PEPCODING

PURSUIT OF EXCELLENCE AND PEACE

MODULE

DYNAMIC PROGRAMMING

+91 11 4019 4461

PepCoding, 3rd Floor, 15 Vaishali,
Pitampura Opposite Metro Pillar 347,
Above Karur Vysya Bank, New Delhi,
Delhi 110034

www.pepcoding.com

/pepcoding

Catalan Based Questions

Q. Catalan Number

* Catalan numbers are sequence of natural numbers.

First few Catalan numbers for $n = 0, 1, 2, 3, 4$ are $1, 1, 2, 5, 14, \ldots$

What?

Find nth Catalan Number.

How? For $n = 4$.

dp →

1	1	2	5	14
0	1	2	3	4

Explanation:

dp[i] is filled for any i as:

For $i = 4$, $dp[4] = (1*5) + (1*2) + (2*1) + (5*1)$

$= 14.$

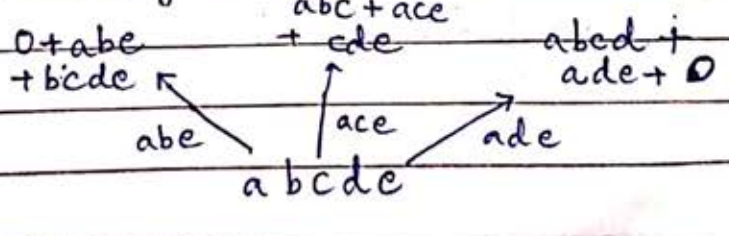
or $C_0C_3 + C_1C_2 + C_2C_1 + C_3C_0$

Q. Minimum Score Triangulation:

What? Given N, which is a N-sided polygon with vertices labelled as given in the array, return the smallest possible total score that you can achieve with some triangulation of the polygon.

How?

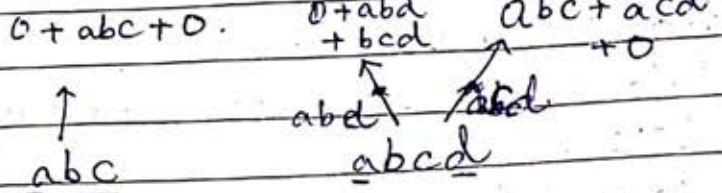
For any given set of vertices, say abcde, a possible triangulation score could come as:



(Keeping ae as two end points of any polygon to be formed.)

Page No.
Date

Relating to catalan, and matrix chain multiplication:



ex: N = 6, arr = {1, 3, 2, 5, 4, 6}

a[0]*a[1]*a[2] = 1×3×2 = 6

	0	1	2	3	4	5
0	0	0	6	16	36	60
1	0	0	0	30	64	124
2	0	0	0	0	40	88
3	0	0	0	0	0	120
4	0	0	0	0	0	0
5	0	0	0	0	0	0

→ answer

Min [0+6+30, 6+10+0 = 36, 16] = 16

because a Δ has atleast 3 sides.

At gap 2, simply put `a[i]*a[i+1]*a[j]`,

else

dp[i][j] = Math.min(dp[i][j], a[i]*a[k]*a[j] + dp[k][j] + dp[i][k])

(k = i+1 to j-1)

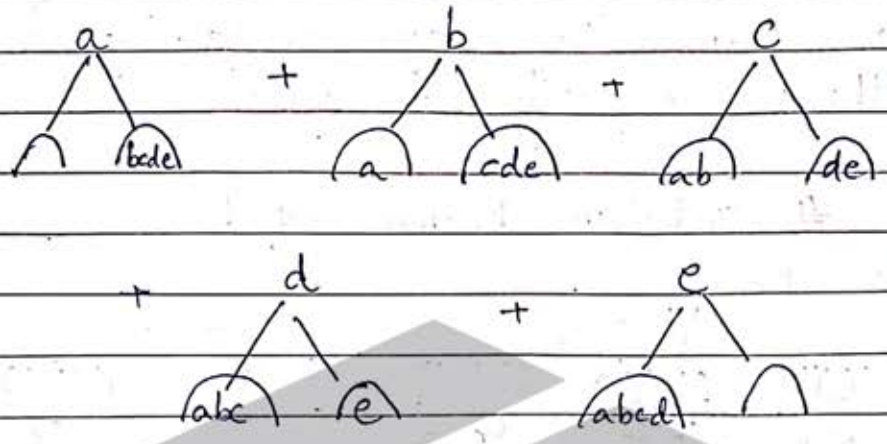
Q. Number of BSTs.

What?

Given n elements, tell how many BSTs can be formed?

s.t. a < b < c < d < e.

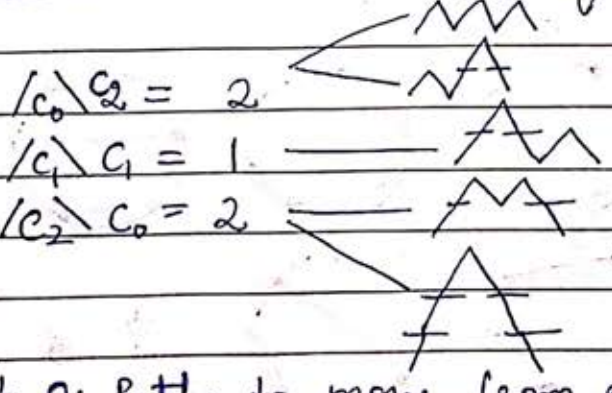
How? # Relate to Catalan Number.



0 elements = $C_0 = 1$.
1 element = $C_1 = 1$
Illy, $C_2 = 2$,
$C_3 = C_0C_2 + C_1C_1 + C_2C_0$
$= 1 \cdot 2 + 1 \cdot 1 + 2 \cdot 1$
$= 5$.

Q. Valleys and Mountains.
What?
You are allowed 2 types of slashes, '/' & '\'. How many maximum shapes can you create, given none of the '/' or '\' can go below ground level (N will be given).

How? Ans is C_3 for '/' & '\'.
→ N = 3.



Catalan Numbers of N.

Q: Paths to move from source to destination, given you can move only above diagonal.

Page No. _____
Date | | | |

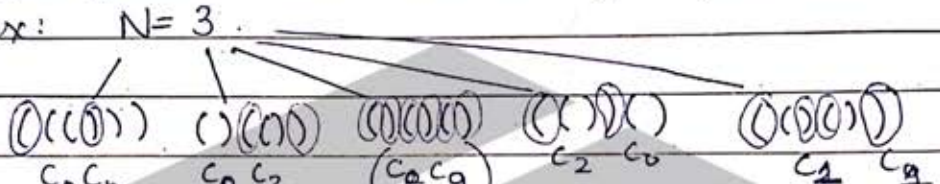
Q. Brackets

What?

Return number of ways containing n pairs of parenthesis correctly matched.

How? # Catalan Number of N

ex: N = 3



$(C_0)(C_2)$

$= ()(()())$

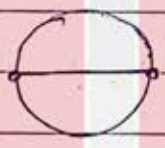
Q. Chords

What?

Given 2n points, tell how many non-overlapping (or non-intersecting points) chords can be made?

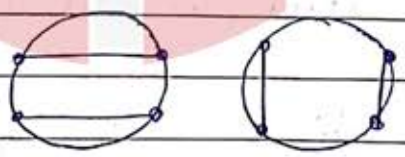
How? # Catalan Numbers

n = 1, 2n = 2



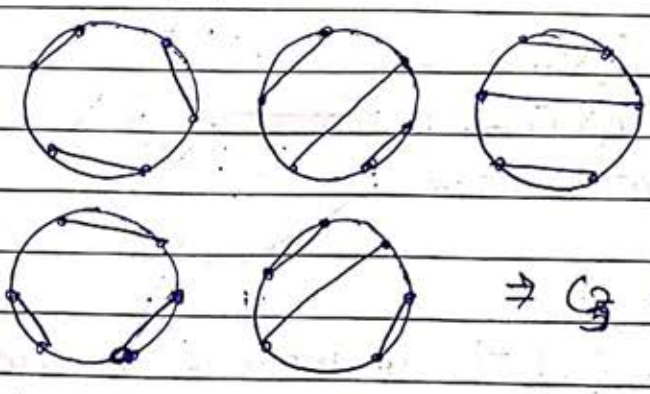
$\Rightarrow C_1 = 1$

n = 2, 2n = 4



$\Rightarrow C_2 = 2$

n = 3, 2n = 6



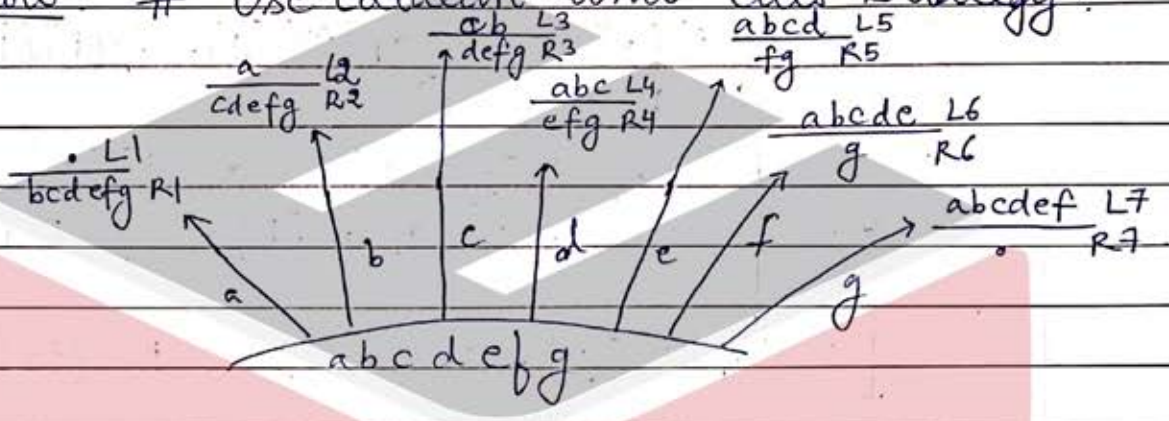
$\Rightarrow C_3 = 5$

Q. Optimal BST

What?

Given sorted array keys of search keys and array frequency, construct a binary tree of all keys such that total cost of all the searches is as small as possible.

How? # Use Catalan and Cuts Strategy.



ans = min (L1+R1, L2+R2, L3+R3 + L4+R4, L5+R5, L6+R6, L7+R7) + sum of all frequencies (for i to j).

For sums, you can use prefix sum array.

	a	b	c	d	e	f	g
a	fa						
b	∝	fb					
c	∝	∝	fc				
d	∝	∝	∝	fd			
e	∝	∝	∝	∝	fe		
f	∝	∝	∝	∝	∝	ff	
g	∝	∝	∝	∝	∝	∝	fg

min (fb, fa) + fa + fb.

min(dp[b][c], dp[a][b]) + fa + fb + fc

+ min of all other possible cuts.

Page No. _____
Date | | | |

ex: keys = {10, 12, 20}
freq = {34, 8, 50}

	0	1	2
0	34	50	142
1	0	8	66
2	0	0	50

```
for (gap > 0)
    dp[i][j] = min( dp[i+1][j], dp[i][j-1])
    for (k = i to j-1)
        dp[i][j] = min ( dp[i][j],
                         dp[i][k] + dp[k+1][j]
    dp[i][j] += psa[j] - psa[i-1]
```

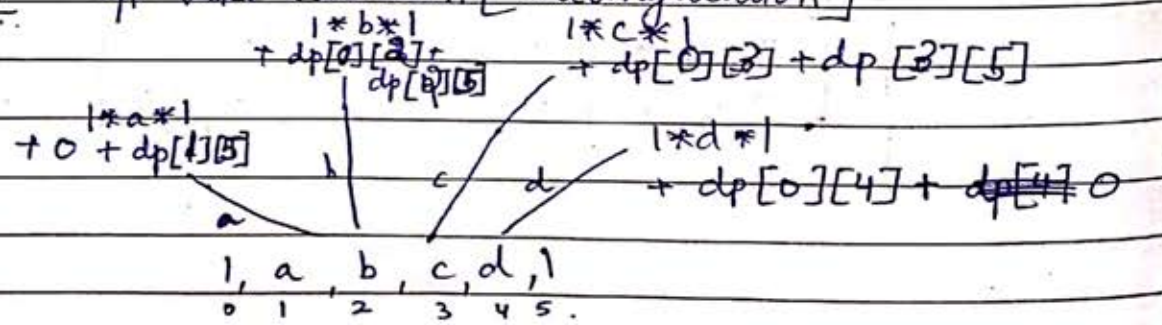
Q. Burst Balloons

What?

You are given n balloons, each having a number on it. You need to burst all balloons, s.t. On every burst you get nums[left] * nums[i] * nums[right] coins. After burst, left & right become adjacent; burst in a way such that coins collected is maximised.

How? # like catalan [triangulation]

ex:



Page No.
Date

ex: [7, 1, 9, 12, 5, 6] ⇒ make dp of n+2 for 1 at each end.

	0	1	2	3	4	5	6	7
0	0	0	7	126	903	1274	1725	1732
1	0	0	0	63	819	1239	1683	1725
2	0	0	0	0	108	585	1062	1071
3	0	0	0	0	0	540	1008	1062
4	0	0	0	0	0	0	360	432
5	0	0	0	0	0	0	0	30
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

Ans = 1732

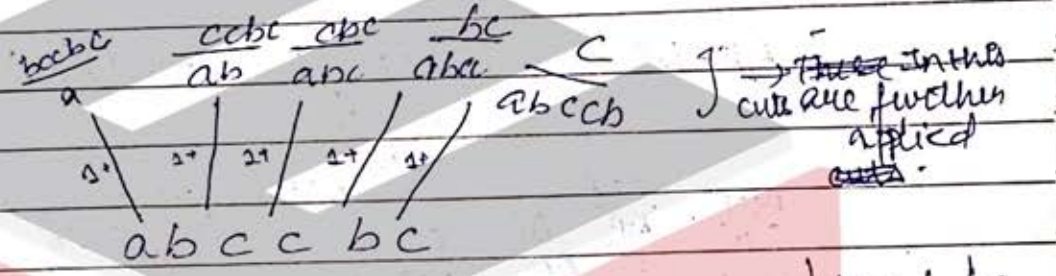
* We append 1, 1 in the two ends of array to make sure that 7, 6 which are the two ends of the original array are included in the burst process.
* At each i,j pair, we take the value of the burst as the max of all the possible bursts possible.

