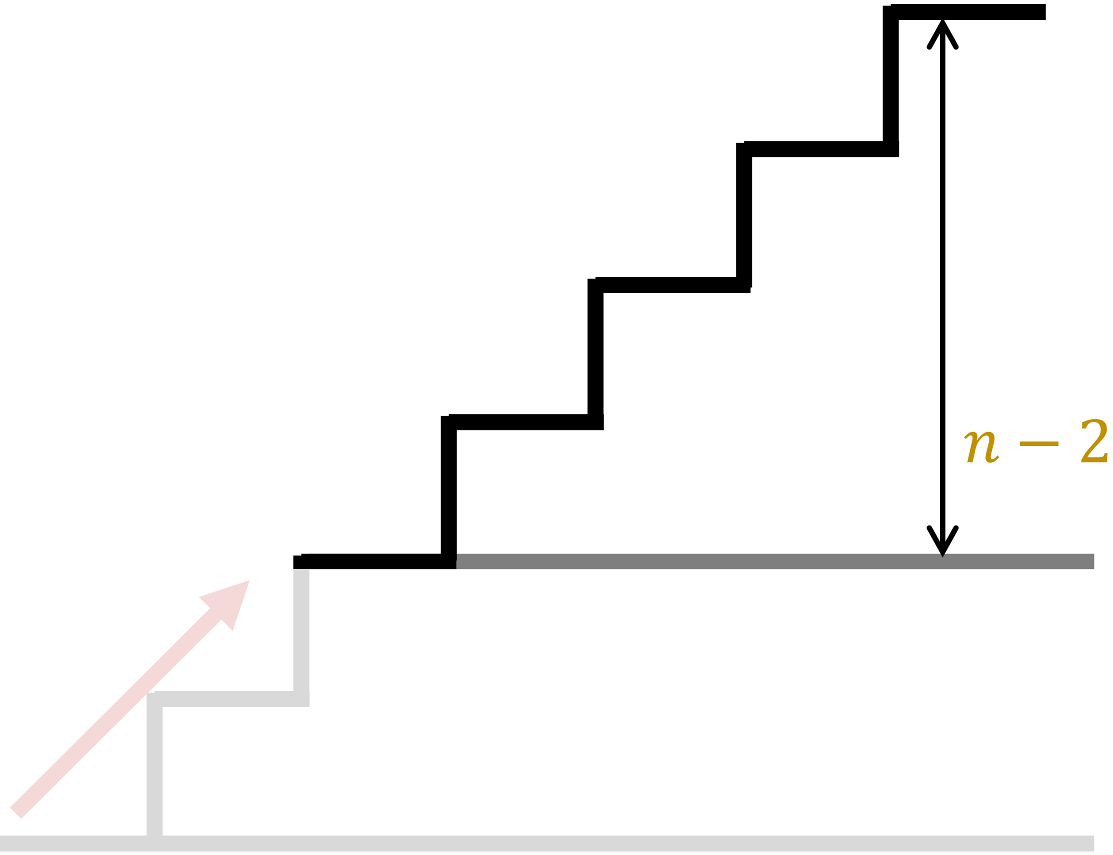
- **Case 1:** Climb one step at first.
  - Then there are $\mathrm{W}[n-1]$ distinct ways to finish the climbing.

Climbing Stair Problem



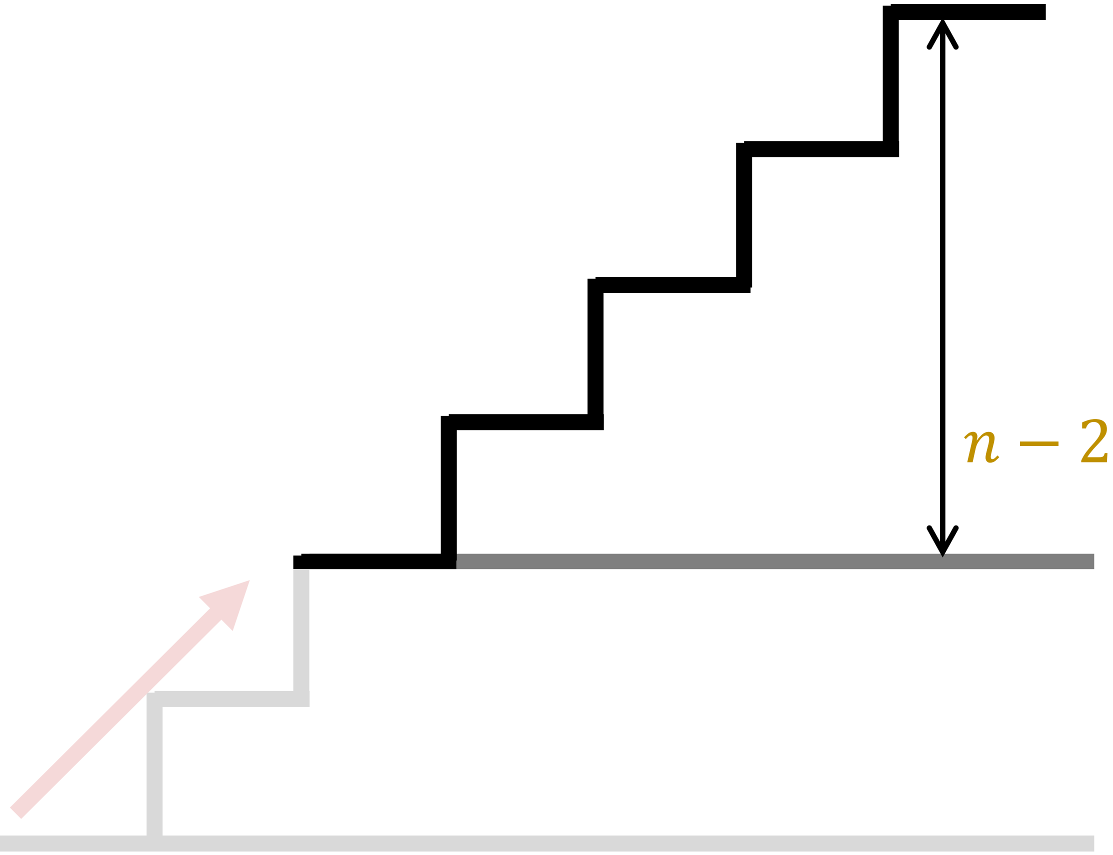
- **Case 1:** Climb one step at first.
  - Then there are $\mathrm{W}[n-1]$ distinct ways to finish the climbing.
- **Case 2:** Climb two steps at first.

