DP Famous Problems

~~~~~~~~

Agenda

~~~~~~~~

1. Longest Increasing Subsequence
2. Russian Doll Envelopes
3. Count of palindromic substrings
4. Palindromic Partition

Hello Everyone
Very Special Good Evening
to all of you 😊😊😊
We will start session
from 9:06 PM

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Longest Increasing Subsequence (LIS)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

arr: [6, 9, 10, 13, 20]   ans: 5

arr: [13, 6, 2, 1]   ans: 1

arr: [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]

0, 8, 12, 14, 15 → length = 5
0, 2, 6, 9, 13, 15 ⇒ length = 6
0, 4, 6, 9, 13, 15 ⟶ length = 6

} longest increasing subsq. = len = 6 Ans

Bruteforce:
* Consider all subseq.
* Check if current subsq. is incr in nature
* if it is, then maximise length

T.C: $O(2^n * n)$

Optimise Approach:

arr[]: [10, 3, 12, 7, 9, 11, 20, 11, 13, 6, 8]

arr[]	10	3	12	7	9	11	20	11	13	6	8
index	0	1	2	3	4	5	6	7	8	9	10
dp →	1	1	2	2	3	4	5	4	5	2	3
	10	3	[10, 12] [3, 12]	[3, 7]	[3, 7, 9]	[3, 7, 9, 11]	[3, 7, 9, 11, 20]	[3, 7, 9, 11]	[3, 7, 9, 11, 13]	[3, 6]	[3, 6, 8]

dp[i] → length of longest increasing subseq, which is ending at index = i.

ans = max of DP → 5 Ans

code #

```
dp[n];
dp[0] = 1,   ans = 1;
for(int i=1 ; i<n;  i++){
    |     max=0;
    |     for( int j=0; j<i; j++){
    |     |     if( arr[j] < arr[i]){
    |     |     |     max = math.max(max, dp[j]);
    |     |     }
    |     }
    |     dp[i]= max+1;
    |     ans= Math.max(ans, dp[i]);
}
return ans;
```

Problem: longest increasing subarray?

[1 2 3 7 | 5 | 3 11 25 30 40 | 20]

$\xrightarrow[4]{}$ $\qquad$ $\longrightarrow$ $\qquad$ $\longrightarrow$

TODO:

T.C: O(n)
S.C: O(1)

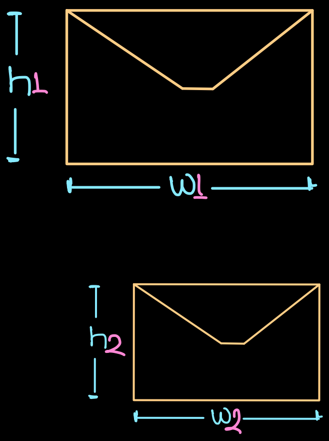
~~~~~~~~~~~~~~~~~~~~~~~~~~

Russian Doll Envelopes

~~~~~~~~~~~~~~~~~~~~~~~~~~

N- Different Envelopes find max. count of envelopes that can be put in a single envelope.

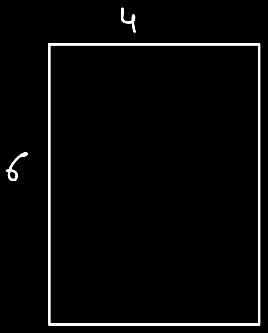
# Rotation of envelope is not allowed.

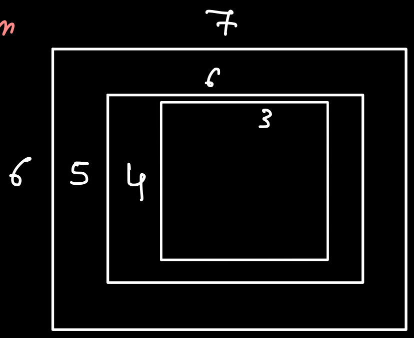


$h_2 < h_1$
$w_2 < w_1$

env.	Height	width	⇒ area = h × w
A	5	6	→ 30 — II ✓
B	6	4	→ 24 — III ✓
C	6	7	→ 42 — I ✓
D	4	3	→ 12 — IV

NOTE: Area can't be a correct factor for arrangement becoz it's combination of h×w





h → [9 5 10 3 4 2]

w → [3 4 8 2 3 7]