CUTS BASED QUESTIONS.

⊗ Minimum Palindromic cut # O(n³)

What?
Find minimum no. of cuts required to make each part a pallindrome.

How?



ans = a | bccb | c

eg: abccbc

	a	b	c	c	b	c
a	0	1	2	2	1	2
b	0	0	1	1	0	1
c	0	0	0	0	1	1
c	0	0	0	0	1	0
b	0	0	0	0	0	1
c	0	0	0	0	0	0

→ b & b are equal and cc is also palindrom so 0 no. cuts.

→ gap=0 no cuts required

* move on gap strategy.

* Make every cut from all possible location i to j to get min cost

```
gap = 0 to n
    i=0, j=gap+i to j<n
        if (char[i]==char[j] && dp[i+1][j-1]==0)
            dp[i][j] = 0;
        else
            k=i; k<j; k++
                dp[i][j] = min(dp[i][k] + dp[k+1][j])
        dp[i][j] += 1
```

Matrix Chain Multiplication

What?

Given sequence of matrices find the most efficient way to multiply these matrices. Find all order of multiplication and result is best (with less multiplications)

How?

ABCDE → A*(BCDE) or (AB)(CDE) or (ABC)(DE)

⇓

BCDE multiplied in optimized way and then A multiplied to result.

dimension = [10 20 30 40 50 60]

A = 10×20, B = 20×30, C = 30×40, D = 40×50, E = 50×60

	A	B	C	D	E
A	0	6000	18000	38000	68000
B	0	0	24000	48000	88000
C	0	0	0	60000	96000
D	0	0	0	0	120000
E	0	0	0	0	0

Ans = 68000

→ 18000 + 1200 + dim[i] * dim[gap] * dim[gap+1]

→ dim[i] * dim[j] * dim[k]

→ no multiplication

Move in same strategy as ~~matri~~ minimum palindromi cut

For ABCD → $\min$(A×(BCD), (AB)*(CD), (ABC)*D)

Page No. _____
Date | | | |

Q. Boolean Parenthesization

What?

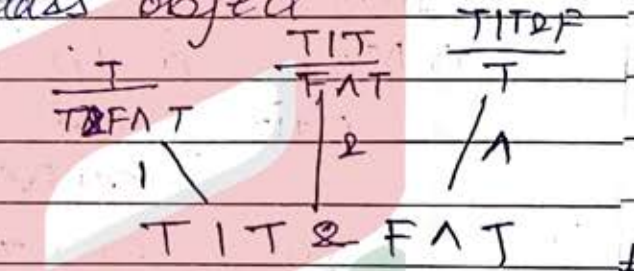
Given a boolean expression with symbols 'T' & 'F' and operators filled b/w are '&', '|', '^'. Count number of ways we can parenthesize the expression so that value of expression evaluates to true.

How?

eg: symbol = {T, T, F, T}
operator = {|, &, ^}

→ Make a dp of pair class object

pair → True count (TC)
→ False count (FC)
→ All count (AC)



	T	|	T	&	F	^	T
T	1,0		1,0		1,1		
T	x		1,0		0,1		
F	x		x		0,1		1,0
T	x		x		x		1,0

LP1 ⇒ Left Pair 1
RP1 = Right Pair 1

(T|(T&F)) } possible options
((T|T)&F)

TC = Addition of all TC of option
FC = Add. of all FC of options

For OR (|) ⇒ TC = LP.AC * RP.AC − LP.FC * RP.FC
FC = LP.FC * RP.FC
AC = TC + FC

For And (&) TC = LP.TC * RP.TC
FC = LP.AC * RP.AC − LP.TC * RP.TC
AC = TC + FC

For XOR (^) TC = LP.TC * RP.FC + LP.FC * RP.TC
FC = LP.TC * RP.TC + LP.FC * RP.FC
AC = TC + FC

Page No.
Date

Same cut strategy TC & FC is found on the basis of formula above.

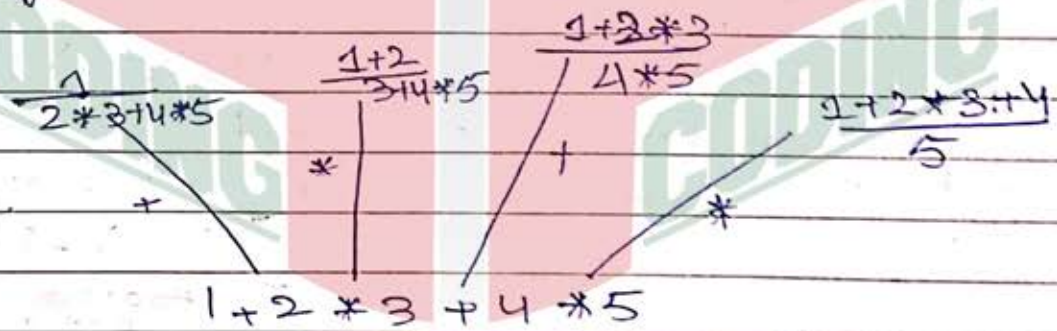
Ans = 4
Here TC will be result

Q Minimum And Maximum Sum Values. With expression.. of '*' & '+'.

What?
Find minimum and maximum value that can be computed of a given expression

How?
1) Make a dp of pair object
pair → min
→ max

eg: 1+2*3+4*5



		1	+	2	*	3	+	4	*	5
	1	1-1		3-3		7-9				
+	2	1		2-2		6-6		10-14		
*	3	x		2		3-3		7-7		23-35
+	4	4		2		x		4-4		20-20
*	5	x		x		1		1		5-5

1+(2*3) = 7
(1+2)*3 = 9

4*5 (operation b/w 2)

↳ no itself is min & max

8 Friends Pairing Problem

What?

Given n friends, each one can remain single or can be paired up. Each friend can be paired only once find total no. of ways in which they can be paired or remain single

How?

eg: n = 8

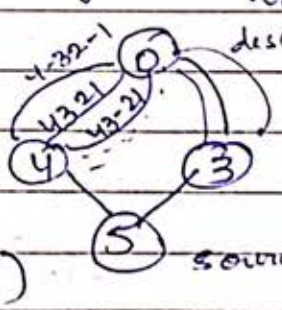
X	1	2	4	10	26	76	232	764
0	1	2	3	4	5	6	7	8

Ans = 764

$dp[i] = dp[i-1] + dp[i-2] * (i-1)$

- $dp[i-1]$: When current i^{th} person is single So it can make $dp[i-1]$ combinations.
- $dp[i-2] * (i-1)$: current element is paired It can pair with $(i-1)$ people and remaining can be arranged in $dp[i-2]$ ways.

*each box contains way to arrange i people.



$\therefore f(n) = f(n-1) + f(n-2) * (n-1)$

Page No. _____
Date

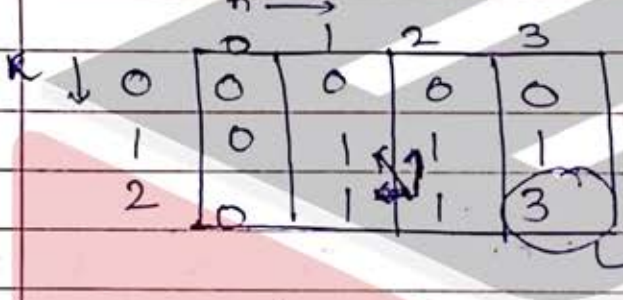
8 Number of ways to partition a set into K subsets

What?

A set has n elements, find number of ways to partition the set into K subsets.

How?

$n = 3$, $K = 2$.



Ans = 3

$dp[i][j] = dp[i-1][j] * K + dp[i-1][j-1]$

$f(n, K) = K * f(n-1, K) + f(n-1, K-1)$

Note: when $K = 1$ or $i == j$ value = 1

Because when $i = 1$ only one set so can be divided in 1 way only and same goes when elements are equal to no. of subsets than (all 1 element in each set).

Each element has 2 choice

Form a new set → $F(n-1, K-1)$ → only these ways.

Be part of other set → $F(n-1, K) * K$ → has K options

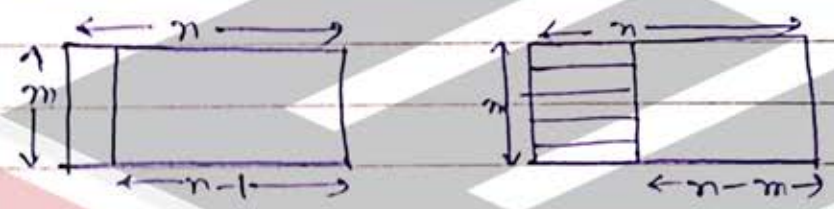
Page No.
Date

Q Count the number of ways to tile floor

(What?)

A floor of size n x m needs to be filled with tiles of size 1 x m. Count no. of ways to tile the given floor using 1 x m tiles.
Tile can be placed horizontally or vertically.

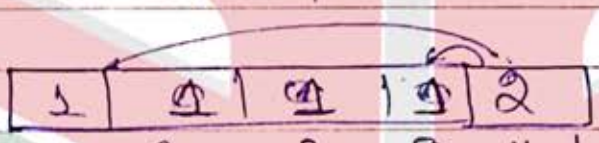
(How?)



vertical placement

$$f(n) = f(n-1) + f(n-m)$$

eg: n = 4 m = 4



place all horizontally or all vertically.

* $dp[i] = dp[i-1] + dp[i-m]$

Q

Coin Change

(I) Permutations

What?

Given a set of coins find all the arrangement (permutations) with sum of coins equal to target (Duplicacy allowed). Infinite supply of each coin.

How?

→ For each sum check for each and every coin.

coins | 2 | 3 | 5 | 6 | sum = 7

	0	1	2	3	4	5	6	7
dp:	1	0	1	1	1	3	3	5
			2.	3.	22.	2.3	6.	23.
						3.2	222	232
						5.	33.	25
								322.
								52

Ans = 5

For 7

7 − 2 = 5 dp[5] = 3 ⇒ 23.2, 32.2, 5.2.

7 − 3 = 4 dp[4] = 1 ⇒ 22.3

7 − 5 = 2 dp[2] = 1 ⇒ 2.5

7 − 6 = 1 dp[1] = 0 ⇒ no combination

Ans = 5

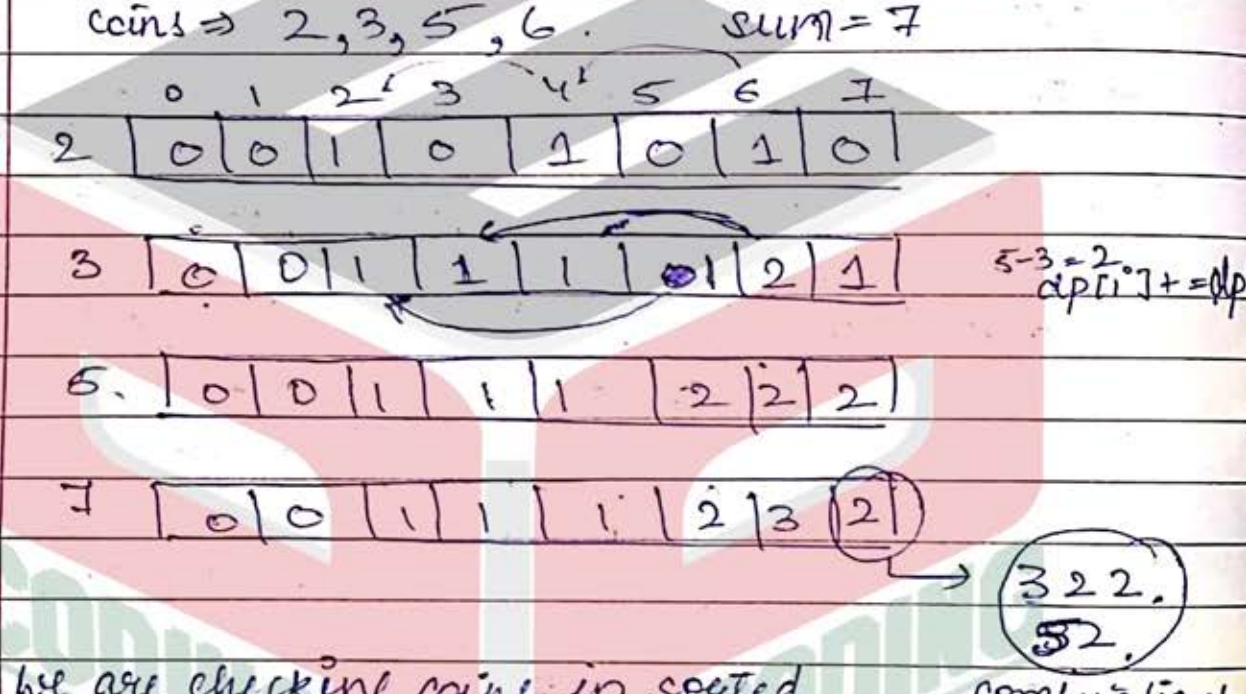
Why?

For each sum we are considering all options i.e why permutations are obtained

(1) Combinations

What?
Given a set of coins. find all selections that will give sum = target.

How?
For each coin update all the sums.



We are checking coins in sorted order hence we are not getting repeated combination ie we will never get 32 only 23 as 2 is checked before 3

Page No.
Date

RELATED QUESTIONS

8 Unbounded Knapsack

What?

You are given a certain item weights and prices and a bag capacity find maximum profit you can make by using these items and filling the bag.

* Duplicacy of each item is allowed.
  ie one item can be taken multiple time.

How?

Just like coin combination, always consider the maximum profit combination

eg: weights | 2 | 5 | 1 | 3 | 4 |
prices | 15 | 14 | 10 | 45 | 30 |

	0	1	2	3	4	5	6	7
2-10	0	10	20	30	40	50	60	70
2-15	0	10	20	30	40	50	60	70
3-45	0	10	20	45	55	65	90	100
4-30	0	10	20	45	55	65	90	100
5-14	0	10	20	45	55	65	90	100

For 3-45 at idx = 3 3-3 = 0 dp[0] + price[3]
= 45

max(45, 30) = 45

idx = 7 · max(100, 70)

↳ 45 + dp[7-3] = 45 + 55 = 100

Page No.
Date

Q Combination Sum version 4

What?

