LIS [Longest Increasing Subsequence]

Q Given an array, find the length of LIS! → Strictly increasing!

A: [2, 3, 5, 1, 10, 3, 7] → 4

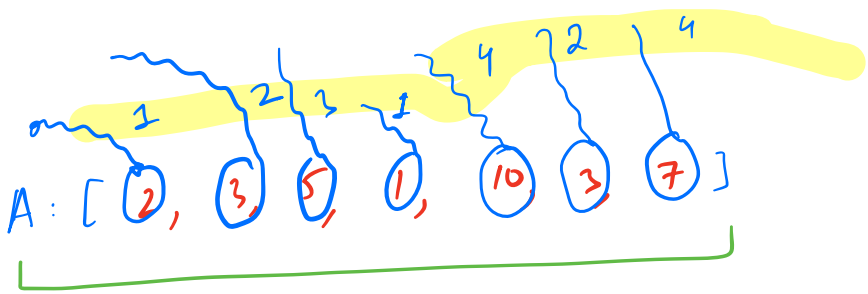
I) BF

Generate all sub-seq → 2^N

→ check × N

$N \cdot 2^N$

II)



ANS = MAX = 4

dp(i) → the length of the LIS ending at ith index

RR

$$dp(i) = \max_{j=0}^{j<i} \{ 1 + dp(j) : A_j < A_i \}$$

$$ANS = \max_{i=0}^{i=N-1} dp(i)$$

$[0 - N-1]$

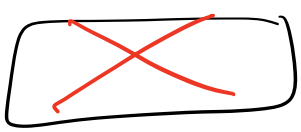
→ #US → N

TRPS → ~N

$TC = O(N^2)$

$SC = O(N)$

```
int dp[N] = {-1};

int LIS(i) {
    
    TC
    if (dp[i] != -1) ret dp[i];
    ANS = 1;
    f(j=0 → i-1) {
        if (Aj < Ai) {
            ANS = MAX(ANS, LIS(j) + 1);
        }
    }
    dp[i] = ANS;
}
```

```
f(i=0 → N-1)
    ANS = MAX(ANS, LIS(i));
```

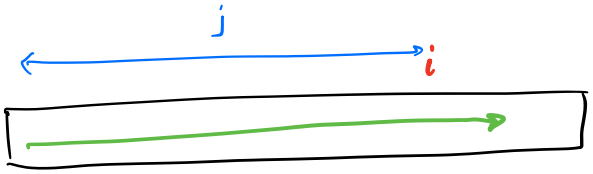
	0	1	2	3	4
A	2	3	5	5	6
dp	1	2	3	3	4

$O(N^2)$

ret ANS;

}

Bottom Up →



$\forall_i$ dp[i] = 1

```
for ( i = 1 ⟶ N-1 ) {
    for ( j = 0 ⟶ i-1 )
        if ( Aj < Ai )
            dp[i] = MAX( dp[i], dp[j] + 1 );
```

TC = O(N²)

SC = O(N)

NOTE :

1) BIN SEARCH

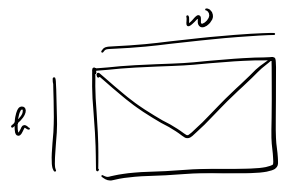
2) Some Range Query DS

TC = O(N log N)

SC = O(N)

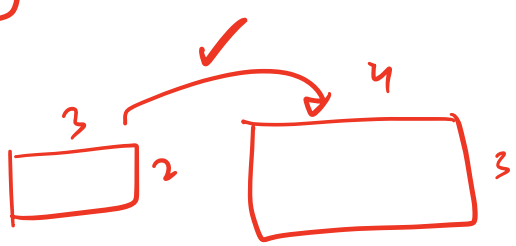
RUSSIAN DOLL ENVELOPE

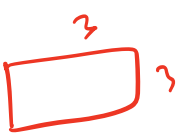
Given N envelopes having width & height
w[] h[]

