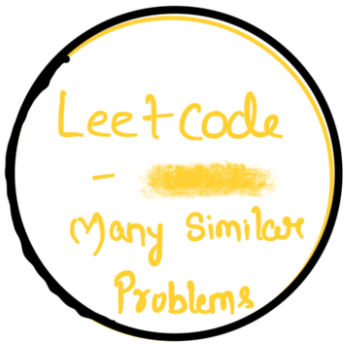
Dynamic Programming

Video - III

Note :- This playlist is on for explanation of dns & olutions.

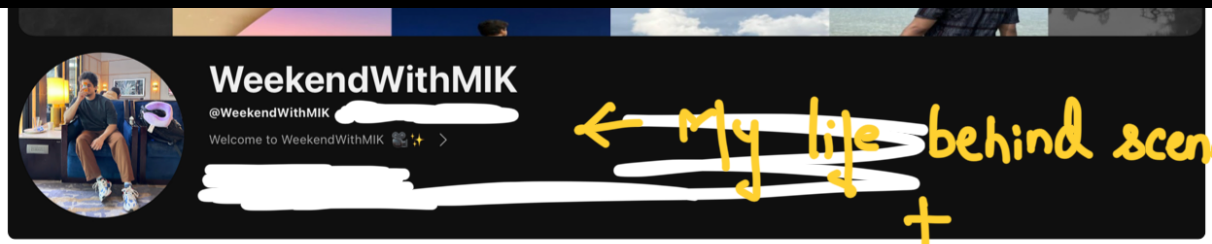
See my "DP Concepts & dns" playlist for understanding DP from scratch...



codestorywithmik

CSwithMIK

codestorywithMIK



← My life behind scenes
+
Tech News/updates

Motivation :-

A good life needs some bad days too.

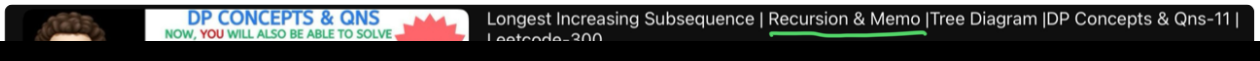
Never wait for perfect time.

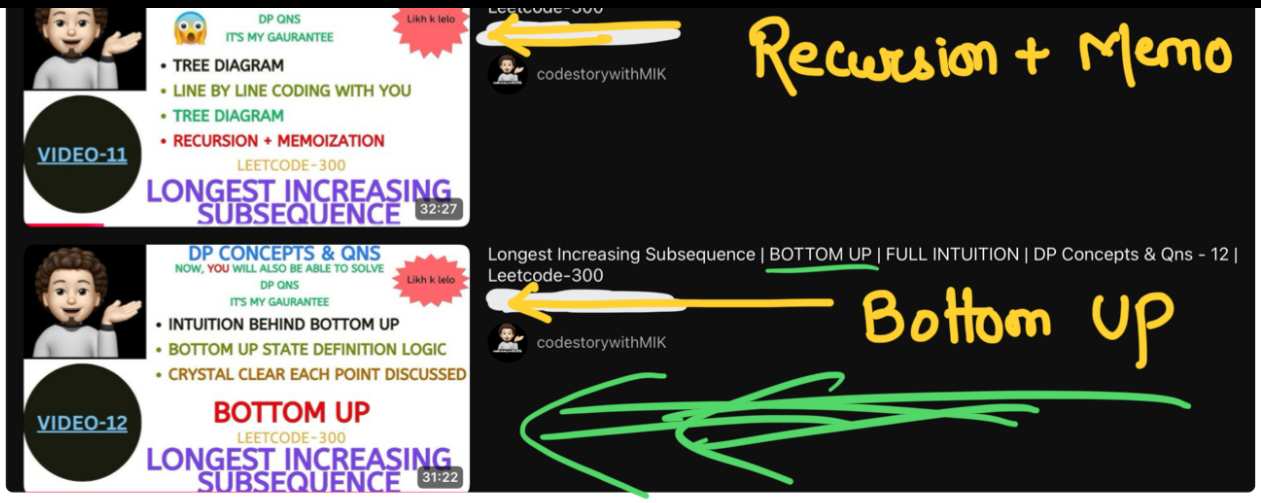
Life goes faster than you think.

START Now ...

MIK...

Print the Longest Increasing Subsequence (LIS)





Recap of LIS using Bottom up :-

nums = { 10, 9, 2, 5, 3, 7, 101, 18 }

0 1 2 3 4 5 6 7

~~2, 3, 7, 101~~
2, 3, 7, 18

2, 5, 7, 101
2, 5, 7, 18

dp[i] = longest increasing subsequence ending at index i

nums = { 10, 1, 2, 5, 3, 7, 101, 18 }

0 1 2 3 4 5 6 7

i

dp = { 1, 1, 2, 3, 1, 1, 1, 1 }

```
dp[n] = {1};
int LIS = 1;
for( i=0;  i<n;  i++) {
    for(j= 0;  j<=i-1;  j++) {
        if (nums[j] < nums[i]){
            dp[i] = max(dp[i], dp[j]+1);
            LIS = max(LIS, dp[i]);
        }
    }
return LIS;
```

Printing the LIS ...

nums = { 10, 9, 2, 5, 3, 7, 101, 18 }

dp = { 1, 1, 1, 2, 2, 3, 4, 4 } LIS=4

PrevIndex = { -1, -1, -1, 2, 2, 3, 5, 5 }

{ 101, 7, 5, 2 }

reverse = { 2, 5, 7, 2 }

18, 7, 5, 2,

→ 2, 5, 7, 18

```
Parent = {-1};
for (int i = 0;  i<n;  i++) {
    for(j = 0;  j<i ; j++) {
        if (nums[j] < nums[i]) {
            if (dp[j]+1 > dp[i]) {
                dp[i] = dp[j]+1;
                Parent[i] = j;
```

```
        }
    }
}
```

```
if (LIS < dp[i]) {
    LIS = dp[i];
    LIsidx = i;
}
```

LISindex

```
vector<int> result;
while (LISindex != -1) {
    result.push_back(nums[LISindex]);
    LISIndex = parent[LISindex];
}

//reverse result;
return result;
```

Why is LIS pattern Special ???

```
for (int i=0;  i<n;  i++) {
    for(j=0;  j<i ; j++) {
        if (nums[j] > nums[i]) {
            if (dp[j]+1 > dp[i]) {
                dp[i] = dp[j]+1;
                if (LIS < dp[i]) {
                    LIS = dp[i];
                }
            }
        }
    }
}
```

① Longest subsequence → "Increasing"

② Longest subsequence → "Decreasing"

③ nums[j] ? nums[i]

↓

My logic