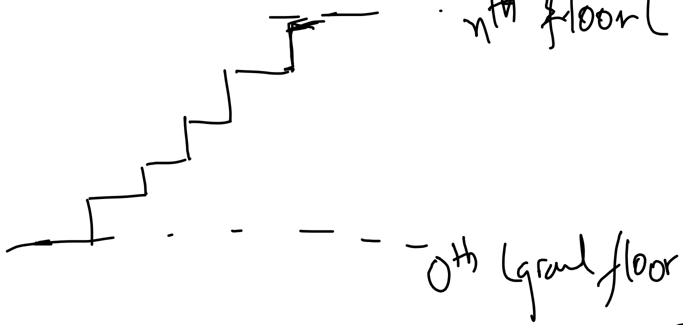
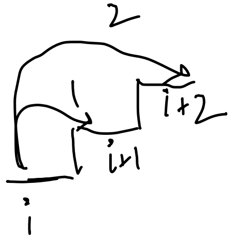
Dynamic Programming 1

Thursday, 8 July 2021 3:04 PM

① Recursion



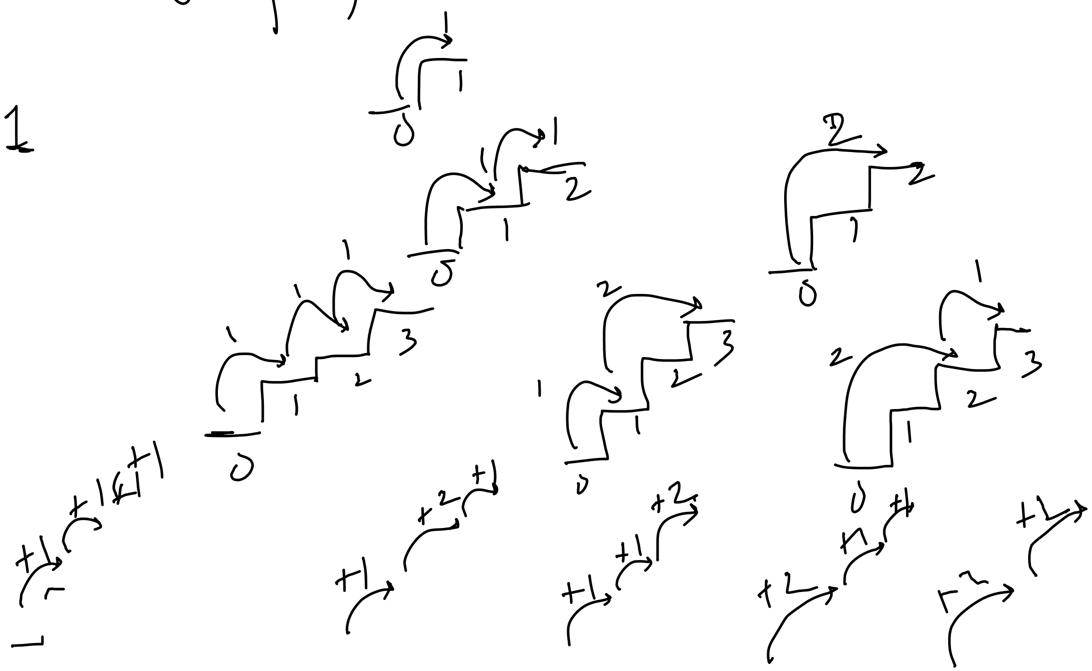


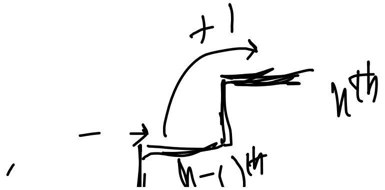
$n=1 \Rightarrow 1$

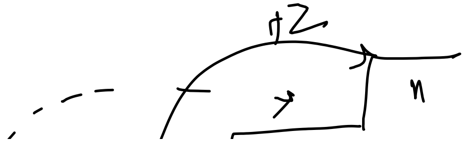
$n=2 \Rightarrow 2$

$n=3 \Rightarrow 3$

$n=4 \Rightarrow 5$

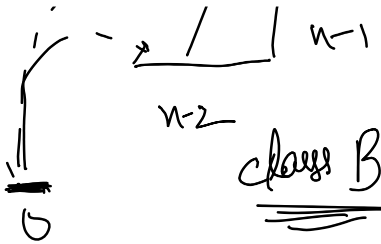






$n-4^{th}$ step

or Class A = No of ways of $n-1$



class B = No of ways $n-2^{nd}$ step

$\rightarrow S = \{P_1, P_2\}$ Partion $\rightarrow \boxed{P_1 \cup P_2 = S \text{ and } P_1 \cap P_2 = \phi}$

