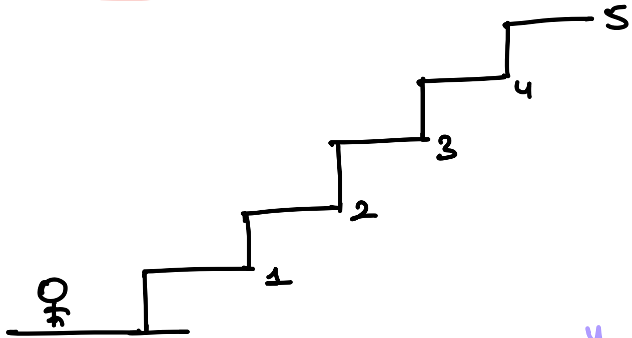
... contd

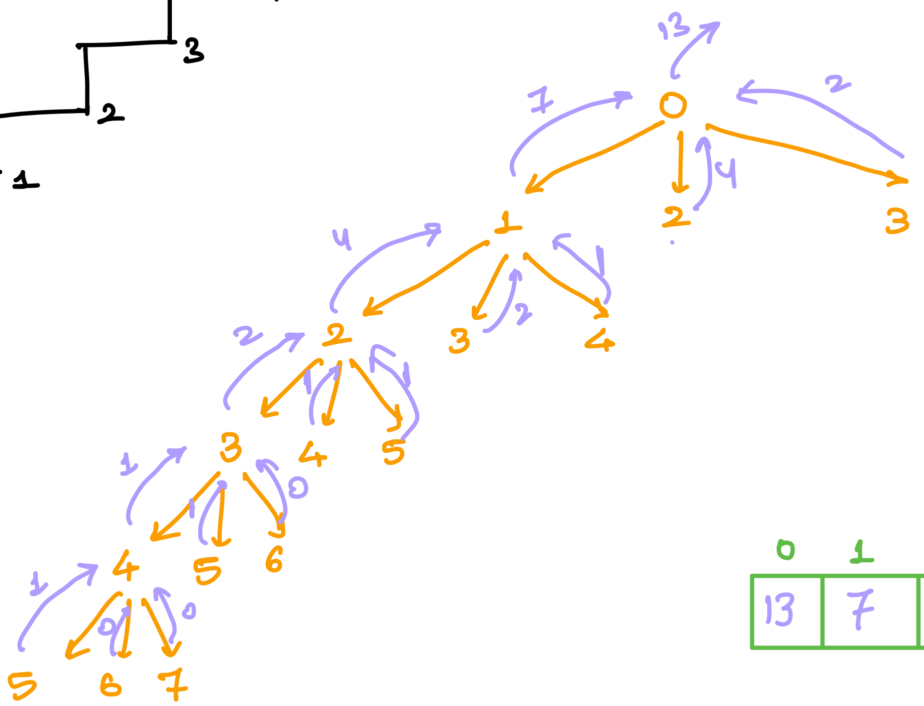
Climbing Stairs

TD DP:



no. of ways Gf → 5th floor ?

1,2,3 jump



0	1	2	3	4
13	7	4	2	1

n=5 index: 7 (n+2) size: n+3

0	1	2	3	4	5	6	7
13	7	4	2	1	1	0	0

BU DP(Iteration)