Given an array of Integer and a target find all possible permutations in which element from array can amount to target.

How?

Same as permutation of coin change.
Here coins are elements of array.

Q Linear Equation of N variables

What?

You are given a linear equation with N variables find all possible values of these variables.

How?

eg: $2x + 2y + 3z = 4$.

Here coins are coefficients $[2, 2, 3]$
Sum = 4.

* Solve in same manner as combinations of coin change.

Target - Sum and Based Questions.

Q. Target - Sum

What?

Given an array, and a target; return if a subsequence exists where sum of elements in subsequence equals target.

How? # Build a dp of [n][target+1] where n is array length.

For every box → ✓ True if dp[i-1][j] is True.
→ ✓ True if dp[i-1][j-arr[i-1]] is True.

ex: [2, 3, 1, 5, 6], Target = 10.

arr \ Sum →	0	1	2	3	4	5	6	7	8	9	10
	0	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
2	1	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
3	2	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗
1	3	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
5	4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

10 can be made.

There are 2 sets which make 10.
1) [3, 1, 6]
2) [2, 3, 5].

* Reverse engineer to find paths.

* For negative numbers, make dp where columns range from −target to target.

Why? Every element either plays and checks if target − (its value) can be made, or doesn't play if elements before it have already completed target.

Page No. _____
Date

Q. 0-1 Knapsack:

What?

You are given certain item weights and prices, and a bag capacity. Return maximum price you can make by using these weights and filling the bag completely / maximum total value.

* Each item either contributes completely, or doesn't contribute at all; duplicacy not allowed.

How?

ex: weights = [2, 5, 1, 3, 4] W = 7
prices = [15, 14, 10, 45, 30]

	0	1	2	3	4	5	6	7
-	0	0	0	0	0	0	0	0
2-15	0	0	15	15	15	15	15	15
5-14	0	0	15	15	15	15	15	29
1-10	0	10	15	25	25	25	25	29
3-45	0	10	15	45	55	60	70	70
4-30	0	10	15	45	55	60	70	75

Max profit = 75
[3, 4]

Like Target Sum, dp[i][j] = Math.max (dp[i-1][j] , dp[i-1][j - ~~pri~~ weights[i-1]]

If profit made by elements except you was greater

If profit made by you + left over weight profit is greater.

Page No.
Date

Q. Min partitions.

What?

Partition in a set into two subsets such that difference of subsets' sums is minimum. Return such minimum difference.

How?

1) First create a dp of the same type as in target sum ~~take~~ for [n] [(target)/2 + 1].
Here target = sum of all elements in array.

2) After dp is filled,

```
i = n/2 to 0
if (dp[n][i] == true)
    if (sum - dp[n][i] < diff)
        diff = abs(dp[n][i] - sum);
return diff.
```

boolean array just telling if sum can be made or not.

// we check in last row only, for an index i, abs(value in dp - sum) gives min diff as answer.

Longest Pallindromic Subsequence (LPS)

What?

Given a sequence of length n, find the length of longest palindromic subsequence.

How?

1) Make a dp of nxn where n is string length.
2) Each box of dp stores count of substrings (palindromic) from i to j where i is row and j is col.
3) Move on gap strategy (ie diagnol by diagnol)

eg: S= abckycbc.

dp:

	a	b	c	k	y	c	b	c
a	1	1	1	1	1	3	5	5
b	x	1	1	1	1	3	5	5
c	x	x	1	1	1	3	3	3
k	x	x	x	1	1	3	3	3
y	x	x	x	x	1	1	1	3
c	x	x	x	x	x	1	1	3
b	x	x	x	x	x	x	1	1
c	x	x	x	x	x	x	x	1

→ ans.

Here this box represents string ckycb max length pallindrom Here is cbc of 3 len

```
gap=0 to n
    i=0, j=gap+i to n
        if char(i) == char(j)
            strj[i][j] = 2 + strg[i+1][j-1]
        else
            strj[i][j] = max(strg[i+1][j-1], strg[i][j])
```

Page No.
Date

Why?

First and last character

for any string. S → lps(S) = lps(c1 + r + c2)

4. combination can be possible → x lps(r) x
x lps(r) c2
c1 lps(r) x
c1 lps(r) c2

→ when character c1 & c2 are equal whatever result lps(rest of the string) was given will be added 2.

→ when unequal possible solutions can be find
→ lps(c1 + r) = strg[i][j-1] or lps(r) + c2 = strg[i+1][j]
max of these will be taken as we want max length.

Both cases cover x lps x case.

Q Minimum Number of deletions

What?
Find minimum no. of deletions required to make complete string a pallindrome

How?
1) Find the length of longest pallindromic subsequence using LPS.
2) Subtract LPS length from string length.

eg: S = abcKycbc
lps(S) = 5
min deletions = 8 - 5 = 3 Ans.

Page No. _____
Date | | | |

Q. Largest Common Subsequence

What?

Given two strings S and P find the longest common subsequence.

How?

1) Make a dp storage of size. N+1 x M+1 where N is size of string S & M is size of string P.

S = abcd P = abed.

	a	b	e	d	x
a	3	2	1	1	0
b	2	2	1	1	0
c	1	1	1	1	0
d	1	1	1	1	0
x	0	0	0	0	0

If c_1 & c_2 are equal strg[i][j] = 1 + strg[i+1][j+1]

Here ~~substrings~~ subsequence bcd & bed which is 2.

Algo:

```
i = N to 0
    j = M to 0
        if char(i) = char(j)
            strg[i][j] = 1 + strg[i+1][j+1]
        else
            strg[i][j] = max(strg[i+1][j], strg[i][j+1])
```

Here each box represents length of largest common ~~substring~~ subsequence from ~~that~~ subsequence formed by (i to n) & (j to m).

Page No.
Date

Q Count Palindromic Subsequence.

What?

Given a string of length N find count of all the palindromic subsequences.

How?

1) Make a dp of size N×N
2) Gap strategy is used to traverse

eg: s: acbkca

	a	c	b	k	c	a
a	1	2	3	4	8	17
c	0	1	2	3	7	8
b	0	0	1	2	3	4
k	0	0	0	1	2	3
c	0	0	0	0	1	2
a	0	0	0	0	0	1

→ 6-2=4

Ans = 17

Each character its sub-is a palindrome !

```
gap=0 to n
  i=0, j=gap+i to n,
    if char(i)==char(j)
      strg[i][j]=1+strg[i][j-1]+strg[i+1][j]
    else
      strg[i][j]=strg[i][j-1]+strg[i+1][j]-strg[i+1][j-1]
```

* Each box contain count of substring (i, j)'s all palindromic subsequence.

Q Count and longest Palindromic Substring.

What?

Given a string of length N find total no. of palindromic substrings and also the largest palindromic substring of that string.

How?

→ Make a 2-D dp of N×N size

→ Move on gap strategy:

Eg: abccbc.

→ largest substring = bccb

	a	b	c	c	b	c
a	T	F	F	F	F	F
b	F	T	F	F	(T)	F
c	F	F	T	T	F	F
c	F	F	F	T	F	T
b	F	F	F	F	T	F
c	F	F	F	F	F	T

→ when i & jth char not equal so not a palindrom

→ when characters i & j are equal check for inbetween string.

↳ longest substring is a palindrome

→ update count for each True value.

→ since we are travelling on gap last box to get true will be the largest substring. here bccb. at gap 4.

Q Minimum no. of Insertions

What?

Count minimum no. of Insertion required to make whole string a palindrome.

How?

→ Make a dp of n×n where n is length of string

→ Every box contains no. of insertions for that substring a palindrome

Page No.
Date

eg: a c b b c b

	a	c	b	b	c	b
a	0	1	2	2	1	2
c	∞	0	1	1	0	1
b	∞	∞	0	0	1	1
b	∞	∞	∞	0	1	0
c	∞	∞	∞	∞	0	1
b	∞	∞	∞	∞	∞	0

→ when ith & jth char not equal $\min[(i+1, j), (i, j-1)] + 1$

→ when ith & jth character same look diagonally down.

↳ 1 length substring palindrome so no insertion.

Ans = 2

2 ~~deleti~~ Insertions are required.

a b c b b c b a
↑ ↑

Longest Increasing Subsequence

LIS

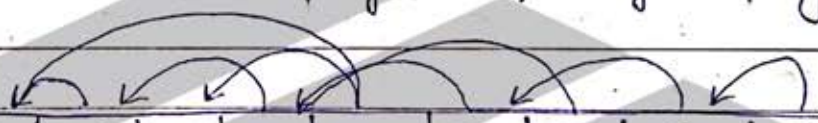
What?

Find length of longest increasing subsequence in a given assay.

How?

Approach used : Dynamic Programming.

Ex:



Original:	10	22	9	33	21	50	41	60	80	1
DP:	1	2	1	3	2	4	4	5	6	1
	10.	10. 22.	9.	10 22 33	10 21	10 22 33 50	10 22 33 41	10 22 33 50 60	10 22 33 50 60 80	1

↑ answer.

Why?

A box in the dp is defined as containing the maximum length LIS from starting (index 0) to the current position.

Each current element appends itself to the back of the maximum of all subarrays with largest element smaller to current element.

Complexity : $O(n^2)$ – Time
$O(n)$ – Space

Page No. _____
Date | | | |

Questions related to LIS.

Q. Maximum Sum Increasing Subsequence (MSIS)

What?

Find maximum sum increasing subsequence in given array.

How?

Ex:

Original:	10	32	-90	55	21	-110	27	8	-5
DP:	10	42	-90	97	31	-110	58	8	-5
	10	10 32	-90	10 32 55	10 21	-110	10 21 27	8	-5

↑ answer = (97) : [10, 32, 55]

Why?

Using same LIS technique, the box now contains the sum formed by maximum increasing subsequence from index 0 to current index.

Complexity : $O(n^2)$ — Time
$O(n)$ — Space.

Q. Longest Bitonic Subsequence

What?

Given an array, find the longest bitonic subsequence you can form, i.e; subsequence which first increases, then decreases.

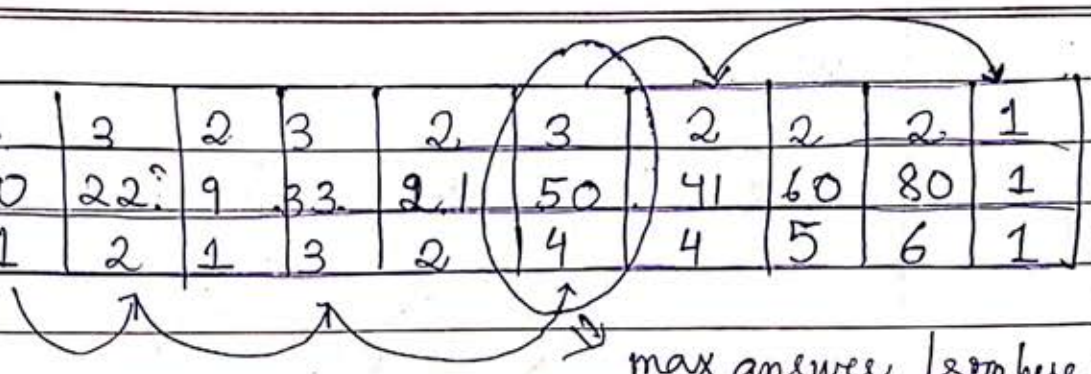
How?

Find both LIS & LDS of the given array, ans is max(lis[i] + lds[i] - 1) at a given i.

Page No.
Date

ex:

lds:	3	3	2	3	2	3	2	2	2	1
arr:	10	22	9	33	21	50	41	60	80	1
lis:	1	2	1	3	2	4	4	5	6	1



max answer from here.

Longest bitonic subsequence = 4 + 3 - 1 = 6.

↓

10, 22, 33, 50, 41, 1.

* Related: Maximum Sum Bitonic Subsequence (MSIS)

Q. Minimum number of deletions to make sorted sequence.

What?

Return number of deletions to be done to make given sequence sorted.

How?

Step 1. Find lis.

Step 2. Return n (array length) - lis (length).

ex:

	7	1	5	2	3	4
lis:	1	1	2	2	3	4

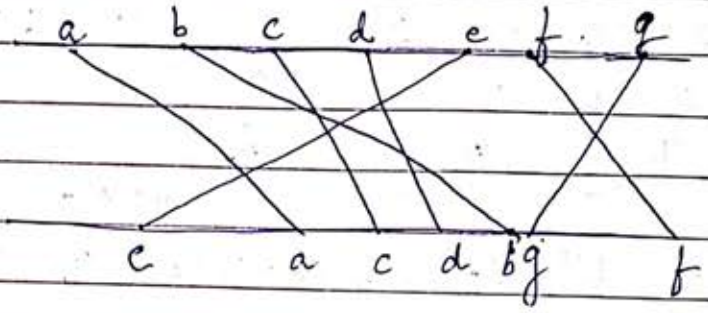
∴ #Deletions = 6 - 4 = 2 (7, 5)

Q. Maximum non-overlapping bridges.

What?

Given the starting and ending of bridges, return the maximum no. of non-overlapping bridges that can be constructed.

How?



Step1: Sort on starting points
Step2: LIS on end points.

e	a	c	d	b	g	f
1	1	2	3	2	4	4
e	a	a	a	a	a	a
		c	c		c	c
			d	b	d	d
					g	f

∴ Maximum 4 bridges can be constructed

Q. Russian Doll Envelopes.

What?

Return maximum number of envelopes that can fit into another much like a russian doll.

How?

Step1: Sort on widths.
Step2: Lis on heights.

Given: widths: 2 | 1 | 7 | 10 | 8
heights: 9 | 4 | 1 | 7 | 3

After sorting:

	a	b	c	d	e
	1	2	7	8	10
	4	9	1	3	7
lis →	1	2	1	2	2
	a	a,b	c	c,d	c,d,e

⇒ Ans = 3

8. Activity Selection — Greedy (like LIS).

What?

Given starting and ending times of various activities, return maximum number of activities that can be done by a person in a party.

How? → very similar to bridges, optimised by using greedy approach.

Step 1. Sort on end points.