$\Rightarrow |S| = |P_1| + |P_2|$

Total No of ways to Reach nth step = |Class A| + |Class B|

Fibonacci Series

Recursion $C(n) = C(n-1) + C(n-2)$

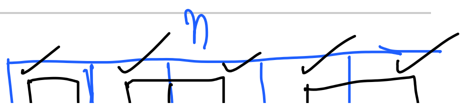
Tiling Problem

Dynamic Programming

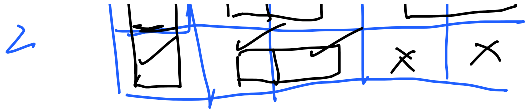
Some tiling problems

We have an n×2 grid to be tiled.

<-------- n --------->



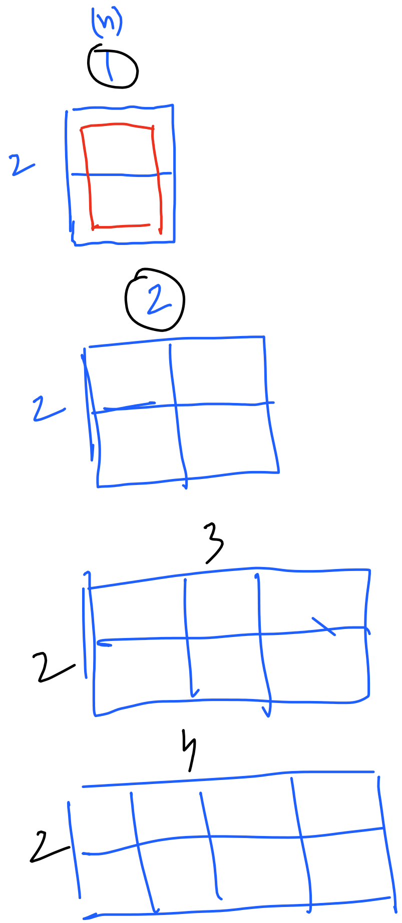
```
 -- -- -- -- -- -- --
|  |  |  |  |  |..|  |
 -- -- -- -- --    --
|  |  |  |  |  |..|  |
 -- -- -- -- -- -- --
```



We have with us a supply of rectangular tiles of size 2×1. Each tile can be rotated and laid horizontally or vertically.

