→ L.I.S

→ Russian Doll Envelopes

→ Count of palindromic substrings.

→ Palindromic partition

Longest Increasing Subsequence

arr → [10 3 12 7 2 9 11 20 11 13 6 8]

[3, 7, 9, 11, 20] → 5

[3, 7, 9, 11, 13] → 5

idea. → Consider all increasing subsequences.

[Back-tracking]

⇓

T.C → $O(2^N)$

[lis[i] = length of longest increasing subsequence ending at idx - i]

arr →	10	3	12	7	2	9	11	20	11	13	6	8
	0	1	2	3	4	5	6	7	8	9	10	11
lis[N]	1	1	2	2	1	3	4	5	4	5	2	3
	"10"	"3"	"10,12" or "3,12"	"3,7"	"2"	"3,7,9"	"3,7,9,11"	"3,7,9,11,20"	↓ 3, 7, 9, 11	3,7,9,11,13	↓ 2, 6	2, 6, 8

#pseudo.code.

```
lis [N] ;   // initialise with 0
ans = 0

for( i = 0; i < N; i++) {
        max -> 0
        for( j = i-1 ; i >= 0 ; i--) {
            if ( arr[j] < arr[i]) {
                    max = Max(max , lis[j])
            }
        }
        lis[i] = max + 1
        ans = Max( ans, lis[i])
}
```

T.C -> $O(N^2)$

S.C -> $O(N)$

$O(\underline{N \log N})$

length. [P.S.S.]

```
main( — ){
        int ans = 1 , int dp[N] ;
        for( i = 0; i < N; i++) {
                ans = Max(ans , lis( arr[], i,dp)) ;
        }
        return ans.
}
```

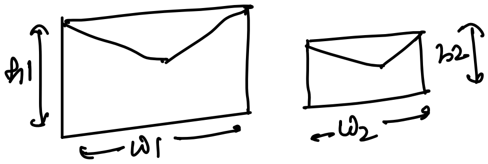
```
int  lis( int arr[ ], int i, int [] dp){
{
        if( dp[i] != -1) {return dp[i]}
        max = 0
         for( j = i-1 ; j >= 0 ; j--){
         {
                  if( arr[j] < arr[i]){
                  {
                         max = Max( max, lis( arr[], j))
                  }
         }
         dp[i] = max+1 ;
}
```

Russian Doll Envelopes-

N- different envelopes.

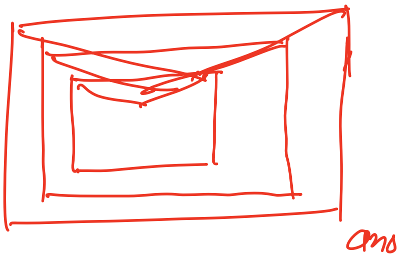
Find max count of envelopes that can be put in a single envelope.

Note→ Rotation of envelope is not allowed.



$$\begin{bmatrix} h_1 > h_2 \\ w_1 > w_2 \end{bmatrix}$$

	h.	w	
A →	5	6	✓
B →	6	4	
C →	6	7	✓
D →	4	3	✓



ans = 3

$$\begin{array}{ll} h \rightarrow & \begin{bmatrix} 9 & 5 & 10 & 3 & 4 & 2 \\ 3 & 4 & 8 & 2 & 3 & 7 \end{bmatrix} \\ & \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \\ w \rightarrow & \end{array}$$

~~Greedy~~ →

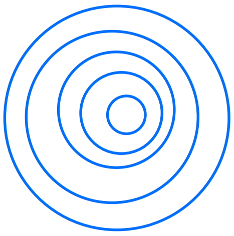
$$\begin{matrix} 10 & 9 & 3 \\ 8 & 3 & 2 \end{matrix} \quad \boxed{3}$$

$$\begin{matrix} 10 & 2 \\ 8 & 7 \end{matrix} \quad \boxed{2}$$

$$\begin{matrix} 10 & 5 & 4 & 3 \\ 8 & 4 & 3 & 2 \end{matrix}$$

[ans = 4]

x→ $\overset{✓}{5}$, $\overset{✓}{6}$, $\overset{\times}{7}$, $\overset{✓}{3}$, $\overset{✓}{9}$, $\overset{✓}{7}$, $\overset{\times}{5}$



Single dimension
greedy ✓

$$\begin{array}{ll} h \to & \multirow{2}{*}{$\begin{bmatrix} 10 & 3 & 4 & 9 \\ 8 & 2 & 3 & 2 \end{bmatrix}$} \\ w \to & \end{array}$$

$$\begin{array}{lcccc} \underline{\text{area}} \to & 80 & 6 & 12 & 18 \end{array}$$

Greedy on area ✗

Sort envelopes on the basis of height.

