DYNAMIC PROGRAMMING Level-1

DP Class-1 (LIVE)

11 sep 2024

→ not data structure.

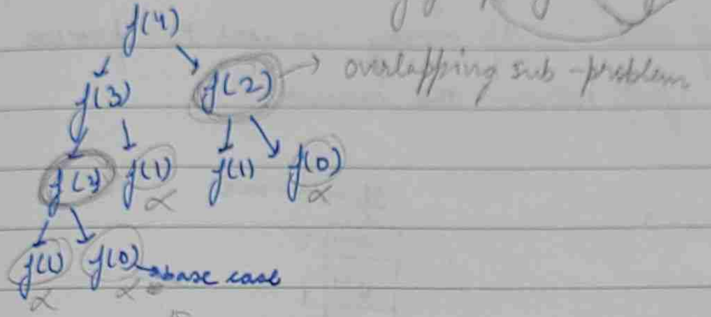
→ it is a technique ~~as~~ for problem-solving.

→ When to apply : ⓐ Overlapping sub-problems

ⓑ Optimal sub-~~sub~~ structure. (badi problem choti problems se solve krjayge)

↳ eg: $f(n) = f(n-1) + f(n-2)$

eg: rec tree

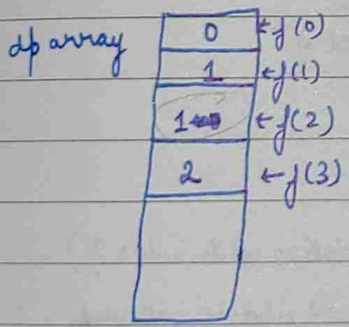


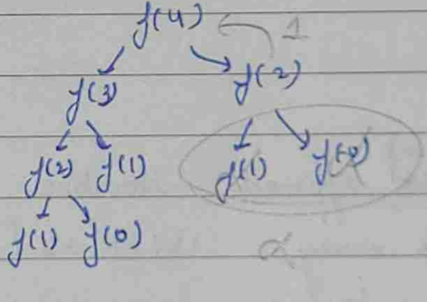
leetcode 509.

Q. Fibonacci number

f(2) isme do bar solve ho raha hai

→ DP : jo bhi ek baar solve Karliya usko dubara Karne ki need nhi hai
↳ kyuki mai answer store Karlunga.





eg: chanakya
→ galti → fir kabhi na karna → raja
↳ repeat galti → lallu

DP questions approach

↳ Top-down → rec + memoization (→ ans store karlena)

↳ Bottom-up (iterative) → loop ~~+ store ans~~ ~~data structure~~

↳ space-optimisation (if possible)
↳ pattern find

① Top-down approach : recursion + memoization

→ check ki kitne parameters function ke change ho rhe hai rec call me

① findout → 1D/2D/3D dp

based on that create dp array

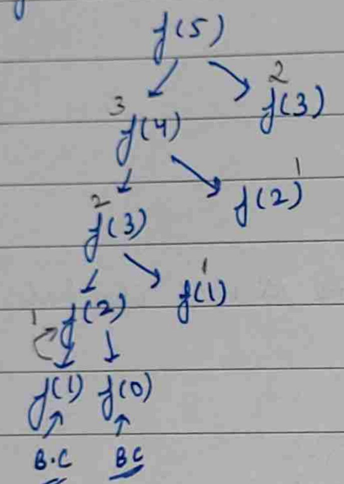
↳ pass in function.

② ans in recursive code → store in dp array and return in dp array.

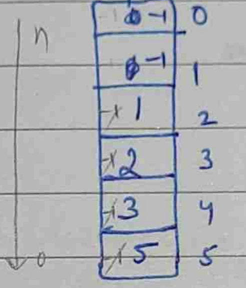
base case ke baad

③ ↳ if ans already exists in dp array : no need to process again, just return the answer.

eg: n=5



dp → n+1 size



normal rec : exponential calls.

dp : O(n)

SC : O(n)

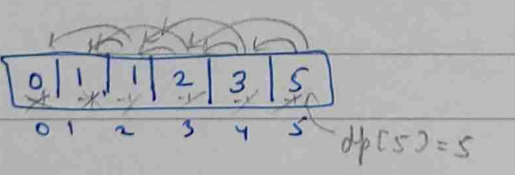
② Bottom-~~down~~ up approach : iterative

① create dp array (1D/2D/3D / size? / initialize with value?)

② analyse base cases in recursive approach → update array

③ check parameter, reverse it, copy-paste logic.