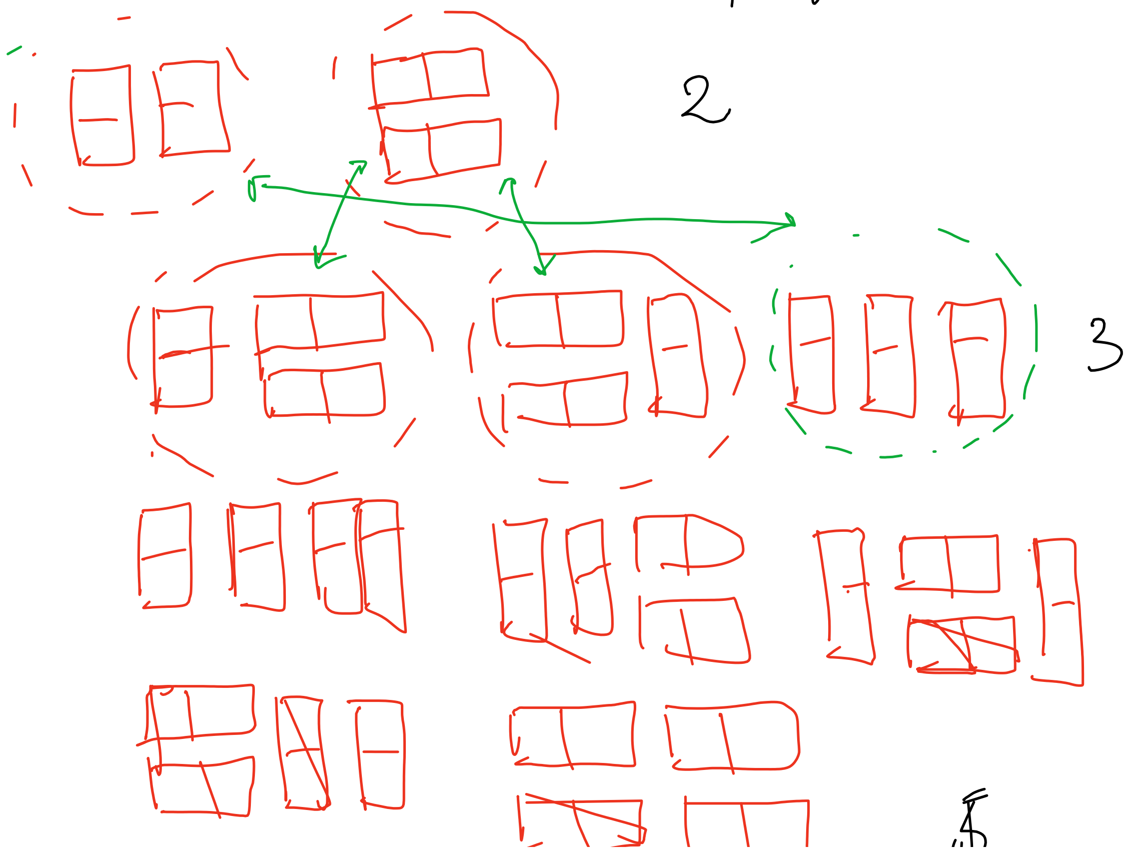
```
 -- --
|  |  |
 -- --
```

How many ways can we tile the n×2 grid using these tiles?



No of Ways

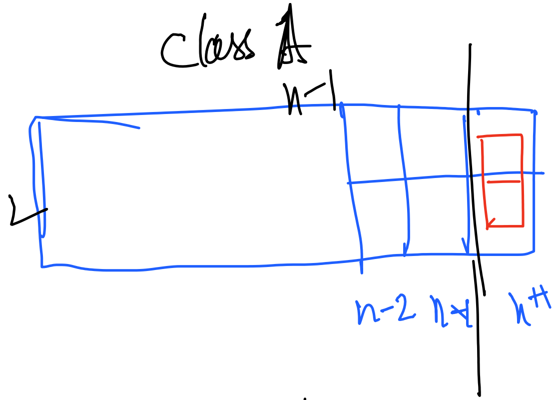
1

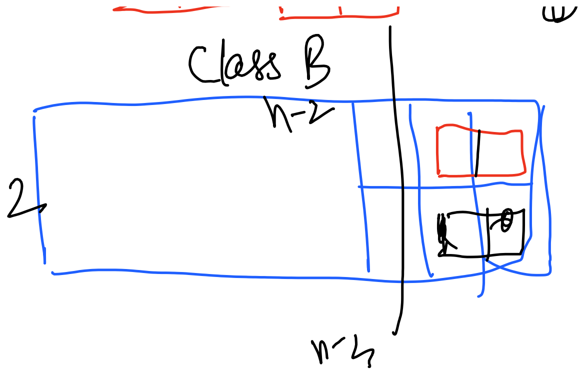


No of Ways

2

3





$$|S| = |\text{Class } A| + |\text{Class } B|$$

$$C(n) = C(n-1) + C(n-2)$$

A more complicated tiling problem

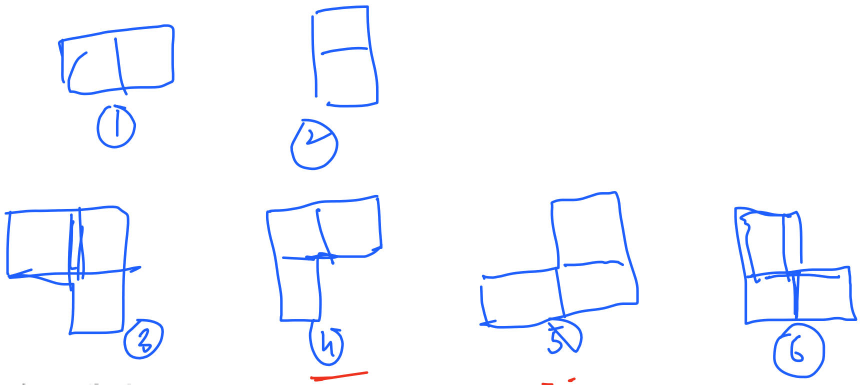
Again we want to tile an n×2 grid, but we have two types of tiles:

- A 2×1 tile as before

```
 -- --
|  |  |
 -- --
```

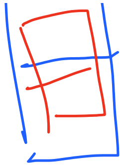
- An L-shaped tile covering 3 squares

```
 -- --
|  |  |
 -- --
   |  |
    --
```



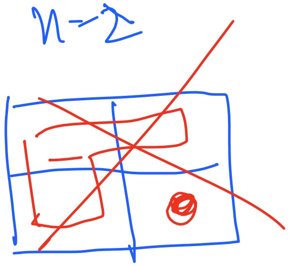
How many ways can we tile the n×2 grid using these tiles?

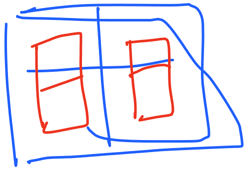
n=1

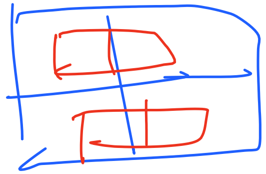


1

n=2

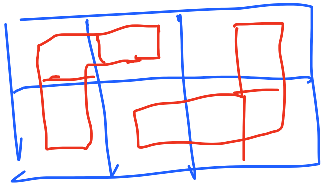






2

n=3

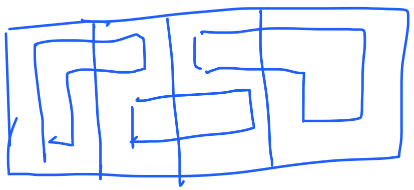


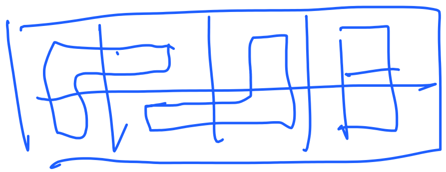
3

4

C(n)   n×2 grid

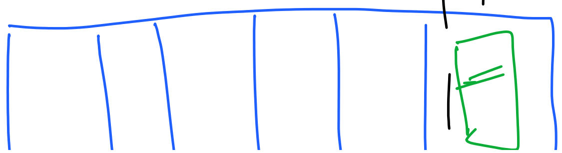
n=4





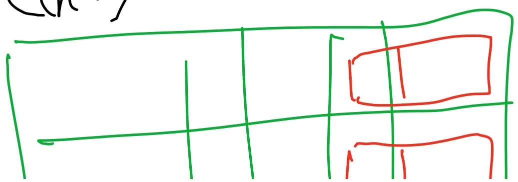
n   C(n-1)

