Java Week 1

Monday, 12 October 2020 12:30 PM

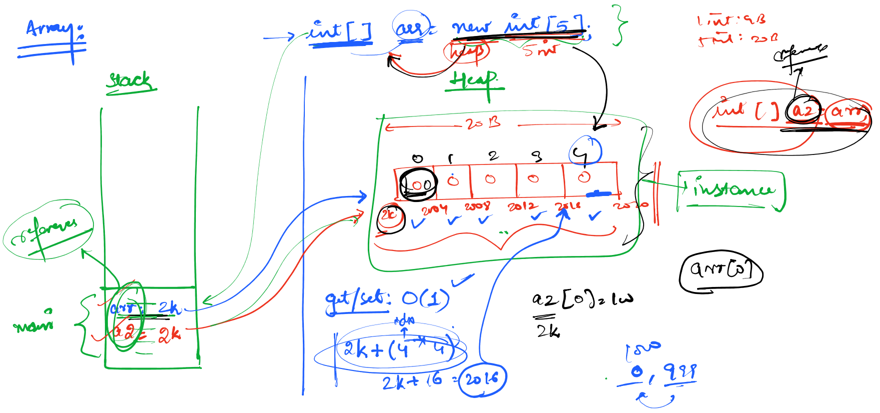
- → Quiz
- → Assignments
- → Hackathons

Java: Platform Independent
→ Byte Code

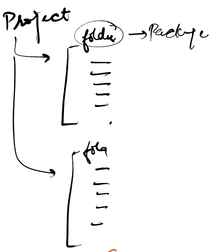
Primitives: byte, short, int, long, double, float, char, boolean

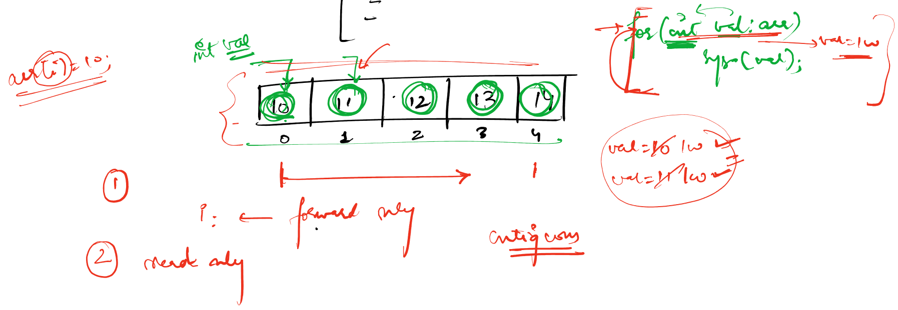
References

Array:

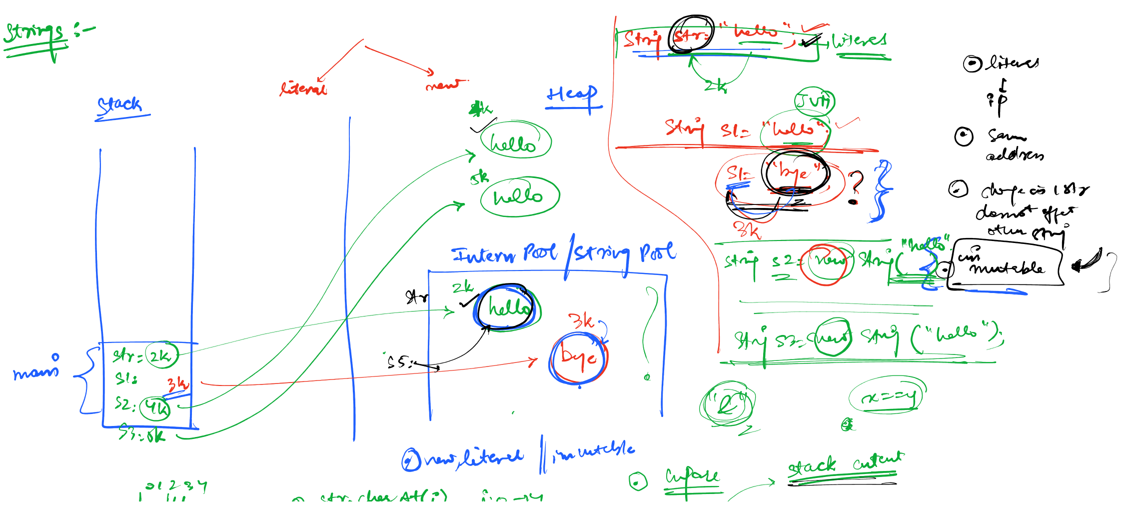


Codes: Eclipse





Strings :-



str: hello (indices 0 1 2 3 4)

(new, literal || immutable)

- str.charAt(i) i: 0 → 4
- str.substring(2) llo
- str.ss(0, 3) : hel (si included, ei excluded)

- compare: stack content
- str == s2 (2k ≠ 4k)
- str.equals(s2) → true / false

n 10^5 times — assume: O(n)

10^5 instructions total

1 GHz → 10^9 ins/sec → < 1 sec

string addition → O(n)

```java
long start = System.currentTimeMillis();
int n = 100000;
String s = "";
for (int i = 1; i <= n; i++) {
s = s + i;
}
long end = System.currentTimeMillis();

System.out.println(end - start);
```

n^2 — O(n) — 12ms

String str = "hello";

String str1 = str + "a"; (str: 2k, str1: 3k, hello a: 3k, hello: 2k, IP) O(n)

s = "" (1k)

i=1: "1" → 0+1 = 1 ✓

i=2: "12" → 1+1 = 2 ✓

i=3: "123" → 2+1 = 3 ✓

i=4: "1234" → 3+1 = 4 ✓

⋮

i=n: "1 … n" → n−1+1 = n ✓

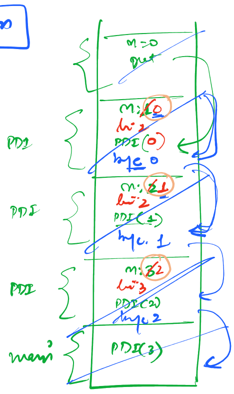
1+2+3 … n = $\frac{n(n+1)}{2} \approx n^2$

n^2: $(10^5)^2 = \frac{10^{10}}{10^9} = \sim 10$ sec

String s = "abc" + "de" + "def";

"abcdef" → after IP & inside str → abcdef / already

Recursion



hi 3
hi 2
hi 1
bye 0
bye 1
bye 2

```java
public static void PDI(int n) {

if (n == 0) {
return;
}

System.out.println("hii " + n);
PDI( -- n);
System.out.println("bye " + n);
}
```

--n → n changes

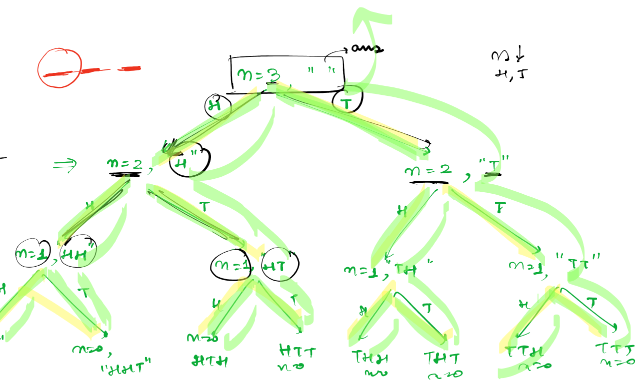
n−1 → n ✗

Q1: Coin Toss

n=3

HHH
HHT
HTH
HTT
THH
THT
TTH
TTT

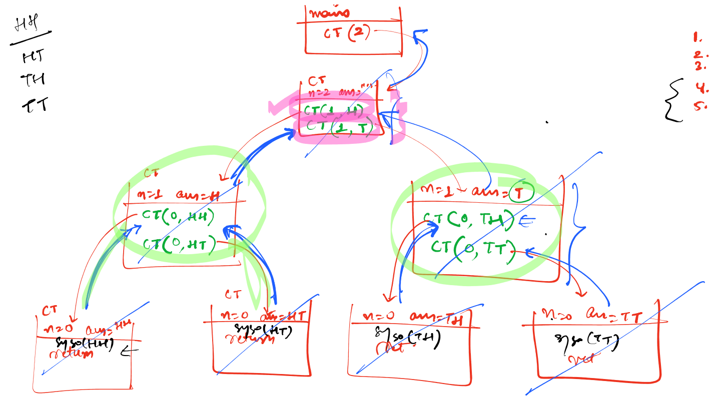
n ↓ H, T



HH / HT / TH

main: CT(2)

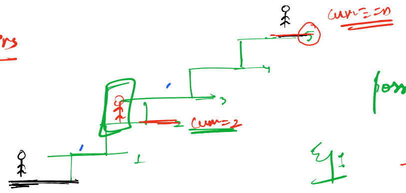
```java
if(n == 0) {
    System.out.println(ans);
    return;
}
coinToss(n-1, ans + "H") ;
```



```
1. if(n == 0) {
2.     System.out.println(ans);
3.     return;
4. }
5. coinToss(n-1, ans + "H") ;
   coinToss(n-1, ans + "T") ;
```

Q2: Climb stairs

options
- 1 jump ✓
- 2 jumps ✓
- 3 jumps ✓



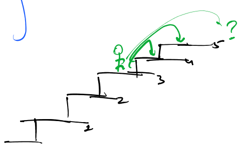
n=5

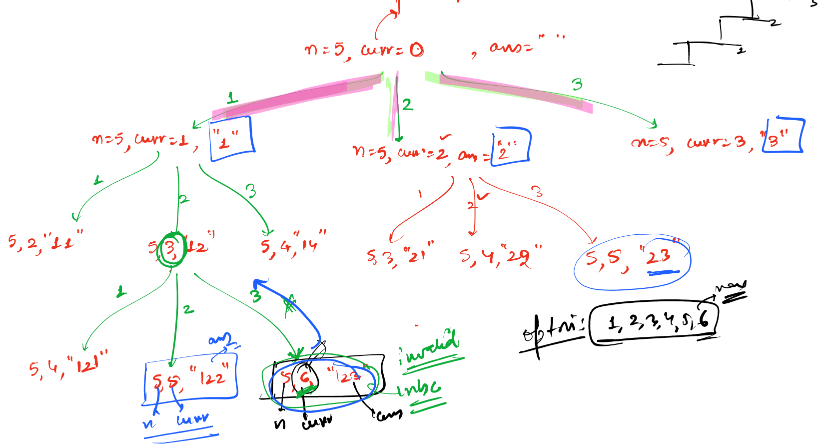
0th → nth stair

possible ways of reaching to the nth stair ?

Eg:
- 11111
- 1112
- 32
- 1211

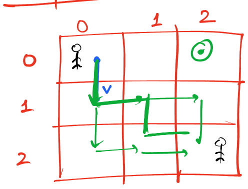
print





optmi: (1, 2, 3, 4, 5, 6)

Q3: Maze Path



er : 2
ec : 2

(0,0) → (2,2)

all possible ways of reaching from (0,0) → (er, ec)

options: 1H, 1V

+ve moves

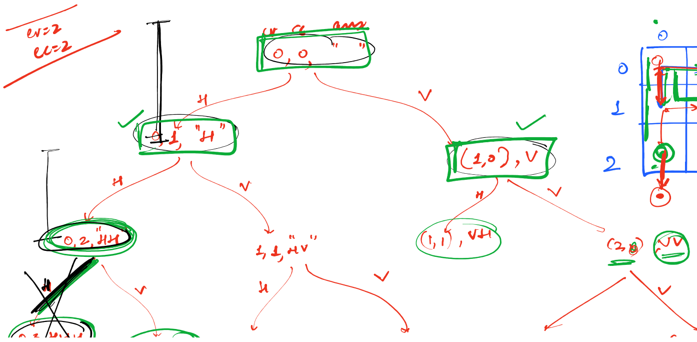
cr, cc

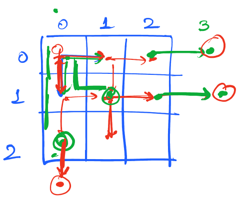
(0,2)

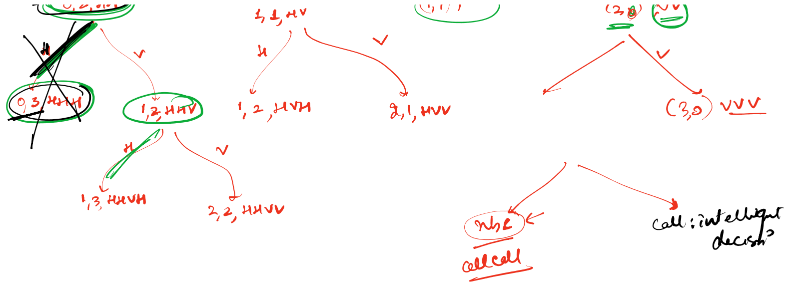
Results:

HHVV	VHHV
HVHV	VVHH
HVVH	VHVH

er=2
ec=2







Q4: Valid Parenthesis

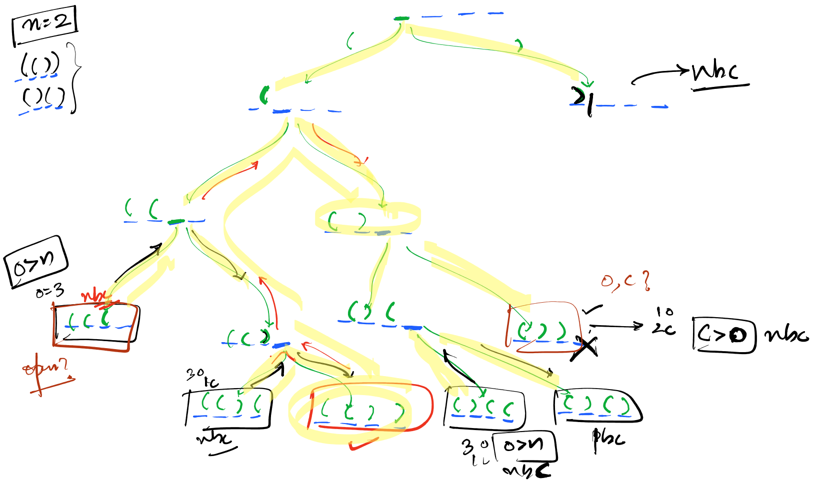
Bracket: ( )   { }✗ [ ]✗

$\boxed{n=3}$ → 3 open brackets, 3 close brackets

n=2
(( ))
( ) ( )

Eg:
((( )))
(( ) ( ))
(( )) ( )
( ) (( ))
( ) ( ) ( )
} 5 valid configuration

invalid:
) (( )
)) ((



Homework:
1) Assignment: HB ✓
2) Quiz: Recursion ✓

Backtracking =

n i t i n

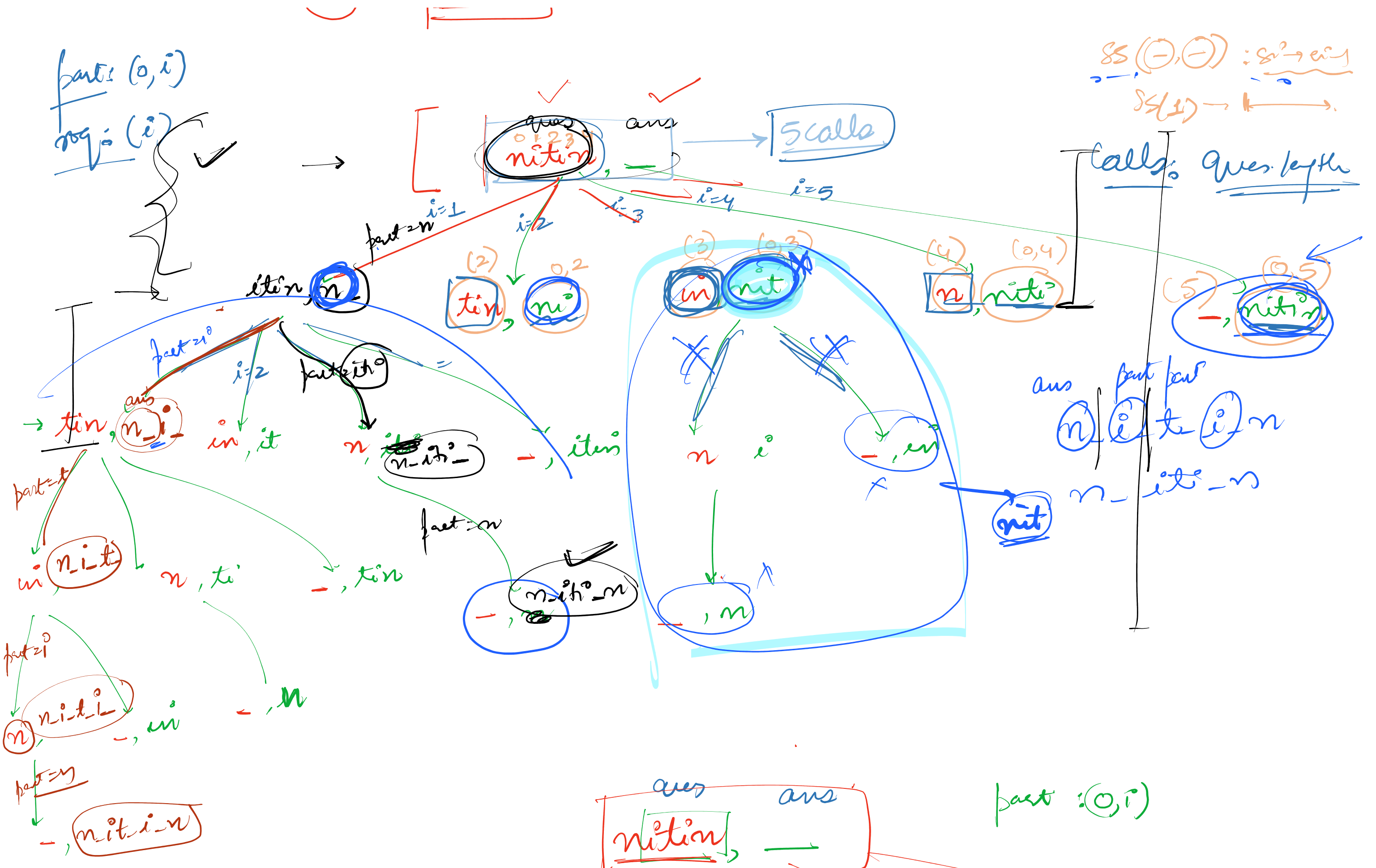
1) n i t i n   parts: palindrome

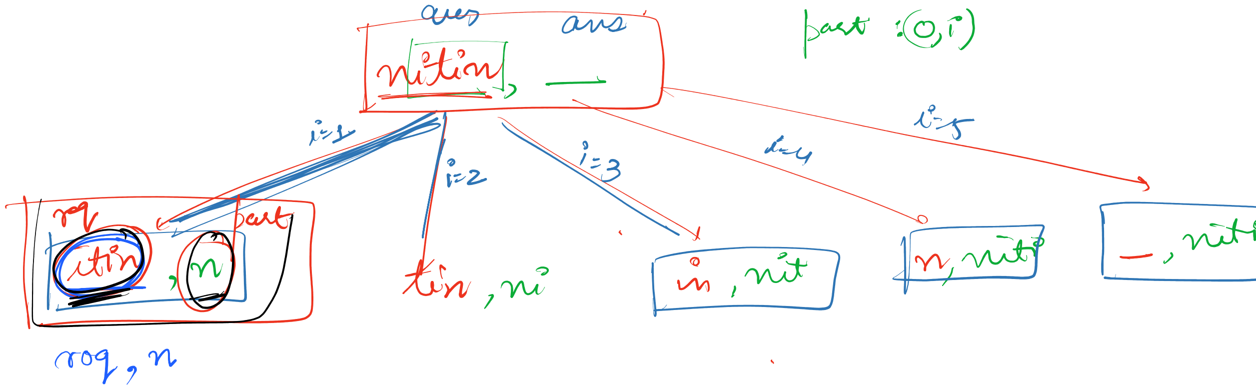
2) n iti n
part part part

3) nitin

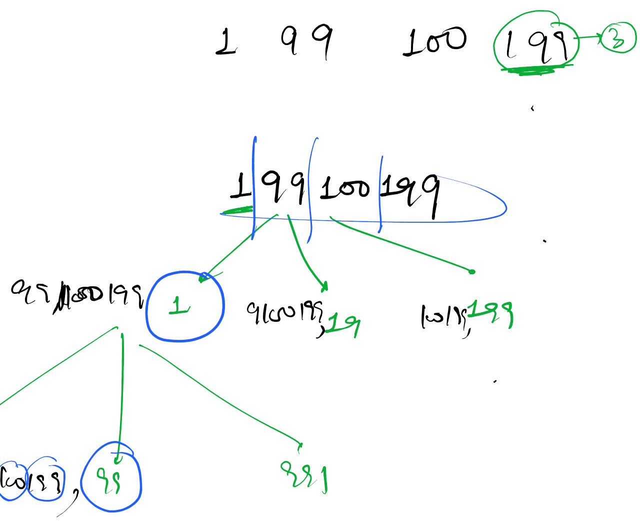
parts: (0, i)

SS(—,—) : si → ei





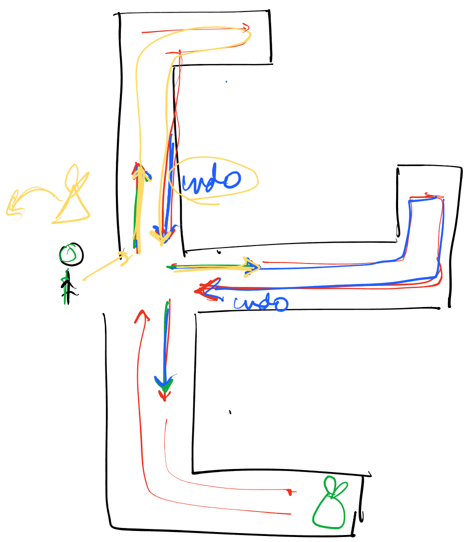
199100199

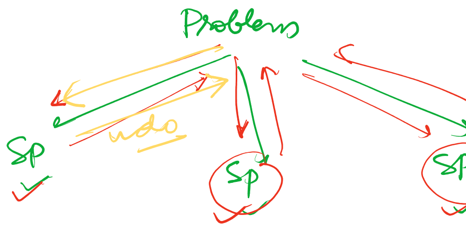


Backtracking:

Problems

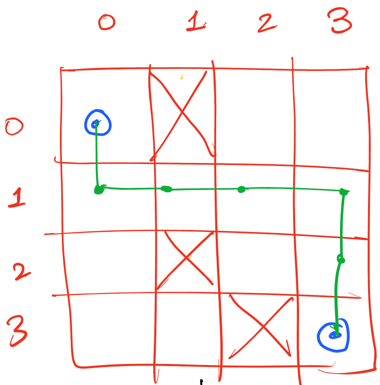
Backtracking:





1 Blocked Maze

T
L R
D



(0,0) → (3,3)

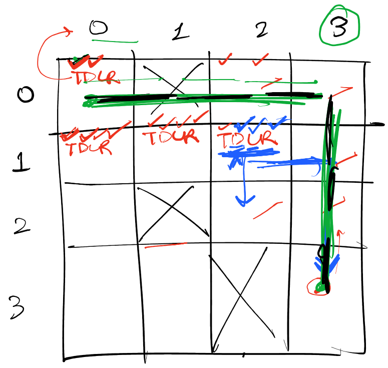
ways ?

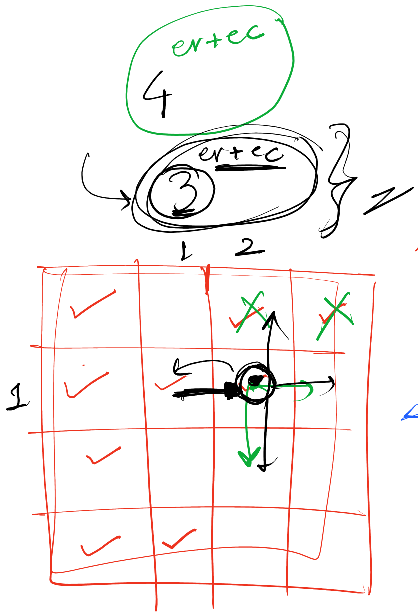
DRRTRDDD
DRRDRD
DRRRDD

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

T
L R
D

TDLR



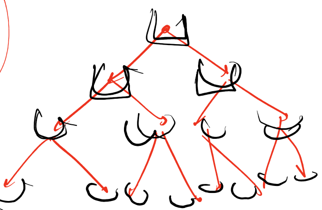


Sudoku Solver:

0 1 2 3 4 5 6 7 8 9 (cr, cc)

3		6	5		8	4		
5	2							

(work done in 1 fxn frame) → 1



0,0

(0,1)
ret

cr, cc+1

0.8 1

Solvers:-

1 → 9

3		6	5		8	4		
5	2							
	8	7					3	1
		3		1			8	
9			8	6	3			5
	5			9		6		
1	3					2	5	
							7	4
		5	2		6	3		

cr=7

cr%3

7%3

cr,cc

cr

cr-cr%3

cr%3

Eg:
7chocolates
3students

3 4 5 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
5 3 4 2 8 6 1 7 9

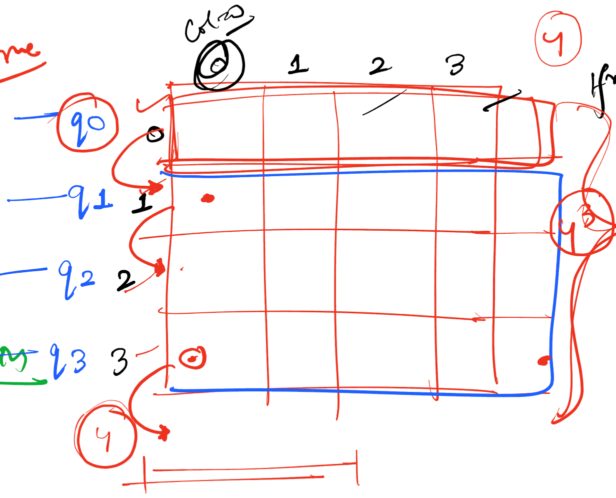
N Queens:-

boolean 2D
place: true

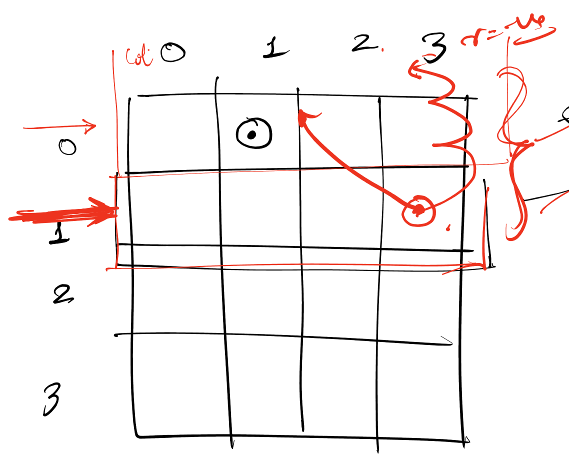
$\boxed{\text{1 Row: 1 Queen}}$

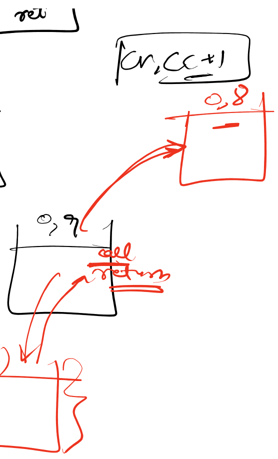
Total Configurations?