↳ main loop



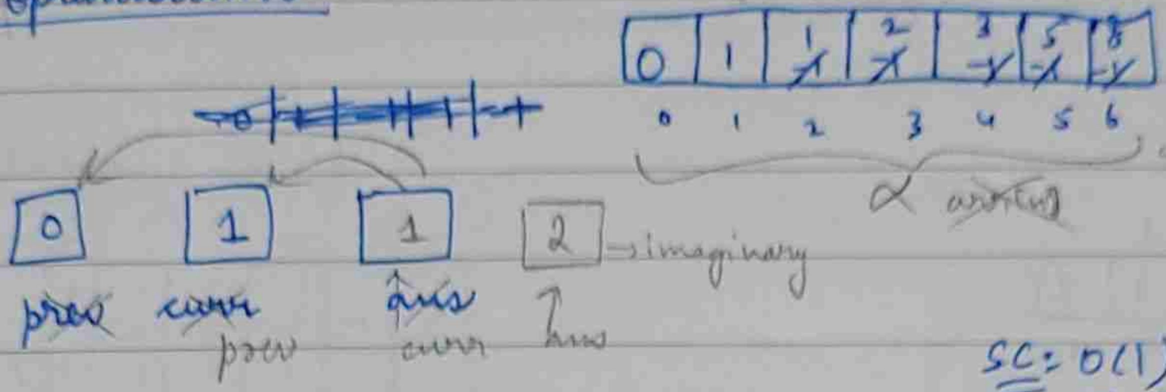
dp[5] = 5

∘ dp array me initial state shuru me hi daal dete hai & baki sari states uske basis pe update karte jayenge

SC : O(n)

TC : O(n)

③ Space optimisation



ans not dependent on whole array, only dependent on last 2 index.

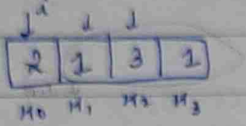
SC = O(1)
TC : O(n)

DP Class-2 (LIVE)

16sep2024

leetcode198.

Q. House Robber



i $\xrightarrow{\text{adjacent}}$ i+1 X

i+2 ✓

include → 2 + remaining : recursion → i+2

exclude → 0 + " " → i+1

} max → final Ans ✓

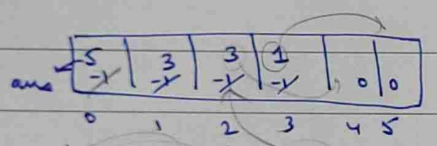
TC : O(n)

SC : O(n)

- in tabulation, while adding /subtracting index in dp array, make sure it is valid.

tabulation: n=4   2|1|3|1

n+2 → dp

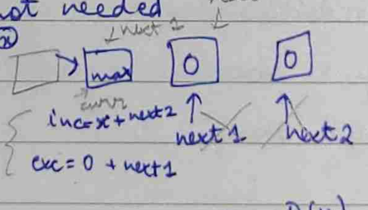


inc = 3+2 → 5
exc = 0+3 → 3

inc = 3 + 0
exc = 0+1

inc = 1+1 → 2
exc = 0+3 = 3

Space-optimisation : array not needed



max { inc = x + next2
exc = 0 + next1

TC : O(n)

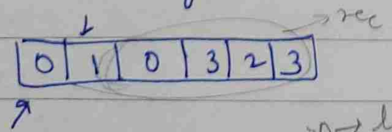
SC : O(1)

next 2 = next

next 1 = curr

leetcode300.

Q. Longest Increasing subsequence



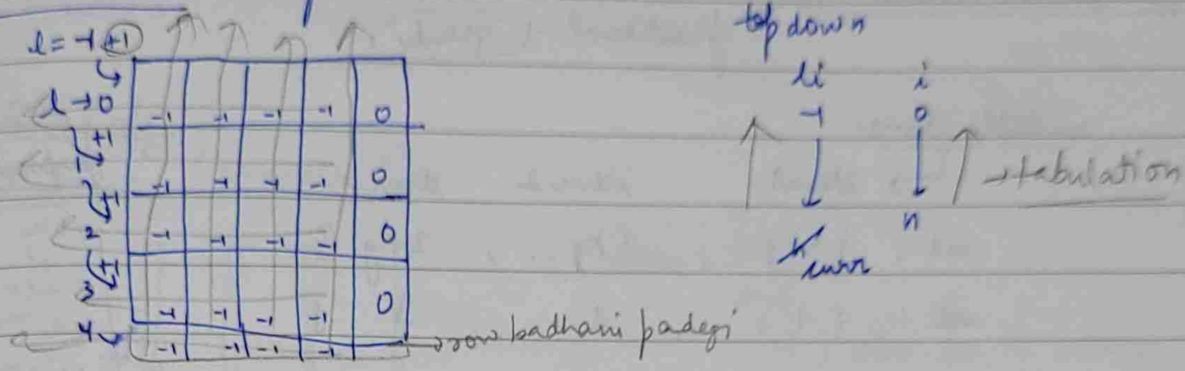
inc → if curr > last element

include → {0} + rec → 0 → last

exclude → α + rec → INT_MIN

- when confused whether to take values or index, prefer <u>index</u>.

tabulation : 2D dp

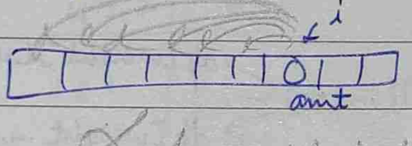


-1	-1	-1	-1	0
-1	-1	-1	-1	0
-1	-1	-1	-1	0
-1	-1	-1	-1	0
-1	-1	-1	-1	

leetcode 322.