- Size ?
- TD BC → BU fill work start

n=5

+ve bc: 5

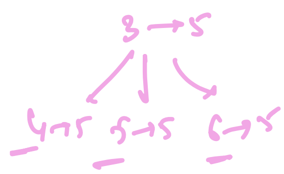
-ve bc: 6, 7

- Cell meaning

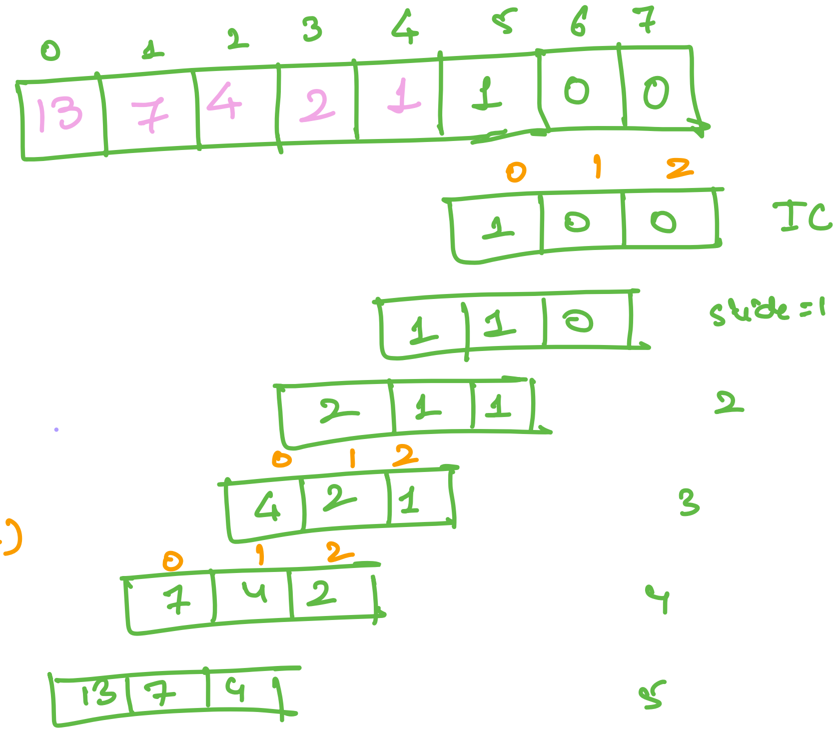
index → ith stair → nth stair ways ?

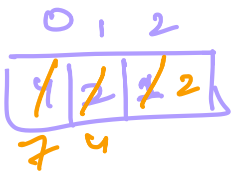
- filling drn ? ⇄
- fill
- final answer.

4



n=5





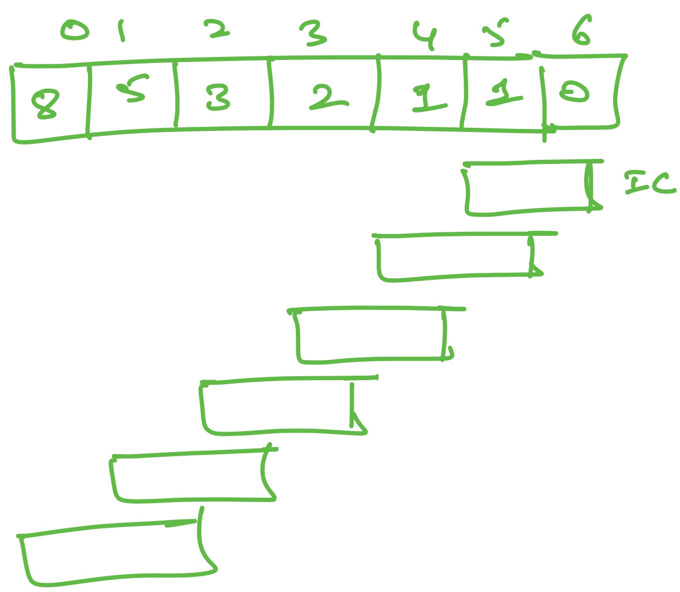
sum = s[0]+s[1]+s[2]
= 7

strg[2] = strg[1]

strg[1] = strg[0]

strg[0] = sum

n=5



CS → count based

Longest Common Subsequence (LCS)

Given 2 strings find the length of longest subsequence which is present in both the strings.

Subsequence

→ Sequence that appears in the same relative order but not necessarily contiguous.