256 configurations



kill

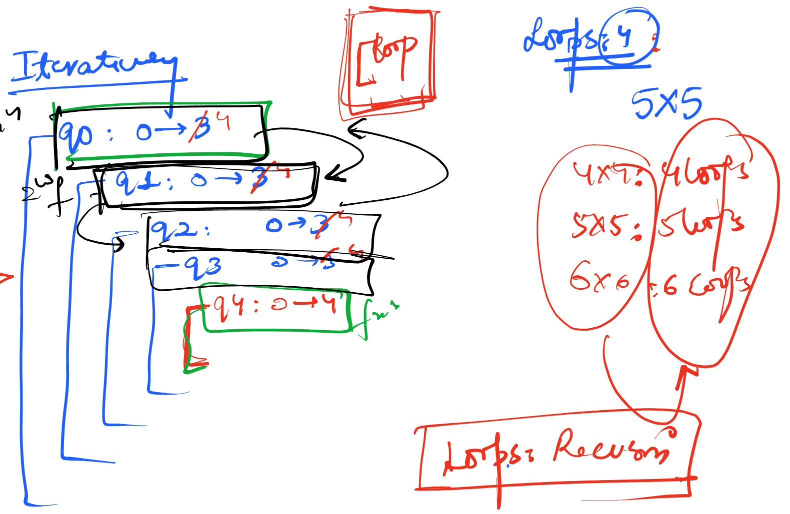




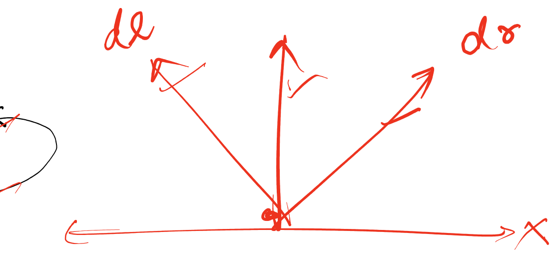
cr, cc+1
0,8
0,9
call
return

$7/3 = 2$

$7 \% 3 = 1$



Iteratively
q0 : 0 → 3 4
q1 : 0 → 3 4
q2 : 0 → 3 4
q3 0 → 3 4
q4 : 0 → 4
Loop
Loops : 4
5X5
4X4 : 4 Loops
5X5 : 5 Loops
6X6 : 6 Loops
Loops : Recursion



dl
dr
X

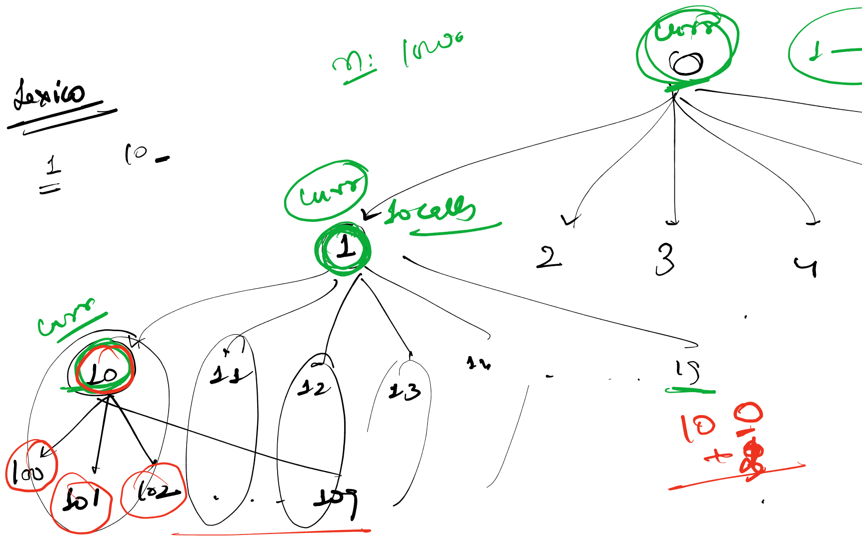
→ Quiz: 5:30 pm: 20 mins } → MCQ

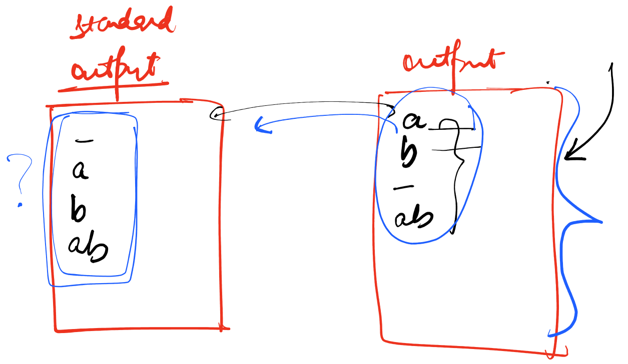
→ Assignments: Thursday (DP)

→ Thursday: 12:30 pm: DP

Lexico

1 10



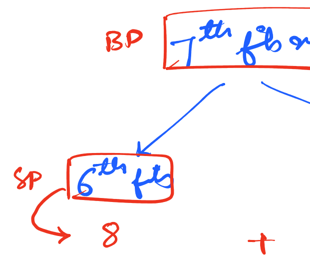


Dynamic Programming

Fibonacci

0^{th}	1^{th}	2^{th}	3^{th}	4^{th}	5^{th}	6^{th}
0,	1,	1,	2,	3,	5,	8,

Recursion:



Stack

Lexico

9

5 . . . 9

10 → 100

→ 101

→ 102

→ 103

109

curr * 10 + i

i : 0 → 9

Stack

7th

13 . . .

n : nth fib no ?

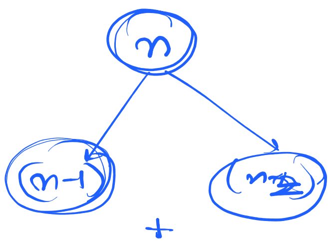
↓

7 : 13

nth ?

sp 5th fib

5 = 13

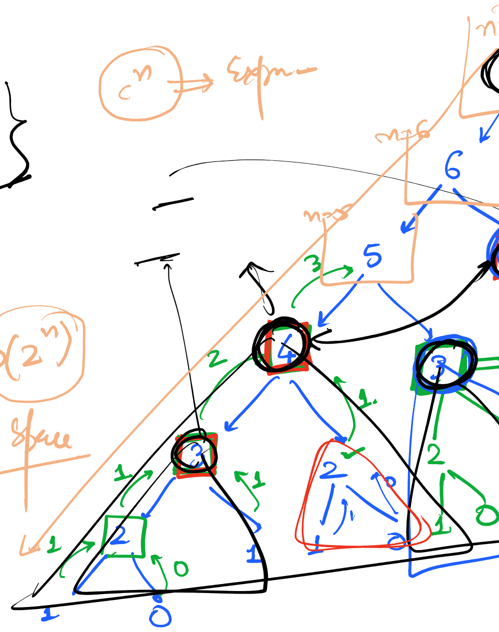


DP:

overlapping subproblems

Space

$c^n \rightarrow$ Exp...



TC: $2^n \cdot 1 = O(2^n)$

SC: Rec Extra Space

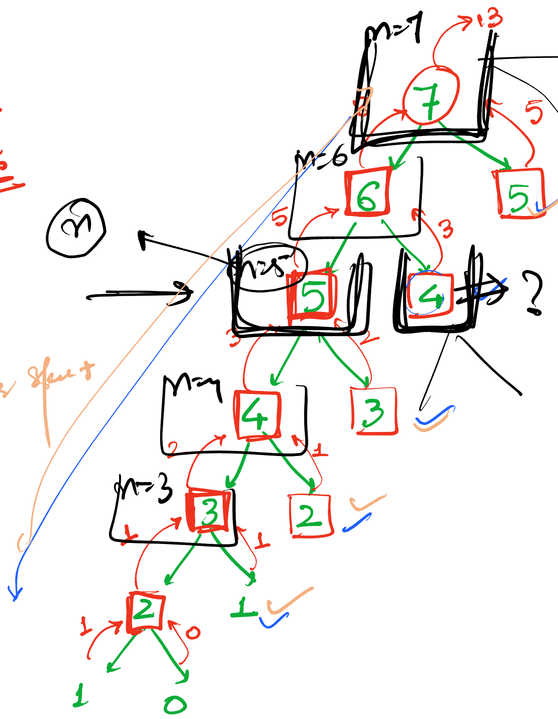
DP

- TD (Memoization)
- BU (Tabulation) ✓

Fib

TD DP:

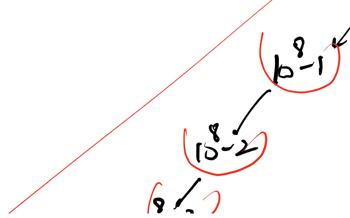
(n)



TC: $O(n)$

SC: Rec Extra Space + $O(n)$

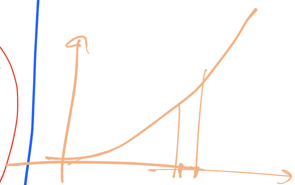
Problems



19^2

$$\begin{array}{r} 19 \\ \times\ 19 \\ \hline 171 \\ 19\times \\ \hline 1 \\ \hline \end{array}$$

19^{27}

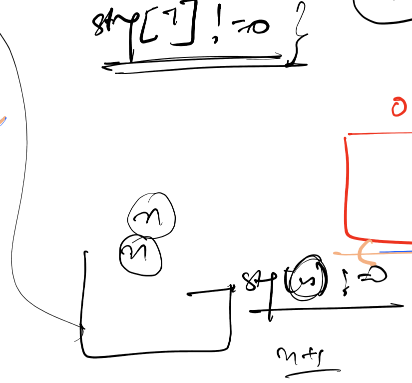


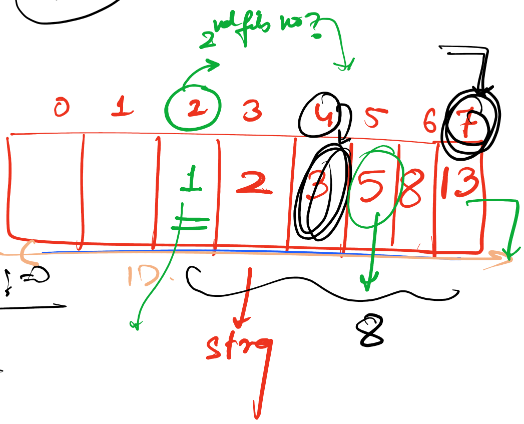
Rec
↓
TD
↓
BU

$n==0 \,||\, n==1$ ✗

strg[7] != 0

!=0

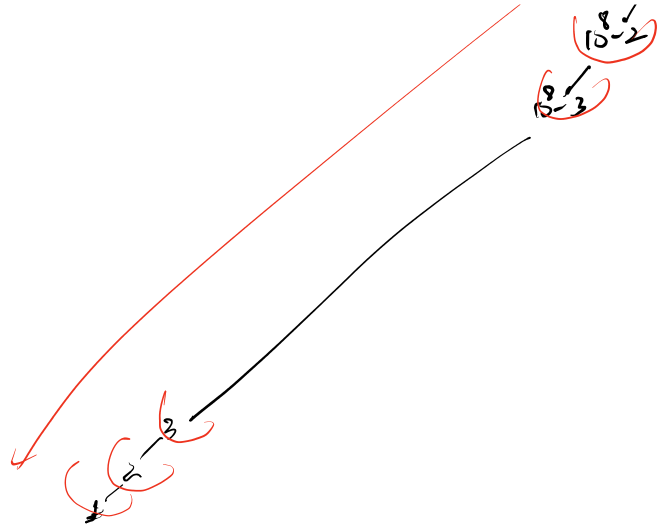




strg

$\boxed{10^8}$

$n = \boxed{10^8}$



Bottom up

→ cell meaning? : TD ?