Q. <u>coin change</u>

dp [amt] → dp [amt - coin]



space optimisation X

DP class-3 (LIVE)

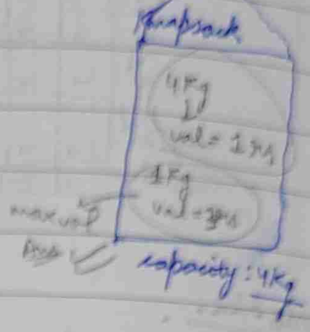
18sep 2024

(gfg) Q. Knapsack Problem → 0-1 → dp ✓
→ fractional → greedy

multiple items
↳ item1 item 2 item 3

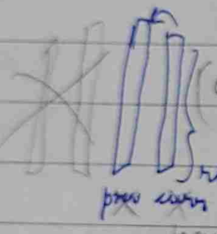
wt : {4Kg , 5Kg , 1Kg }
val : { 1 , 2 , 3 }

→include / exclude



TC: O(n×W)
SC : O(N×W)

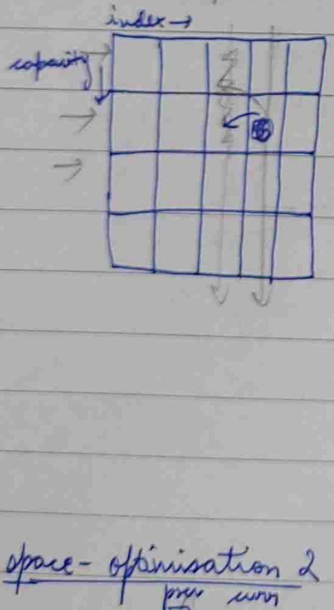
space-optimisation
↳ dependency
↳ dp [weight][index] → dp [weight][index-1]
→ dp [weight - wt[index]][index-1]



(only dependent last wale pe)
2 hi array needed hai bas
rows = capacity+1

SC : O(W)

⇒ 2d array not needed ✓



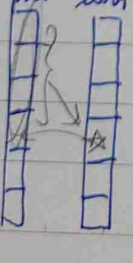
dp[wt][in]
↳ dp[wt][in-1]
↳ dp[wt -][in-1]
<=wt

```
for(weight = 0 → capacity) {
    for( 1 → n-1) {
    —

}
```

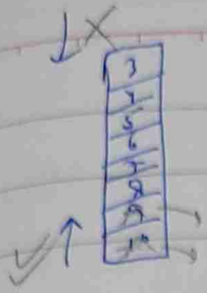
} row-wise loop chl raha hai
⇒ col-wise kro

space-optimisation 2



dependent
curr[weight]
↳ prev[weight]
→ prev[weight - wt[in]]

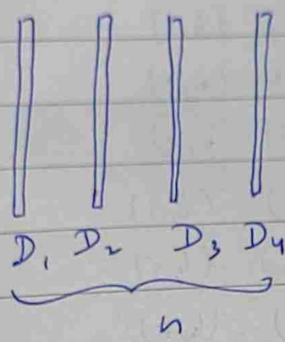
2 array use hi mt kro → usi me over-write kardo



TC : O(n)

leetcode 276 (gfg)

Q. Paint Fence (premium ☹)



colors : 3 ← k
RGB
↳ 1 color can't come three or more times

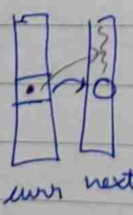
K=3 → RGB

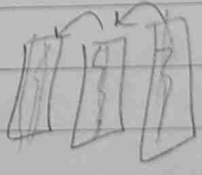
	n=1	n=2	n=3	n=4
same	R G B	RR GG BB	→ RR R → GG → BB G → RR → BB B → RR → GG X X X ↳ RR GG BB F(n-2) X(K-1)	X X X X F(n-2) X(K-1) RR GG BB
diff	ans = 3	RG RB GR GB BR BG	RR → (K-1) GG → (K-1) BB → (K-1) RG RB GR GB BR BG F(n-1) X(K-1) X X X ↳ R G B	X X X X ↳ R, G, B F(n-1) X(K-1) F(n-2) X(K-1) + F(n-1) X(K-1) ⇒ (K-1) X [F(n-1) + F(n-2)]

leetcode 416

Q. Partition Equal subset sum

SO $dp[t][i] \rightarrow dp[t][i+1]$
$\quad\quad\quad\quad \rightarrow dp[t-nums[i]][i+1]$





SO 2. $curr[t] \rightarrow next[t]$
$\quad\quad\quad \rightarrow next[t-nums[i]]$

single array

leetcode 1155.