Step 2. Include first activity always, then include subsequent activities if end of previous was before starting of current.

ex: st a b c d e f g h

end a b d c f g e h

end: a b c d e f g h

ends sorted:

a	b	d	c	f	g	e	h
1	X	2	X	3	4	X	5
✓	b.st	✓		✓	✓		✓
	← a.et						

st: start time, et: end time.

⇒ 5 activities a, d, f, g, h can be done.

Page No.
Date

Q. Arithmetic - Slices I

What?

Return the number of arithmetic slices in an array, slice is a sequence of atleast 3 numbers and has same difference between consecutive numbers. (Subarrays as slices.)

How? → Very much like lis, just check if you have same difference with your previous element.

Ans = sum of ans array
= 1 + 2 + 3 + 1 + 2 + 1 + 2
= 12

dp[i] = dp[i+1] + 1 if it makes a slice.
Notice that each subarray is an AP.

Q. Arithmetic - Slices - II

What?

Find all arithmetic slices such that they are of atleast 3 length and ~~can be~~ are subsequences of the given array where each element has same differend with its consecutives.

Page No. _____
Date

How?

Maintain a HashMap for each element where key is gap and value is count of elements with the gap prior to the current element.

ex:

2	-2	4	0	6	8	10	-1	3	6	7	8	10
	-4=1	6=1	-4=1	6=1	2=3	2=4	-11=1	1=1	3=2	1=2	1=3	2=7
		2=1	2=1	2=2	8=1	4=2	-9=1	-7=1	7=1	4=2	2=6	3=2
			-2=1	8=1	4=1	10=1	-7=1	-5=1	-4=1	8=1	5=2	4=4
				4=1	10=1	6=2	-1=1	-3=1	-2=1	-3=1	9=1	7=1
						12=1	-5=1	3=1	0=1	-1=1	-2=1	11=1
						8=1	-3=1	-1=1	6=1	7=1	8=1	0=1
								5=1	2=2	3=1	8=1	8=1
								1=1	8=1	9=1	4=1	0=1
									4=1	5=1	10=1	6=2
											6=1	12=1

all subsequences formed.

1 (2,4,6)
+ 2 (2,4,6,8) (4,6,8)
+ 3 (2,4,6,8,10) (4,6,8,10) (6,8,10)
+ 1 (2,6,10)
+ 1 (-2,4,10)
+ 1 (0,3,6)
+ 1 (2,4,6)
+ 1 (-1,3,7)
+ 2 (2,4,6,8) (4,6,8)
+ 6 (2,4,6,8,10) (4,6,8,10) (6,8,10) (2,4,6,8,10) (4,6,8,10) (6,8,10)
+ 1 (4,7,10)
+ 1 (2,6,10)
+ 2 (6,7,8) (6,7,8)
+ 1 (-2,3,8)
+ 2 (2,4,6,8) (4,6,8)
+ 1 (2,6,10)
+ 1 (-2,4,10)
+ 3 (2,4,6,8,10) (4,6,8,10) (6,8,10) =)

* For each element, it calculates gap with all previous elements, and adds in its hashmap gap vs count+1 where count is the count stored for the same gap in the element seen.

* If no count for same gap found, add gap vs 1 in current hashmap.
* If more than 2 previous elements have same gap with the current element, add the count2 in the gap vs count1 which was already stored.
* for each element, ans += count obtained from previous maps (if > 1).

Q. Word Break

What?

Given a string and a dictionary (list of strings) tell if it is possible to make a sentence from string.

How?

ex: string : ilikemango

dictionary : { sung, man, go, like, i, sam, mango, samsung }.

Use LIS strategy.

i	l	i	k	e	m	a	n	g	o
T	F	F	F	T	F	F	T	F	T

i in dictionary.

1) il → not in dictionary. (∵ i is true)
2) il → not in dictionary.

1) g not in dictionary (∵ n is T)
2) mang not in dictionary (∵ E is T)
3) likemang not present (∵ i is T)
4) ilikemang not in dictionary.

variations: how many maximum words & sentences can be formed?

* In T/F case, we check for all the words that could be present in dictionary where previous word (till T.) is seen in dp array, and from that index +1 to current index, substring is checked.

Page No. 11
Date

for ex: for [n]

i like → e was true, so we checked 'man' in dictionary & found it.
TFFFT

In count words, we would, in this case, increase word count by 1.

for words & sentences:

ex: s: ilikemangosamsung

d: { i, man, go, sam, like, sung, mango, samsung }.

(+2) o was end of a word; m was end of a word, (+2).

	i	l	i	k	e	m	a	n	g	o	s	a	m	s	u	n	g
sentences:	1	0	0	0	1	0	0	1	0	2	0	0	2	0	0	0	4
words:	1	0	0	0	2	0	0	3	0	4	0	0	5	0	0	0	6

for g: 4 sentences are:
i - like - man - go - sam - sung
i - like - mango - sam - sung
i - like - man - go - samsung
i - like - mango - samsung.

g → m ('ilike mango sam' was already made in 2 ways, & sung was in dictionary, ∴ +2 sentences). - ①
→ o ('i like mango' was already made in 2 ways, & samsung was in dictionary ∴ +2 sentences). - ②

⇒ g = ① + ②
= +2 + 2
= +4

Page No.
Date

Q. Maximum sum Alternating Subsequence.

What?

Given an array, find longest M.S.A.S such that it is of the type increasing - decreasing - increasing

How?

We use L.I.S strategy and create dp array from end; # also uses include-exclude.

4	3	8	5	3	8
0	1	2	3	4	5

Lengths (sums):

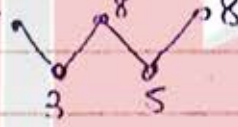
1 (4)	4 (3-8-5-8)	3 (8)	2 (5-8)	2 (3-8)	1 (8)
5 (4-3-8-5-8)	4 (3-8-5-8)	3 (8-5-8)	3 (5-3-8)	1 (3)	1 (8)
0	1	2	3	4	5

← increasing subsequence

← decreasing subsequence

Ans: 5 length subsequence

⇒ (4 - 3 - 8 - 5 - 8)



Sum = 28

* At every ith box, for j = i+1 to end, if increasing sequence is to be found, use decreasing boxes and vice versa.
* Since start is ↘ always, answer will always be in decreasing dp[0].

Page No. _____
Date

Q. Highway Billboard Problem.

What?

Given highway of M miles, task is to place billboards in such a way that revenue is maximized; with a restriction that no two billboards can be placed within t miles or less than it.

How?

ex: M = 20

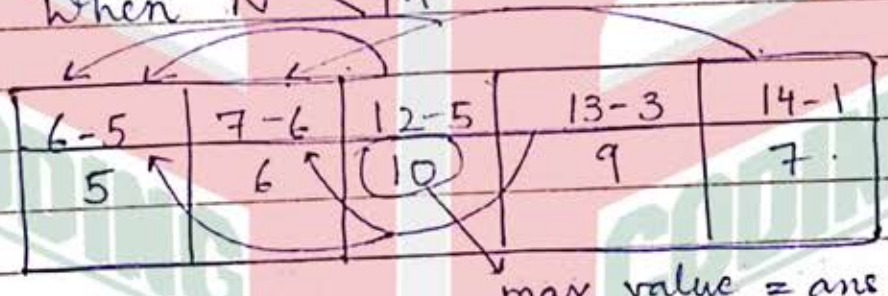
X = { 6, 7, 12, 13, 14 } → billboards

R = { 5, 6, 5, 3, 1 } → revenue.

1) If $N^2 > M$, use Miles method, else LIS.

2) n log n approach using TreeSet.

1) When $N^2 \leq M$



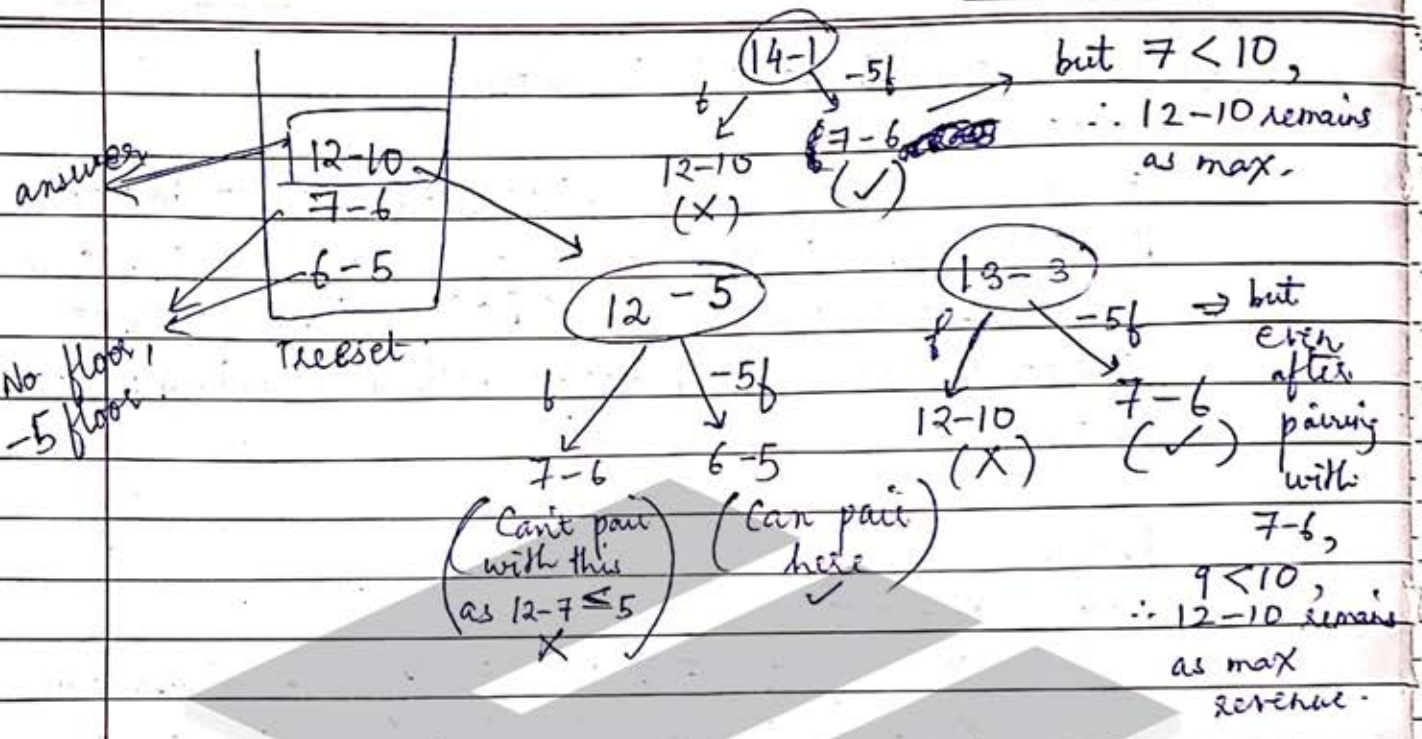
max value = ans = 10.

2) When $N^2 > M$.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	5	6	6	6	6	6	10	10	10	10	10	10	10	10	10

ans = dp[M] = 10.

3) Use Treeset, for each pair, if its floor and -t floor does not exist, push pair, else let it have maximum profit.



Q. Word wrap problem.

What?

Given a sequence of words, and limit on characters in one line (line width), put line breaks in given sequence such that lines are printed in a way to minimize score.

Score = (Sum of all lines where one line cost = (No. of extra spaces in the line at the end)^3).

Don't include last line's score.

How? You can use two strategies, either LIS from left or LIS from right.

For sum of scores, use a prefix sum array / 2D dp.

For a word, gap is calculated as:

ex: geeks = 5 length, max length = 15.

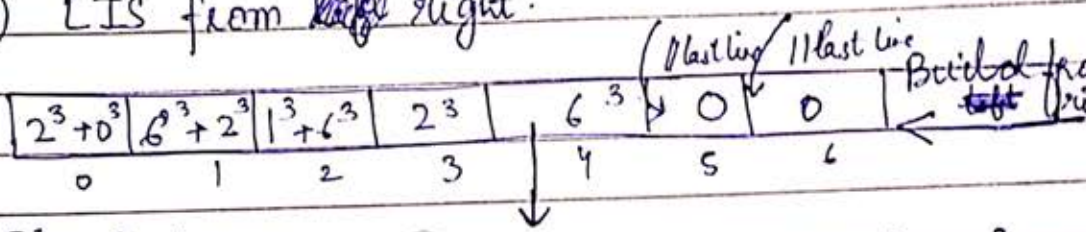
∴ gap in a line having "geeks" = $(15-5)^3$
$= 10^3$

0 1 2 3 4 5 6

Geeks for geeks presents word wrap problem.

lengths :	5	3	5	8	4	4	7

I) LIS from ~~left~~ right.



dp[0] = 8 ⇒ ans.

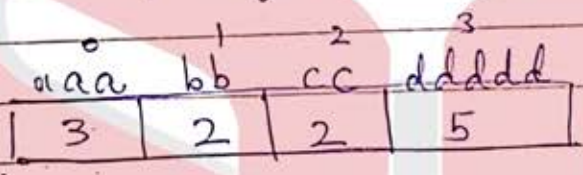
Ans: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
L1 → geeks for geeks.
L2 → presents word
L3 → wrap problem

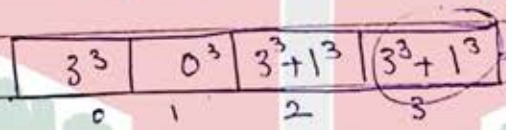
I) 4 → L1 $(15-4) = 11^3$
5, 6 → L2 0^3

II) 4, 5 → L1 $(15-9)^3 = 6^3$ ✓ min of all
6 → L2 0^3

III) 4, 5, 6 → cannot come in one line.

II) example for LIS from left. M=6





Ans: ⇒ dp[3] = 9+1 = 10

Ans:
L1 → aaa (0 1 2)
L2 → bb cc (0 1 2 3)
L3 → ddddd (0 1 2 3 4)

INCLUDE EXCLUDE STRATEGY BASED QUESTIONS.

Q Count of Binary strings with no consecutive zeroes

What?
Count the no: of binary strings that can be formed of size n such that no zero appear consecutively.

How?