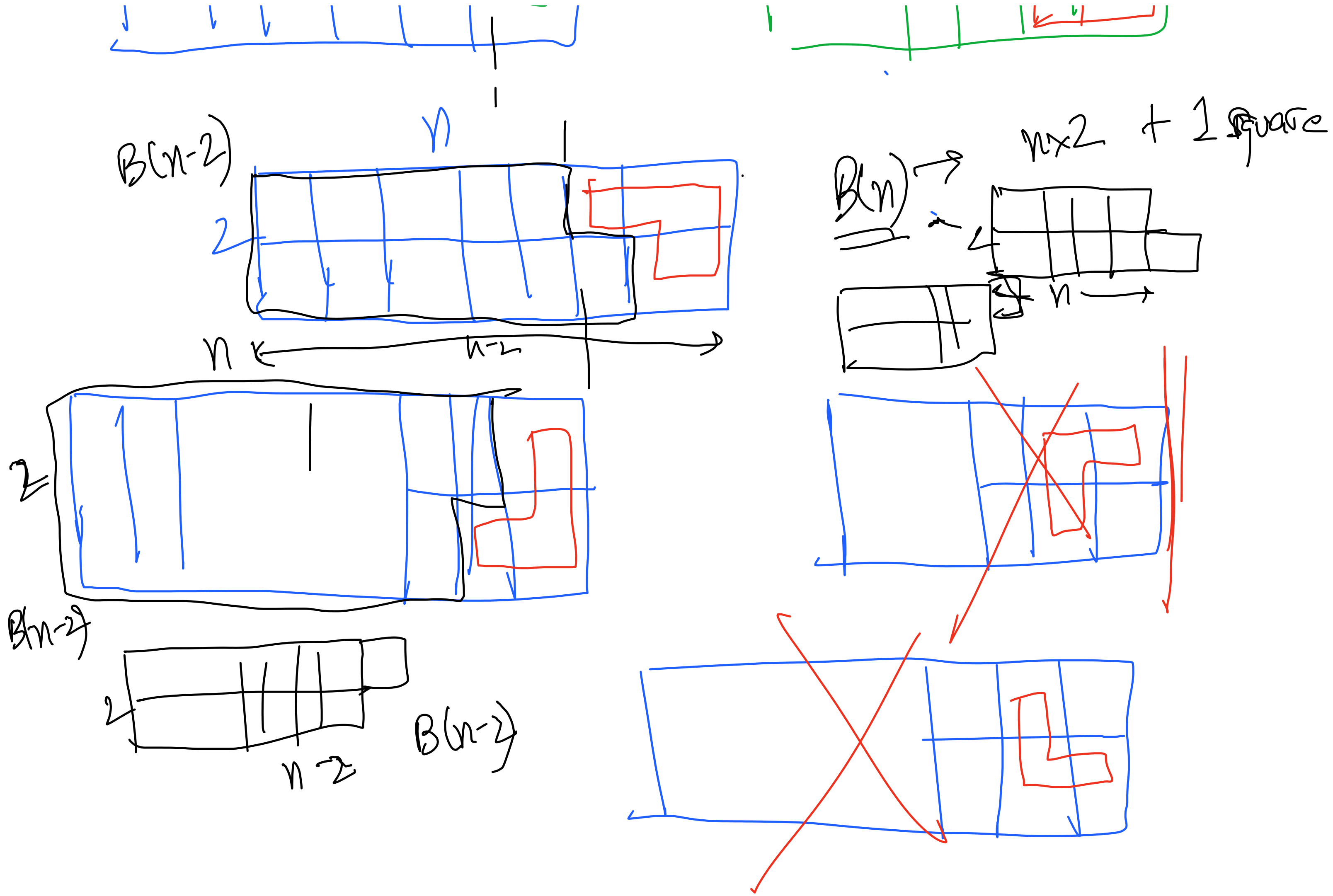
2



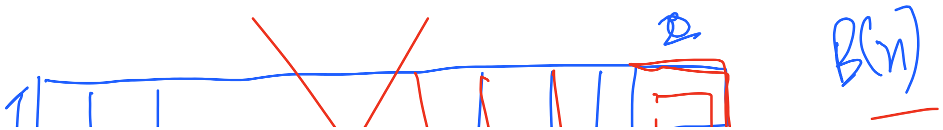
C(n-2)   n

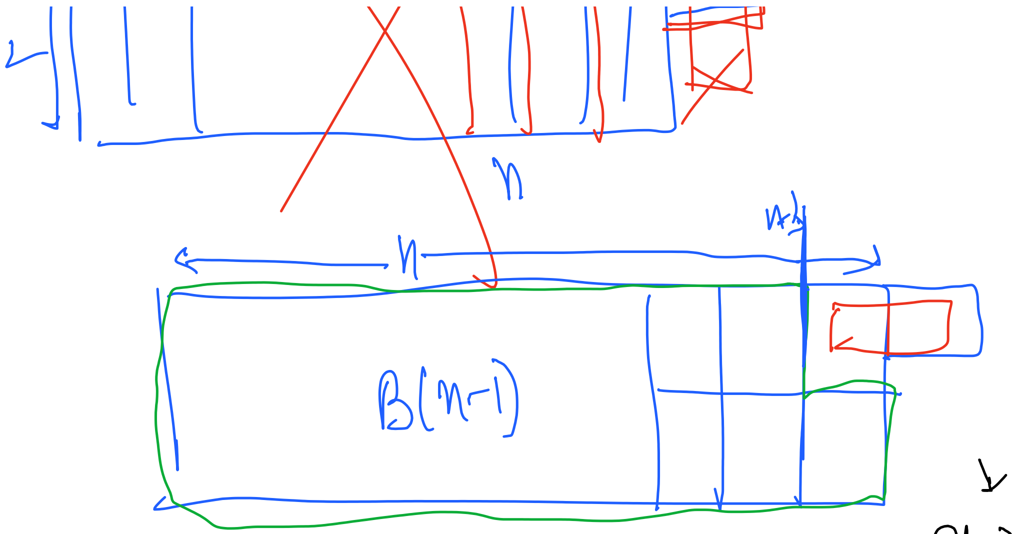
2



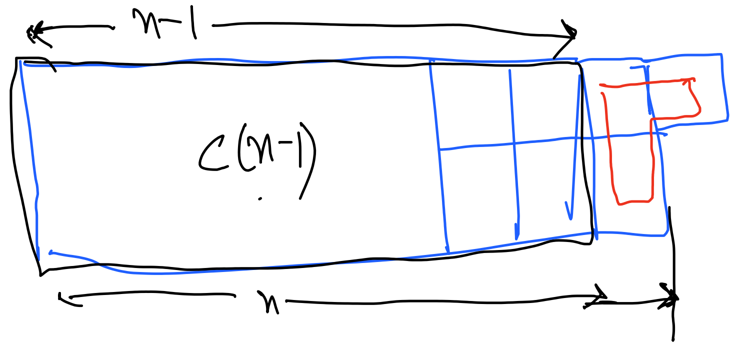


$$C(n) = C(n-1) + C(n-2) + 2B(n-2)$$





$$\rightarrow \quad B(n) = B(n-1) + C(n-1)$$



$$C(n) = C(n-1) + C(n-2) + 2B(n-2) \quad (1)$$

3 C 4
1 3

2 Bs
1 C

$$B(n) = C(n-1) + B(n-1) \quad (2)$$

$$B(n-2) = C(n) - C(n-1) - C(n-2)$$

$$\frac{\quad}{2}$$

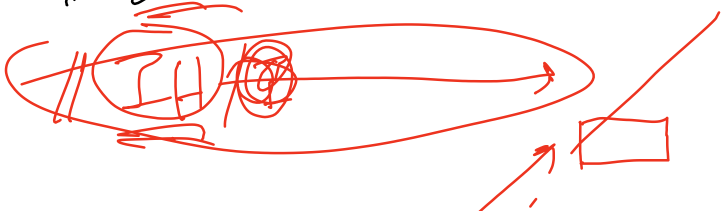
$$\frac{C(n+2) - C(n+1) - C(n)}{2} = C(n-1) + \frac{C(n+1) - C(n) - C(n-1)}{2}$$