Q. Number of dice rolls with target ~~sum~~ sum.

→ n : no. of dice
→ K → faces
→ target sum

} total possible ways? to generate

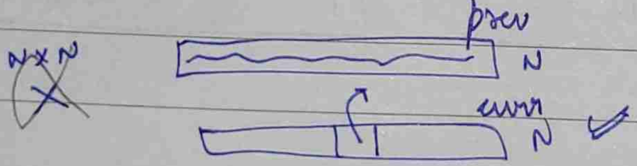
eg : n=2, K=6 , t=9 4 ways ✓

{1,1}	{2,1}	{3,1}	{4,1}	{5,1}	{6,1}
{1,2}	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)
{1,3}	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)
{1,4}	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)
{1,5}	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)
{1,6}	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)

eg : 10 dices

{1 _ _ _ → rec }
{2 _ _ }
{3 _ _ }
{4 _ _ }
{5 _ _ }
{6 _ _ }

SO : dp[N][t] ⟶ dp[N-1][t-i]



DP class-4 (LIVE) 19sep2024

leetcode 1143.

Q. longest common subsequence

eg : abcde
$\overset{\uparrow}{i}$

ace
$\overset{\uparrow}{j}$

text 1 [i] == text 2[j] } match
↓
1 + rec

text 1 [i] != text2[j] } no match
↓
0 + rec

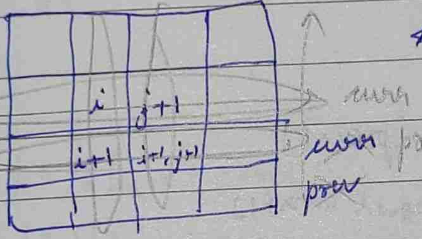
TC: O(N×M)

SO.

$$\begin{tikzcd} & dp[i+1][j+1] \\ dp[i][j] \arrow[ur] \arrow[r] \arrow[dr] & dp[i+1][i] \\ & dp[i][j+1] \end{tikzcd}$$

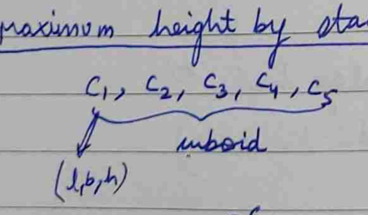
TC: O(N×M)

* loop : SC : O(m)



SO 2

$$\begin{tikzcd} & prev[j] \\ curr[j] \arrow[ur] \arrow[r] \arrow[dr] & curr[j+1] \\ & prev[j+1] \end{tikzcd}$$

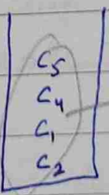


~~not possible~~

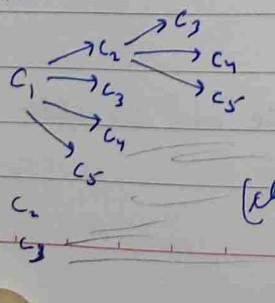
leetcode 1691.

Q.* maximum height by stacking cuboid.

$\underbrace{c_1, c_2, c_3, c_4, c_5}_{\text{cuboid}}$
↓
(l,b,h)



→ as per rules
return max height.



(check all cases using rec)
as per rules

same pattern (LIS)
prev → curr → next
curr > prev next > curr

{ - - - }

sort
↳ nlogn → ↳ O(1) ← (3 element ka array hai)

eg : c_1 → {10, 5, 20} $\xrightarrow{sort}$ {5, 10, 20}

c_2 → {15, 3, 7} → {3, 7, 15}

c_3 → {20, 50, 15} → {15, 20, 50}

l ↓ b ↓ h ↓

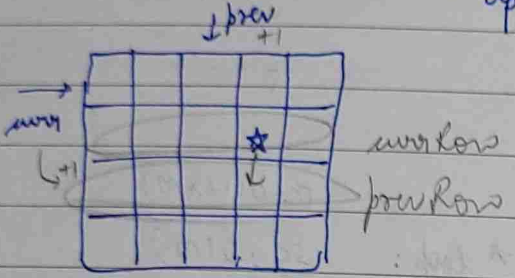
max → height

} sort → {3, 7, 15}
{5, 10, 20}
{15, 20, 50}

↳ Sabse bada answer ho skta hai

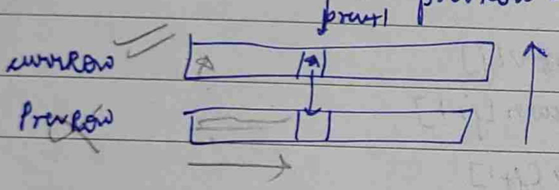
SO.

dp [curr] [prev+1] → dp [curr+1] [prev+1]
↳ dp [curr+1] [curr+1]

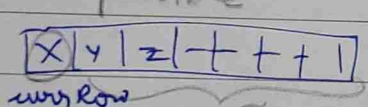


SO2

currRow [prev+1] → prevRow [prev+1]
↳ prevRow [curr+1]



SO3 → not possible



leetcode 354.

