Competitive Programming

19, 20

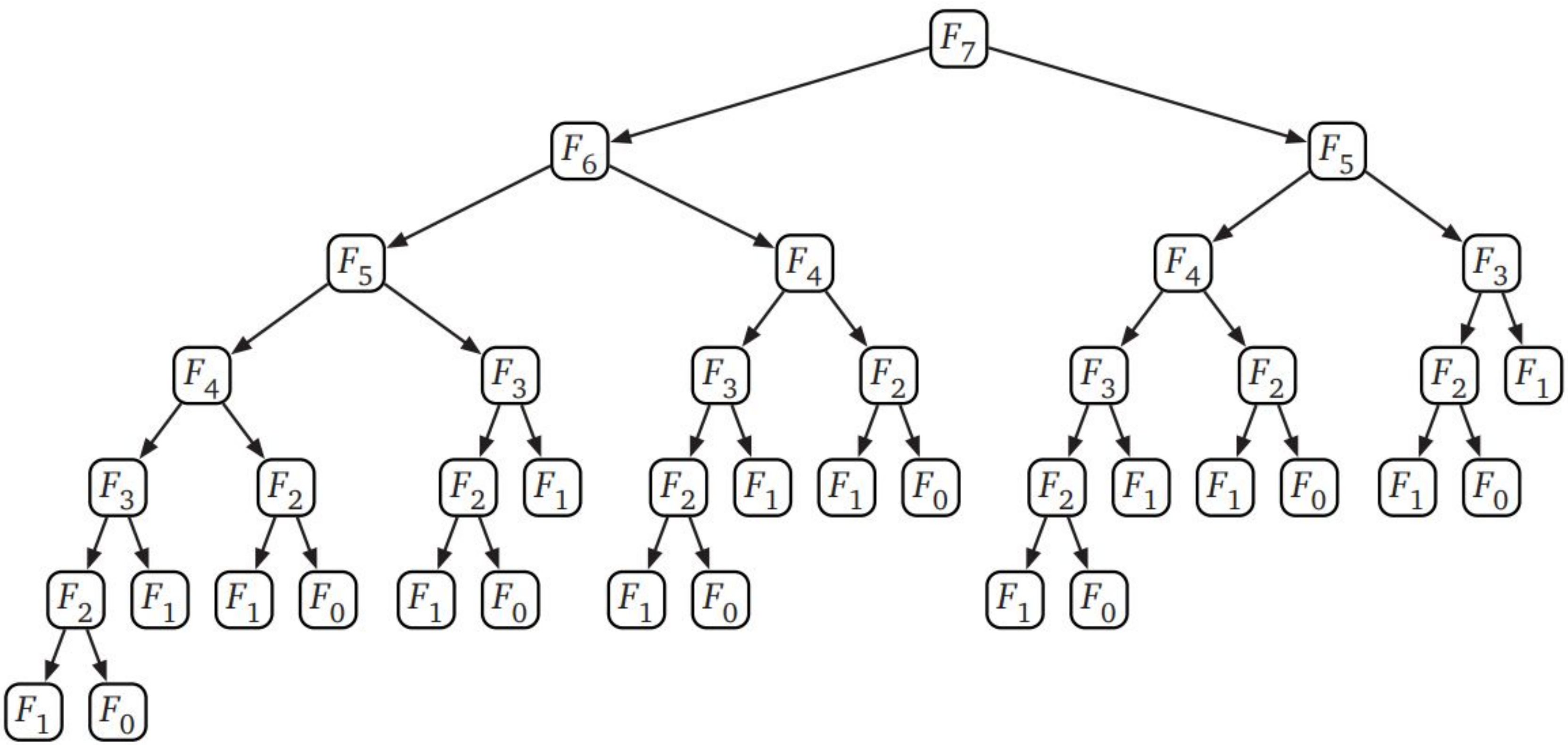
Dynamic Programming

Bhavik Dhandhalya

1. Write a recursive program to print Fibonacci numbers.

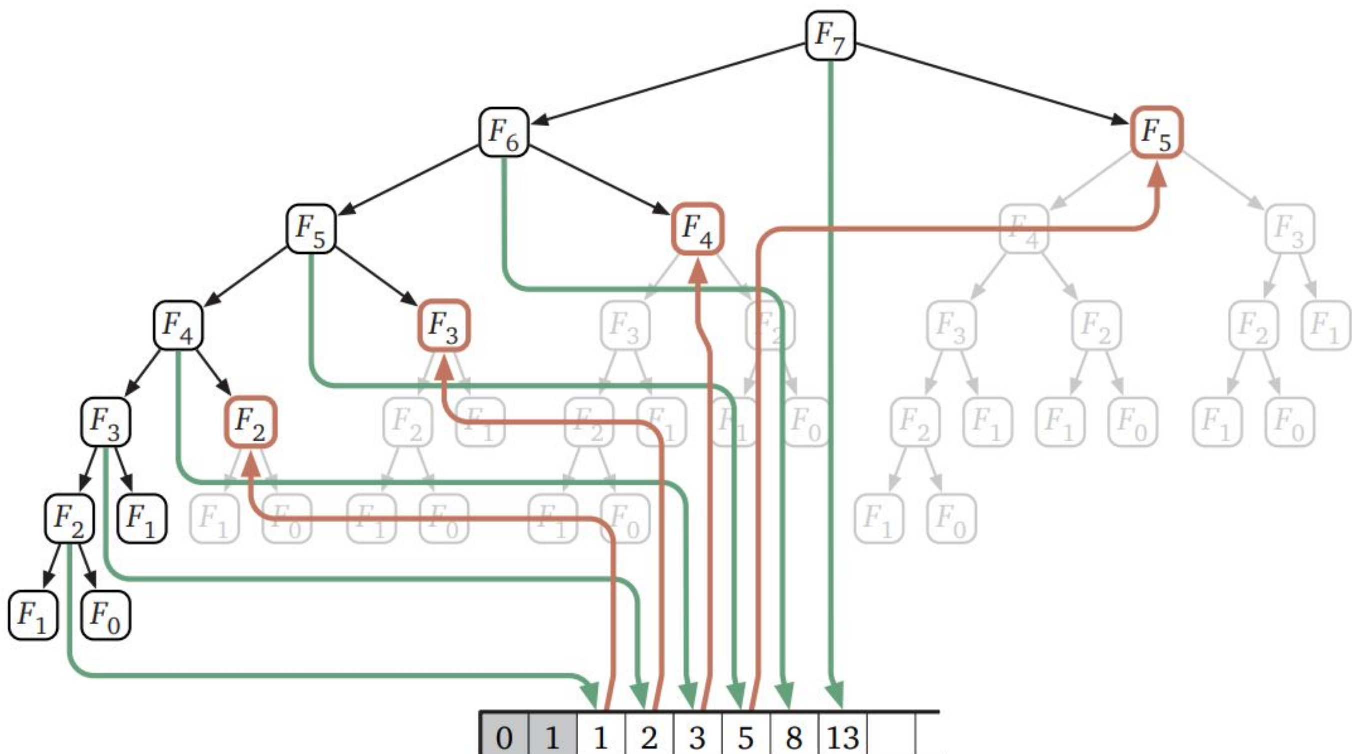
Time complexity?

```
RecFibo(n):
    if n = 0
        return 0
    else if n = 1
        return 1
    else
        return RecFibo(n − 1) + RecFibo(n − 2)
```



Bhavik Dhandhalya

```
MemFibo(n):
    if n = 0
        return 0
    else if n = 1
        return 1
    else
        if F[n] is undefined
            F[n] ← MemFibo(n − 1) + MemFibo(n − 2)
        return F[n]
```



Bhavik Dhandhalya

```
ITERFIBO(n):
    F[0] ← 0
    F[1] ← 1
    for i ← 2 to n
        F[i] ← F[i−1] + F[i−2]
    return F[n]
```

Bhavik Dhandhalya

How to Approach DP problems ?