	1	2	3	4	5	6	7	n=7
eg:0	1	1	2	3	5	8	13	← Zeros
1	1	2	3	5	8	13	21	← Ones

3+5

Ans = 13 + 21 = 34

Zeros represents strings ending with zero's
Ones represents strings ending with 1's

for any length x
string ending with zero Zeros = Count of string of lenght (x-1) ending with 1'.

(Ones = Z) nz = Ones. ∵ (no 2 zero can come together)

For any length x string ending at 1 are no = one's + zero's as one can be appended to both.

Page No.
Date

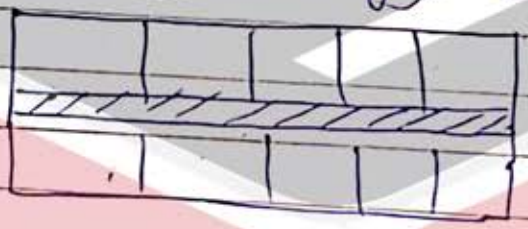
Q Building space Building

What?

We are given no. of sections and each section has 2 plots on either sides of the road. Find all ways to construct building such that no two building are together.

How?

Apply the same strategy of Include & Exclude.



sections = 5

(1's) Space → represent if current spot is a space.

(0's) building → ways in which that spot is a building

(Improv strategy)

building can be arranged such

	1	2	3	4	5
(0's) build	1	1	2	3	5
(1's)	1	2	3	5	8

sp

13 → this will give arrangement for one side of road.

Same arrangement can be done on other side of road.

⇒ Ans = 2*13

① Find arrangement for one side using Include and exclude considering spaces 1's & building as 0's

② Twice the res for answer.

Page No.
Date

Q Maximum Sum Subsequence without Adjacent

What?
Find the maximum sum subsequence such that no two adjacent element come in same subsequence.

How?

eg:

	5	6	10	100	106
Inc:	5	6	15	106	112
Exc	0	5	6	15	106

prev Exc + current value.

Ans = 112

prev Inc.

Inc: Means if our element is Included (so prev Exclude sum is added to it)

Exc: It Means that our element is Excluded (so prev Include is the current Exclude)

Q Paint House

What?
Given n no. of houses each can be painted with 3 different colours red, blue & green. cost of painting each house is given. You need to find minimum amount to spend painting all the houses such that no 2 adjacent house have same color.

How?
r → It represents in how many ways previous houses can be colored such that current house is colored red.

Page No. _____
Date

Similarly for 'g' and 'b'

Eg:

	R	G	B
H1	1	5	7
H2	5	8	4
H3	3	2	9
H4	1	2	4

new r = min(g, b) + cost(Red) at i°

new g = min(r, b) + cost of green at House i

new b = min(r, g) + cost of blue at House i

⇒

	H1	H2	H3	H4
r	1	10	8	(8) → Ans
g	5	9	7	10
b	7	5	18	11

* We are taking minimum of other colors because only those combinations will be valid.

Ans = min(r, g, b) ⇒ 8.

Paint House Version 2

What?

Given n houses that can be painted from K different colors, such that no two adjacent houses have same color. Cost of painting each house with each color is given.

How?

For eg:

	H1	H2	H3	--	Hn
color1	c_1	$\min(c_2 - c_K) + c_1$			
color2	c_2	$\min(c_1, c_3 - c_K) + c_2$			
color3	c_3				
⋮	⋮	⋮			
colorK	c_K	$\min(c_1 - c_{K-1}) + c_K$			

① For House 1 cost will be same as given costs

② For House 1 to n Find minimum from prev cost of all other colors and add to current cost

For House i & color j

cost = min (cost of all color 0 to j-1 & cost of all color j+1 to k) + cost[i][j]

Note* To find minimum keep track of smallest and second smallest of all previous cost. If min smallest cost is from same color then choose second smallest cost and add curr cost otherwise add smallest cost.

Q Paint Fence

What?

Given a n post fence. find the cost to paint it with k colors such that not more than 2 adjacent posts has same color.

How?

eg: n = 8 , K = 3

	1	2	3	4	5	6	7	8	
S	0	3	6	18	48	132	360	984	]+
D	3	6	18	48	132	360	984	2688	

S: combinations such that current color and previous color is kept same

D: combinations such that current color and prev color is kept different.

Ans = 984 + 2688 = 3672.

Page No. _____
Date

current same (S) = prev Different (D)

because
$\cdots x_1 \Rightarrow \cdots x_1 x_1$
$\cdots x_2 \Rightarrow \cdots x_2 x_2$
$\cdots x_3 \Rightarrow \cdots x_3 x_3$
x_1, x_2, x_3 are colors
D combinations

curren Different (D) = (prev same (S) + prev Diff (D)) * (K-1)

each combination ending with color X has (K-1) color options to make last 2 different.

Q Count substrings of nature $a^+b^+c^+$ in a given string

what?
count all the substring of a given string which are in form $a^+b^+c^+$ where '+' represents one or more occurrence of that character.

How?
eg: s = abcabc.

	a	b	c	a	b	c
countA	1	1	1	3	3	3
countB	0	1	1	1	5	5
countC	0	0	1	1	1	(7) → ans

countA = 2 * (countA) + 1
countB = 2 * (countB) + countA
countC = 2 * (countC) + countB.
↓ Previous

Page No. _____
Date | | | |

Why?

	a	b	c	a	b	c
a	1 (a)	1 (a)	1 (a)	3 (a, aa')	3 (a, aa')	3 (a, aa')
b	0	1 (ab)	1 (ab)	1 (ab)	5 (abb', ab, ab', aa'b, a'b)	5 (ab, a'b, ab', abb', aa'b')
c	0	0	1 (abc)	1 (abc)	1 (abc)	7 (aa'b'c', abc, a'b'c', abb'c, abcc', abc', ab'c')

For count C. we are adding 2 * countC because we can make all the combination made by old C with new c (c') and countB is added as c' can be appended at the end of these strings to make them of $a^i b^j c^j$ type.

Similarly with countA + countB.

Some other Related Questions

Q Buy and sell stocks with only K transaction allowed.

Q Buy and sell stocks with ~~only~~ cooldown time

Q Buy and sell stock with transaction fee.

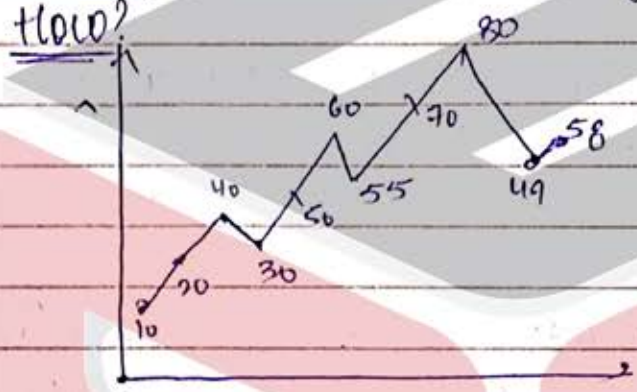
Buy and Sell Stocks (6 variations)

1) With Infinite transactions

What?

Find the maximum profit that can be earned when infinite transactions can be done.
only one allowed buy & buy should not overlap

How?



when
arr[i+1] > arr[i]
then buy.
profit += arr[i] - arr[i-1]

(20-10) + (40-20) + (50-30) + (60-50) + (70-55)
+ (80-70) + (58-49)

= 94

2) With only 1 transaction

What?

Find the maximum profit that can be earned when only 1 transaction is allowed.

How?

eg: 2 5 9 10 7 8 15 3 6 12

minElement	2	2	2	2	2	2	2	2	2	2
curr profit	2	5-2=3	7	8	5	6	13	1	4	10
max profit	0	3	7	8	8	8	13	13	13	13

curr Element − minElement.

Since only 1 transaction is allowed we find min price at which we can buy. and calculate profit.

min = minimum (min el , current element)
curr profit = curr element - min element.

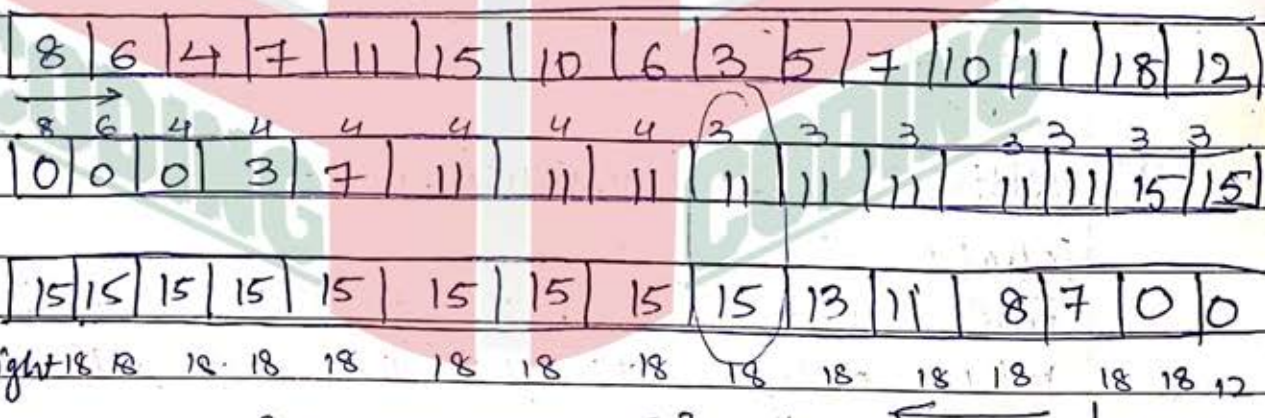
3) Atmost 2 transaction allowed.

What?

Calculate maximum profit when atmost 2 transactions can be done.

How?

2 dp's will be built first containing profits calculated with min and other containing profits with max.



→ maxprofit from left indicates maximum profit earned when selling on that box or before.

→ max profit from right indicates when selling (at max) and buy are current box.

Page No.
Date

(4) Buy and Sell stocks with K transaction

What?
Maximum profit that can be earned doing atmost K transactions.

How?
→ Make a 2d-DP that maps all n elements on column & K on rows i.e (K*n) size.

eg: | 9 | 2 | 3 | 7 | 8 | 3 | 10 | 11 | ⇒ prices.

K=3

t ↓	9	2	3	7	8	3	10	11
0	0	0	0	0	0	0	0	0
1	0	0	1	5	6	6	8	9
2	0	0	1	5	6	6	13	14
3	0	0	1	5	6	6	13	14

Ans = 14

Here every box represents maximum profit if. i transaction allowed.

→ dp[i][j] = max(dp[i-1][x] + price[j] - price[x]) where x is from 0 to j-1

Also dp[i][j] = max(dp[i][j-1], dp[i-1][j])

→ To perform current transaction maximum of all profits earned in previous transaction is taken and cost (profit) from current transaction is added.
→ We also check previous value is same row because maybe profit earned from that is more than current.

Page No. _____
Date | | | |

Note: When k is greater than n/2 use infinite transaction (version1) method because to earn profit from n prices maximum transaction needed are n/2.

(5) With Cooldown Period

What?
Find the maximum profit that can be earned when 1 day cooldown is required after every transaction (sell)

How?
This is done by include & Exclude Strategy.

eg: | 9 | 1 | 3 | 10 | 1 | 4 | 8 | 6 |

	9	1	3	10	1	4	8	6
buy	-9	-1	-1	-1	1	5	5	5
sell	0	0	2	9	9	9	13	13
cool	0	0	0	2	9	9	9	13

max of these = 13

→ max(2, 10-1) = 9

Buy → Maximum profit when bought at current price
Sell → Maximum profit when sold at current price
Cooldown → Maximum profit when at this price cooldown is done!

Buy = max (cool-price(curr) , buy)
Sell = max (sell , price(curr) + buy)
Cool = max (sell, cool)

(6) With transaction fees

What?

Maximum profit that can be earned when transaction fees is applied on each transaction

How?

Include Exclude strategy is used.

eg:

	9	1	3	10	1	11	6	9
buy	-9	-1	-1	-1	6	6	6	6
sell	0	0	0	7	7	8	10	13

Fee = 2 ₹

Ans = 13

(8, 6+6-2)

sell = max (sell , price + buy) - transaction fee)

buy = max (buy , sell - price)

Costs Paths

RELATED QUESTIONS.

Q Min Cost Path

What?

Given a 2-D board find the minimum cost required to reach from starting position (0,0) to ending pos (N-1, M-1) where board dimensions are N×M. 2 possible moves are Horizontal (1 step) & vertical (1 step)

How?

1) Make a dp storage of dimension N×M

eg:

	0	1	2	3	4
0	5	2	8	1	7
1	0	5	0	4	3
2	3	1	2	8	6
3	6	0	7	0	8
4	1	2	4	9	5

```
i=N-1 to 0
    j=M-1 to 0
        dp[i][j]
            = min(dp[i+1][j]
                  , dp[i][j+1])
              + cost[i][j];
```

dp:

	0	1	2	3	4
0	29	28	30	26	29
1	24	26	22	5	22
2	24	21	29	21	19
3	26	20	20	13	13
4	21	20	18	14	5

dp[0][0] contains the res.

* Here every box indicates minimum cost from that box to destination.

Page No.
Date

Reverse Engineering (To print paths with min cost)

0	29	28	30	26	29
1	24	26	22	5	22
2	24	21	21	21	19
3	26	20	20	13	13
4	21	20	18	14	5

For any cell check which dp[i+1][j] or/and dp[i][j+1] has equal value to dp[i][j] - cost[i][j] At this position.

dp[i+1][j] & dp[i][j+1] both are equal to dp[i][j] - cost so both are added to Queue as potential paths.

→ we will use queue and apply BFS technique for reverse Engineering

Algo [remove mark* work add children]

↳ based on above

q

0,0	0,1	1,0	1,1	2,1	2,2	3,1	3,2	4,1	4,2	4,3
29	24	24	21	21	20	20	20	20	18	...
	H	V	VVH	VVH	VVHV	VVHV	VVHVH	VVHVV	VVHVHV	

In this way path will be build paths are.

→ VVHVHVHH

→ VVHVVHHHH

We used Queue because of call stack etc

Page No.
Date

GOLD MINE Problem.

What?

Given a n*m matrix which contains some positive integer that represents amount of gold. Find out the maximum amount of gold he can collect when 3 moves allowed right up, right, right down

How?