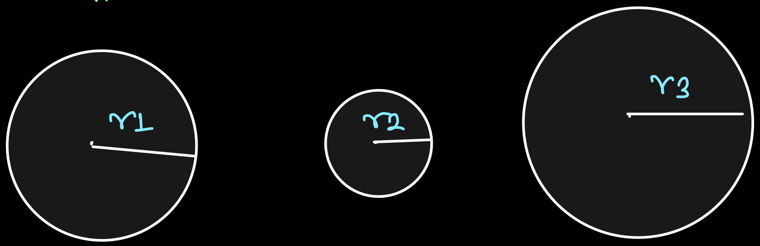
Sorting :
- * only ht → we will lose the data of correct envelope.
- * only wd → "
- * Area → ✗
- * ht/wd → ✗
  division

array of radius: [5, 6, 7, 3, 9, 4, 9]

→ Sort the array

[3, 4, 5, 6, 7, 9, 9]

→ 3, 4, 5, 6, 7, 9



r3 > r1 > r2 → we can place one circle on another with crossify each othe

If single dimension is there then sorting works.

But if we have 2-dimension available.

→ Sort one dimension and apply LIS on second dimension

while sorting one dimension, make a pair of (height, width)

h→ [9 5 10 3 4 2]

w→ [3 4 8 2 3 7]

Sort on the basis of ht →

ht→[2 3 4 4 5 9 10]

wd→[7 2 3 7 4 3 8]

→ LIS on width, make sure that high (j) is also smaller.

```
#code:   1. Sort on the basis of ht

         2. dp[0]=1,   ans=1;
            for(int i=1; i<n; i++){
                max=0;
                for(int j=0; j<i; j++){
                    if( wd[j] < wd[i]  && ht[j]< ht[i]){
                        max= Math.max(max, dp[i]);
                    }
                }
                dp[i]= max +;
                ans= Math.max(ans, dp[i]);
            }
         return ans;
```

T.C: $O(n^2)$

S.C: $O(n)$

10:13- 10:25pm Break

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Count of palindromic substrings

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Given a String, for every substring check if it is palindromic or not

String Str = "abac"
0 1 2 3

if length = n

count of substring = $\frac{n(n+1)}{2}$

i.e. $= \frac{4 * 5}{2} = 10$

a   ab   aba   abac
b   ba   bac
a   ac
c

ei

si	0	1	2	3
0	T	F	T	F
1	×	T	F	F
2	×	×	T	F
3	×	×	×	T

expected o/p

Bruteforce:

Consider all of the substrings for every substring check if it is palindromic or not.

s=0, e=1 to n-1 → n-1
s=1, e=2 to n-1 → n-2
s=2, e=3 to n-1 → n-3
⋮
s=n-2, e=n-1 to n-1 → 1

total str = 1+2+3+..n-1

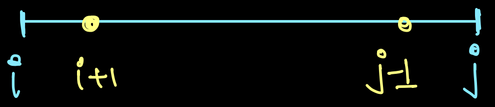
$= \frac{n(n-1)}{2}$

T.C: O(n^2) → for substring only.

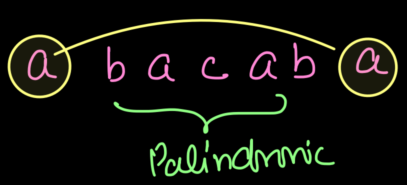
Now we have to check if that substring is palindromic or not

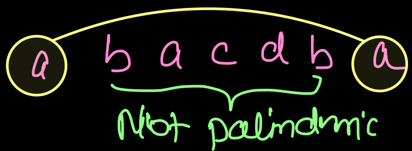
final T.C: O(n^3)
S.C: O(1)

optimised approach!



if(str[i] == str[j]) → answer depends on substring from i+1 to j-1





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

else {

// str[i] != str[j]