Q. Russian doll envelopes

↳ (same to last ques bas ye alag hai ki individual array sort nhi kar skte)

↓

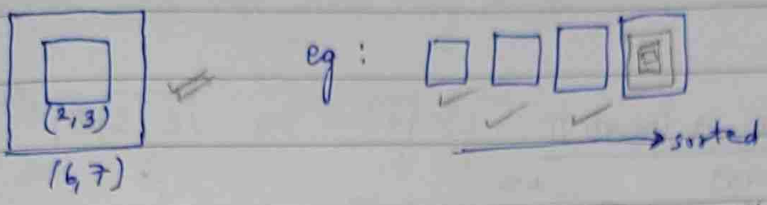
TLE (even after SO2)

↳ (do from assignment)

↳ binary search

↳ isse jyada optimise nhi ho skta

envelopes ⇒ [[5,4], [6,4], [6,7], [2,3]]



~~inc~~ inc/exc. sorting acc. to width : 2,3 | 5,4 | 6,4 | ~~6,7~~

pick ⇒ 2,3 → (5,4) → (6,4)

not pick (5,4) → (6,4)

sort : 2,3 | 5,4 | 6,7 | 6,4 (width inc ; $w_i = w_j$ ⇒ height dec.)

↳ left ke saare ajaye isliye

2 3
5 4
6 7] jaha same hai width waha dec height-wise.
6 4

width → LIS

length = 3

3, 4, 7, 4

eg : 4,5 | 4,6 | 6,7 | 2,3 | 1,1

sort ⇒ 1,1 | 2,3 | 4,5 | 4,6 | 6,7

↓'h'

1 | 3 | 5 | 6 | 7

↳ length = 5 ✗

1,1 | 2,3 | 4,6 | 4,5 | 6,7

↓'h'

1 | 3 | 6 | 5 | 7 ⇒ 1 3 6 7

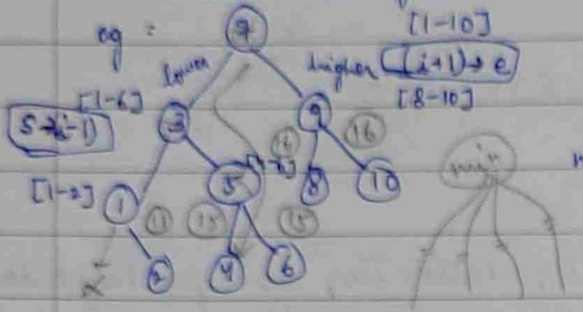
length = 4 ✓

DP Class-5 (LIVE) 23sep2024

[Merge Intervals]

leetcode 375.