Possible directions → right up, right, right down

Given matrix:

	0	1	2	3
0	4	1	0	3
1	3	0	2	8
2	6	4	1	6
3	2	0	2	7
4	1	6	3	4

	0	1	2	3
0	15	11	8	3
1	17	10	10	8
2	20	14	9	6
3	18	10	9	7
4	17	16	10	4

last column, as it is.

```
j = M-1 to 0
    for i = 0 to N-1
        dp[i][j] = Max(dp[i+1][j+1], dp[i-1][j+1], dp[i][j+1]) + gold[i][j]
res = max of first column.
```

* Every box represents max gold that can be collected from that box + further

Why?

As he needs to collect maximum gold it will take maximum from possible position.

Page No. _____
Date | | | |

Reverse Engineering To print path

	0	1	2	3
0	15	11	8	3
1	17	11	10 →	8
2	20 →	14	9	6
3	18	10	9	7
4	17	16	10	4

Path = R R D R

→ For this box make call to position where gold = dp[i][j] - gold[i][j]

Keyboard Questions.

Q. 2 Keys Keyboard.

What?

You are allowed 2 operations on a keyboard:
1) Copy all that copies entire workspace – C
2) Paste copied characters – P
* Initially only one character X is present on the workspace.

For a given n, get n X's on workspace & return least number of steps to do it.

Examples: Initially given $n = 1$: X.

n	workspace	steps
1	X	0
2	XX	CP (2)
3	XXX	CPP (3)
4	XXXX	CPCP (4)
5	XXXXX	CPPPP (5)
6	CPCPP / CPPCP gives XXXXXX in (5).	
⋮		

How?

We use sieve of eratosthenes for this question. Intially sieve integer array of 0 to $n+1$ is made where initially each box stores itself as its smallest prime factor.
Then we use sieve method to cancel all non-prime numbers and if any value has been cancelled once, i.e., if at a time sieve[i] ≠ i, then no further cancellations are done for ith position.

Page No.
Date

For ex: $n = 10$.

(1) loop from here

table loop (2) starts here

0 - 0	5 - 5	10 - 10 2
1 - 1	6 - 6 2	
2 - 2	7 - 7	
3 - 3	8 - 8 2	
4 - 4 2	9 - 9 3	

∴ Answer ⇒ Now calculate steps:

```
while (n != 1)
    count += scive[n]
    n = n / scive[n]
return count
```

n = 10	count
10	0
5	2 (10/2)
1	7 (5/5)

Ans for 10 = 7 which is CPCPPPP
XXXXXXXXXX

Q. 4 Keys Keyboard

What?

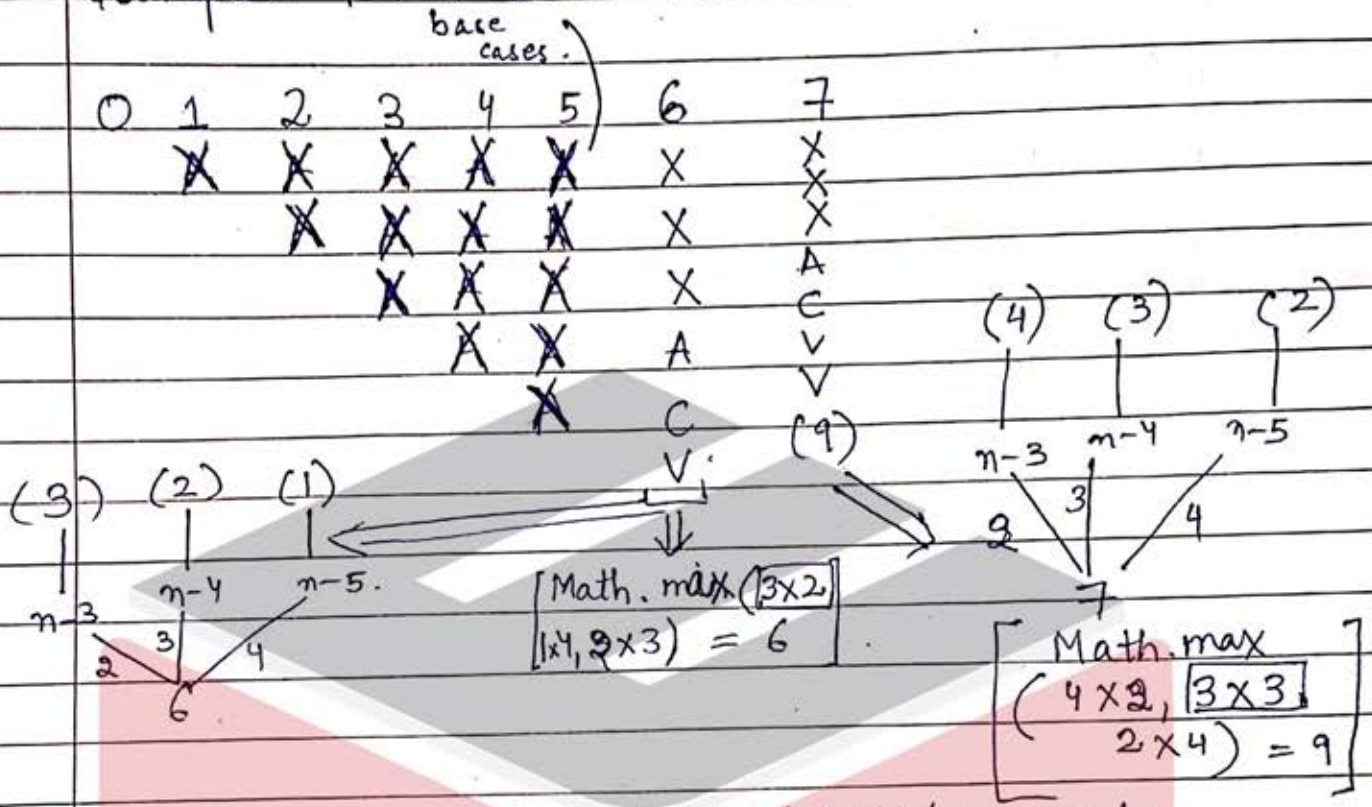
You are allowed 4 keys' functions:

1) A : Print one 'A' on workspace. (X)
2) (Ctrl+A) : Select whole workspace. (A)
3) (Ctrl+C) : Copy selection to buffer. (C)
4) (Ctrl+V) : Add buffer to workspace. (V)

Return the max characters you can produce in given n.

Page No.
Date

Example: For n = 7:



∴ n = 7 can use XXX ACVV to produce 9 characters at max.

For each block where $i \geq 6$, $dp[i]$ = Math.max($2 * dp[i-3]$, $3 * dp[i-4]$, $4 * dp[i-5]$).

For $i \geq 6$, we look at $i-3$, $i-4$, $i-5$ only because all other solutions will be included in these 3 calls.
This is just like fibonacci where ans at i is calculated by $i-1$, $i-2$ because all other calls are included in these two.

Minimum Jumps and related questions.

Q. Find minimum jumps.

What?

Find minimum jumps required to go from source to destination n, given the jumps that can be made from each place.

How?

→ least out of all subsequent path jumps.

n = 10

arr

3	5	0	1	2	0	3	1	0	1	0
0	1	2	3	4	5	6	7	8	9	10

Jumps: dp array of [n] size.

dp:

4	3	11	4	3	11	2	12	11	1	0
0	1	2	3	4	5	6	7	8	9	10

Minimum 4 jumps needed.

store ass. length at place where no jumps can be made.

For paths: (Extension)

dp:

4	3	0	1	1	0	1	0	0	1	1
0	1	2	3	4	5	6	7	8	9	10

4 paths.

* You can also print paths using reverse engineering.

for paths, dp[n] = 1, but for jumps, dp[n] = 0 as 10 to 10 is a path, not a jump.

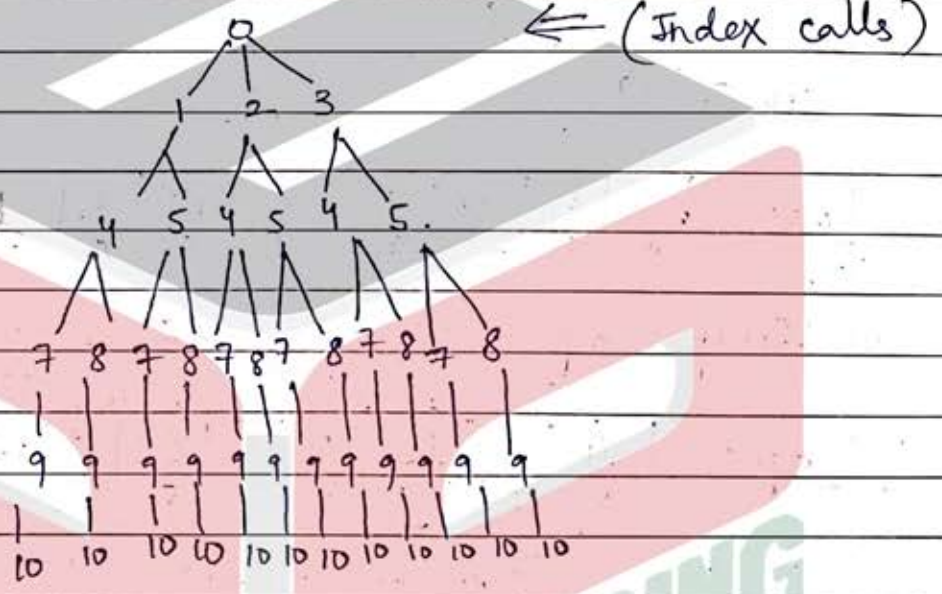
* for printing all paths with minimum jumps:

given:

3	5	3	2	4	3	0	2	1	1	0
0	1	2	3	4	5	6	7	8	9	10

dp:

5	4	4	4	3	3	11	2	2	1	0
0	1	2	3	4	5	6	7	8	9	10



Q. Minimum steps to minimize n as per given condition.

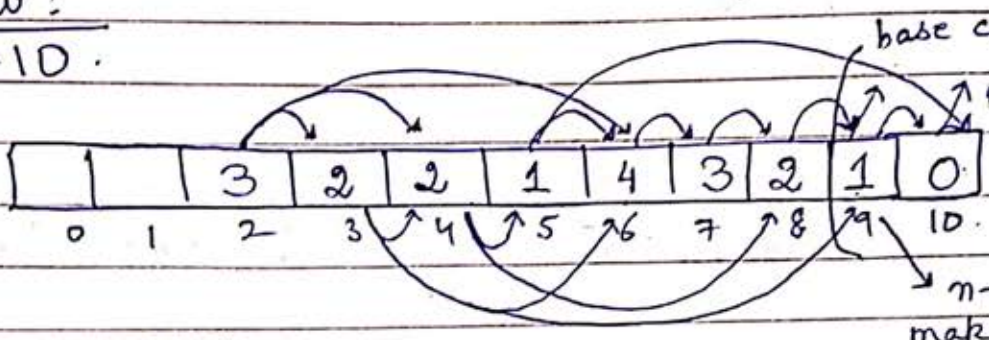
What?

Return minimum steps to convert n → 1 such that you can:

1) Divide n by 2.
2) Divide n by 3
3) Decrement n by 1.

How?

n = 10.



For each box

dp[i], dp[i] = min of dp[i+1]+1,
dp[2*i]+1 // if 2*i ≤ n.
dp[3*i]+1 // if 3*i ≤ n.

Q. Minimum Operations.

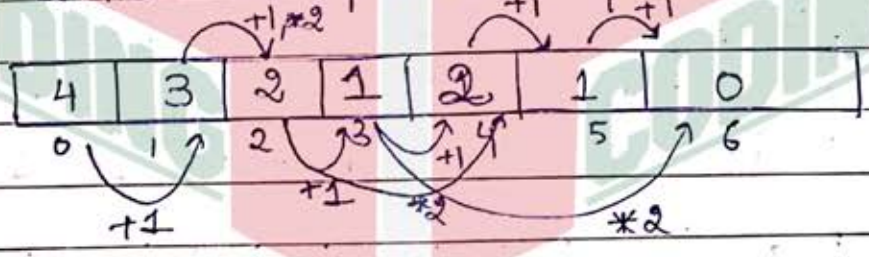
What?

Tell minimum operations to convert N → 0 given: (or 0 → N)

1) Double number
2) Add one to number.

How? ex: n = 6.

Same as previous question.



∴ 4 operations needed.

0 +1 1 +1 2 +1 3 *2 4.

1 2 3 4

Page No. _____
Date

Q. Climb stairs

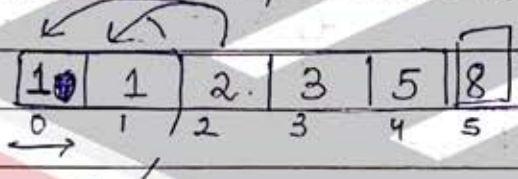
What?

Given n that is no. of stairs and that you are allowed to take only two steps or one step at a time, count ways to reach the nth stair.

How?

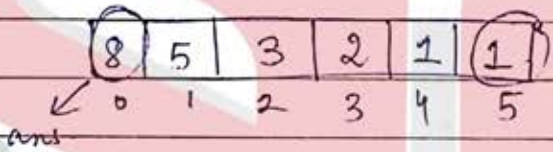
ex: $n = 5$.

For each stair, make 2 jumps at $n-1$ & $n-2$.



$dp[i] = dp[i-1] + dp[i-2]$.

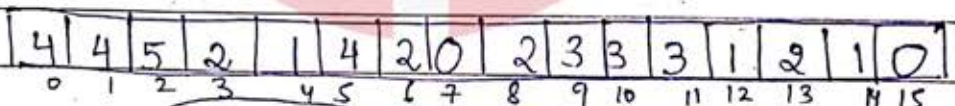
You can also be building it from right to left.



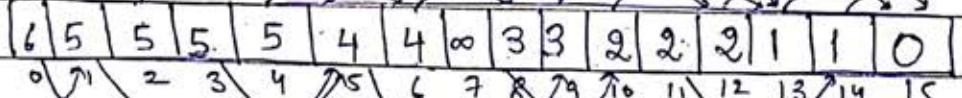
⇒ ways :- base case is for jumps it is 0.

