Dynamic Programming

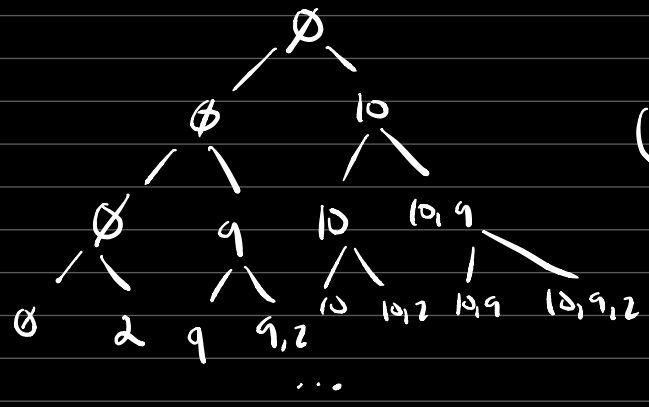
Longest Increasing Subsequence (subset)

[10, 9, 2, 5, 3, 7, 101, 18]

{2, 3, 7, 101}

{2, 5, 7, 101}

Approach 1: find all subsets.



(2^n)

Answer

$O(2^n)$ exponential

then check if each subset is increasing $O(n)$

{50, 3, 10, 7, 40, 80} (0 1 2 3 4 5)

3, 10, 40, 80

3, 7, 40, 80

→

{50, 3, 10, 7, 40, 80, 41, 42}

3, 10, 40, 41, 42 ✓

3, 10, 40, 80 ×

not necessarily consec

F(5) = 1

F(4): 1 or 2

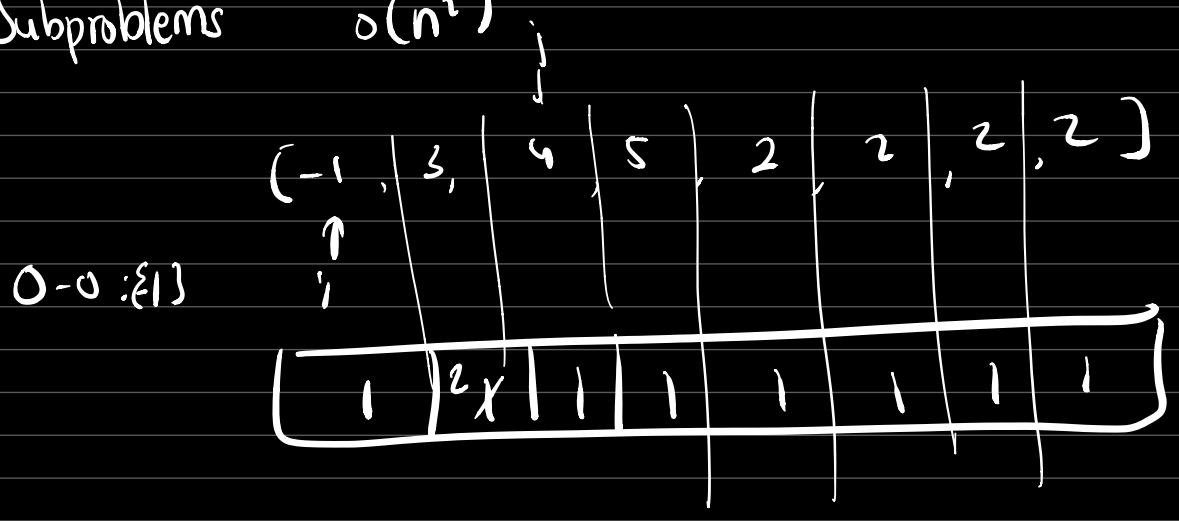
F(4) = 1 + {include 80}

or

1 + (not include 80)