Q: Guess Number Higher or Lower II

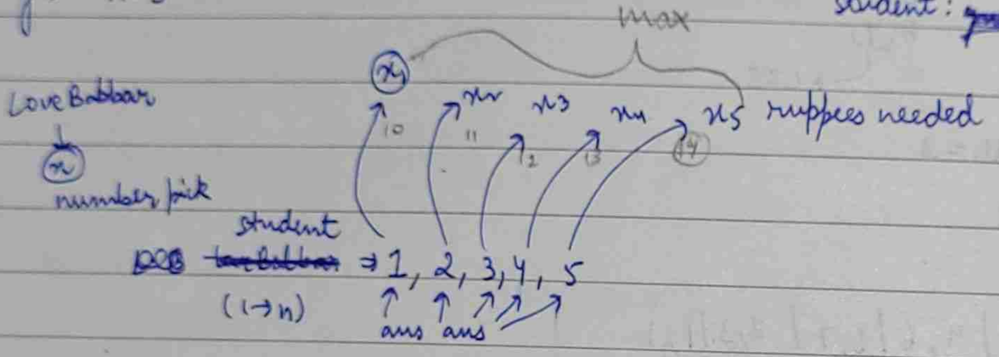


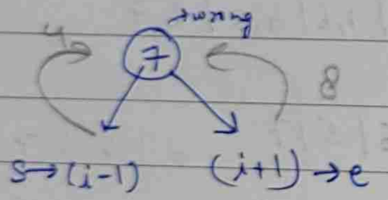
+7
+3 ⇒ 15
+5

money pay : 11 or 15 or 15 or 16 or 16 ⇒ guarantee win

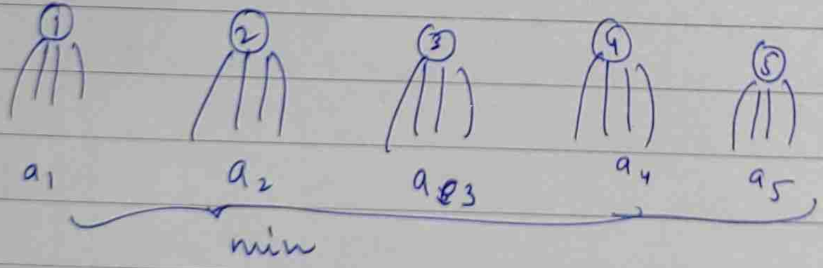
eg: n=5

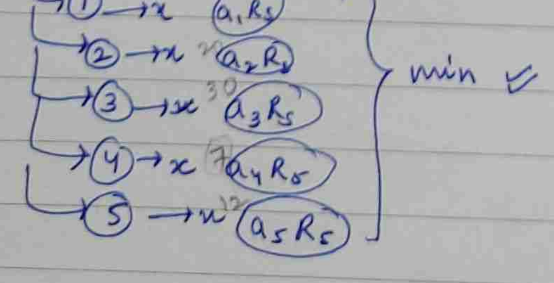
student : ~~guess~~ guess

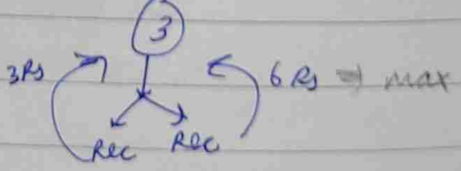




wrong : $x + \max(Rec(s, i-1), Rec(i+1, e))$
↑
penalty

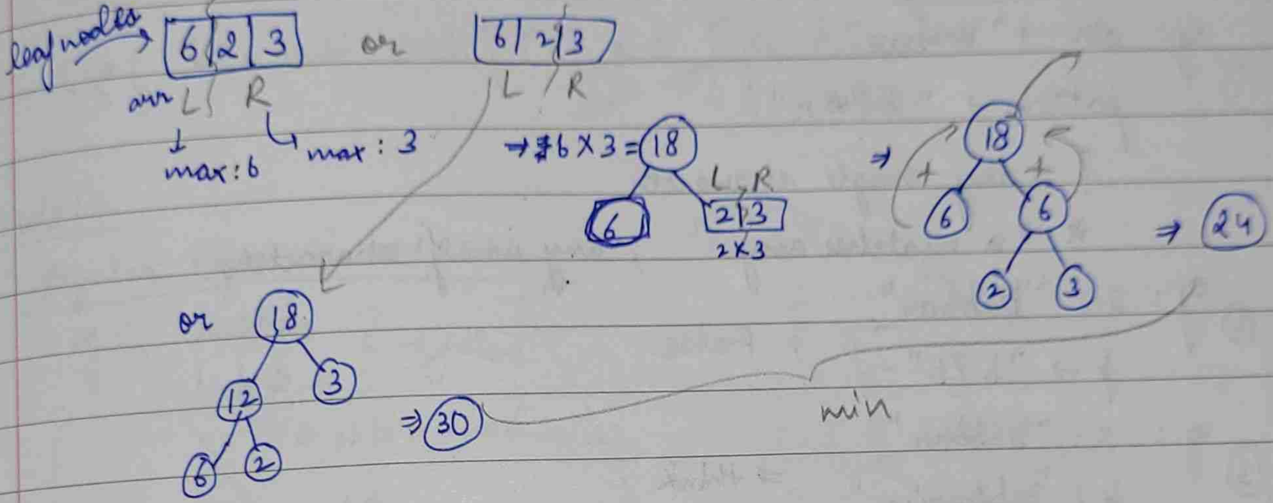






* leetcode 1130 :

Q. minimum cost tree from leaf values



i j

eg : | 5 | 10 | 12 | 4 | 6 |
0 1 2 3 4

→ Tree1 → N.L → sum1
→ T2 → N.L → S2
→ T3 → N.L → S2
→ T4 → N.L → S3

} min → final Ans

<0,1> → max
<0,2>
<0,3>

pair<int, int>

{0,0} → 5
{0,1} → 10
{0,2} → 12
{0,3} → 12
{0,4} → 12
{1,1} → 10

{1,2} → 12
{1,3} → 12
⋮

DP class-6 (LIVE)

24 sep 2024

leetcode 44. → OA

Q. Wildcard matching

① eg: str → "babbar" i
pattern → "b?*r" j } True (i & j dono bhar nikl gaye)

? → any single character
* → matches any " , any no. of characters.