$$C(n+2) - C(n+1) - \cancel{C(n)} = \cancel{2}C(n-1) + C(n+1) - \cancel{C(n)} - \cancel{C(n-1)}$$

$$\boxed{C(n+2) = 2C(n+1) + C(n-1)} \leftarrow \text{Ans}$$

Bottom-Up DP (An iterative way)   Induction

//Base Case



Memory

```
int f[n+1];
f[0]=1;
f[1]=1;
```

→ Base Cases

Sum(n+1);

Sum[0] = 0;

[illegible]

```
f[1]=1;
for(int i=2;i<=n;i++)
    f[i] = f[i-1] + f[i-2];
f[n] // has now the answer to the staircase problem
```

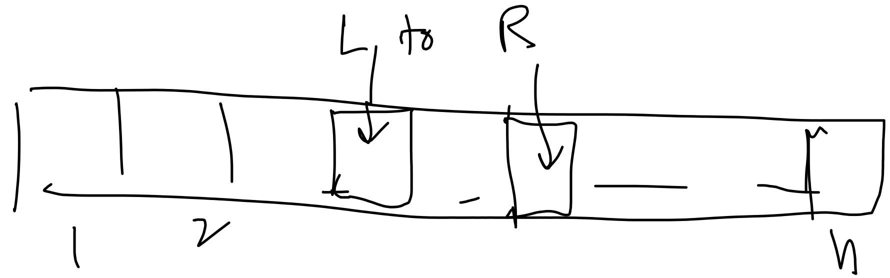
+ c

sum[R] - sum[L-1]