TC: $O(n)$

SC: $O(n)$

→ array size? : $n+1$

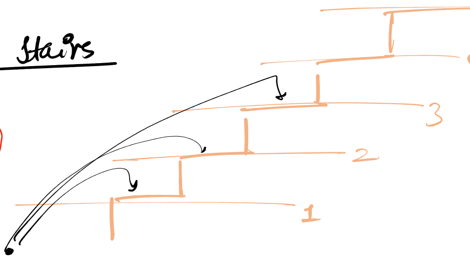
→ filling dir? ?

→ TD: BC ✓
BU: fib ✓

→ fill logic?

BU space efficient: TC: $O(n)$
SC: $O(1)$

Q: Climb Stairs



Rec: $\text{calls}^{ht} * (\text{Work})$

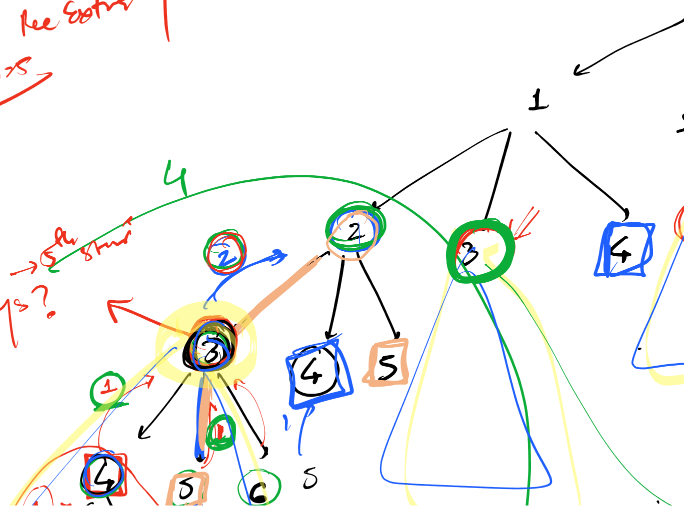
$3^n * 1$

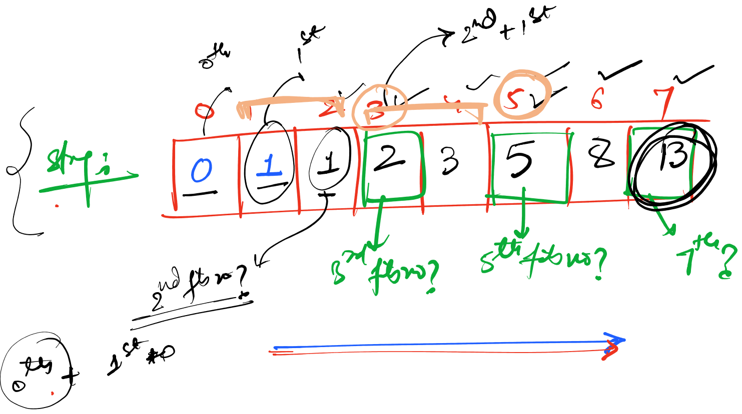
TC: $O(3^n)$

SC: Rec extra space

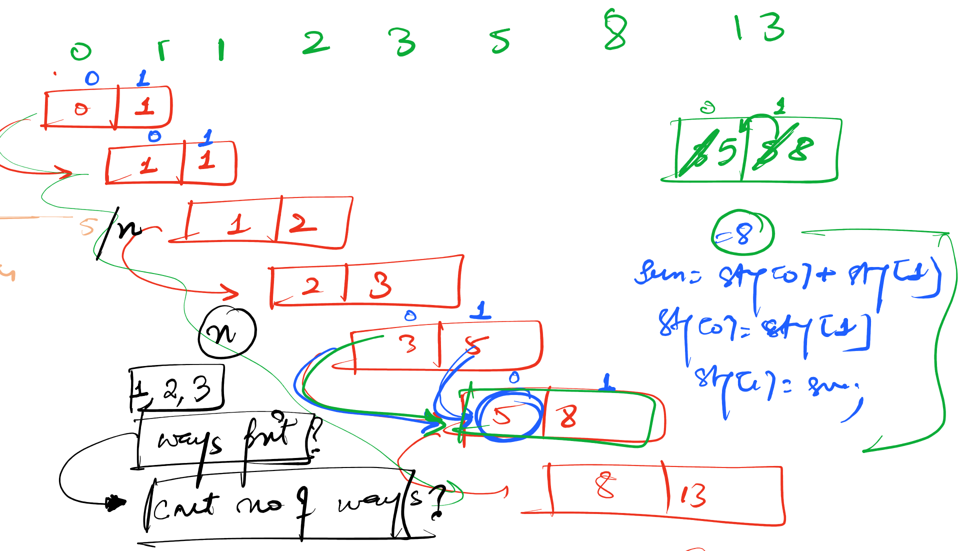
$n > 5$

3rd stair → 5th stair
ways?

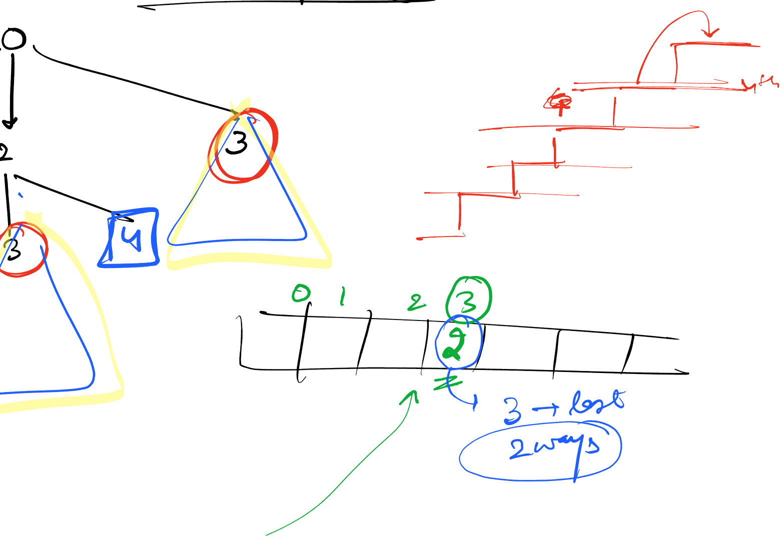




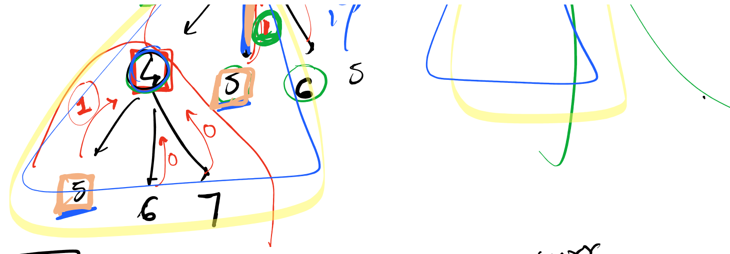
0 1 1 2 3 5 8 13
2nd + 1st
2nd fib no?
3rd fibno?
5th fib no?
7th?



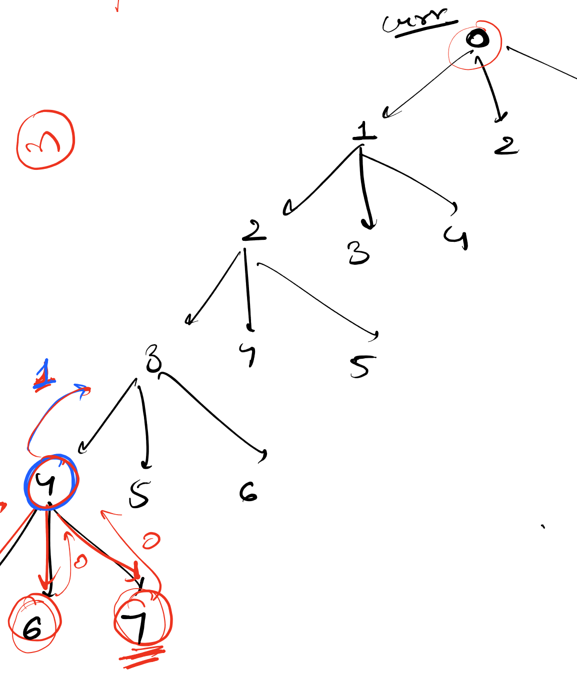
0 1 1 2 3 5 8 13
sum = str[0] + str[1]
str[0] = str[1]
str[1] = sum;
n
1, 2, 3
ways fnt?
Cnt no of ways?



0 1 2 3
3 → lost
2 ways



$\boxed{m=5}$



Bottom up:

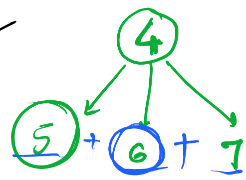
→ cell meaning ? no. of ways

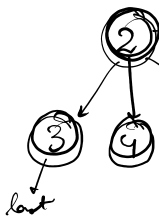
→ array size ? $\boxed{m:5}$ 1 index & size $[n+3]$

→ filling dir^n ?

→ TD BC : BU fill

→ fill logic ?



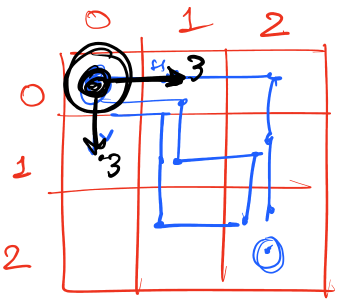


BU SC: SC: O(1)

TC: O

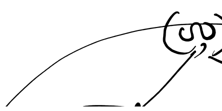
SC: O

Maze paths

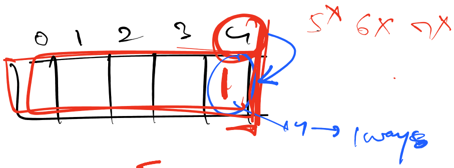


er = 2
ec = 2

6 path



2 ways

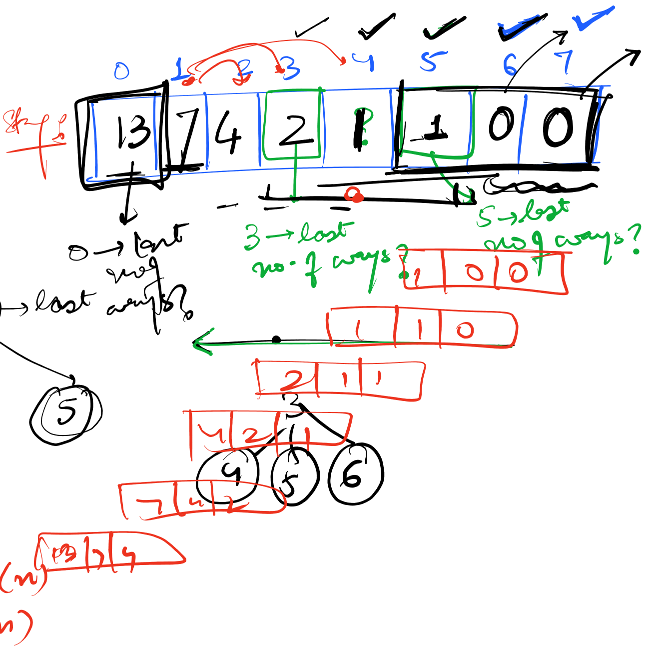


Size: 5
↳ n

TC: n unique problem * work

$n * 1 = n$

SC: Rec Stack Space + n

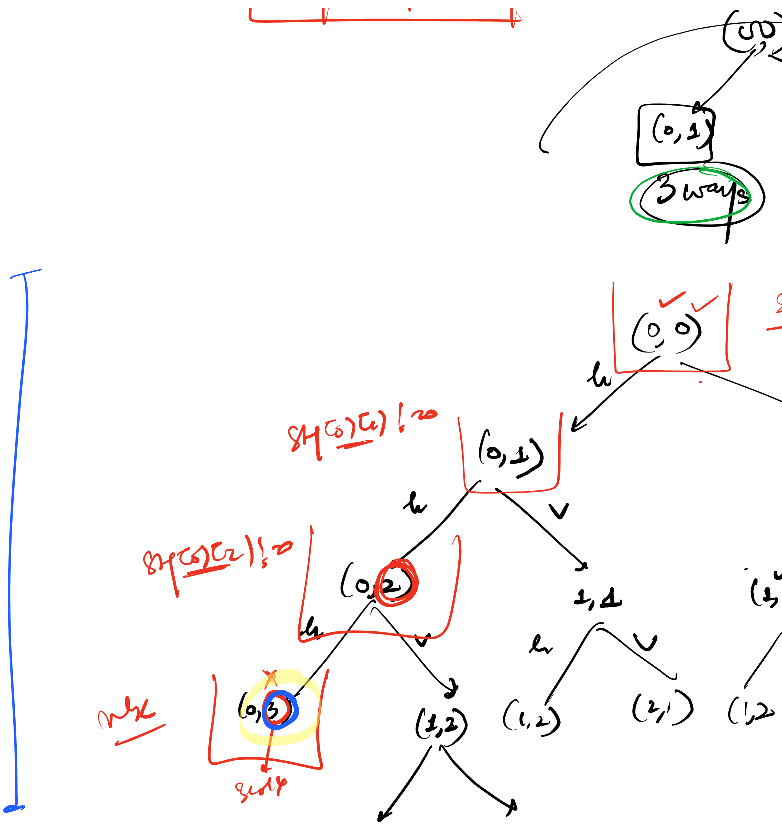


ways?

