CompSci 161
Winter 2023 Lecture 10:
Dynamic Programming:
Interval Scheduling

Warm-Up Question 1

Fib(n : non-negative integer)

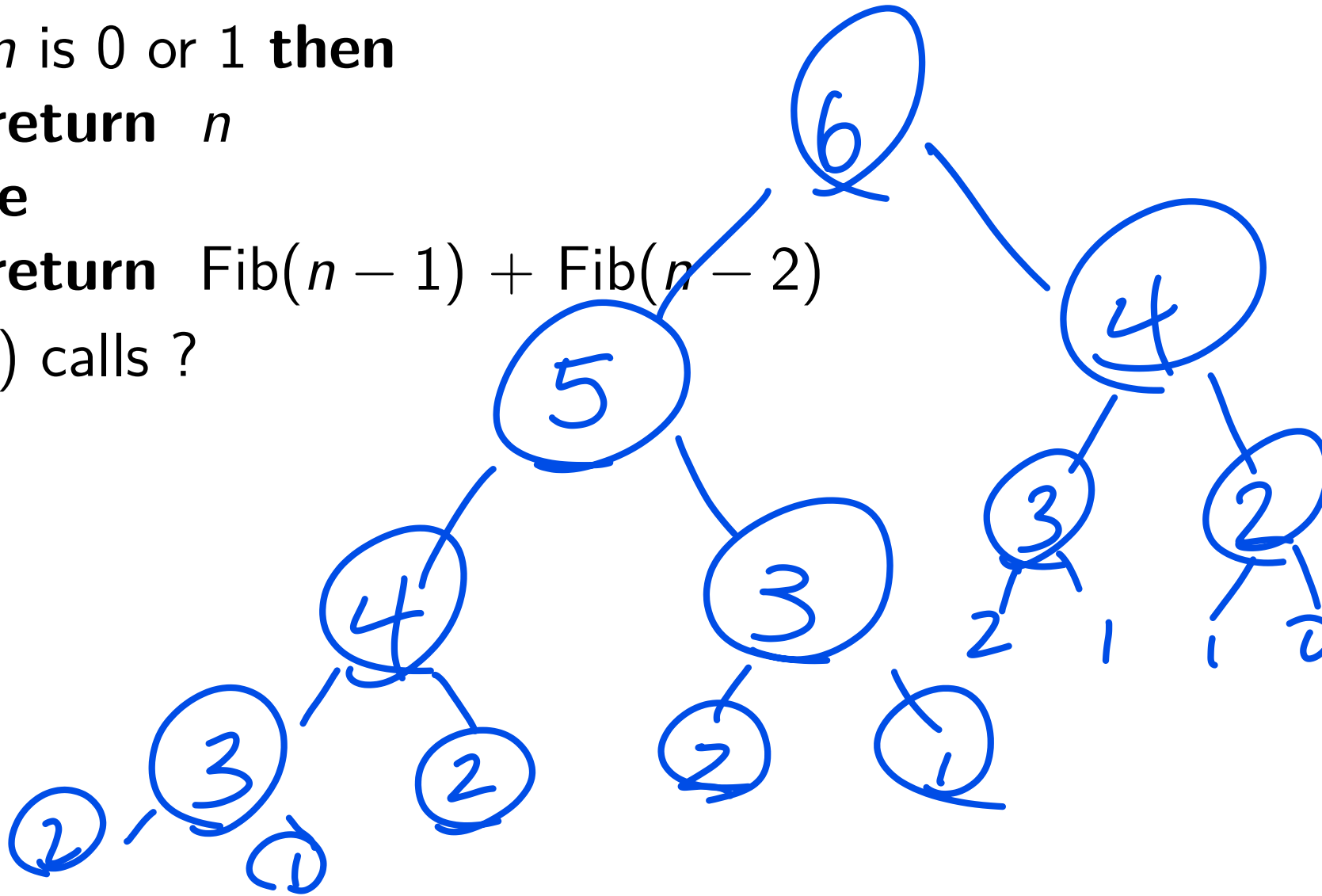
if n is 0 or 1 **then**

return n

else

return Fib($n-1$) + Fib($n-2$)

