Longest Increasing Subsequences

arr[] = {10, 9, 2, 5, 3, 7, 101, 18} → Length return

Strictly Increasing

because.

arr[] = {8, 8, 8} → ans = 1

Approach:

Brute: Generate all the Subsequence and store it and check every increasing subsequence length and find Longest Length.

Recurrence:

(I) Index
(II) Explore
(III) Maxlen
(IV) Basc

$f(i, prev)$

if $(i == n-1)$ return 0;

not take = $f(i+1, prev)$;

take = 0

if $(arr[i] > arr[prev] \;||\; prev == -1)$

take = $1 + f(i+1, i)$;

return max(not take, take);

Tabulation: dp[n][n+1] = 0

1. Base Cause

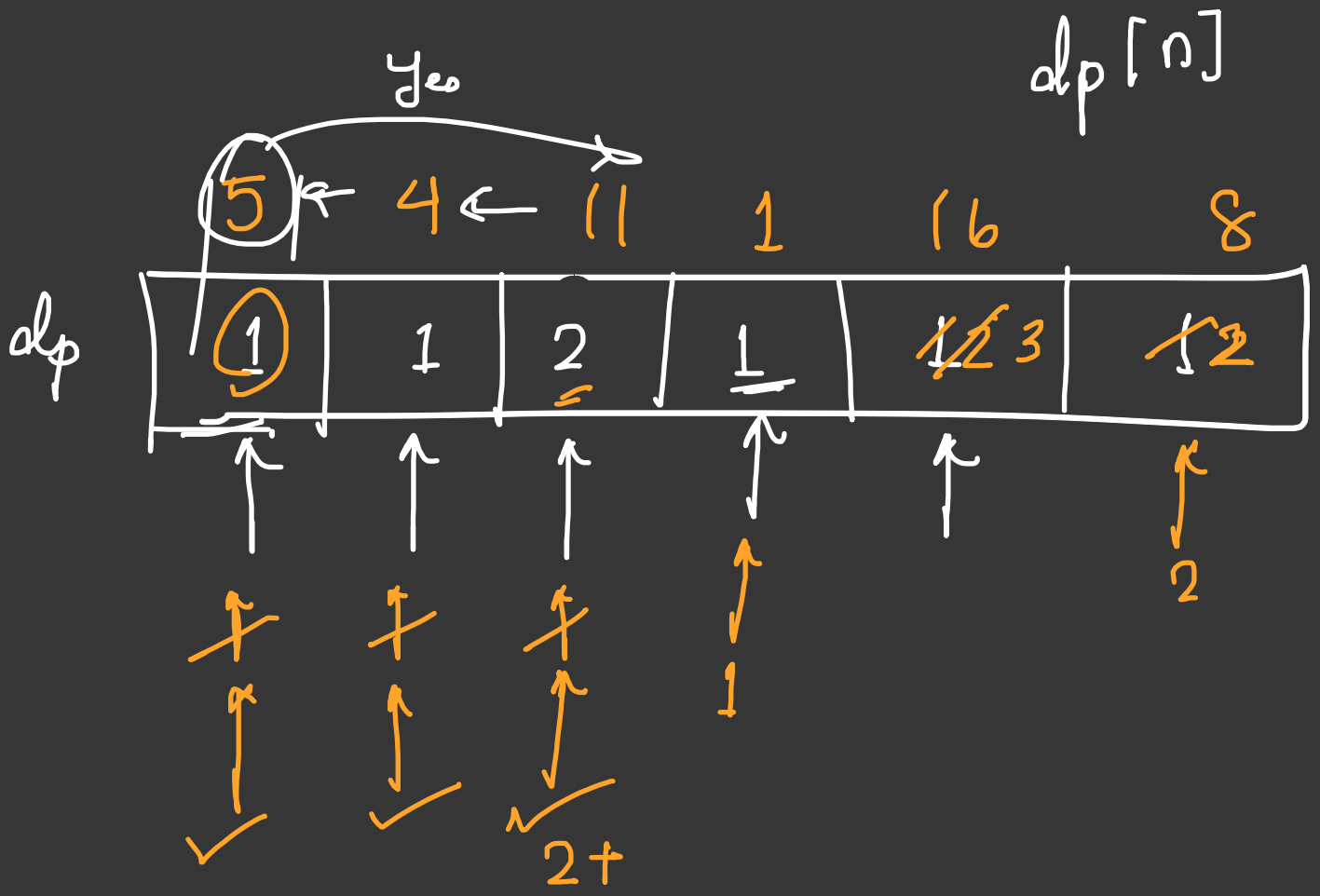
dp[n][n+1] = 0 x baseCase

2. Nested loop with changing parameters

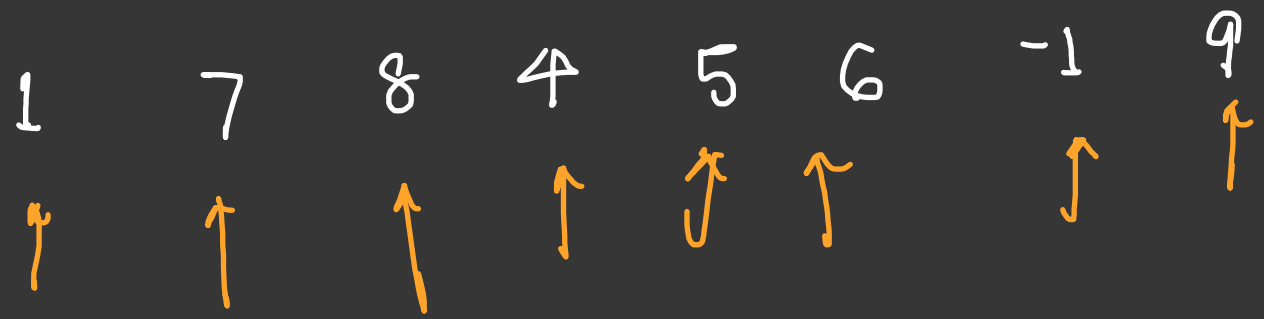
for(i = n → 0
prev_ind = i - 1 → -1

O(N) Solution

5 4 11 1 16 8



Binary Search:



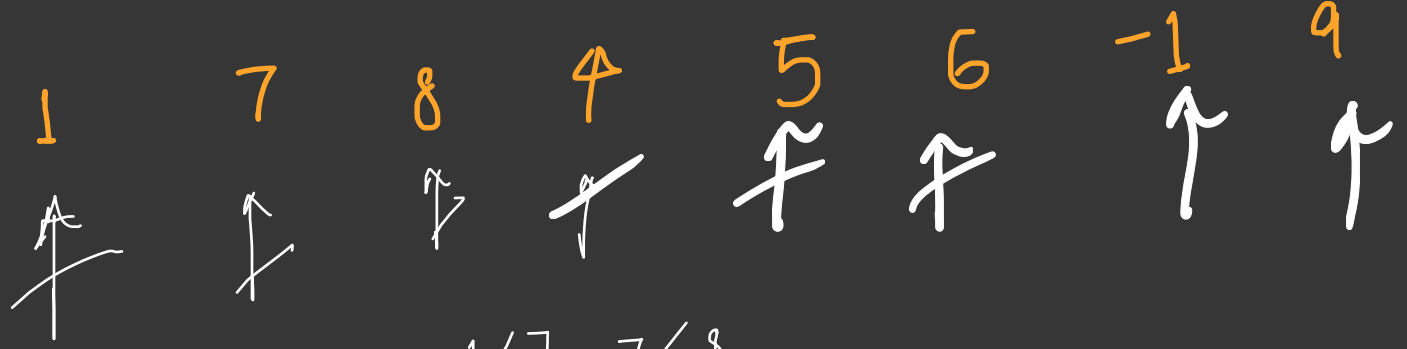
{1, 7, 8, 9} → 4

{1, 4, 5, 6, 9} → 5

{-1, 9} → 2

Instead of creating subsequence different as we require only length so, we overwrite the subsequence.