Count? no. of ways

s/ways

6 ways



Rec: TC: 2^{r+c}

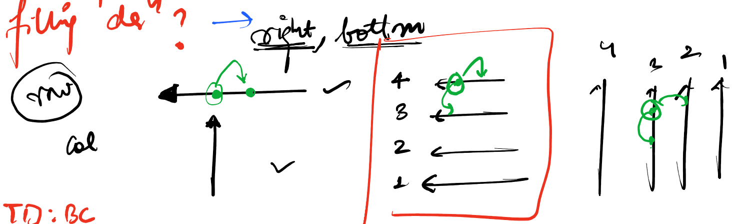
TD: TC: $(r \times c)$

Bottom Up:

→ cell meaning ? : Count or no. of ways to reach

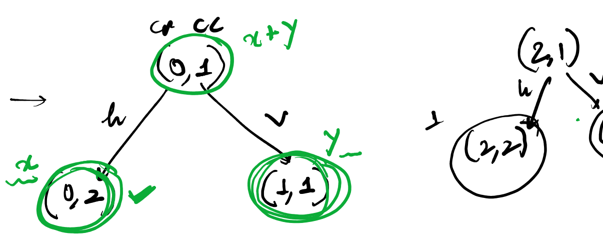
→ array size ? → $(r+2)(c+2)$

→ filling dir^n ? → right, bottom



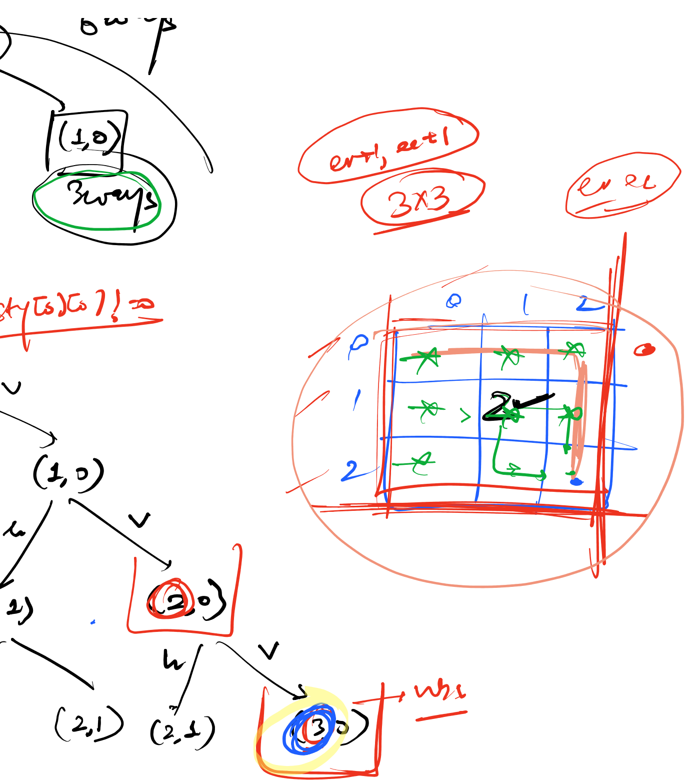
→ TD: BC
BU: fill

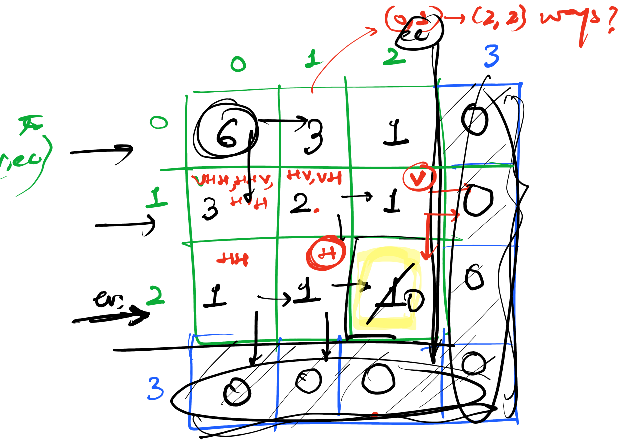
→ fill logic ? →

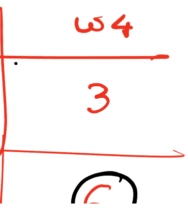


Wine Problem:

	W1	W2	W3
yr=1	1	4	2







Max profit

Customers

$1 + 6 + 6 + 16 = 29$

Greedy Approach

[Best Soln

[But Soln

[But

yr=1	1		
yr=2	~~2~~	8	4
yr=3	~~3~~	12	6
yr=4	~~4~~	16	~~8~~

eg:

	w1	w2	w3
yr=1	2	3	5
yr=2	4	6	10
yr=3	6	9	15
yr=4	8	12	20
yr=5	10	15	25

Recursion

yr=1 → {2, 3, …

→ $\{3, 5, 1, 4\}_{x} + 2$

first sell: max profit

visually break

yr=1

{4, 4}

5x5

nxn

yr=2

si=1
ei=4

48

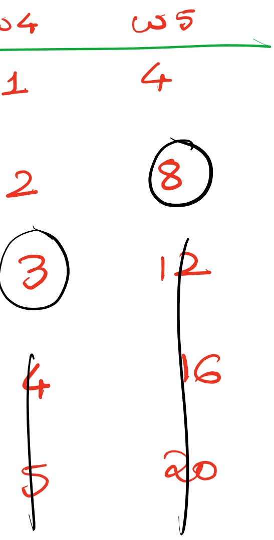
si=2
ei=4

$1+6+6+16 = 29$

Best Soln

Best

Globally Best Soln



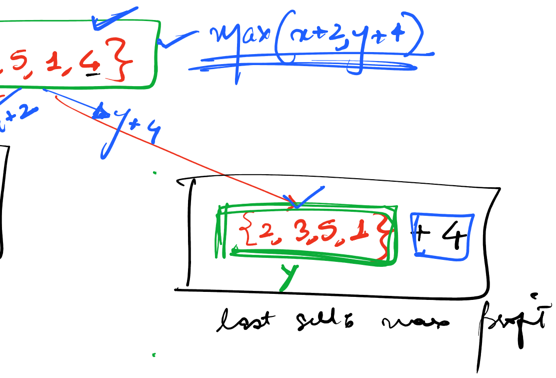
$2+6+12+4+25$

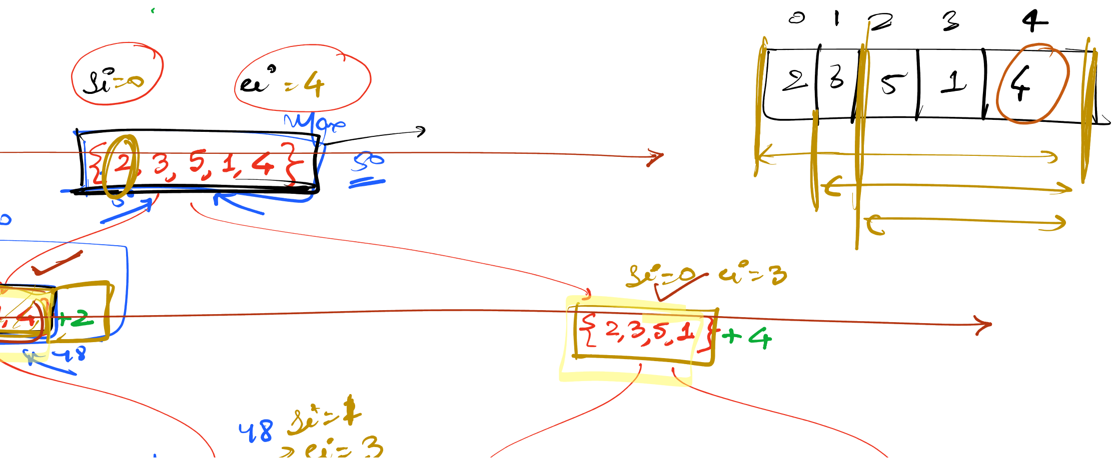
$= 14+10+25$

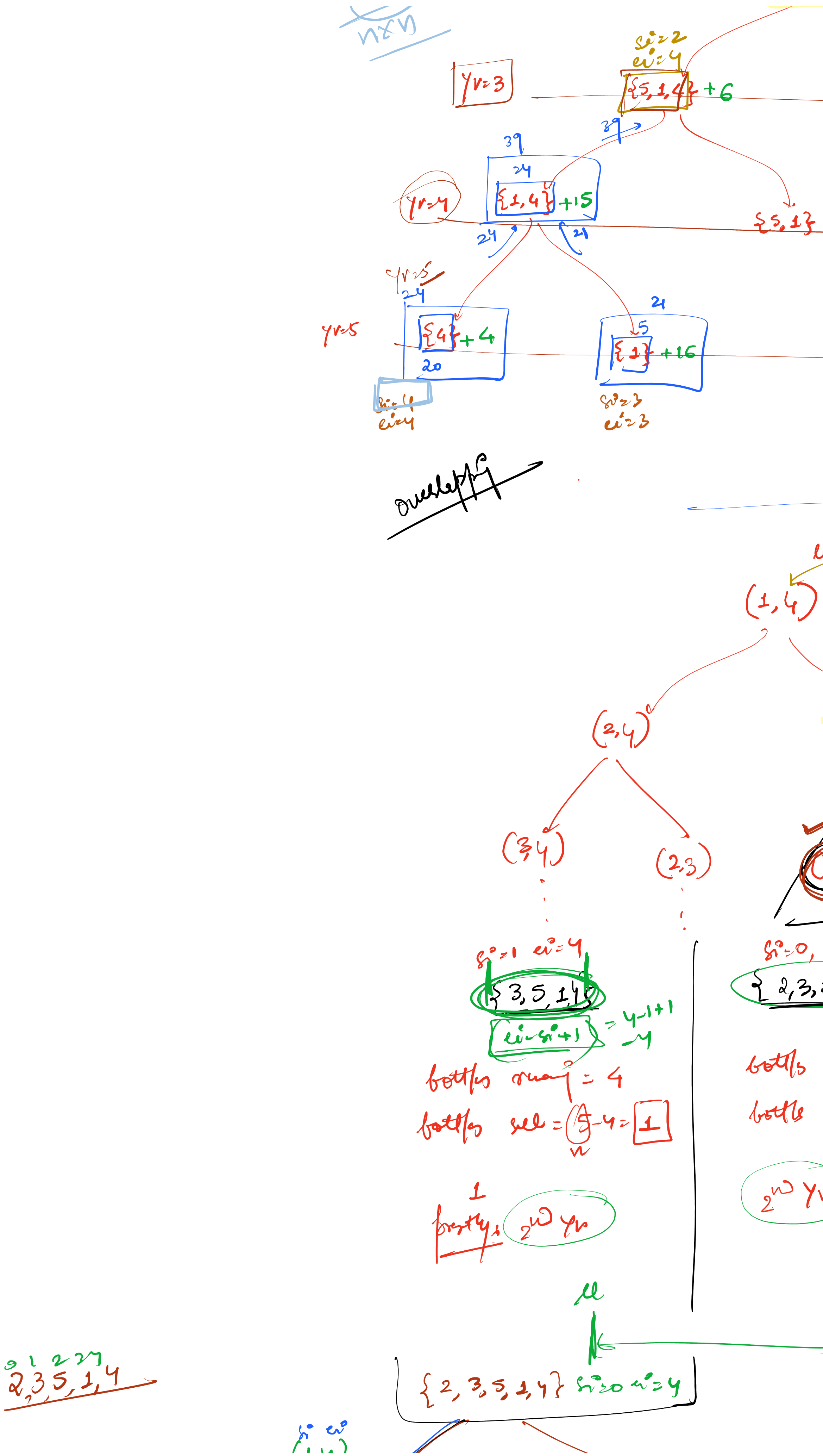
$= 49$

$2+8+3+12+25 = \boxed{50}$

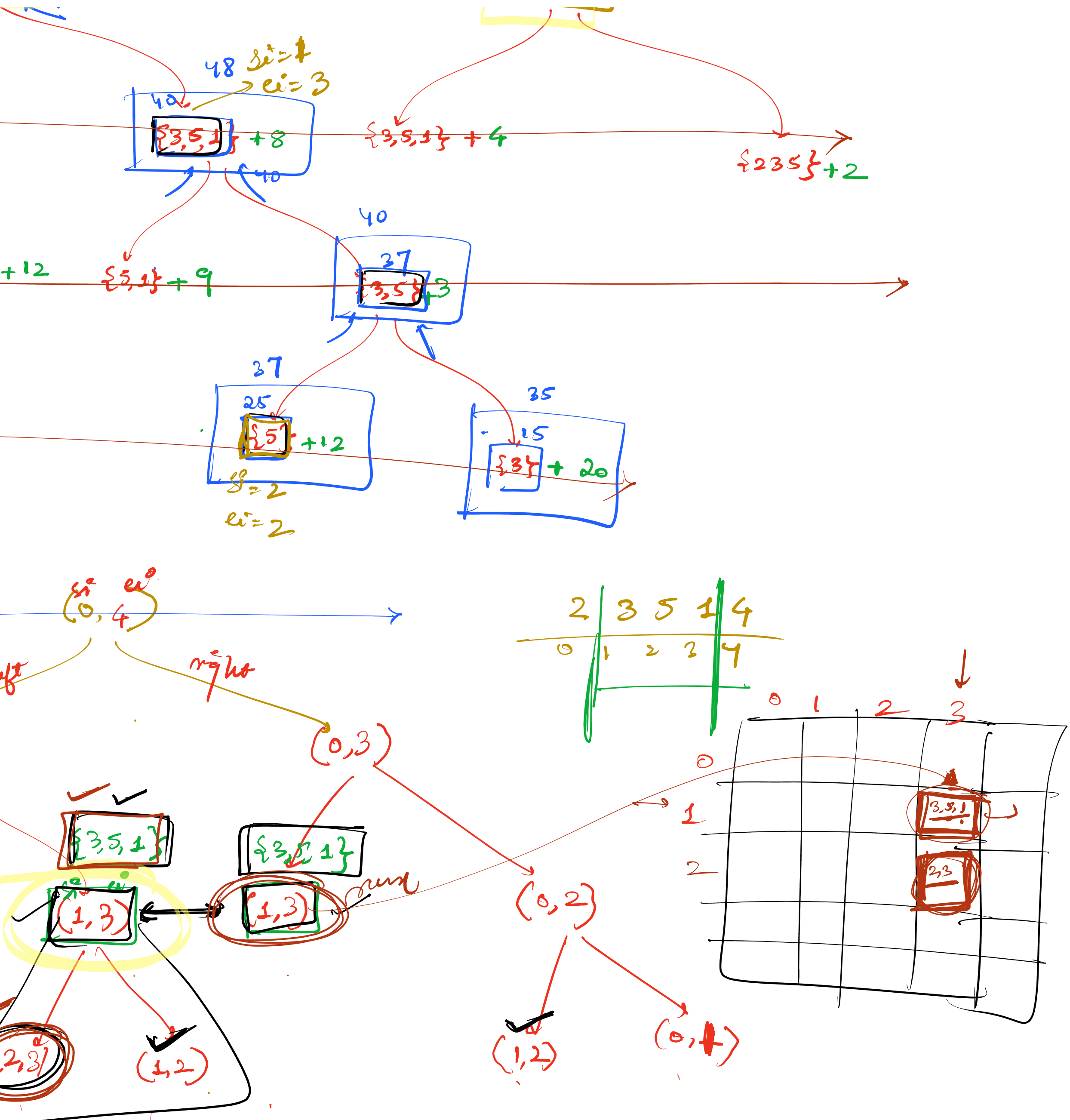
Greedy ✗







n×n
yr=3
si=2
ei=4
{5,1,4}+6
39
24
yr=4
{1,4}+15
{5,1}
yr=5
{4}+4
20
21
{1}+16
si=1
ei=4
si=3
ei=3
overlapping
(1,4)
(2,4)
(3,4)
(2,3)
si=1 ei=4
{3,5,1,4}
ei-si+1 = 4-1+1 = 4
bottles remaining = 4
bottles sell = 5-4 = 1
1
profit, 2nd yr
si=0,
{2,3,
bottles
bottle
2nd yr
{2,3,5,1,4} si=0 ei=4
0 1 2 3 4
2,3,5,1,4
si ei



ei = 3

{3,5,1}

remj = 4 ⇒ 3-0+1 = 4

sell: 5-4 = 1 → 1yr

si = 2 ei = 3

2	3	5	1	4
0	1	2	3	4

{5,1}

bottles remj = 2   3-2+1=2.

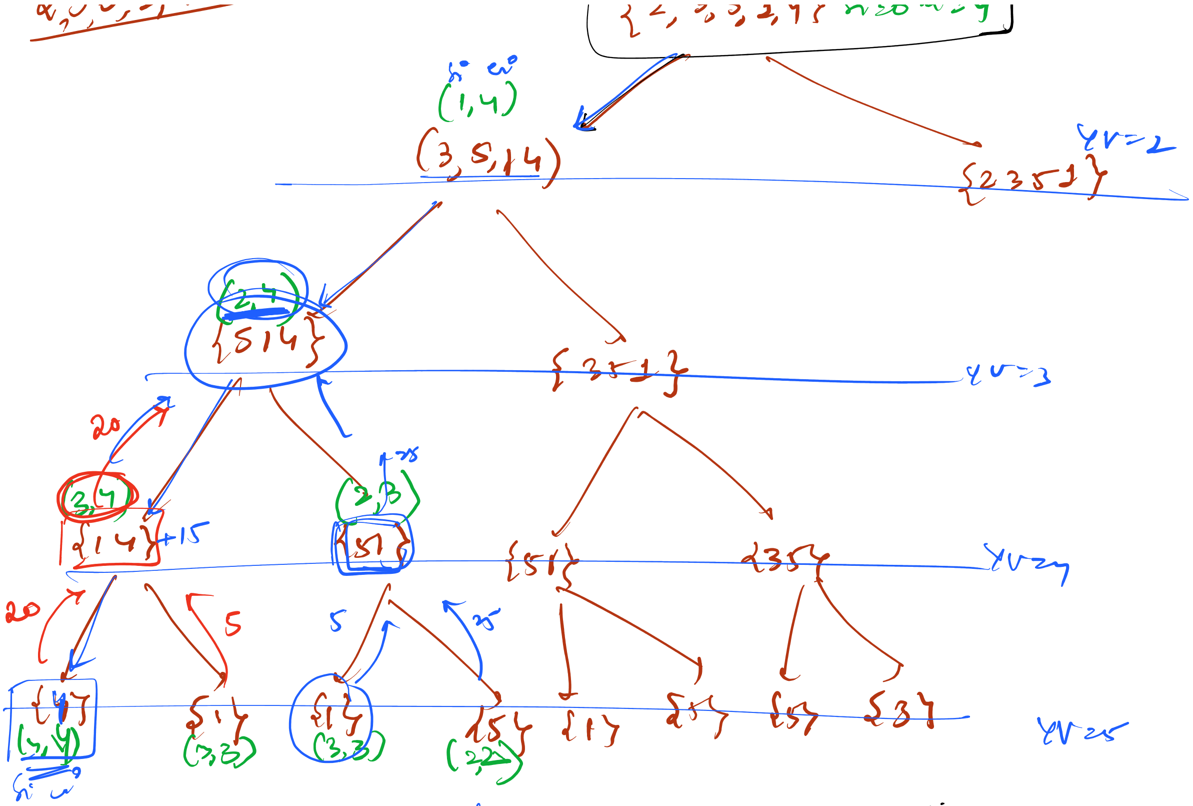
sell = 5-2 = 3

4th yr

bottles remj = $ei - si + 1$

sell = $n - (ei - si + 1)$

p. yr = $\boxed{n - (ei - si + 1) + 1}$



→ DP lecture: 12:30 pm

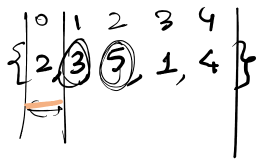
→ wp BU try

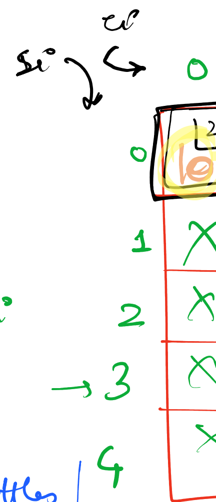
→ Assignments Rec: Sunday

→ Codes

→ DP Assignment: Tomorrow

Bottom up





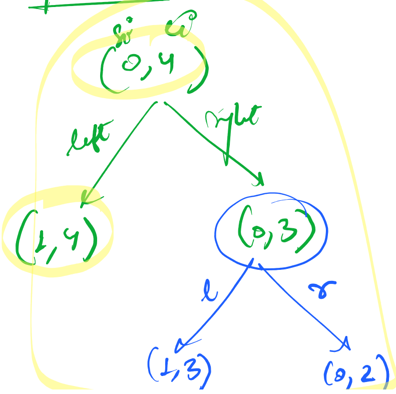
→ Cell meaning

→ array size : n x n

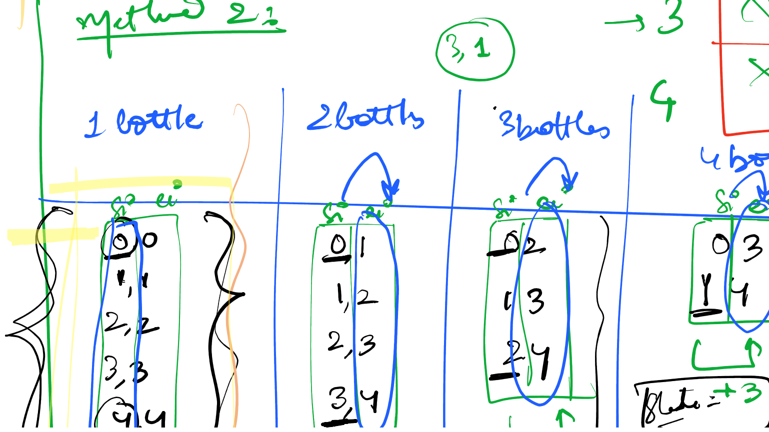
→ filling dir: diagonally

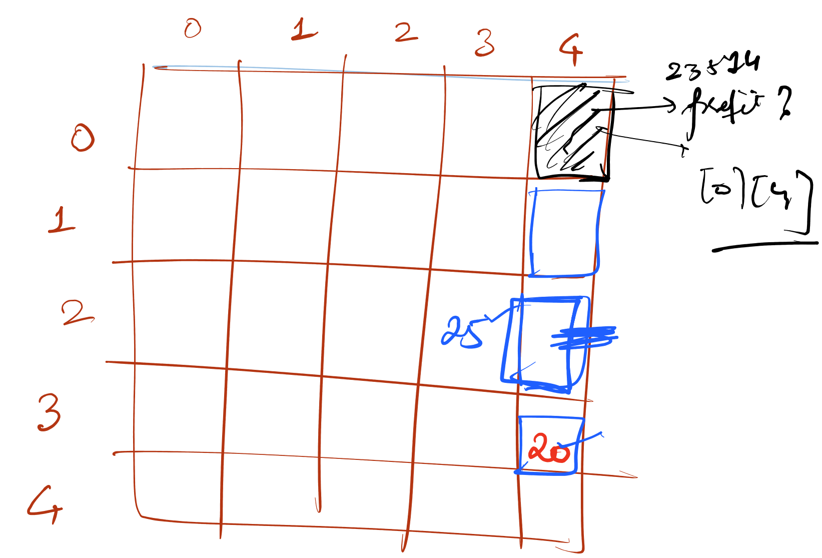
si > ei

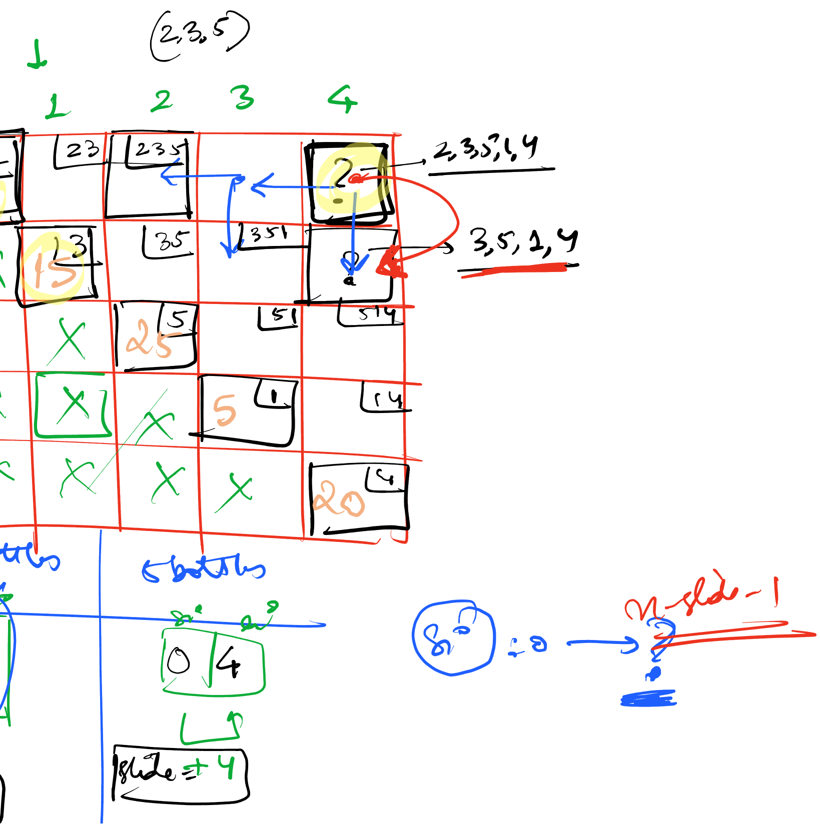
Method 1

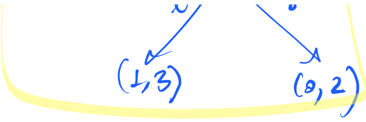


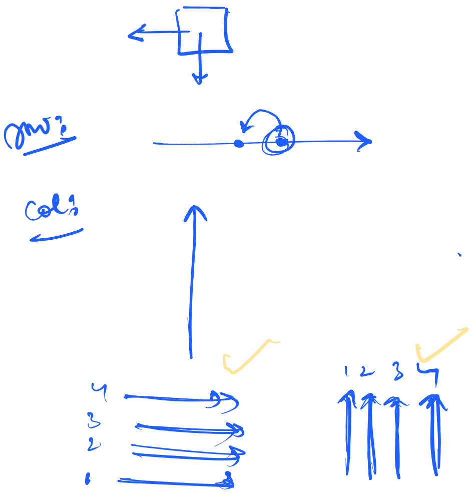
Method 2:

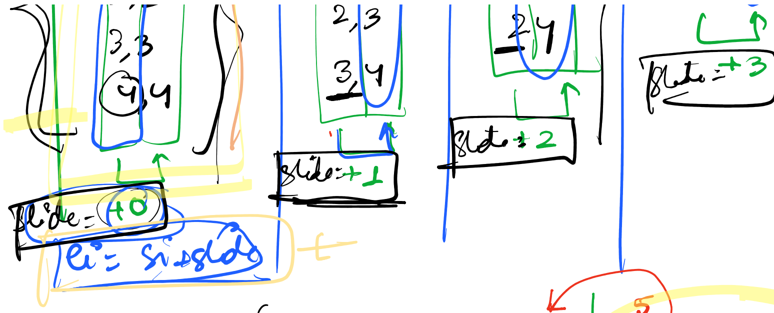



23514
prefix ?
[0][4]
25
20


(2,3,5)
2,3,5,1,4
3,5,1,4
slide = 4
n-slide-1







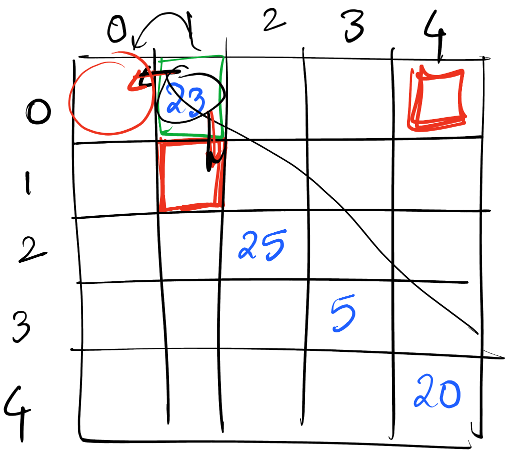
slide	si ending point	n - slide
0	4	5 - 1
1	3	4 - 1
2	2	3 - 1
3	1	2 - 1
4	0	1 - 1

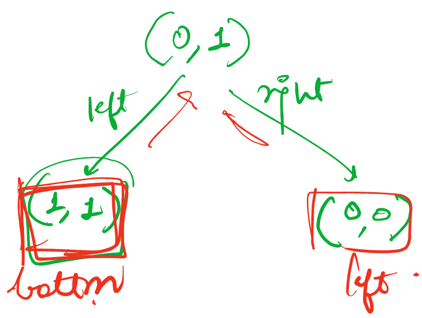
→ TD 3C
BU fill

→ fill? $\boxed{\text{TD}}$

$\{2, 3, 5, 1, 4\}$
0 1 2 3 4

for (i





}

slide = 4

si ending point = n - slide - 1

0 1 2 3 4

5 - 1 - 1

0 1 2 3

= 4

bottom + 5 + 2 * 4 = 15 + 8 = 23

10 + 12 = 22

```
nt slide = 0; slide <= n - 1; slide++) {

 for (int si = 0; si <= n - slide - 1; si++) {

         int ei = si + slide;

         int yr = n - (ei - si + 1) + 1;

         if (si == ei) {
             strg[si][ei] = arr[si] * yr;
         } else {

             int left = strg[si + 1][ei] + arr[si] * yr;
             int right = strg[si][ei - 1] + arr[ei] * yr;

             int res = Math.max(left, right);

             strg[si][ei] = res;

         }

 }
```

Wildcard Pattern Matching

Given a text and a wildcard pattern, implement wildcard pattern matching algorithm that finds if wildcard pattern is matched with text. The matching should cover the entire text (not partial text).

The wildcard pattern can include the characters '?' and '*'

'?' – matches any single character

'*' – Matches any sequence of characters (including the empty sequence)

src → alphabets ⟷ pattern → alphabets
→ ? : 1 char
→ * : blank, no chars, multiple

For example,

```
Text = "baaabab",
Pattern = "*****ba*****ab", output : true
Pattern = "baaa?ab", output : true
Pattern = "ba*a?", output : true
Pattern = "a*ab", output : false
```

```
Input =         ba aab ab
Pattern = *****ba*****ab
Output : true
```

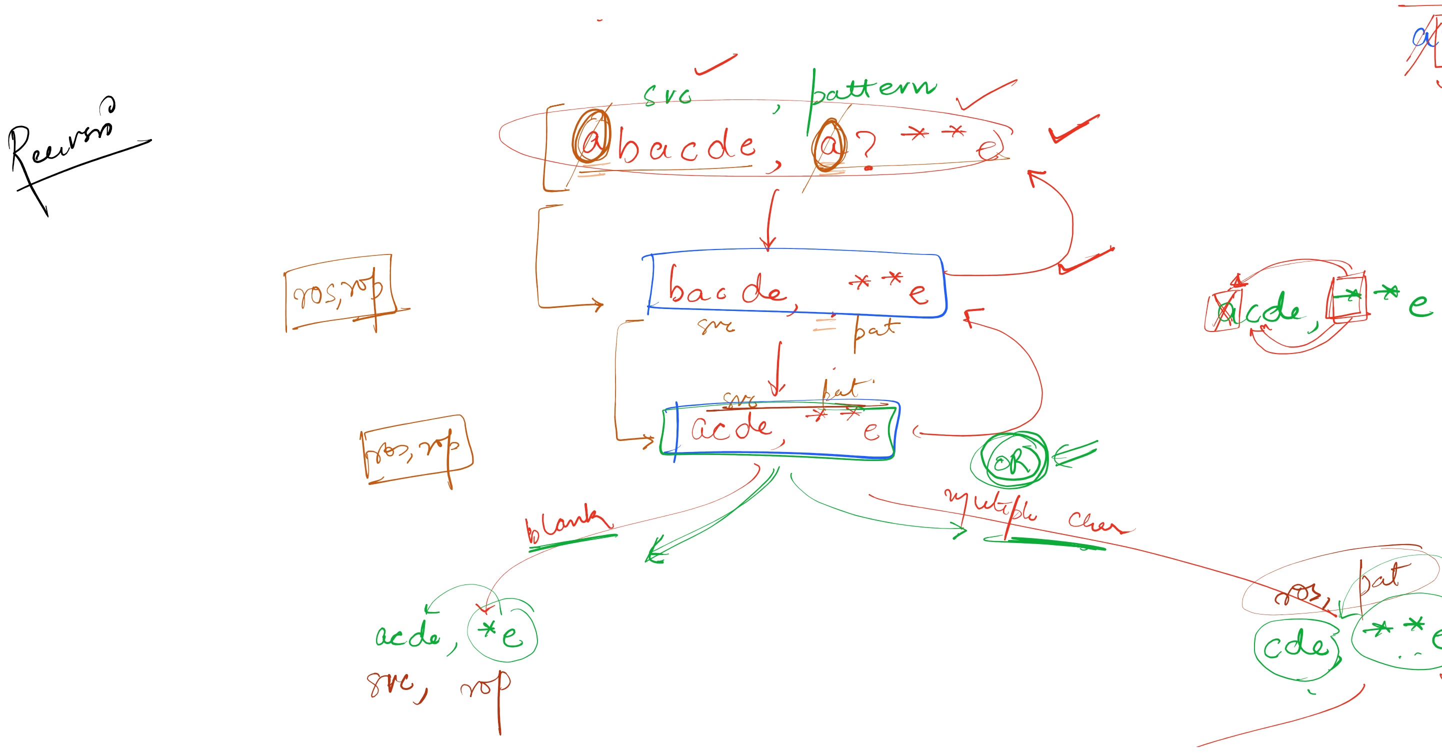
No matching text

```
Input =         baaabab
Pattern =       a *   ab
Output : false
```

```
Input =         ba aab ab
Pattern =       ba  *  a?
Output : true
```

? → 1 char
* → blank
→ 1 char
→ m.c.

Each occurrence of '?' character in wildcard pattern can be replaced with any other character and each occurrence of '*' with a sequence of characters such that the wildcard pattern becomes identical to the input string after replacement.



. char

n

s

bcde

pos

* : blank
~~sigle char~~
m·char

m.c.

src, rop

Char match, ? : sos, rop : diagonal

* : (src, rop) : blank : right

(sos, pat) : my.c. : bottom

: false

cde, *e

abc, d*

src pat

_, _ : true

abc, _ : false

{ _, ab*? : false }

_, *-*-*-* blank : true

abcdef

bcdef

Visually

0 1 2 3 4 5

2k abcdef

idx : 2 : cdef

idx = 4 : ef

idx = 0 : abcdef

idx = 5 : f

idx = 6 : _ :

abacde , a?**e }

2k abacde 0 1 2 3 4 5

sidx pidx

m.c.

de, →*→e

blank

de, *e

idx → end

idx == s.length

blank

? * *e

1 2 3 4

ok

abacde , a?**e

bacde , ?**e

acde , **e

blank

m.c.

acde, *e

cde, **e

2k

svidx pvidx

0, 0

1 1

2, 2

2 3

2DArray

overlapping

0, 0

abcd, ***d

svidx pvidx

0, 1

abcd, **d

1, 0

bcd, ***d

blank

m.c.

abcd, *d

svidx pvidx

1, 1

bcd, **d

blank

svidx pvidx

1, 1

bcd, **d

m.c.

cd,

pvidx

svidx

src

0

0

1

2

3

4

false

1, 2

1, 2

0

→ result compute

→ true → 2

→ false → 1

true

false

boolean | int

if (strg[svidx][pvidx] != -1)

== -1

false de

return

}

call

0 1 2 3

a b c d

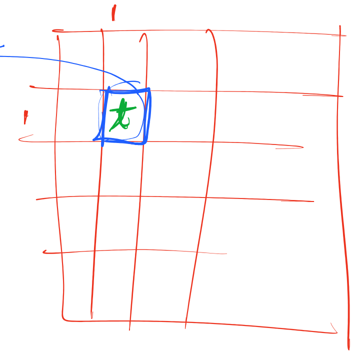
0 1 2 3 4 5

? * ? c * *

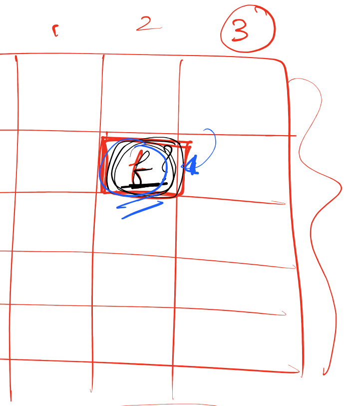
pat 0

Bottom up :

3,2.



src.l → p.l



boolean ⇒ t/f

Boolean = null

class

1 2 3 4 5 6

Bottom up :