h →	1	2	3	4	4	5	7	10	10	12	15
w →	10	3	7	9	11	20	11	6	13	8	2

① Sort the arr accordly to height

② Apply L.I.S on width [h1 != h2]

T.C → O(N²)

S.C → O(N)

Q: Given a string. For every substring, check if that is a palindrome or not. (ans → 2D array)

s = a b c b
0 1 2 3

ei →

si ↓

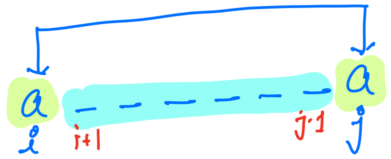
	0	1	2	3
0	t	f	f	f
1	✗	t	f	t
2	✗	✗	t	f
3	✗	✗	✗	t

ans.

Idea 1. * Consider all substrings & → $\frac{N(N+1)}{2}$
iterate & check if that substring is a palindrome. → N

$$\boxed{\begin{array}{l} T.C \rightarrow O(N^3) \\ S.C \rightarrow O(1) \end{array}}$$

Observation



gap = 0	gap = 1.	gap = 2	gap = 3
0, 0	0, 1	0, 2	0, 3
1, 1	1, 2	1, 3	
2, 2	2, 3		
3, 3			

```
boolean dp[N][N];

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

T.C → $O(N^2)$
S.C → $O(1)$

→ count of all palindromic substrings.

T.C → $O(N^2)$, S.C → $O(N^2)$

Q: Find min no. of cuts to partition the string such that all the partitions are palindrome.

Eg→ x x | y

ans = 1

x | a | b a a b | p   ans = 3.

a | b c b   ans = 1.

a | b c b | b

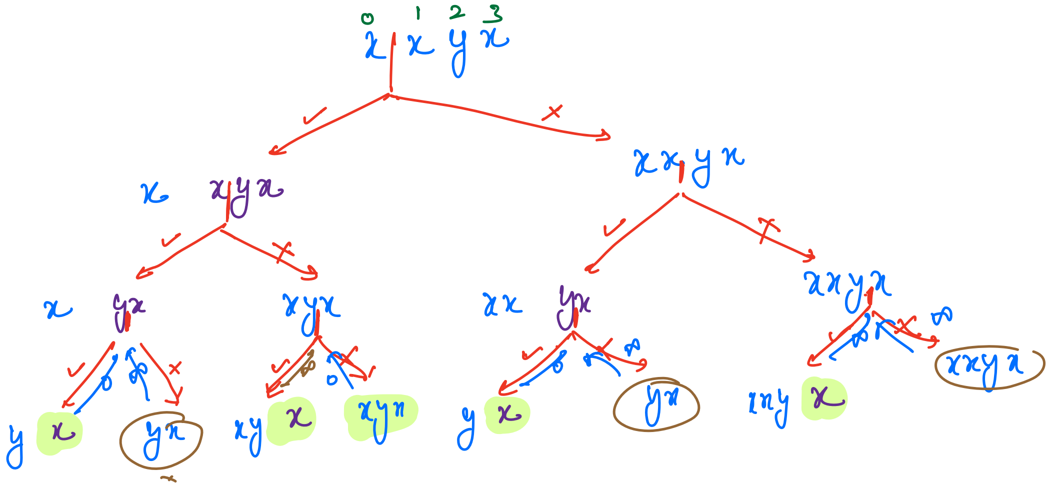
ans = 2

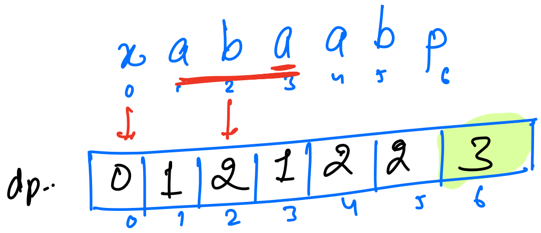
a b c d

✓ ✓ ✓

✗ ✗ ✗

ways = $2^{n-1} = 2^3 = 8$.





dp(i) → min cuts required for string [0, i]

→ make the cut only when you are getting palindrome.

pseudo-code.

```
dp[N], P[][]; → // for all substrings if they are palindrome or not.
dp[0] = 0    //edge case.

for( j = 1 ; j < N ; j++) {
        if( P[0][j] == true){ dp[j] = 0 }
        else {
            min → ∞
            for( i = 1 ; i <= j ; i++){
                if( P[i][j] == true){
                    min = Min(min, dp[i-1])
                }
            }
            dp[j] = min + 1
        }
}
dp[N-1];
```

$T.C \rightarrow O(N^2)$
$S.C \rightarrow O(N^2)$

→ [trees, l.l, hashmap, sorting & searching,]