$dp[i][j] = false.$

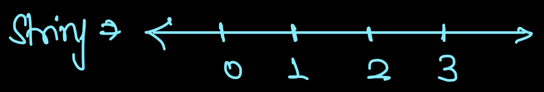
Str = abac

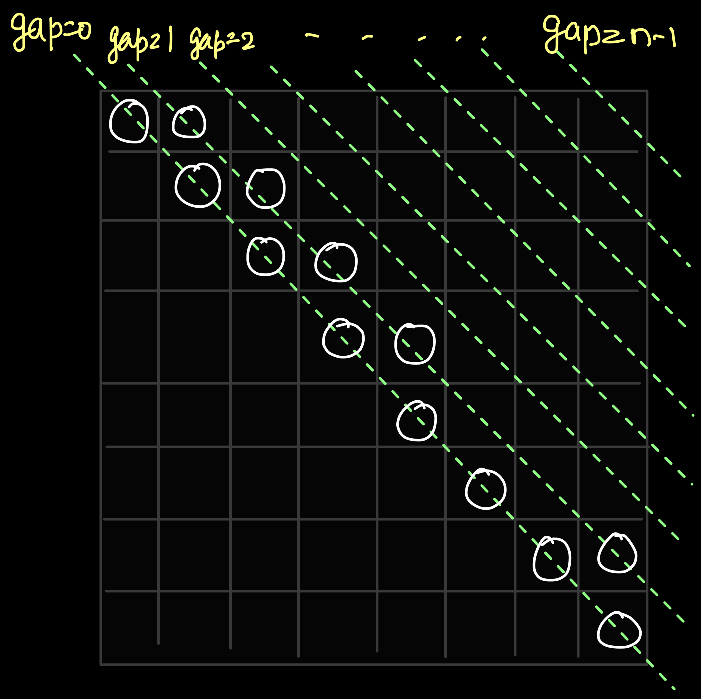
substring length = 2

Ch1 → Ch2

Same → T

	0	1	2	3
0	T	F	T	F
1		T	F	F
2			T	F
3				T





gap= 0	gap= 1	gap= 2	gap=3 .
0,0	0,1	0,2	0,3
1,1	1,2	⋮	⋮
2,2	2,3		
3,3	3,4		4,7
⋮	⋮	5,7	
7,7	6,7		

```
booleam [N][N];

for(int gap=0;   gap < n;   gap++) {
    for(int i=0, j=gap;   j<n;   i++ , j++) {
        if( gap==0) {  dp[i][j]= true; }
        else if( gap==1 ) {  dp[i][j]= str[i] == str[j]; }
        else {
            if( str[i] == str[j] ) {  dp[i][j]= dp[i+1][j-1]; }
            else {  dp[i][j] = false }
        }
    }
}
return dp;
```

T.C: $O(n^2)$

S.C: $O(n^2)$ → for ourselve

: $O(1)$ → if problem demands for that array.

---

count of all palindromic substrings → no. of 'trues' are count of palindromic substring.

longest palindromic substring → Iterate from gap= n-1 to gap=0

→ once found any true return gap+1.

Palindromic Partition

find the min no. of cuts to partition the strings such that all partitions are palindromic.

Eg: Str ⇒ $x\ x \mid y$ → ans = 1

⇒ $x \mid a \mid b\ a\ a\ b \mid p$ → ans = 3

⇒ $x\ b\ b\ x \mid c$ → ans = 1

⇒ $a \mid b\ b \mid z\ y\ z$ → ans = 2

Greedy → select the longest palindromic substring first ✗ not work

$$c \mid \underbrace{b\ c\ a\ c\ b}_{\text{longest palindromic substring.}} \mid b \mid c$$

