Computer Science 161
Spring 2021 Lecture 12: Dynamic Programming: Introduction

Big Idea: recursion

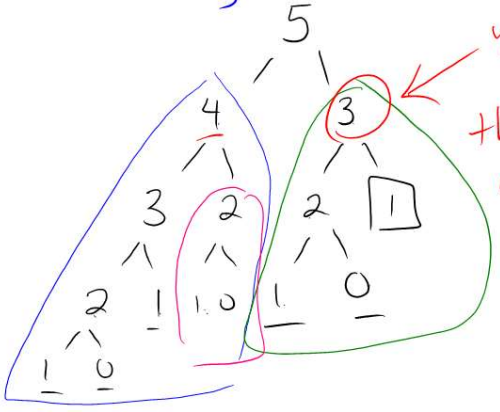
- programming concept
- mathematical concept

2 General Introduction: Computing Fib

0 1 1 2 3 5 8 13 21 34 ...

```
Fib(n)
  if n = 0 or n = 1 then
    return n
  return Fib(n − 1) + Fib(n − 2)
```

How many function calls?

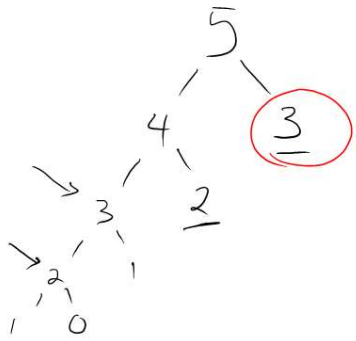


n	# calls
5	15
4	9
3	5
2	3
1	1
0	1

3 General Introduction: Memoizing Fib

Declare a global array `memo[0, ..., n]`
Set all `memo[i] = -1` // means "I have not yet found this value"
Set `memo[0] = 0` and `memo[1] = 1`

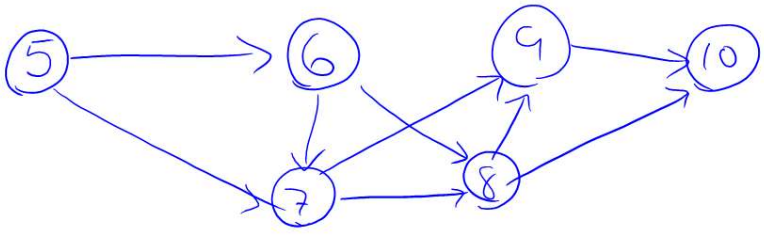
```
fib(n)
    if -1 == memo[n]
        memo[n] = fib(n-1) + fib(n-2)
    return memo[n]
```



4 A topological order for Fib

Suppose we have a directed acyclic graph

- One vertex per recursive call
- An edge (v_i, v_j) means we use i to compute j



Order: 0, 1, 2, ... n-1, n

5

General Introduction: Better Fib

```
Declare Fib[0 . . . n]
Fib[0] = 0
Fib[1] = 1
for i ← 2 . . . n do
  Fib[i] = Fib[i − 1] + Fib[i − 2]
```

invariant: Fib[0...i-1] has the relevant values

- overlapping subproblems