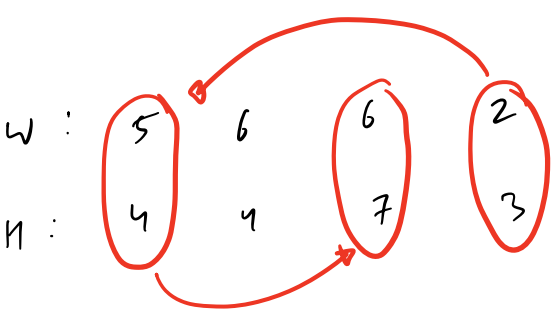


An envelope can fit inside another if and only if the width & height of this envelope is strictly smaller than the other!

find the MAX no. of envelopes that you can chain one inside another!







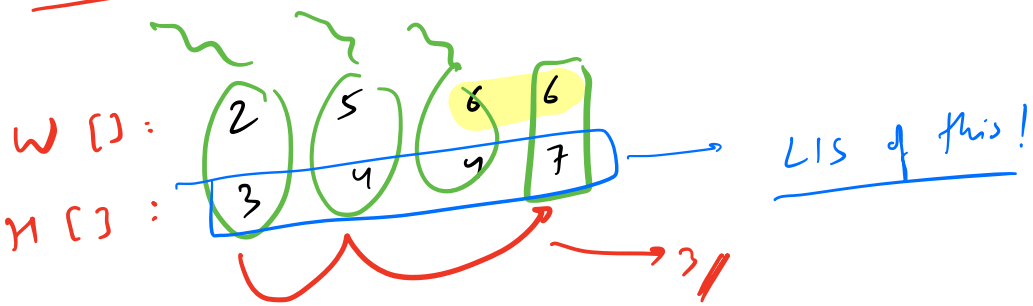
ANS = 3

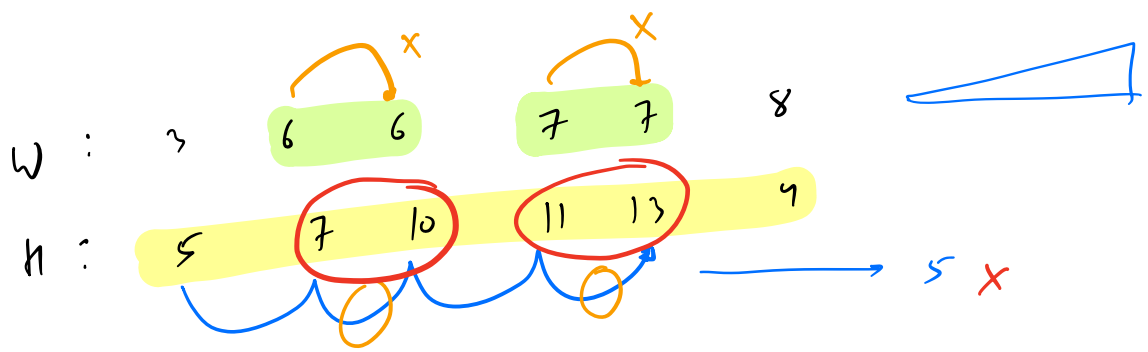
ANS E_{K_1} E_{K_2} E_{K_3} . — E_{K_n}

$W_{K_1} < W_{K_2} < W_{K_3}$ — W_{K_n}

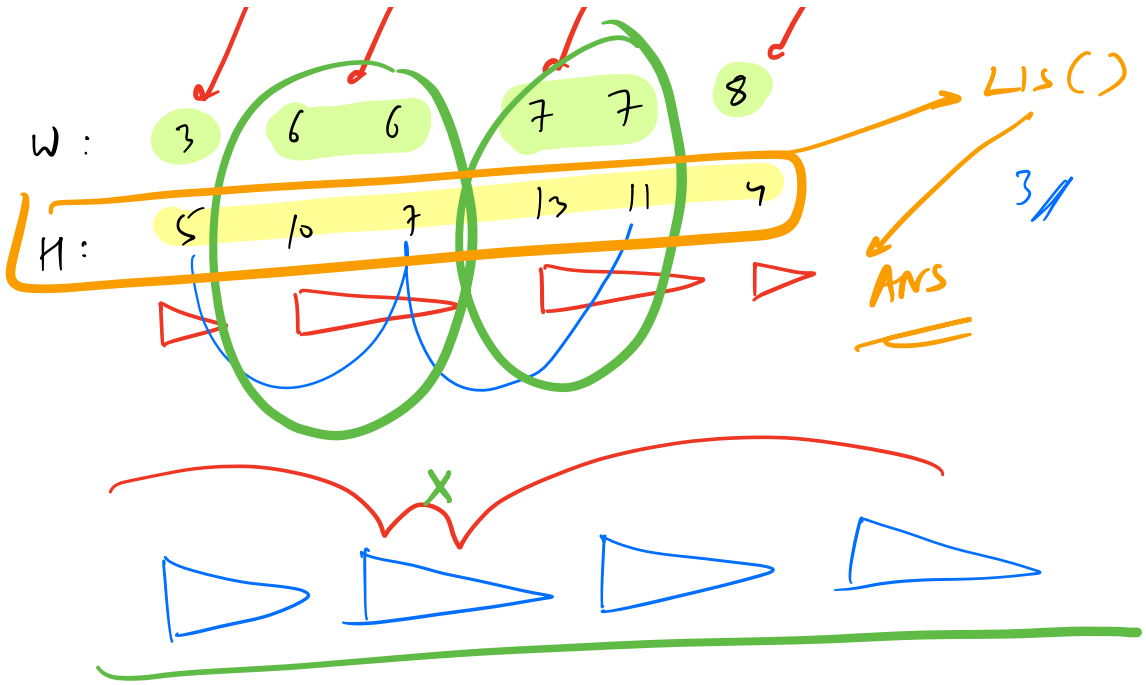
$H_{K_1} < H_{K_2}$ — — H_{K_n}

SORT ON W[]





Idea: Sort in INC order of W[] & to break TIE, DEC order of H[]



$TC \rightarrow N \log N + N^2$