Climbing Stair Problem



- **Case 1:** Climb one step at first.
  - Then there are $\mathrm{W}[n-1]$ distinct ways to finish the climbing.
- **Case 2:** Climb two steps at first.
  - Then there are $\mathrm{W}[n-2]$ distinct ways to finish the climbing.

Climbing Stair Problem



- **Case 1:** Climb one step at first.
  - Then there are $\mathrm{W}[n-1]$ distinct ways to finish the climbing.
- **Case 2:** Climb two steps at first.
  - Then there are $\mathrm{W}[n-2]$ distinct ways to finish the climbing.
- Thus, $\mathrm{W}[n] = \mathrm{W}[n-1] + \mathrm{W}[n-2]$.

Dynamic Programming

Dynamic Programming

Step 1: Break a big problem into smaller subproblems.

- Optimal substructure: $\mathrm{W}[n] = \mathrm{W}[n-2] + \mathrm{W}[n-1]$.

Dynamic Programming

Step 1: Break a big problem into smaller subproblems.

- Optimal substructure: $\mathrm{W}[n] = \mathrm{W}[n-2] + \mathrm{W}[n-1]$.

Step 2: Use a table instead of recursion.

- Record the solutions to the small problems.
- Use the recorded solutions for solving bigger problems.

Dynamic Programming

Step 1: Break a big problem into smaller subproblems.

- Optimal substructure: $\mathrm{W}[n] = \mathrm{W}[n-2] + \mathrm{W}[n-1]$.

Step 2: Use a table instead of recursion.

- Record the solutions to the small problems.
- Use the recorded solutions for solving bigger problems.

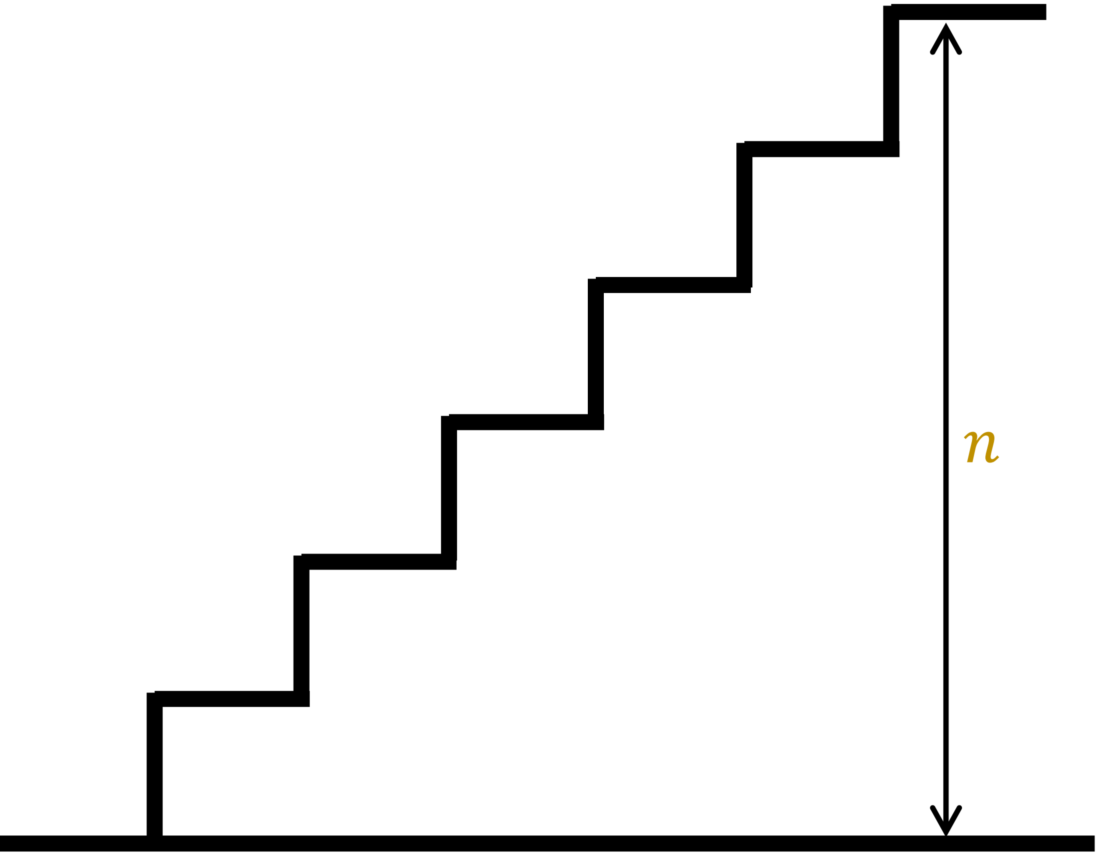
n	1	2	3	4	5	6	7	8	9	10
$\mathrm{W}(n)$	1	2	3	5	8	13	?			

