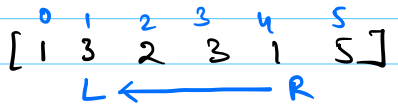
13/9/2023

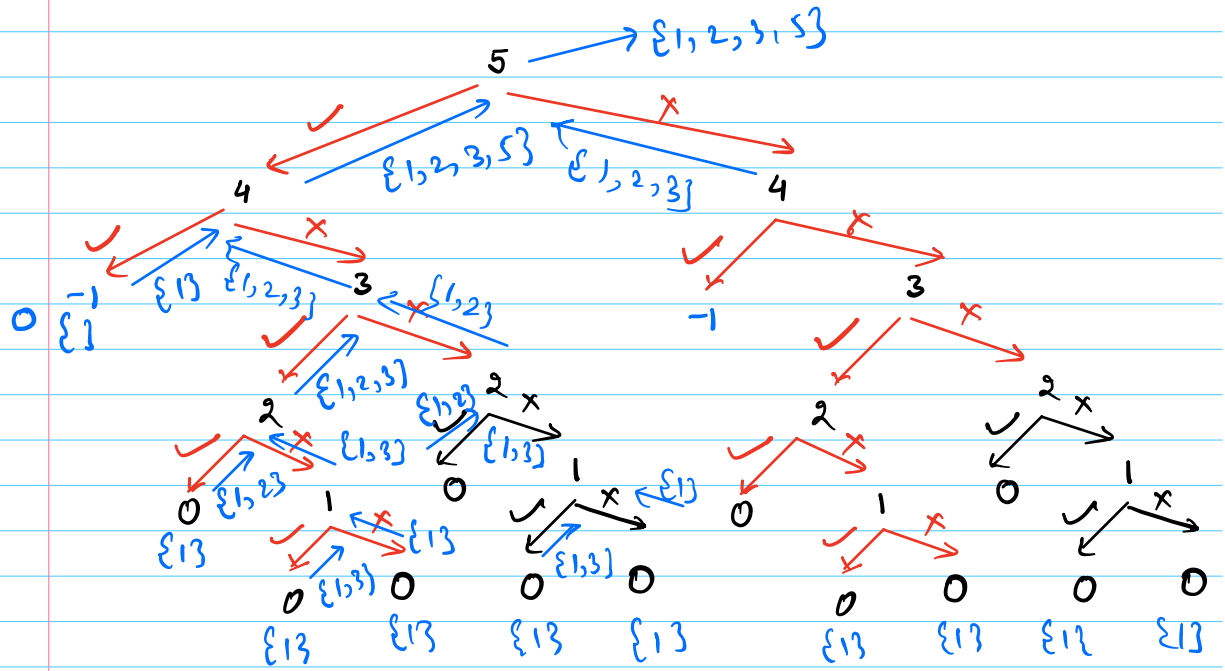
DP-6

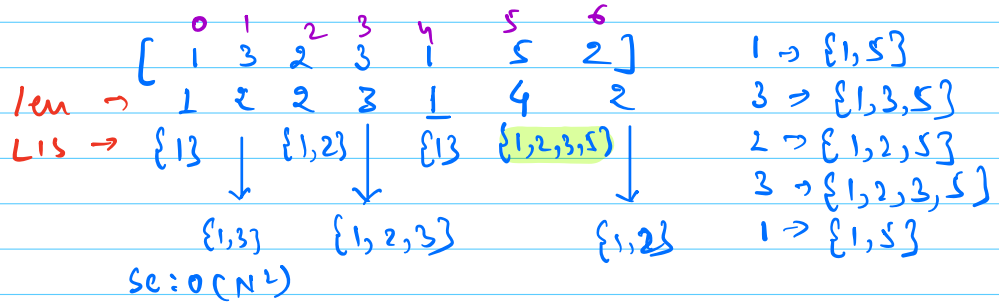
Q Find any longest increasing subsequence (LIS) of the given array A.

A = [1 3 2 3 1 5] Ans = [1, 2, 3, 5]

A = [1, 5, 8, 2, 1, 10] Ans = [1, 5, 8, 10]







LIS[i] → Longest increasing subsequence ending at 'i'

	0	1	2	3	4	5	6
	1	3	2	3	1	5	2
len →	1	2	2	3	1	4	2
Prev →	-1	0	0	2	-1	3	0/4

```
for i → 0 to N-1 {
    len[i] = 1 , Prev[i] = -1
    for j → 0 to (i-1) {
        if (A[j] < A[i] && len[j] >= len[i]) {
            len[i] = len[j] + 1;
            prev[i] = j;
        }
    }
}
```

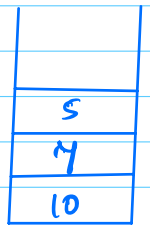
	0	1	2	3	4
A =	8	5	7	10	7
len →	1	1	2	3	2
prev →	-1	-1	1	2	1

5, 7, 10

```
mid = 0
for i → 0 to (N-1) {
    if (len[i] > len[mid]) {
        mid = i;
    }
}

while (ind != -1) {
    st.push(A[mid])
    mid = prev[mid]
}

while (!st.isEmpty()) {
    print (st.pop());
}
```



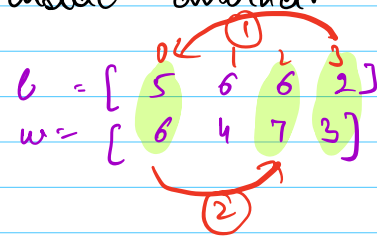
TC: O(N²)
SC: O(N)

Q Find length of LIS → Binary search

Q2 Given N envelopes with their length & width. An envelope i can fit in envelope j if

$l[i] < l[j]$ & $w[i] < w[j]$

Find the max count of envelopes that can fit one inside another.