Q. #Variation: You are given an array where you are given the number of steps you can more from each box. Return number of ~~ways~~ jumps & paths?

How?

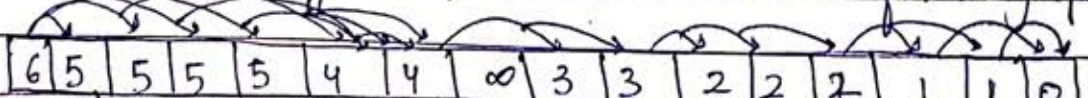


Jumps:



every ith box choses min value out of all places it can go to, and adds +1 for its jump.

Paths:



Reverse engineering

Egg Drop

Q What?

You are given number of floors and number of eggs in which you need to tell which floors are safe to drop egg from, and those which are not.
Return minimum number of ~~eggs~~ trials required to tell the critical floor, the one after which egg breaks.

How?

ex: floors = 10, eggs = 2

eggs ↑ / floors →	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	1	2	2	3	3	3	4	4	4	4

for a given [i,j]

dp[i][j] is filled as:

```
if (i==0 || j==0)
    dp[i][j] = 0
else if (j==1)
    dp[i][j] = 1
else if (i==1)
    dp[i][j] = j
else
    l = 0, r = j-1
    min = INT_MAX
    while (l < r)
        min = Math.min(min, Math.max(dp[i-1][l], dp[i][r]))
        l++
        r--
    dp[i][j] = min + 1
```

ex: dp[2][7] = 1 + min (
max(dp[1][0], dp[2][6]),
max(dp[1][1], dp[2][5]),
max(dp[1][2], dp[2][4]),
max(dp[1][3], dp[2][3]),
max(dp[1][4], dp[2][2]),
max(dp[1][5], dp[2][1]),
max(dp[1][6], dp[2][0]))

= 1 + min(3, 3, 3, 3, 4, 5, 6)
= 3 + 1 = 4

At a given floor, we compute its dp keeping in mind worst luck, i.e. maximum of egg bleak and save probabilities from all floors floor to current floor.

$$\min\left[\max\left(\frac{ms0}{mb\,j-1}\right),\ \max\left(\frac{ms1}{mb\,j-2}\right),\ \ldots,\ \max\left(\frac{msj-1}{mb0}\right)\right]$$

$$\left(\frac{f}{e}\right)_{at\,j}$$

Q. Optimal strategy for a game.

What?

Consider row of n coins with even n. We play a game against an opponent who is equally intelligent by keeping alternating turns. In each turn, a player either selects first or last coin and add its score into his score. He then removes it. Remove maximum amount of money you can definitely win if we move first.

How?

ex: {1, 4, 8, 15, 3, 7, 22, 11}

Using greedy approach for this question:

→ We consider odd sum and even sum for this strategy. At each level, if oddsum > even sum, choose an odd element (index wise), else choose even indexed element.

→ If at a given time either only odds or only evens are visible, choose maximum of them.

// This approach fails in the above testcase.

Use DP approach

* Move in gap strategy.
* For each dp box, $dp[i][j] = max(v1, v2)$
  where $v1 = min(dp[i+2][j], dp[i+1][j-1]) + a[i]$
  and $v2 = min(dp[i][j-2], dp[i+1][j-1]) + a[j]$

here, $dp[i+2][j]$ considering P1 & P2 both chose 2 front elements.

$dp[i+1][j-1]$ considering one of them chose first one and other chose last one.

$dp[i][j-2]$ considering both P1 & P2 chose elements from last.

	0	1	2	3	4	5	6	7	
0	1	4	9	19	12	26	34	37	→ ans
1	0	4	8	19	19	18	41	37	
2	0	0	8	15	11	22	33	33	
3	0	0	0	15	15	18	29	33	
4	0	0	0	0	3	7	25	25	
5	0	0	0	0	0	7	22	18	
6	0	0	0	0	0	0	22	22	
7	0	0	0	0	0	0	0	11	

Page No.
Date

Q. Encodings

What?

You will be given a number where encodings can be done as 1 → A, 2 → B, ..., 10 → J, 11 → K, ..., 26 → Z. Other than this, all numbers are invalid. Return maximum strings you can form after encoding the number, return all lowercase strings.

ex. 1 2 4 3 1 0 2 1

How? Consider multiple cases:

1) x 1 2 y.
   2 can be used as 2, 12.

2) x 3 2 y.
   2 can only be used as 2, 32 is invalid.

3) x 0 2 y.
   2 can only be used as 02 else 0 would make string invalid.

4) x 2 0 y.
   0 can only be used as 20, else 0 alone is invalid.

5) x 3 0 y.
   0 makes it invalid as neither 30, nor 0 are valid.

arr →	1	2	4	3	1	0	2	1	→ answer
dp →	1	2	3	3	3	3	3	6	
	a	ab l	ax abd ld	axc abdc ldc	axca abdca ldca	axcj abdcj ldcj	axcjb abdcjb ldcjb	axcjba abdcjba ldcjba axcju abdcju ldcju	

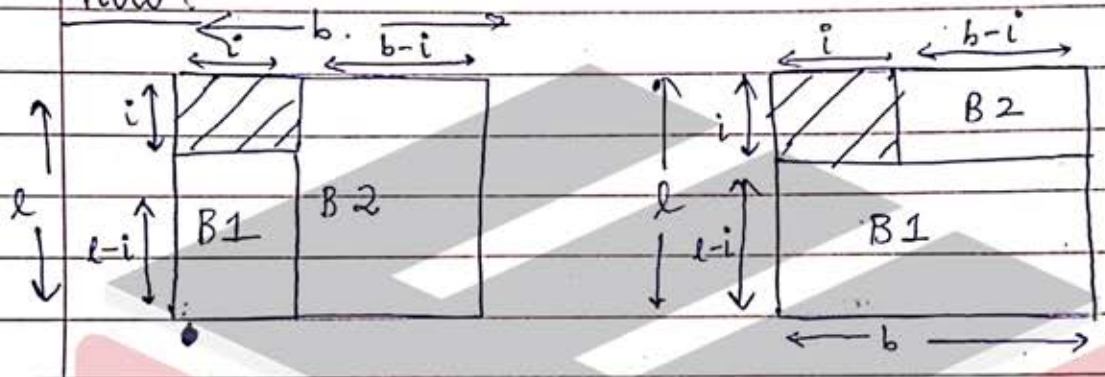
Page No. _____
Date

8. Minimum Number of Squares.

What?

Given a paper of size A X B, return minimum number of squares that can be cut from paper.

How?



ex: L = 4, B = 5.

B+1 →

L+1 ↓	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	2	3	4	5
2	0	2	0	3	2	4
3	0	3	3	0	4	4
4	0	4	2	4	0	**5**

← Memoized

→ ans.

For calculating, we use memoization and gather 2 results: fn → recursive function.

```
lc = 1 + fn(l-i, i) + fn(l, b-i)
rc = 1 + fn(l-i, b) + fn(i, b-i)
your ans = min(lc, rc).
// dp[l][b] = ans.
return ans;
```

Other DP questions

Q. Temple Offerings.

What?

You are given an array of heights of various temples, you need to return the minimum number of garlands you'll need such that a temple whose height is more than its 2 adjacent temples is offered more garlands than them.

* minimum 1 to each temple.

How?

ex: arr = { 1, 4, 3, 6, 2, 1 }
(0, 1, 2, 3, 4, 5)

left → | 1 | 2 | 1 | 2 | 1 | 1 |
(0, 1, 2, 3, 4, 5)

i = 0 to n-1 * Keep arr[0] = 1.

If arr[i] < arr[i+1]

arr[i] = 1 // or maintain count or arr[i+1] = 1 + arr[i].

// and then i+1 gets more than i.

* Keep arr[n-1] = 1.

| 1 | 2 | 1 | 3 | 2 | 1 | ← right
(0, 1, 2, 3, 4, 5)

ans[i] = max(left[i], right[i]);

ans ⇒ | 1 | 2 | 1 | 3 | 2 | 1 | ※
(0, 1, 2, 3, 4, 5)

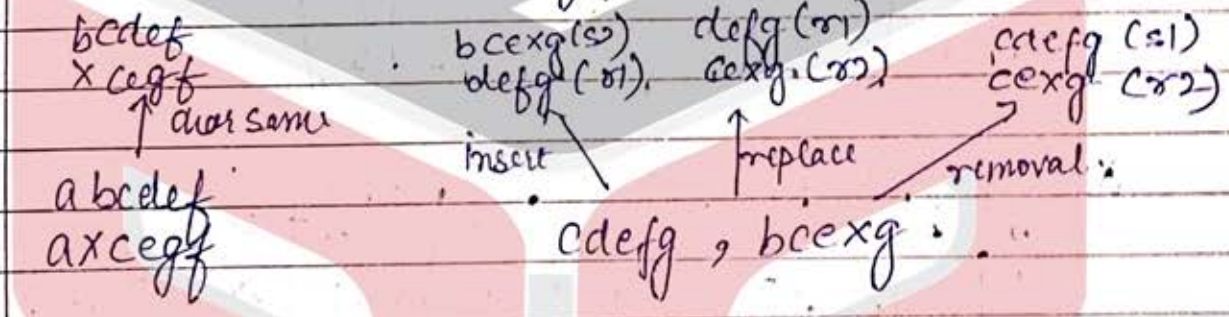
EDIT DISTANCE

What?

Count minimum operations needed to be done on string 2 to make it same as string 1. operations are insert (ie insert a character) delete (delete a char), replace (change value of char)

How?

eg: s1 = abcdef
s2 = axcegf



	a	x	c	e	g	f	-
a	3	4	4	5	5	5	6
b	4	3	3	4	4	4	5
c	4	3	2	3	3	3	4
d	4	3	2	2	2	2	3
e	4	3	2	1	1	1	2
f	5	4	3	2	1	0	1
-	6	5	4	3	2	1	0

insert, ~~replace~~ remove, ~~replace~~ replace

Addition (s2 blank but s1 has characters)

removals (s1 blank but s2 has character)

$c1 == c2 \Rightarrow dp[r1][r2] \Rightarrow dp[i+1][j+1]$

$c1 != c2 \Rightarrow$
- $1 + dp[i+1][j]$ (insert)
- $1 + dp[i+1][j+1]$ (replace)
- $1 + dp[i][j+1]$ (remove)

Maximum 1's Square

What?

Given a 2-D matrix of 0's and 1's find the maximum area square in this matrix

How?

eg:

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

DP:

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

→ when zero So zero as it is.

$dp[i][j] = \min(dp[i+1][j+1], dp[i][j+1], dp[i+1][j]) + 1$

* Every box contains length of square starting from this point

Related question.

Q. Find maximum area rectangle of all 1's of a given matrix.

→ Solve using maximum area histogram → stacks.

Page No.
Date

Q Mobile Numeric Keypad

What?

You are given a numeric keypad and N length find the numbers possible that can be made such that you can only press up, left, down, right to the current button or button itself.

How?

Make an 2 D array to store what keys to press from current key.

1	2	3
4	5	6
7	8	9
*	0	#

Keypad.

0 → 0, 8 , 1 → 2, 4, 1 , 2 → 1, 2, 3, 5 3 → 2, 3, 6
4 → 1, 4, 5, 7 , 5 → 2, 4, 5, 6, 8 , 6 → 3, 5, 6, 9
7 → 4, 7, 8 , 8 → 5, 7, 8, 9, 0 , 9 → 6, 8, 9.

eg: N = 2

	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1	1
1	2	3	4	3	4	5	4	3	5	3
2	7	11	15	11	15	22	15	12	18	12

Summation.

Ans = 138

All possible keys that can be pressed.

dp[2] + d[3] + dp[6] (Previous row)
= 4 + 3 + 4 = 11

Page No. _____
Date | | | |

Rod Cutting

What?

Find the maximum cost that can be obtained by cutting a rod such that each length piece has its own selling price.

How?

eg: len = 8

0	1	2	3	4	5	6	7
3	5	6	15	10	25	12	24
1	2	3	4	5	6	7	8

Each piece of length can be cut in pieces of 1 to i lengths.

$$dp[i] = \max(dp[j] + cost[j-1])$$

(Because of 0 indexing)

dp:

0	1	2	3	4	5	6	7	8
0	3	6	9	15	18	25	28	31

Ans = 31

length of pieces cut.

at 4 ⇒ → 4-1 = 3 cur cost = dp[3] + cost[2]
= 9 + 6 = 15

→ 4-2 = 2 ⇒ dp[2] + cost[1] = 6 + 5 = 11

→ 4-3 = 1 dp[1] + cost[0] = 3 + 3 = 6

→ 4-4 = 0 dp[0] + cost[3] = ~~15~~ 15

↳ whole piece no cut

Caps

What?

Given n persons and K caps, so count the total number of arrangement of ways such that none of them are wearing the same type of cap.

How?

→ A dp of 2-dimension of size $K+1 \times 2^n$ is made where 2^n columns stores the count for that particular bitmask.

eg: Person caps

0 → 1,3
1 → 1,2,3
2 → 0,2

Caps Map

cap → person (which can be worn by which person)

0 → 2
1 → 0,1
2 → 1,2
3 → 0 → 1

← 3 (3 people possible bit mask)

Ans = 6

caps	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)	
0	6	5	4	2	4	3	2	1	2
1	2	2	2	1	4	3	2	1	0,1
2	0	1	1	1	1	2	1	1	1,2
3	0	0	0	0	0	1	1	1	0,1
4 (No cap)	0	0	0	0	0	0	0	0	

111 → person2, person1, person0

For each position check if that person can wear current cap or not by checking the caps map.

Page No.
Date

For cap 2 (row with index 2) (1 & 2 person can wear it)

index 7 (111) → since both 1st & 2nd bit set so next row value

idx 6 (110) → next row value as it, (no body wore this cap)

idx 5 (101) → dp[i][j] = dp[[illegible]][7] + dp[3][5] (101 → 11) = 2

idx 4 (100) → dp[3][6] + dp[3][4] = 2

idx 3 (011) → dp[3][7] + dp[3][3] = 1

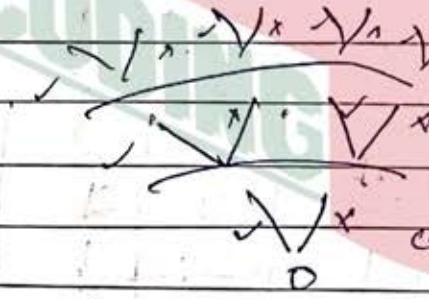
idx 2 (010) → dp[3][6] + dp[3][2] = 1

idx 1 (001) → dp[3][5] + dp[3][1] + dp[3][3] = 1

idx 0 (000) → dp[3][4] + dp[3][2] + dp[3][0] = 0

We are using bitmask here to save space. ~~and~~ through bitmask we can store possible arrangements (2^{32}) for 32 person.