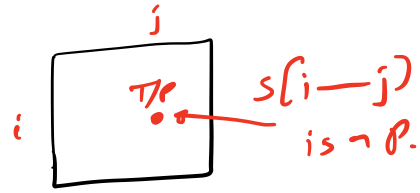
$= O(N^2)$

$SC = O(N)$

Q. Given a string check ∀ substrings if they are palindrome or not!

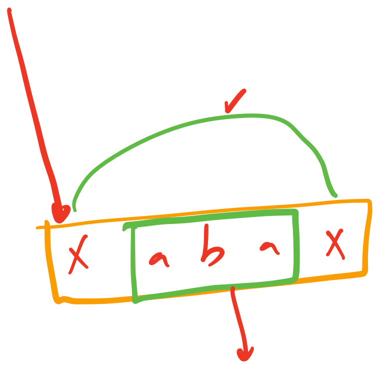
∀ substrings → $\sim N^2$

O/P →



0 1 2 3
b b c b

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

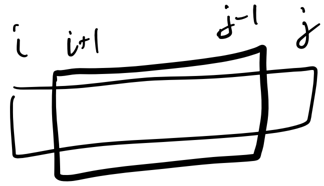


isP(i, j) → true : if the s[i — j] is a Palindrome
false : otherwise

$$isP(i, j) = (s[i] == s[j]) \;\&\&\; isP(i+1, j-1)$$

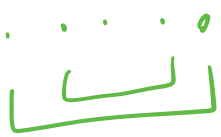
i → [0 – N-1], j → [0 – N-1]

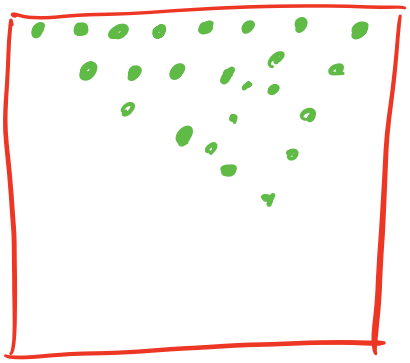
US → N^2



```
f(i = 0 ⟶ N-1)
    f(j = i ⟶ N-1)
        is P(i,j)
```