Ans = 3

l = [3 2 1 3] Ans = 2
w = [2 3 1 3]

Observations:

1) We can change the order of the elements.
2) Sort w.r.t l[i] / w[i]

Ans ⇒ LIS w.r.t w[i] s.t. equal length is not included.

l → [2 5 6 6]
w → [3 6 4 7]

Tc: O(N log N + N²) ≃ O(N²)
Sc: O(N)

len → [1 2 2 3]

Meet at 8:28 am IST

Q Given a string, check if substrings whether the substring is a palindrome. (Ans → 2D array)

s = a b c d
0 1 2 3

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

← Ans

Bruteforce:→ if substring, check if it is palindrome
(N^2) (N)

$\approx O(N^3)$, SC: $O(1)$

Observation :-

a _ _ _ _ _ _ a } is palindrome.
i i+1 j-1 j

Base case →

length	Ans
1	true
2	s[i] == s[i+1]

```
for l → 1 to N { // len        [i, j] ⇒ j-i+1 = l
    for i → 0 to (N-1) { // start        j = l+i-1
        j = l+i-1 // end
        if (j >= N) break;

        if (l == 1) isP[i][j] = true;
        else if (l == 2) isP[i][j] = (s[i] == s[j]);
        else {
            isP[i][j] = (s[i] == s[j] && isP[i+1][j-1] == true)
        }
    }
}
```

TC: $O(N^2)$
SC: $O(1)$

S : abab

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

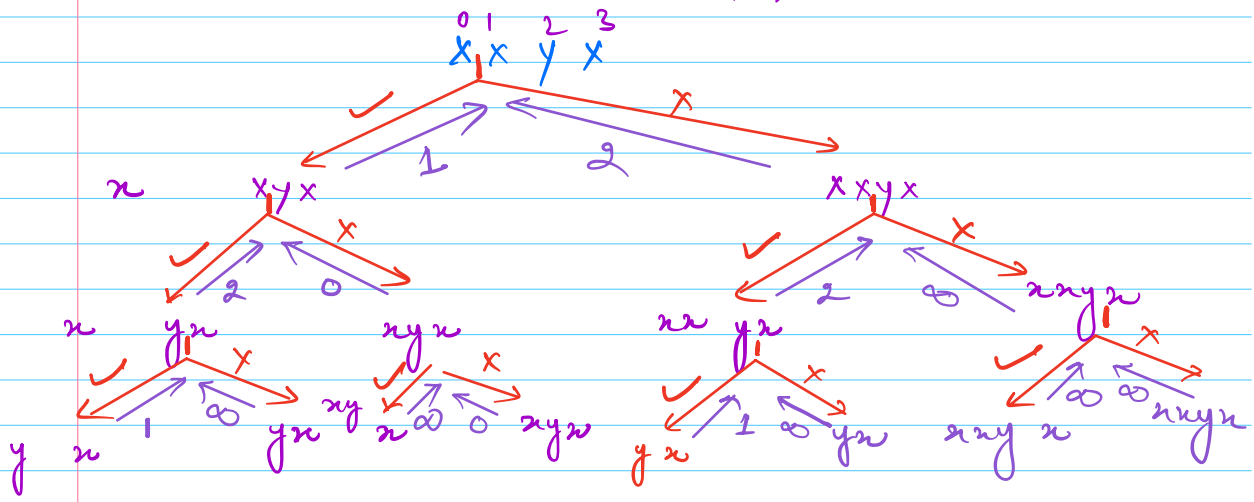
← Ans

Q Find **min cuts** to partition the string s.t. all the partitions are palindrome.

Eg → xxyy , Ans = 1 x | a | b a a b | p

a | b c b | b , Ans = 2 Ans = 3

a | b | c | b 2^{N-1} # ways

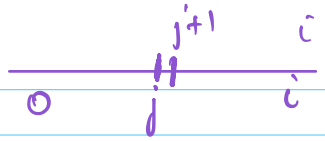


cuts[i] = minimum cuts for string [0 to i]

```
∀i, cuts[i] = INT_MAX
for i→0 to (N-1) {
    if (isP[0][i]) { cuts[i] = 0; }

    else {
        for j→0 to (i-1) {
            if (isP[j+1][i]) {
                cuts[i] = min(cuts[i], cuts[j]+1);
            }
        }
    }

    return cuts[N-1];
}
```



TC: $O(N^2)$
SC: $O(N^2)$

abab → a bab
abab → aba b

S = abab

	0	1	2	3
0	T	F	T	F
1	—	T	F	T ← Ans
2	—	—	T	F
3	—	—	—	T

cuts → [~~MAX~~ ~~MAX~~ ~~MAX~~ ~~MAX~~]
0 1 0 1