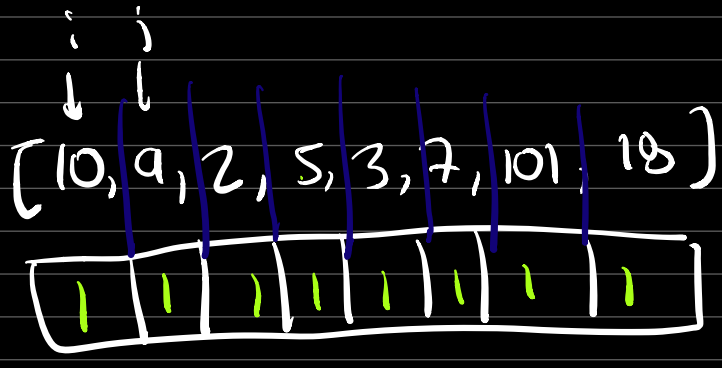
$f(4) = 1 + f(5)$

② Subproblems $O(n^2)$



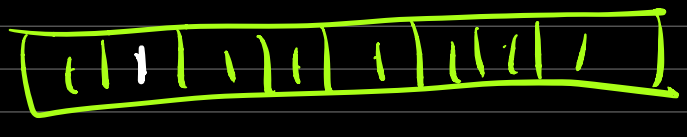
0-1

$4 > -1$

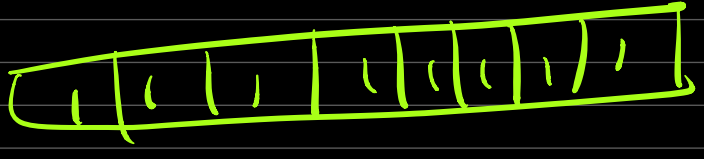


$j=1$
$i=1$

$9 < 10$



$j=2$
$i=0$



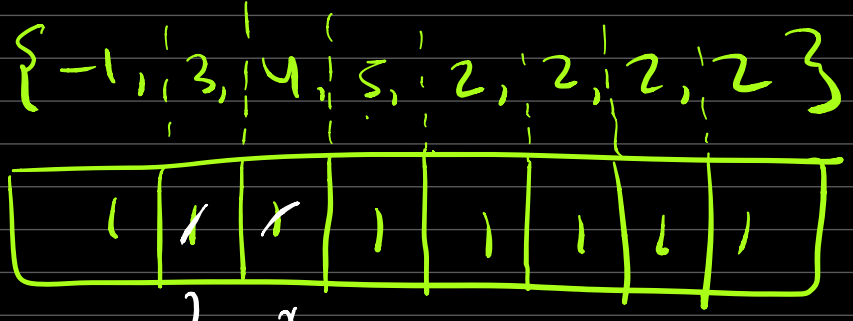
$j=2$
$i=1$

$i=3$

i=0

j=

{-1, 3, 4, 5, 2, 2, 2, 2}



2 2 3

-1, 4

-1, 3, 4

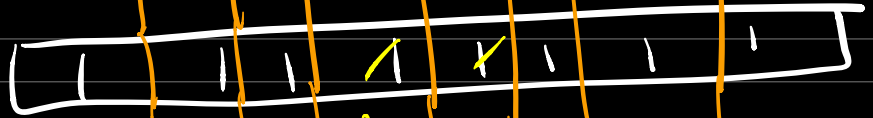
2, 3, -1, 4

1 x 1 1

2 2

3

[10, 9, 2, 5, 3, 7, 101, 18]



2

x 3

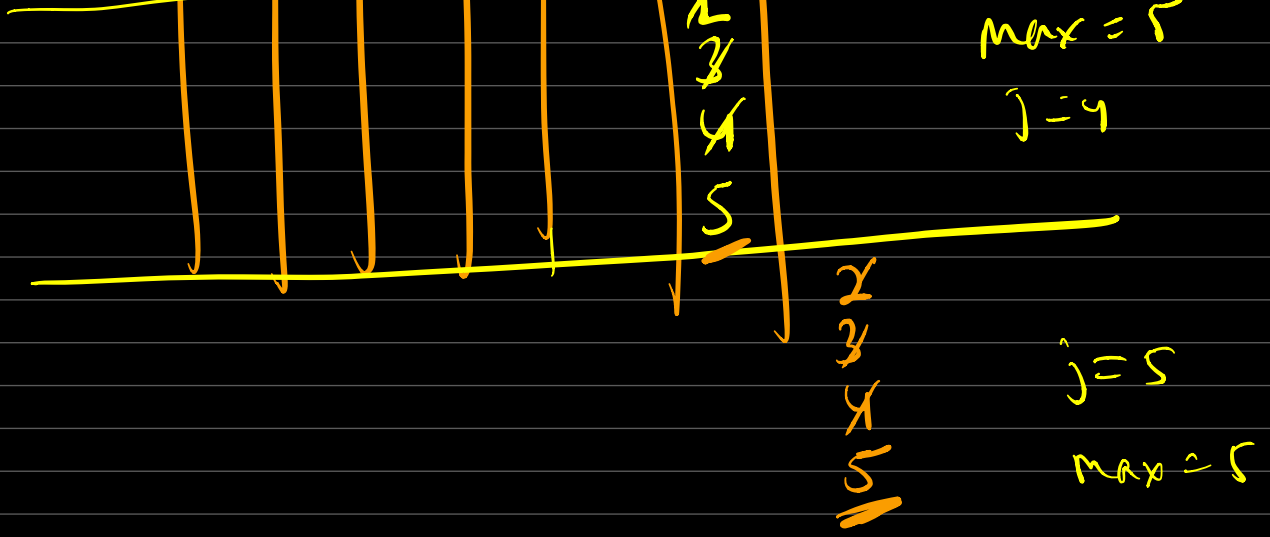
x 3 4

x

j iteration = 1 max = 2

j = 2 max = 3

j = 3 max = 4



return max;

```
class Solution {
public:

    int lengthOfLIS(vector<int>& nums) {
        vector<int> dp(nums.size(), 1);  //dp table for bottom-up
        if (nums.size()==0) return 0;
        int max = 1;
        for(int j=0; j<nums.size(); j++) {
            for(int i=0; i < j; i++) {
                if (nums[i] < nums[j]) {
                    if (dp[i] + 1 > dp[j]) {
                        dp[j] = dp[i]+1;
                        if (dp[j]>max) max = dp[j];
                    }
                }
            }
        }
        return max;

    }
};
```