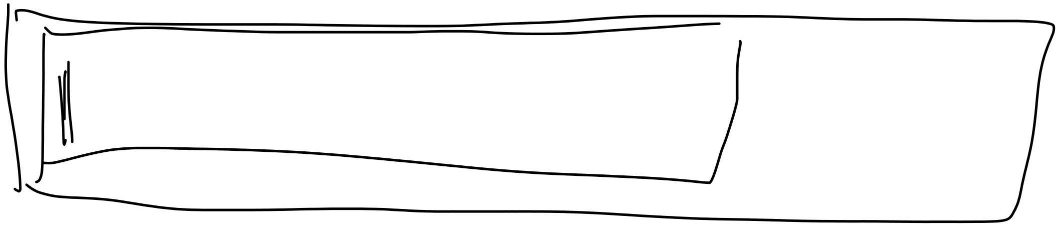
for(int i=1;i<=n)
sum[i] = a[i] + sum[i-1]

Prefix Sums

O(n)



Sum[L to R] = Sum[1 to R] - Sum[1 to L-1]   O(1)



Brute for

Sum[1] → O(1)

[1 to i] → Sum[i] → O(i)

Sum[n] → O(n)

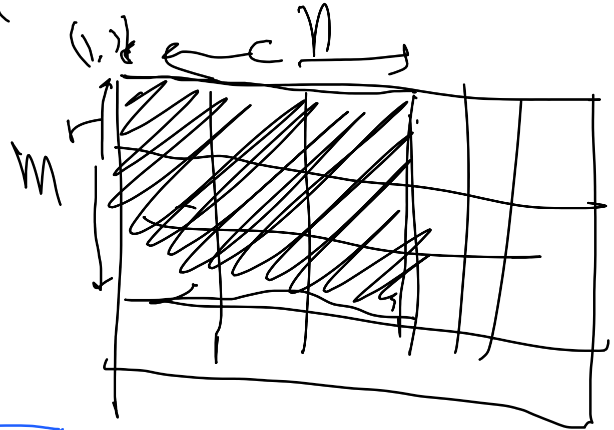
$\frac{n(n+1)}{2}$ $O(n^2)$

Dynamic Programming

$$Sum[i] = a[i] + Sum[i-1] \quad (1)$$

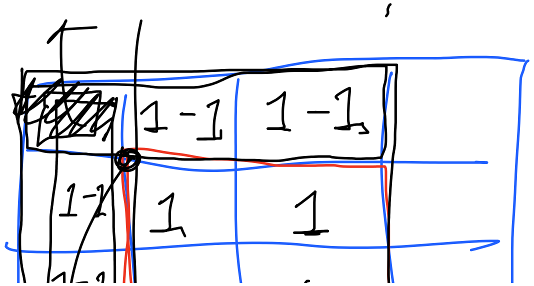
int s

2D prefix Sum

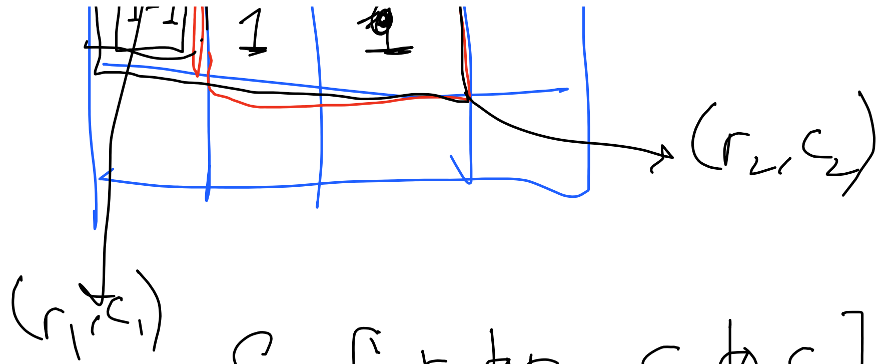


Sum[r][c]

1-1-1+1



Sum[r][c] is the



$$Sum[r_1 \text{ to } r_2, c_1 \text{ to } c_2] = Sum[r_2][c_2] - Sum[r_1][c_2] - Sum[r_2][c_1] + Sum[r_1][c_1]$$

$Sum[n][m]$

$$Sum[i][j] = A[i][j] + Sum[i-1][j] + Sum[i][j-1] - Sum[i-1][j-1]$$

$\Downarrow$