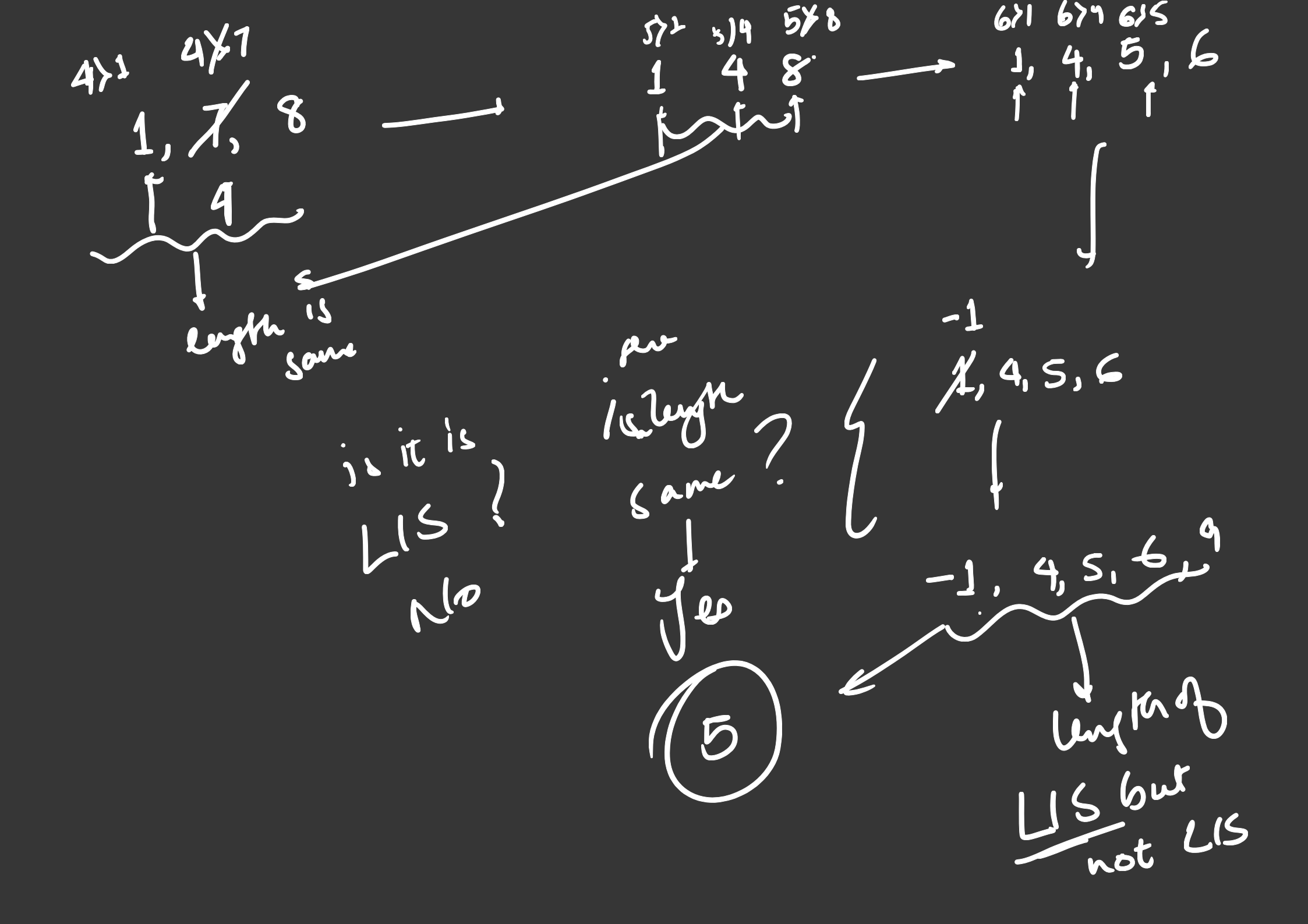
So,



1<7 7<8

1, 7, 8

instead of creating new sequence with, {1, 4} we overwrite in


4>1 4>7
1, 7, 8
9
length is same
5>2 5>4 5>8
1 4 8
6>1 6>4 6>5
1, 4, 5, 6
is it is LIS ? No
per length same ? Yes
5
-1
-1, 4, 5, 6
-1, 4, 5, 6, 9
length of LIS but not LIS

Q. Where is Binary Search?

While overwriting the sequence we have to search the correct place for the number and it as it is sorted we can Binary search.

We us Lower-bound function in C++,

It gives us the ~~first~~ index of curr[i] or index of first element greater than curr[i]