Eg: abcdefg

Subsequence: abc, abg, bdg, aeg, acefg

dbg ✗

Eg:

S1	S2	length of LCS	LCS
abcd	agcfd	3	acd
abc	adcb	2	ab, ac
abc	acd	2	ac

Brute force Approach:

length m abc s1   acd s2 length n

S1 subsequences: [_, a, b, c, ab, ac, bc, abc] → 2^m

S2 subsequences: [_, a, c, d, ac, ad, cd, acd] → 2^n

Common subsequences: [_, a, c, ac]

Time Complexity : $2^m + 2^n + 2^m \cdot 2^n = 2^m + 2^n + 2^{m+n} = \underbrace{O(2^{m+n})}_{\text{Exponential}}$

RECURSION:

Recursive call return: $LCS(s1, s2)$ = s1, s2 LCS length

$LCS(abcd, agcfd) = 3$

Case 1:

s1 abcd, s2 agcfd   a $\boxed{cd}$ (ss) $2+1=3$

↓

bcd, gcfd $\boxed{cd}$ → 2

Case 2:

bcd, gcfd

bcd, cfd — cd: 2

cd, gcfd — cd: 2

longest

0123 abcd, 01234 agcfd   0, 0

↓

bcd, gcfd   1, 1.

1, 2 bcd, cfd

2, 1 cd, gcfd

1, 3 bcd, fd

2, 2 cd, cfd

d, fd 3, 3

1, 4 bcd, d

2, 3 cd, fd

1, 5 bcd, –

2, 4 cd, d

cd, d 2, 4

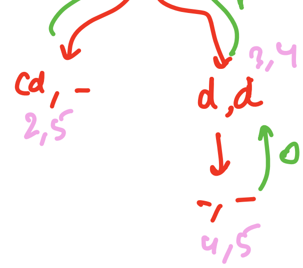
d, fd 3, 3

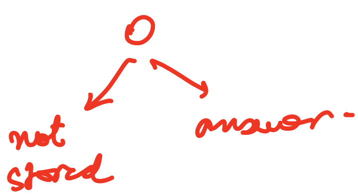
0 1

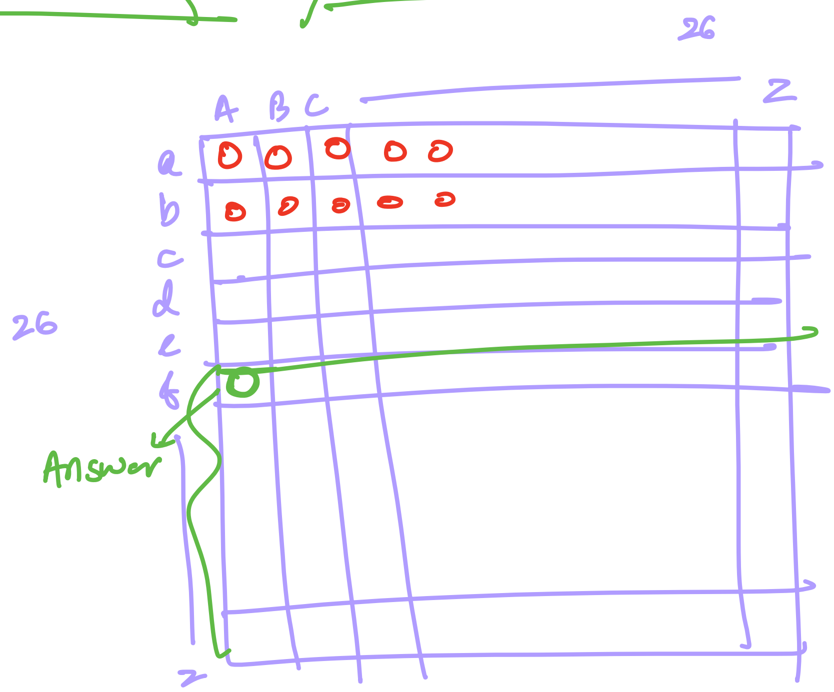
0 1 2 3
abcd

abcd

idx	
0	: abcd
1	: bcd
2	: cd
3 :	d
4 :	–







if 0 is answer
then keep default value in array as -1