Dynamic Programming

Video - 107

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

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

Facebook, Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywith IK

Leetcode - 873

Medium → DP

Motivation :-

No one is perject. Stop running after perjection. You cannot be 100% ready for everything. You just have to keep moving in your own comfortable steady pace. Don't rush, it will all fall in right place in right time.

MIK...

873. Length of Longest Fibonacci Subsequence

Medium | Topics | Companies

A sequence $x_1, x_2, \ldots, x_n$ is *Fibonacci-like* if:

x_1, x_2, x_3, x_4, x_5

- $n >= 3$
- $x_i + x_{i+1} == x_{i+2}$ for all $i + 2 <= n$

Given a **strictly increasing** array `arr` of positive integers forming a sequence, return *the* ***length*** *of the longest Fibonacci-like subsequence of* `arr`. If one does not exist, return `0`.

A **subsequence** is derived from another sequence `arr` by deleting any number of elements (including none) from `arr`, without changing the order of the remaining elements. For example, `[3, 5, 8]` is a subsequence of `[3, 4, 5, 6, 7, 8]`.

Example :- arr = [1, 2, 3, 4, 5, 6, 7, 8]

Output = 5 ← {1, 2, 3, 5.}

arr = [1, 3, 7, 11, 12, 14, 18]

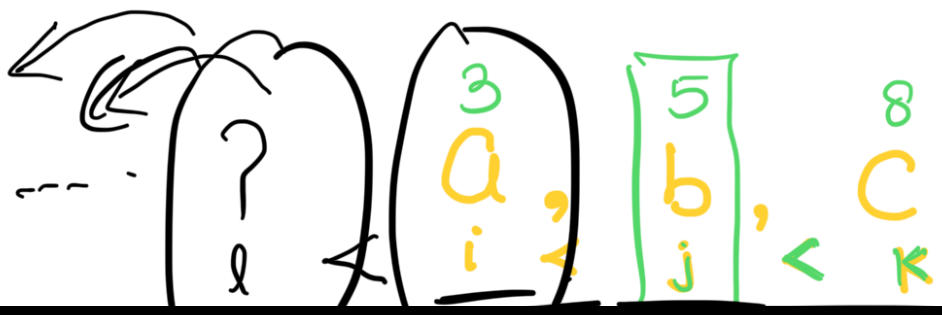
Output = 3 ← {3, 11, 14} ✓
{7, 11, 18} ✓
{1, 11, 12} ✓

Thought Pro

" Let's think from a beginner po "

map = {1→0, 2→1, 3→2, 4→3 ..}

arr = [1, 2, 3(i), 4, 5(j), 6, 7, 8(K)]



$C = a + b \;\; ; \Rightarrow C - b = a$

arr[K] − arr[j] =

8 − 5 = 3

found 3 at index i = 2

$i < j$

arr[1] = arr[j] − arr[i]

5 − 3 = 2

found 2 at index = 1

// Store (element → idx) in the map

```
for ( j = 1 ; j<n ; j+1) {
    for (K = j+1 ; K<n ; K++) {
        length = Solve ( j, K, arr, mp);
        if ( length >= 3) {
            result = max(result, length);
        }
    }
```

}

return result;

```
int Solve( j, K , arr, mp) {

    target = arr[K] - arr[j] ;

    if (mp.count(target) && mp[1 st] < j) {
        int i = mp[target];
        return Solve (i, j, arr, mp) + 1
    }
    return 2;
}
```

kth element

(i) (j) K

arr[j] - arr[i]

map

a → 0
b → 1
c → 2

{ a, b, c }
0 1 2

i j K

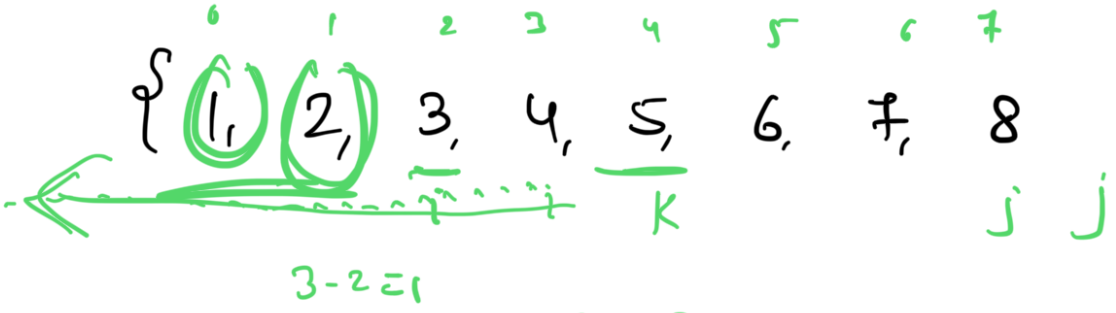
Solve(j,K)

1+2 ← c-b=a

0 < 1

2 ⌊ solve(i,j)

b-a = z



result = 3 4 5

Recursion.

T.C = $O(n^2 * n) = O(n^3)$

S.C = $O(n)$

Why memoization :-??

0 1 2 3 4 5 6 7

{1, 2, 3, 4, 5, 6, 7, 8}

i j K

3+1 = 4

3 Fibon.

t[j][K]

t[i][2] = 3 ;

Bottom UP :-

Solve (i, j, arr, mp)

State Define :: t[i][j] = Max length of Fibb' like seq. ending at (i)(j)

... {... i, j }

for (int j = 1 ; j<n; j++) {

```
for (int k = j+1 ; k < n; k ++) {
    int target = arr[k] - arr[j];
    if (mp.count(target) && mp[target] < j) {
        int i = mp[target];
        f[j][k] = f[i][j] + 1;
    }
    max_length = max(max_ln, f[j][k]);
}
}
return max_length >= 3 ? max_length : 0;
```