Implementation

```
long climbStairs(int n) {
    if (n == 1)
        return 1;
    long W[n+1];
    W[1] = 1;
    W[2] = 2;
    for (int i = 3; i <= n; i++)
        W[i] = W[i-1] + W[i-2];
    return W[n];
}
```

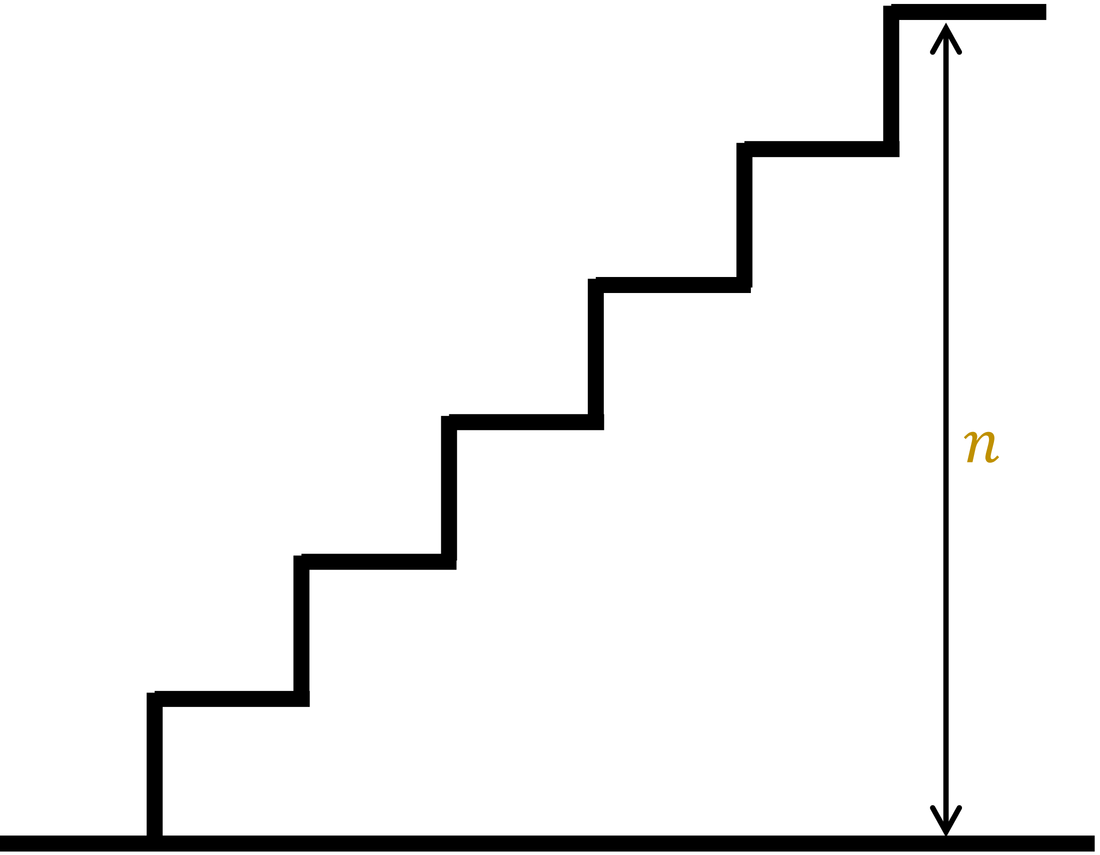
Question

A Different Setting



- You are climbing a stair case. It takes n steps to reach to the top.
- Each time you can climb 1, 2 or 3 steps.
- **Question:** In how many distinct ways can you climb to the top?

A Different Setting



- **Recursion:**

  $$\mathrm{W}[n] = \mathrm{W}[n-1] + \mathrm{W}[n-2] + \mathrm{W}[n-3].$$

- **Why?**

Thank You!