SC = O(N²)





TC = N² × O(1) + N² (vs TRPS)

TC = O(N²)

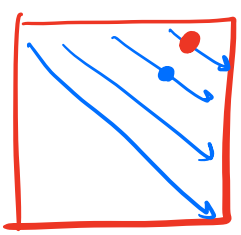
Bottom Up →

Ans of Bigger len string depends on Ans of smaller len string!

```
for(L=1; L<=N; L++){
    for(i=0; i<N; i++){
        j = i+L-1;          // (i, j)
        dp[i][j] = (s[i]==s[j] && dp[i+1][j-1]);
    }
}
```

TC = O(N²)

SC = O(1) ?



PALINDROMIC PARTITIONING

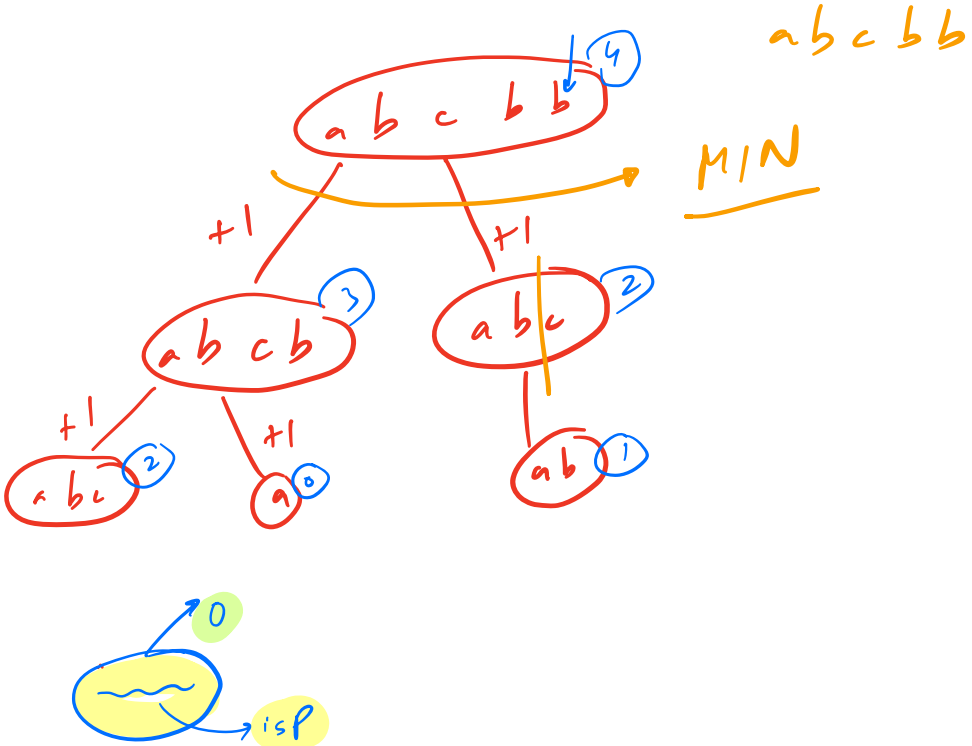
Q

Given a string, Find the min no. of cuts needed s.t after making the cuts, every resultant string is a palindrome!

S = a a | b → 1

S = a b a → 0

S = a | b | c d c | a | b → 4



mC(i) ⟶ min cuts reqd. for the string s[0 — i]

$$mC(i) = \min_{j=0}^{j=i-1} mC(j) + 1 \quad : \; s[j+1, i] \text{ is a palindrome}$$

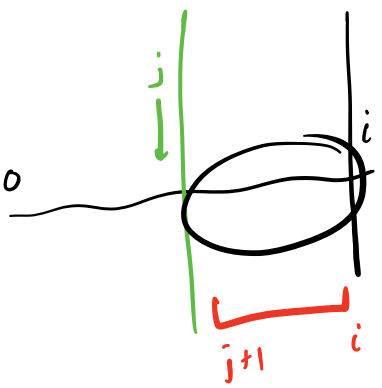
O(i)