② eg: s → "babbar"
p → "b?b" j } False

③ eg: s: "babbar" i
p: "babbaring" ⇒ think

s: "babbar" i
p: "babbar*" j ⇒ check star
↳ agar sare bache huye star hai toh true
nhi toh false

④ logic ↳ match
↳ no match

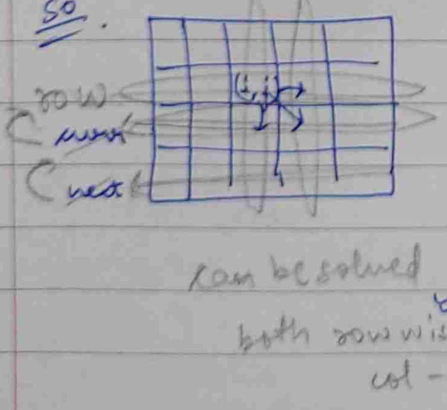
eg: babbar (i)
ba*ar (j)

} * Ko ba maan lia → i+1 → babbar / bab*ar
* Ko "" maan lia → j+1 → False

babbar / babb*ar ← *→b* ; *→"" → False

False ← *←K
"" → True
babbar / babbar

so.



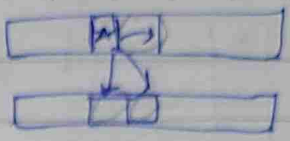
dp[i][j]
↳ i+1, j
↳ i, j+1
↳ i+1, j+1

can be solved both rowwise & col-wise

shifting ⇒ next = curr

SO-2 curr[j] → curr[j+1]
→ next[j]
→ next[j+1]

∝
↳ not possible



} 1array → var
∝

SO ⇒ TC: O(m×n)
SC: O(n)

leetcode 10.

Q. Regular Expression Matching

eg: babbar
bab*r } → false
↳ b, bb, bbb, bbbb - - -

① babbar ↓i
bab*ar ↓j ⇒ true

② j == p.len() ⇒ false

③ i == s.len() ⇒ check star
↳ true
1.* ⇒ false

babbar $\xrightarrow{*\text{""}}$ False
bab*ar → babbar $\xrightarrow{*\text{""}}$ return true
→b babb*ar *→b* → False

* → ""
* → preceding + *

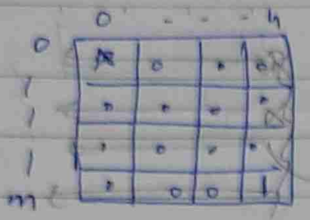
eg: babbar (i)
bab*ar (j) } → babbar (i)
ba--ar (j+2)

① j b* → ""
↳ b*
↳ bb*
↳ bbb* - - - -

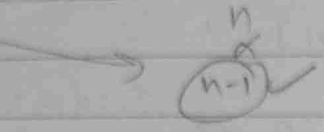
⇒ empty ⇒ f(i, j+2);

② babbar (i)
bab*ar (j) } babbar (i, i+1)
babb*ar (j) currMatch && f(i+1, j);

currMatch ✓

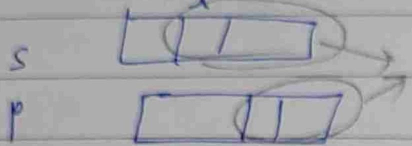


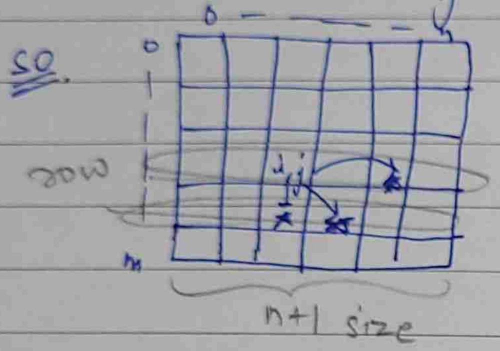
dp[m][n] or dp[i][j]

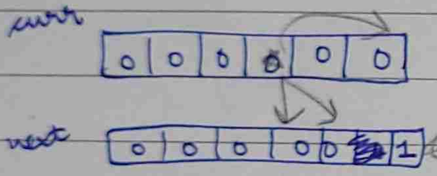


eg: s = " "
p = a* } True

m-1 → cannot assume







last row i==m

↳ dono strings fully process hochuki hai.

DOMS Page No.
Date / /

DP Mega Class (LIVE)

2oct2024

① *→ imp. LIS → dp → $O(n^2)$ ✓ (done in live class)
→ $O(n\log n)$*

② max. height of stacking cuboids → understanding build (done in live class -4)
③ edit distance
④ last stone weight II

leetcode 300.

① Longest increasing subsequence

m① rec → inc /ex
↳ td ✓
↳ BU ✓
↳ SO ✓

M② TC: (nlogn) : DP + binary search

eg: 10 | 9 | 2 | 5 | 3 | 7 | 101 | 18
0 1 2 3 4 5 6 7
↑ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ i

0 → ~~inc~~ {10, 101,
1 → ~~exc~~ {9, 101,
2 → {2, 5, 7, 101,]
3 → {3, 7, 101,
4 → {18,

merge upar se neeche
→

maxlength = ~~1~~ ~~1~~ ~~2~~ ~~3~~ 4

ans ~~9~~ 2 | 3 | | 18 |
| ~~10~~ | ~~5~~ | 7 | ~~101~~ | |

10 | 9 | 2 | 5 | 3 | 7 | 101 | 18
~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ i

((R→L) check for esa element jo just >=3 ho)
↳ lowerbound (L→R)

eg : 10 | 9 | 2 | 5 | 3 | 101 | 108 | 18 | 20 | 23

ans | ~~10~~ | ~~5~~ | ~~101~~ | ~~108~~ | 23 |
~~9~~ 2, 3, 18, 20

→ ans.length() → (no need to store even maxL)
maxL = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5
anything less than ans.length() will not update maxL
(don't bother

leetcode 1691.