→ count ⇒ 3

$$c\ b\ c \mid a \mid c\ b\ b\ c$$

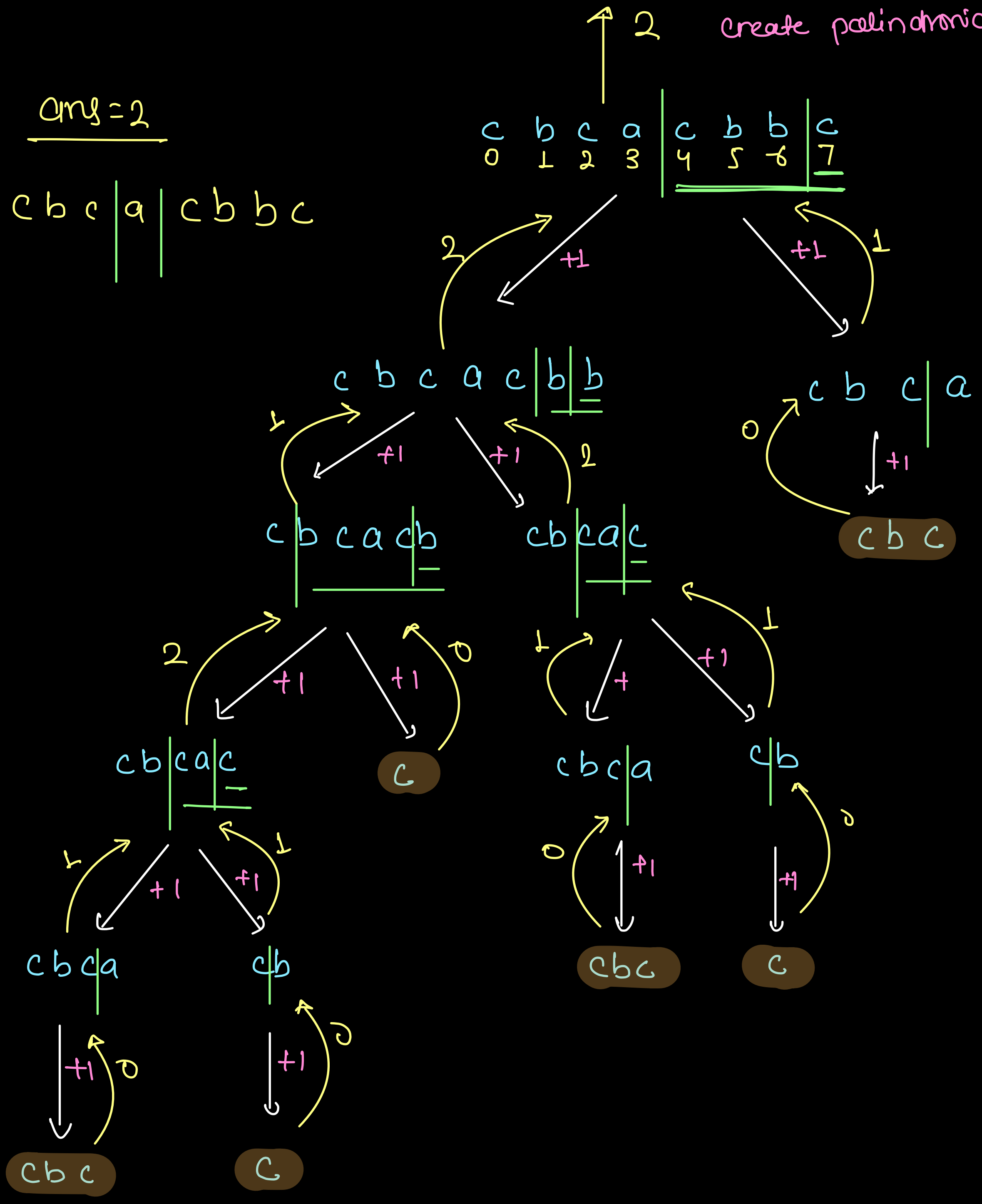
not selecting longest palindromic substring still partition is min.

ans = 2

create palindromic partition

ans = 2

c b c | a | c b b c



pseudocode:

```
dp[N]
int  minCuts( string s , int j ) {
    if( checkPalindromic ( str, 0, j)) {
        return 0;
    }
    if(dp[j] != -1) { return dp(j); }

    min → ∞
    for(int cut=j; cut>0; cut--) {
        if( checkPalindromic ( str, cut , j)) {
            min=Math.min(min, minCuts( str, cut -1));
        }
    }
    return dp[j]= min+1;
}
```

→ dp2[0][j]

Previous problem

check if any substring is palindromic or not → dp2[cut][j]

T.C: $O(n^2 + n^2)$

S.C: $O(n^2 + n)$

bottom up:

3 vs. 1

j ↓

	c	b	c	a	c	b	b	c
	0	1	2	3	4	5	6	7
dp →	0	1	0	1	2	1	2	2 → ans.

dp[i] ⇒ min no. of cuts required to partition the substring (0 to i) such that every partition is palindromic a → 1010

XOR of all cyclic permutat with 'a' & get the count of XOR result of '0'.

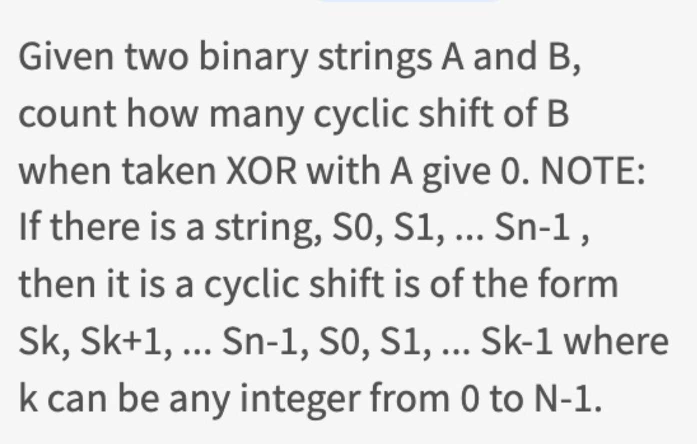
b → 0110

b+b = 2b left shift

0110 0110

0 1 2 3 4 5 6 7

To be discussed →



Given two binary strings A and B, count how many cyclic shift of B when taken XOR with A give 0. NOTE: If there is a string, S0, S1, ... Sn-1 , then it is a cyclic shift is of the form Sk, Sk+1, ... Sn-1, S0, S1, ... Sk-1 where k can be any integer from 0 to N-1.