This is recursion with caps on level. where each cap has choice To be wore or not.



Date 25
DELTA Pg No.

Q Probability that Knight remain on chessboard.

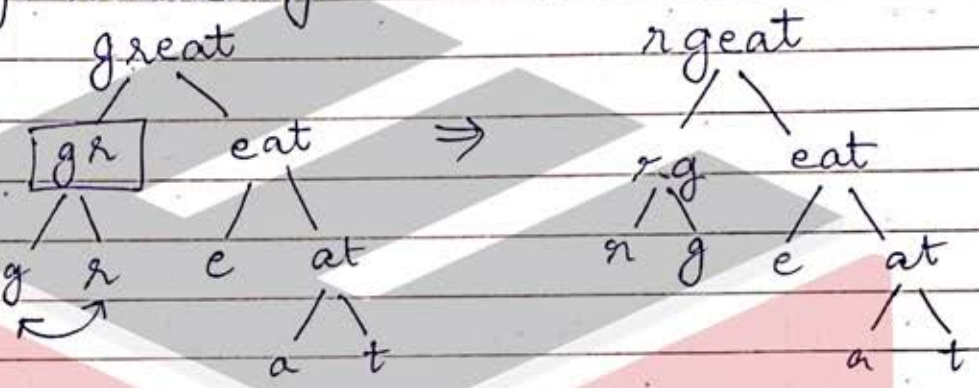
3D DP Questions

Q. Scramble Strings

What?

Given a string s1, we may represent it as a binary tree by partitioning it to two non-empty substrings recursively.

ex:



To scramble a string, we may choose node "gr" and swap its two children, only non leaf nodes can be chosen for swapping its children. Many scramble strings of a word are possible; now you are given two strings s1 and s2, determine if s2 is a scrambled string of s1.

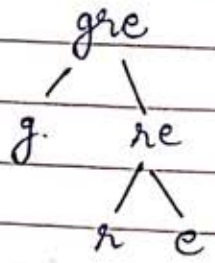
How?

This question is based on using a 3D-dp. Before proceeding to form a dp, we check that if lengths of two strings are not same, then s2 can never be a scramble of s1.

Consider a simple case "gre".

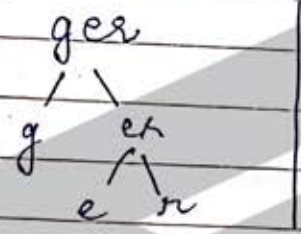
The splits can be done as "g" and "re" or "gr" and "e". So a scrambled string of "gre" can be "ger", "rge", "egr", "reg", "erg" or itself "gre".

gre:

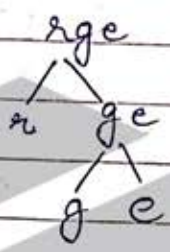


scrambles:

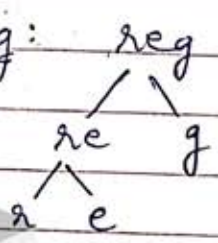
ger:



rge:



reg:



and so on...

In DP, 3 dimensions i, j, k signify i as the index of string 1, j as the index of string 2 and k as the length of the substring being considered.

At k=1, dp looks like:

	e	a	t	r	g
g	x	x	x	x	✓
r	x	x	x	✓	x
e	✓	x	x	x	x
a	x	✓	x	x	x
t	x	x	✓	x	x

Let dp[i][j][k] be whether the substring in s1[i...i+k-1] is a scramble of s2[j...j+k-1] or not.

Since each of these substrings is a potential node in the tree, we need to check for all possible cuts.

For all cuts q where q < k, we have following:

s1[x1 | x2]

i, i+q, i+k-1

which means two possibilities:

s2 [y1 | y2]
j j+q j+k-1

or

s2 [y1 | y2]
j j+k-q j+k-1

where calls are :

dp(i, j, k) = for k = 1,
{ s1.charAt(i) == s2.charAt(j) }
else → for 1 ≤ q ≤ k,
{ ((dp[i][j][q] && dp[i+q][j+q][k-q])
|| (dp[i][j+k-q][q] && dp[i+q][j][k-q]))
}

for 2 words s1 = "great", s2 = "eatrg", dp is :-

k = 1

	0	1	2	3	4
0	X	X	X	X	✓
1	X	X	X	✓	X
2	✓	X	X	X	X
3	X	✓	X	X	X
4	X	X	✓	X	X

k = 2

	0	1	2	3	4
0	X	X	X	✓	X
1	X	X	X	X	X
2	✓	X	X	X	X
3	X	✓	X	X	X
4	X	X	X	X	X

k = 3

	0	1	2	3	4
0	X	X	X	X	X
1	X	X	X	X	X
2	✓	X	X	X	X
3	X	X	X	X	X
4	X	X	X	X	X

k = 4

	0	1	2	3	4
0	X	X	X	X	X
1	✓	X	X	X	X
2	X	X	X	X	X
3	X	X	X	X	X
4	X	X	X	X	X

answer

k = 5

	0	1	2	3	4
0	✓	X	X	X	X
1	X	X	X	X	X
2	X	X	X	X	X
3	X	X	X	X	X
4	X	X	X	X	X

Date
DELTA Pg No.

Q. Cherry Pickup

What?

In an N x N grid representing field of cherries, each cell have 3 possibilities, 0 → empty cell, 1 → cell has a cherry, -1 → obstacle. You can pass through 0 and 1 valued cells. Your task is to collect maximum number of cherries possible by following rules:

a) You can move right / down till (N-1, N-1) cell from (0,0).
b) When you pass through a cherry (1) box, you pick it up and it becomes 0.
c) If there is no valid path between (0,0) to (N-1, N-1), then no cherries can be collected.
d) You can move back to (0,0) once you reach (N-1, N-1) by moving up or left.

How?

Instead of moving from (0,0) to (N-1, N-1) and then back to (0,0), we assume we have 2 people moving from (0,0) to (N-1, N-1). This is because we need 2 max paths that give max cherries, and hence direction does not matter.

We will do this question by using a 3D memoization dp, where for both people, we have 4 possible cases:

a) DD : r1+1, c1, r2+1
b) RR : r1, c1+1, r2
c) DR : r1+1, c1, r2
d) RD : r1, c1+1, r2+1

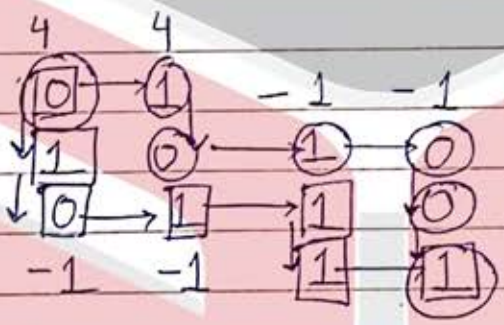
We use only 3 parameters as c2 can be calculated as $c2 = r1 + c1 - r2$.

→ Initialize dp[ ][ ][ ] with Integer.MIN_VALUE.
For each (r1, c1, r2), we fill dp as:

```
dp[r1][c1][r2] = grid[r1][c1] + ((c1 != c2) ?
                                 grid[r2][c2] : 0);
// add grid[r2][c2] only when c1 != c2.
```

Make 4 calls of DD, RR, RD, DR as mentioned and add max of these values into dp[r1][c1][r2].

ex:



Boxes show path by person 1, circles show path by person 2.

Person 1: $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 1 \rightarrow 1 = 5$.

Person 2: $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow \boxed{1} = 2$.

this cherry not counted again.

∴ Total cherries = 7 (5+2)

Date 28
DELTA Pg No.

Q Probability that Knight remains on chessboard.

What?
Given a N×N chessboard and a Knight Position (x,y). The Knight has to take exactly K steps where at each step it chooses from all 8 directions it can move in. Find the probability that Knight remains on the chess board after K steps. It can't enter board again after leaving.

How?
Make a 3-D DP of size K+1 × N × N
Where K is steps that Knight can take

DP stores the probability of remaining on board when k steps are taken and Knight is in i,j th box

eg: K=3 N=4 X=0 Y=0

	0	1	2	3
0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0

K=0

(When K=0 means 0 steps so staying in the board probability 1.0)

K=1

	0	1	2	3
0	0.25	0.375	0.375	0.25
1	0.375	0.5	0.5	0.375
2	0.375	0.5	0.5	0.375
3	0.25	0375	0.375	0.25

$\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8} = \frac{4}{8} = 0.5$

For this position possible position are marked

These positions in previous board (K-1) will give probabilities when K-1 step was taken

Prob/8 ← current Prob. will be sum of all

PEPCODING - PURSUIT OF EXCELLENCE AND PEACE

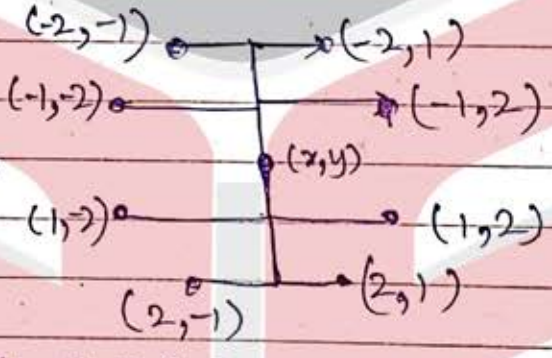
Date
DELTA Pg No.

	0	1	2	3	
0	0.125	0.15625	0.15625	0.125	
1	0.15625	0.15625	0.15625	0.15625	K=2
2	0.15625	0.15625	0.15625	0.15625	
3	0.125	0.15625	0.15625	0.125	

	0	1	2	3
0	0.390625	0.05859375	0.05859375	0.0390625
1	0.5859375	0.0703125	0.0703125	0.05859375
2	0.5859375	0.0703125	0.0703125	0.05859375
3	0.390625	0.05859375	0.05859375	0.0390625

K = 3

Ans = dp[K][x][y] = dp[3][0][0]
= 0.390625 Ans



directions of Knight

new x = dir[r][0] + x
new y = dir[r][1] + y

steps

Pseudo Code

```
k=0 to k=K
    i=0 to n
        j=0 to n
            if k==0   dp[k][i][j] = 1.0;
            else
                r=0 to 8 (directions)
                nx = dir[r][0] + i;
                ny = dir[r][1] + j;
                if (nx & ny valid)
```

dp[k-1]

dp[k][x][y] += dp[k-1][nx][ny]/8.0

Ans dp[K][X][Y]

We are doing probability sum by 8 because out of 8 possible direction only few steps some directions (possibly all) lie inside board. So they are only included.

Q Longest Common Subsequences of 3

What?
Given 3 strings S1, S2 & S3 find the longest common subsequence b/w them.

How?
Same approach as longest Common Subsequence of 2 strings but here for 3 strings.

Make a 3-D DP which 3 strings at its 3 dimensions

dp size is S1.length+1 × S2.length+1 × S3.length+1

*When a character is same in all strings S1, S2 & S3 it means. for string S1 remaining string r1
for string S2 remaining string r2
for string S3 remaining string r3

dp[i][j][K] = dp[K+1][j+1][K+1] + 1
↳ as a common character found.

In other case when all three character are not equal. 3 cases will cover all cases.

1) s1 s2 r3 → string 3 loses a character
2) s1 r2 s3 → string 2 loses a character
3) r1 s2 s3 → string 1 loses a character

eg: s1 = abcdf
s2 = aebdf
s3 = abdcef

	a	b	d	c	e	f	-
a	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0
-	0	0	0	0	0	0	0

K=5 (K=s1.length)

s1 = '-' Blank.

	a	b	d	c	e	f	-
a	1	1	1	1	1	1	0
e	1	1	1	1	1	1	0
b	1	1	1	1	1	1	0
d	1	1	1	1	1	1	0
f	1	1	1	1	1	1	0
-	0	0	0	0	0	0	0

K=4

s1 = 'f-'
K=

s1.char(i) = s3.char(j) = s1.char(k)

	a	b	d	c	e	f	-
a	2	2	2	1	1	1	0
e	2	2	2	1	1	1	0
b	2	2	2	1	1	1	0
d	2	2	2	1	1	1	0
f	1	1	1	1	1	1	0
-	0	0	0	0	0	0	0

K=3

s1 = 'df-'

	a	b	d	c	e	f	-
a	2	2	2	1	1	1	0
e	2	2	2	1	1	1	0
b	2	2	2	1	1	1	0
d	2	2	2	1	1	1	0
f	1	1	1	1	1	1	0
-	0	0	0	0	0	0	0

s1 = 'cdf' / K=2

a b d c e f -

	a	b	d	c	e	f	-
a	3	3	2	1	1	1	0
e	3	3	2	1	1	1	0
b	3	3	2	1	1	1	0
d	2	2	2	1	1	1	0
f	1	1	1	1	1	1	0
-	0	0	0	0	0	0	0

k=1

S1 = 'bcdf'

	a	b	d	c	e	f	-
a	4	3	2	1	1	1	0
e	3	3	2	1	1	1	0
b	3	3 → 2	1	1	1	0	
d	2	2	2	1	1	1	0
f	1	1	1	1	1	1	0
-	0	0	0	0	0	0	0

k=0

Here a = a = a

S1.char(k) = S2.char(i) = S3.char(j)

1 + dp[k+1][i+1][j+1]

S1 = 'abcdf'

Ans = 4

abdf

```
k = s1.length to 0
    i = s2.length to 0
        j = s3.length to 0

            if s1.char(k) = s2.char(i) = s3.char(j)
                dp[k][i][j] = dp[k+1][i+1][j+1]

            else
                dp[k][i][j] = max( dp[k+1][i][j],
                                   dp[k][i+1][j],
                                   dp[k][i][j+1] )
```