1. Identify subproblems

2. Identify Base cases (states)

3. Try writing Recursive solution first

4. A. Write memoization of recursive function **OR**
   B. Convert Recursive solution

1-dimensional DP Example

Problem:
given n, find the number of different ways to write n as the sum of 1, 3, 4

Example:
for n = 5, the answer is 6
5 = 1 + 1 + 1 + 1 + 1
= 1 + 1 + 3
= 1 + 3 + 1
= 3 + 1 + 1
= 1 + 4
= 4 + 1

Draw tree of all possible choices.

1-dimensional DP Example

<u>Recurrence</u> is then

$$D_n = D_{n-1} + D_{n-3} + D_{n-4}$$

Solve the base cases

- $D_0 = 1$
- $D_n = 0$ for all negative n
- Alternatively, can set: $D_0 = D_1 = D_2 = 1$, and $D_3 = 2$

<u>Code:</u>

```
dp[0] = dp[1] = dp[2] = 1; dp[3] = 2;
for(i = 4; i <= n; i++)
   dp[i] = dp[i-1] + dp[i-3] + dp[i-4];
```

Stairs

You are climbing a stair case and it takes A steps to reach to the top.
Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

A = 2
Output 1:
2
Explanation 1:
[1, 1], [2]
Input 2:
A = 3
Output 2:
3
Explanation 2:
[1 1 1], [1 2], [2 1]

Solution

LCS

Problem:
given two strings x and y, find the longest common

subsequence (LCS) and print its length

Example:
- x: A**BC**BD**AB**
- y: **B**D**CAB**C
- "BCAB" is the longest subsequence found in both sequences, so the answer is 4

Try drawing Tree

LCS

- Define subproblems
  - Let D_{ij} be the length of the LCS of $x_{1\ldots i}$ and $y_{1\ldots j}$
- Find the recurrence
  - If $x_i = y_j$, they both contribute to the LCS
    - $D_{ij} = D_{i-1,j-1} + 1$
  - Otherwise, either x_i or y_j does not contribute to the LCS, so one can be dropped
    - $D_{ij} = \max\{D_{i-1,j}, D_{i,j-1}\}$
  - Find and solve the base cases: $D_{i0} = D_{0j} = 0$

LCS

```
for(i = 0; i <= n; i++) dp[i][0] = 0;
for(j = 0; j <= m; j++) dp[0][j] = 0;

for(i = 1; i <= n; i++) {
    for(j = 1; j <= m; j++) {
        if(x[i] == y[j])
            dp[i][j] = dp[i-1][j-1] + 1;
        else
            dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
    }
}
```

Unique Paths in a Grid

Given a grid of size m * n, lets assume you are starting at (1,1) and your goal is to reach (m,n). At any instance, if you are on (x,y), you can either go to (x, y + 1) or (x + 1, y).
Now consider if some obstacles are added to the grids. How many unique paths would there be?
An obstacle and empty space is marked as 1 and 0 respectively in the grid.
Example :
There is one obstacle in the middle of a 3x3 grid as illustrated below.

```
[
  [0,0,0],
  [0,1,0],
  [0,0,0]
]
```

The total number of unique paths is 2.
Solution

Min Sum Path in Triangle

Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

For example, given the following triangle

```
[
     [2],
    [3,4],
   [6,5,7],
  [4,1,8,3]
]
```

The minimum path sum from top to bottom is 11 (i.e., 2 + 3 + 5 + 1 = 11).

Solution

Homework

Jump Game Array	Easy
Longest Increasing Subsequence	Easy
Maximum path sum I	Easy
Edit Distance	Medium, Classical Problem