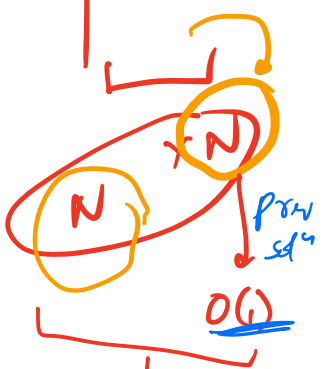
0 — N-1

#US → N

TRPS → ~ N^2

TC = $O(N^3)$





O(N)

Idea: first solve the prev problem

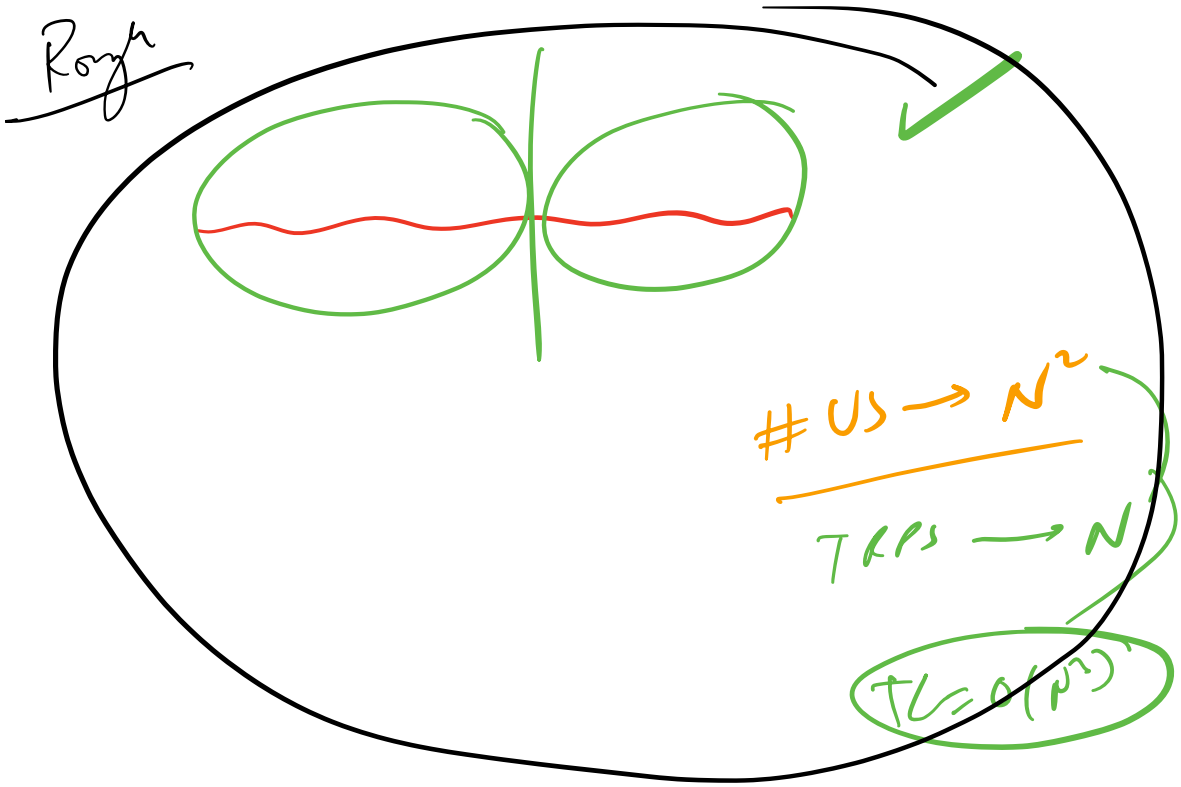
isP[ ][ ] Ready!

TRPS → O(N)

TC = $N^2 + N^2$

TC = $O(N^2)$

SC = $O(N^2)$


Roy
US → N²
TRPS → N
TC = O(N³)