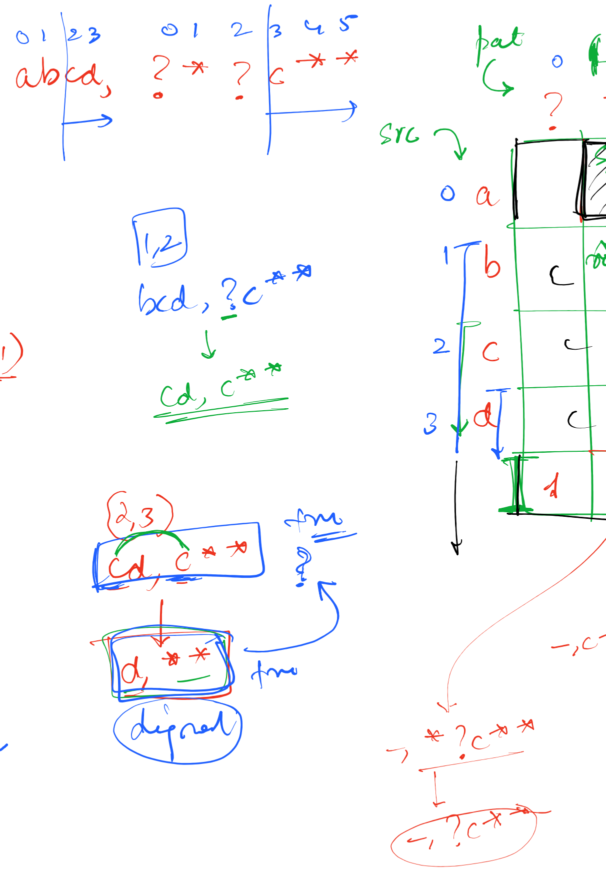
→ cell meaning

→ array size

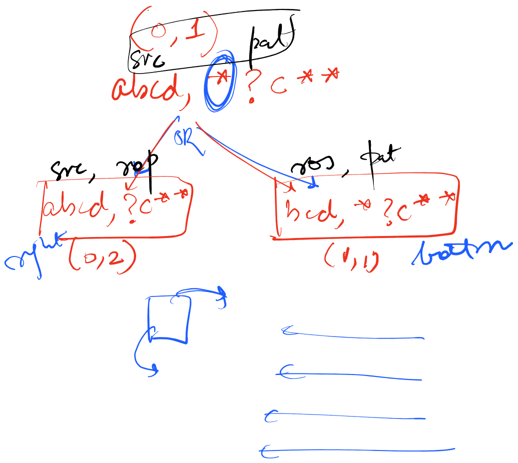
4 x 6

5 x 7

(s.l+1) x (p.l+1)



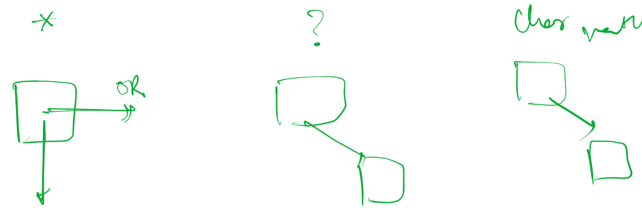
→ filling dirn

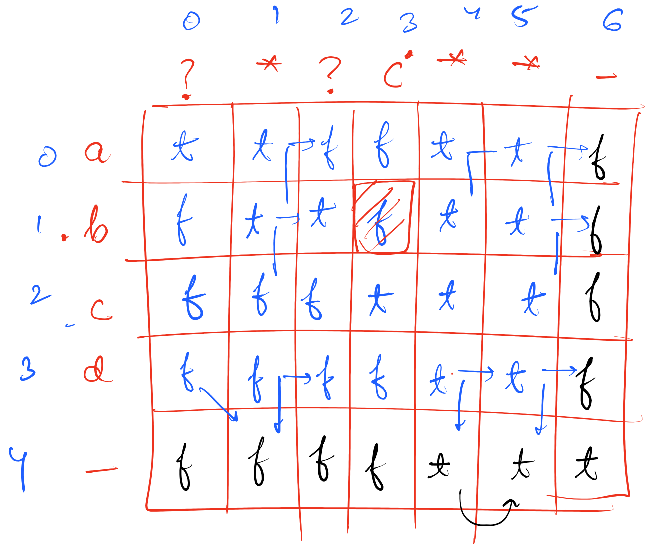


→ TD BC

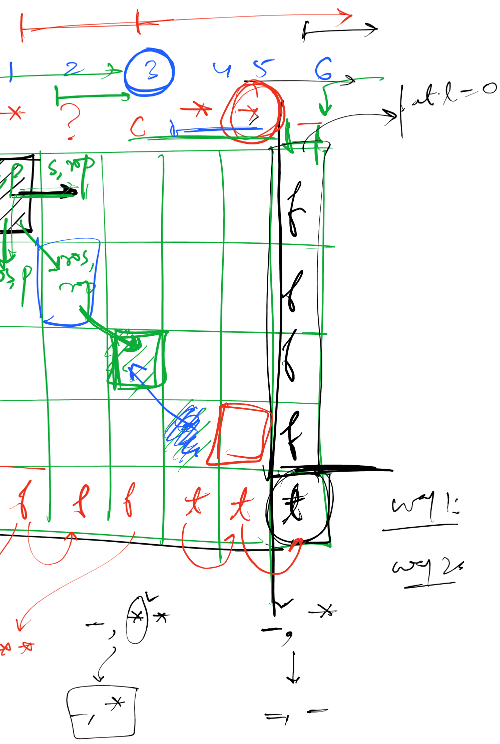
BU fill

→ filling logic





The first few Catalan numbers for n = 0, 1, 2, 3, … are **1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862**



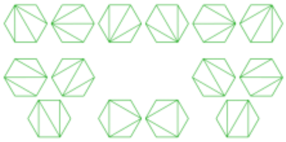
false

Applications :

1. Number of possible Binary Search Trees with n keys.

2. Number of expressions containing n pairs of parentheses which are correctly matched. For n = 3, possible expressions are ((())), ()(()), ()()(), (())(), (()()).

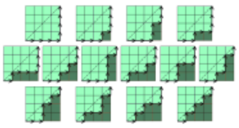
3. Number of ways a convex polygon of n+2 sides can split into triangles by connecting vertices.



4. Number of full binary trees (A rooted binary tree is full if every vertex has either two children or no children) with n+1 leaves.

5. Number of different Unlabelled Binary Trees can be there with n nodes.

6. The number of paths with 2n steps on a rectangular grid from bottom left, i.e., (n-1, 0) to top right (0, n-1) that do not cross above the main diagonal.

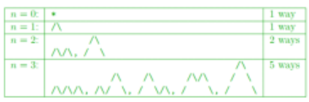


7. Number of ways to insert n pairs of parentheses in a word of n+1 letters, e.g., for n=2 there are 2 ways: ((ab)c) or (a(bc)).
For n=3 there are 5 ways, ((ab)(cd)), (((ab)c)d), ((a(bc))d), (a((bc)d)), (a(b(cd))).

8. Number of Dyck words of length 2n. A Dyck word is a string consisting of n X's and n Y's such that no initial segment of the string has more Y's than X's. For example, the following are the Dyck words of length
6: XXXYYY XYXXYY XYXYXY XXYYXY XXYXYY.

9. Number of ways to tile a stairstep shape of height n with n rectangles. The following figure illustrates the case n = 4:

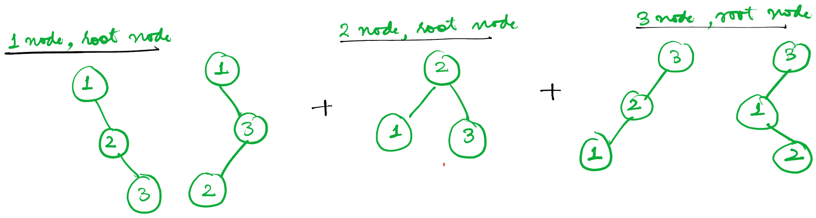
10. Number of ways to form a "mountain ranges" with n upstrokes and n down-strokes that all stay above the original line.The mountain range interpretation is that the mountains will never go below the horizon.



Mountain Ranges

n=3 ① ② ③ no. of bsts ?

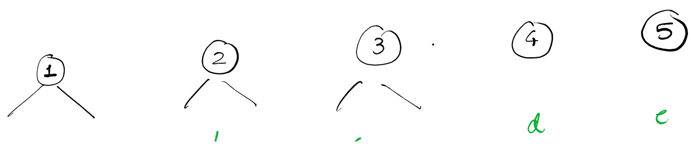
1 - - - n

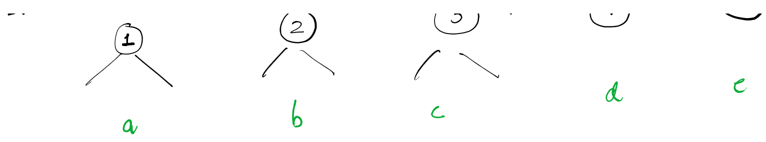


5 bst's

2 + 1 + 2 = $\boxed{\text{5 bst's}}$

n=5





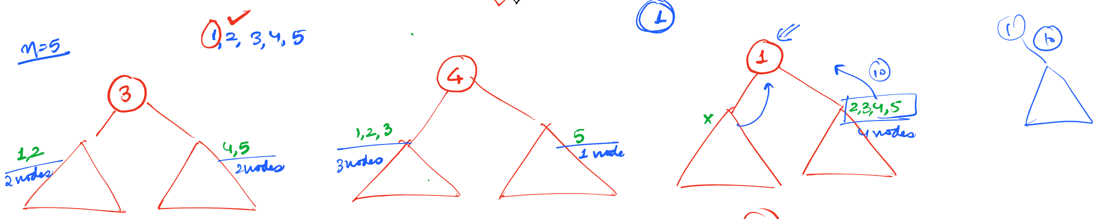
total bsts = no of bsts obtained by having [1 node] as root node → a
+ [2nd node] → b
+ [3rd node] → c
+ [4th node] → d
+ [5th node] → e

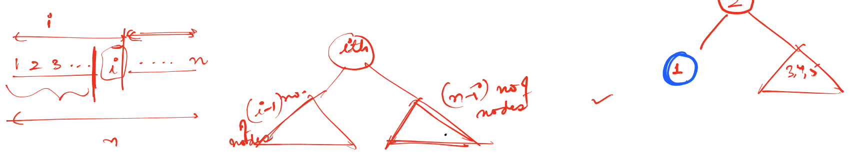
= $\boxed{a+b+c+d+e}$

total bsts = $\sum_{i=1}^{n}$ no of bsts obtained by having ith node as root node

n=5

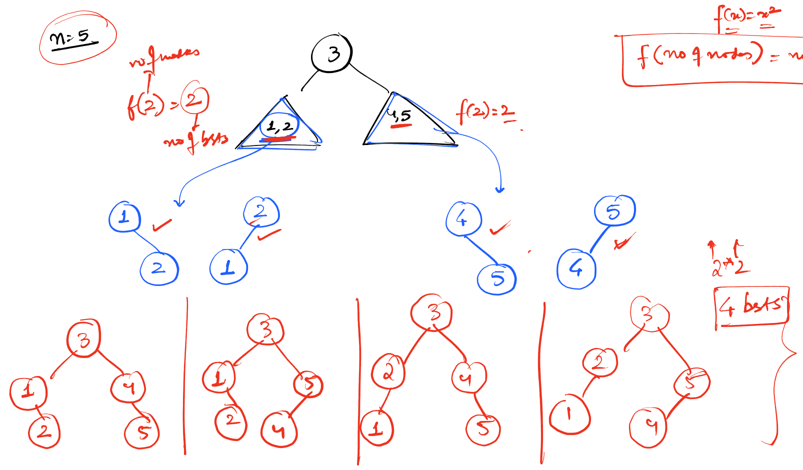
1, 2, 3, 4, 5

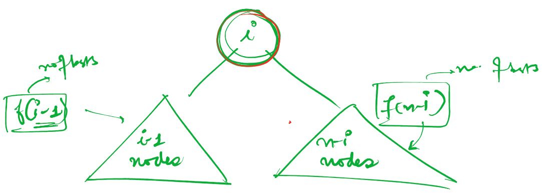




Eg: n=5

f(no of nodes) = no of bsts





no of bsts obtained by having i node as root node: $f(i-1) * f(n-i)$

no of bsts with n node : $\sum_{i=1}^{n} f(i-1) * f(n-i) = \frac{2n\,C\,n}{n+1}$

Catalan Series

n=3

$\frac{^6C_3}{4} = \frac{6!}{3!\,3! \times 4} = \frac{4\times5\times6}{2\times3\times4} = 5$

f(i) {

Recall: no of nodes (no of bsts)

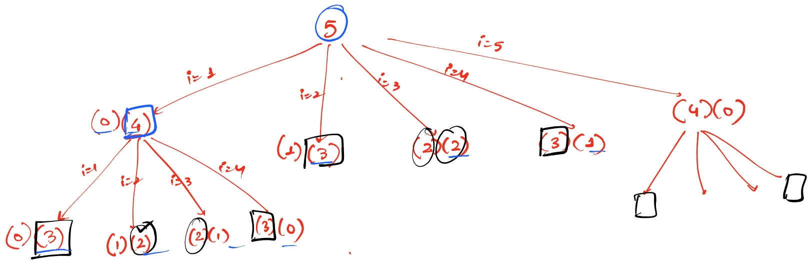
f(i) {

ret (i-1) nodes bsts

}

Rec calls: no of nodes

Leaf: no of bsts



Bottom up

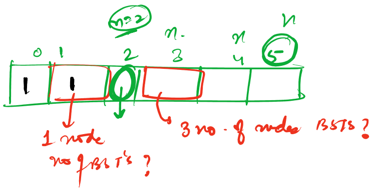
→ cell meaning

→ array size? 1D → n+1

→ filling dir

→ TD BC, BU fill

→ fill logic?



Catalan Series:

0 1 2 3 4 5 6 7 8 9

1,1,2,5,14,42,132 429,1430 4862,16796,58786 208012 742900,2674440

→ DP Assignments

→ Deadline: Sunday: Contest freeze: 5 pm

→ Contest: 20's: Recursion + DP: 12 noon

→ Monday: oops, Stack, Queue 12:30 pm

→ Tmx