② Max height by stacking cuboid

eg : 50, 45, 20,
95, 37, 53
45, 23, 12

$\xrightarrow{\text{sort}}$

20, 45, 50 → fixed height
37, 53, 95
12, 23, 45

↑ ↑
l l
Kisi ko kuch bhi maanlo

0th index se sort → by default 0 index sorting ⇒ done in class - 4

0 based auto-sorting — 1st index se sort

$v_1 \Rightarrow [a, b, c]$ → height
$v_2 \Rightarrow [d, e, f]$ → height
⇒ good sort

① a & ~~b~~d → whichever → less ⇒ put first
a == ~~b~~d ⇒ automatically sort based on 1st index
↳ b & e ke hisab se
b == e → c, f ke hisab se

1 based sorting → problematic

$V_1 \Rightarrow \{a, b, c\}$ (0 ① 2)
$V_2 \Rightarrow \{d, e, f\}$
⇒ a, d ko prefer hi wahi kara
⇒ not good sort

① b, e ⇒ b ≠ e
↳ b and e ke hisab se
↳ whichever will be less, return true
b == e
↳ c, f ke hisab se.
c == f
↳ random

eg : $V_1 \Rightarrow [13, 50, 50]$
$V_2 \Rightarrow [18, 50, 50]$

0 based : $V_1 \rightarrow V_2$

1 based : possible that $V_2 \rightarrow V_1$ (V_1 comes before V_2)
(par V_2 toh bada hai)

⇒ but if you want to do 1 based sorting ⇒ write comparator such that it considers all ~~tha~~ three elements.
→ phle check a[0] then check a[2] kyuki V_1, V_2 — toh phle se hi sorted hai

leetcode 1049

3) last stone weight II

eg : 2|7|4|1|8|1
0 1 2 3 4 5

-2+4

2|7|1|8|1

-7+8

2|1|1|1

(-2+4-1)

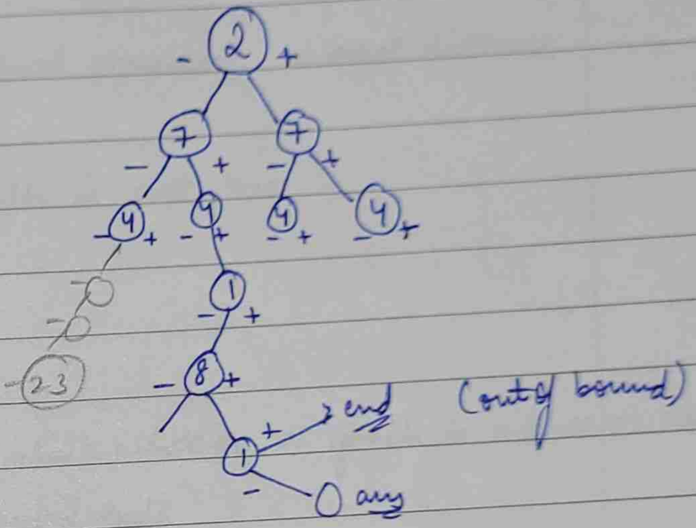
1|1|1 (-7+8-1) → ① ans

φ

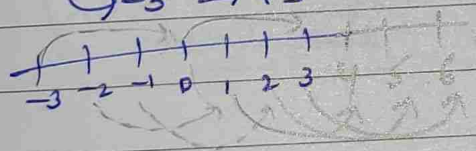
$x \le y$
↳ $x == y$ ↳ NULL
↳ $x < y$
↳ $y-x$

M1 RE Tree
↓
leaf node

all these leaf nodes in rec tree, having sum <0 are ignored.



eg : [1,2]
↳ -3 → +3



columns ⇒ 0, 1, 2, 3, 4, 5, 6 ⇒ 7 indexes ✓

3×2+1 ✓

(4) ~~Edit distance~~

M2 2|7|4|1|8|1

partition: G1 group2

$|\Sigma G1 - \Sigma G2| \ge 0$

or you can say $|\Sigma G1 - \Sigma G2|$ should be minimised

recursively

G1 : {2, 7, 1, 1} ⇒ 11 sum

G2 : {4, 8} ⇒ 12

↳ automatically ban jayega

$|11-12| \ge 0$ ✓

leetcode 72

(4) edit distance

↳ diy