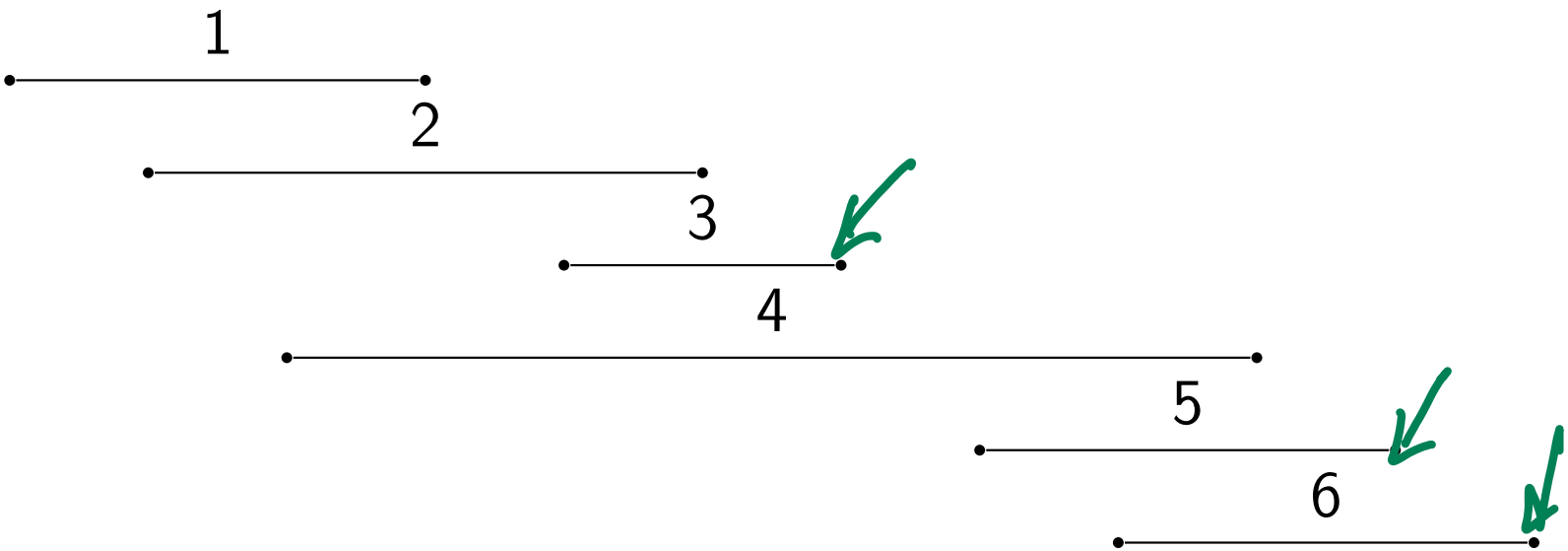
Fib(5) calls ?



Warm-Up Question 2

- Given n intervals, $1 \ldots n$,
  - each has start time s_i and finish time f_i.
- For each interval, compute a value $p[i]$
  - $p[i] = j$ means j is the *latest* f_j such that $f_j \leq s_i$
  - If no intervals end before s_i, then $f[i] = 0$.
- Intervals are already sorted by finish time.

Example:

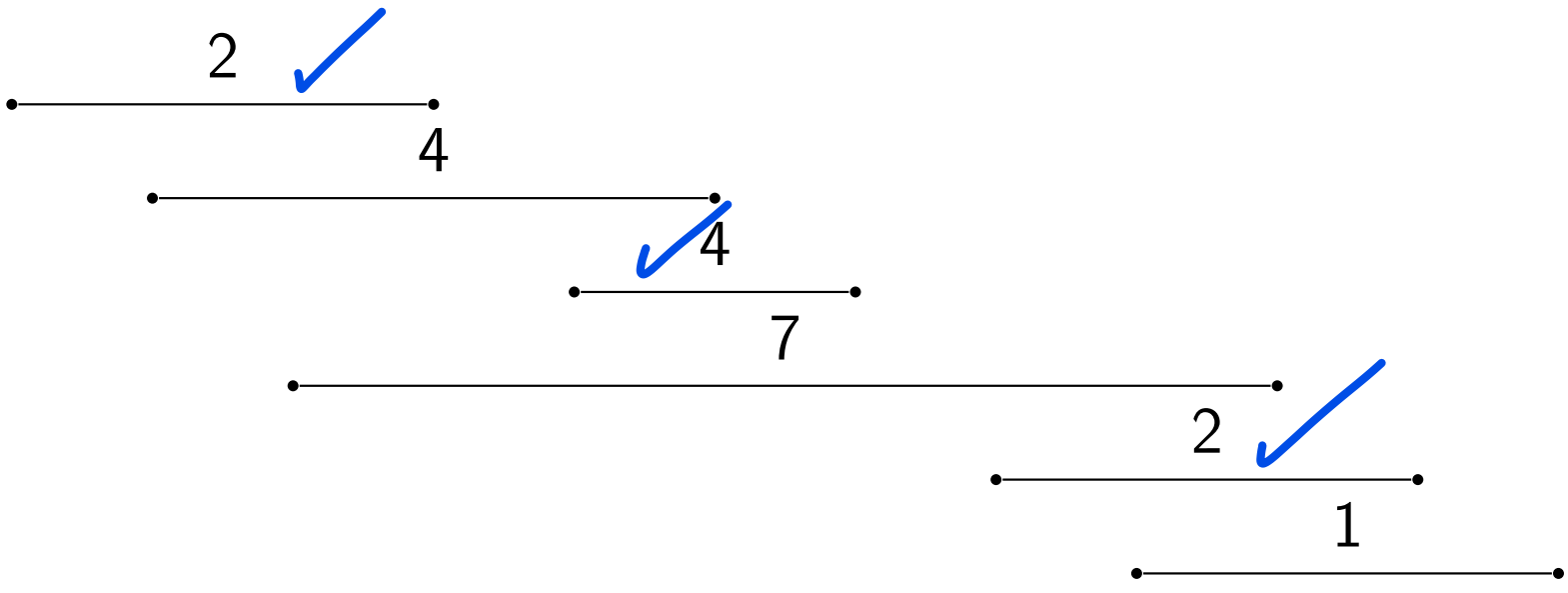


Warm-Up Question 2

Warm-up(int n, intervals $[s_1, f_1]$, $[s_2, f_2], \ldots [s_n, f_n]$)
  Sort intervals by finish time (if not already)
   for each interval $[s_i, f_i]$
      binary search for latest $f_j \leq s_i$

Interval Scheduling Problem Statement

- Which classes should take next quarter?
- The classes all meet once a day,
  - at different times and lengths
  - are worth different amounts of credits.
- Maximize amount of credits earned in quarter
- Without having to skip any classes



Interval Scheduling: Recursive Solution

index of last ending interval under consideration

- Key : your friend will take class i xor won't

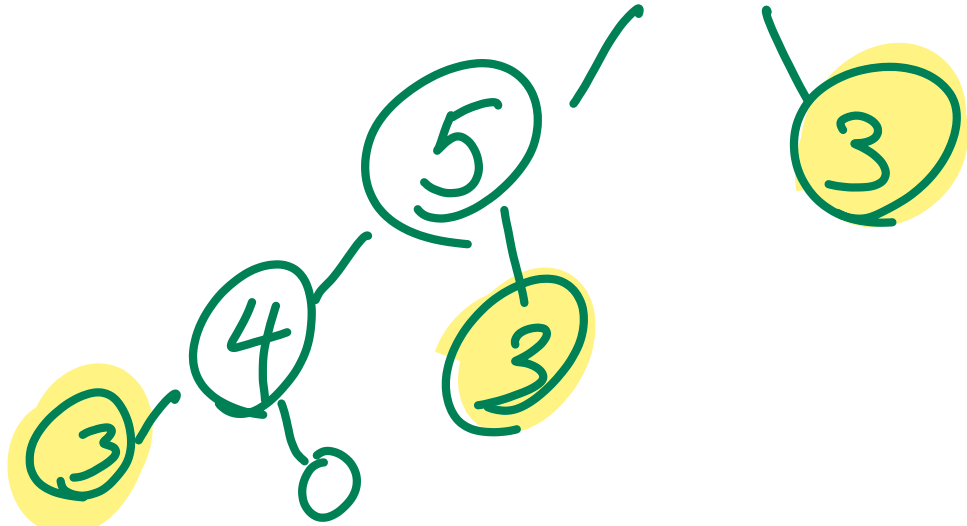
```
WIS(i) // opt # of credits, intervals 1...i
    // Base Case:
    if (0==i) return 0;
    // If my friend doesn't take class i:
a=  value_if_not_taken = WIS(i-1)

    // If my friend takes class i:
b=  value_if_taken = v_i + WIS(p[i])   (see warm up)

    //return something:
    return max(a, b)
```

Interval Scheduling: Recursive Implementation

```
WIS(i)
  if i is 0 then
    return 0
  // value_if_not_taken = WIS(i − 1)
  // value_if_taken = v_i + WIS(p[i])
  return  max( WIS(i − 1), v_i + WIS(p[i]) )
```

recursive substructure

- To solve: call `WIS(6)` for this input.



overlapping subproblems

Interval Scheduling: Iterative Solution

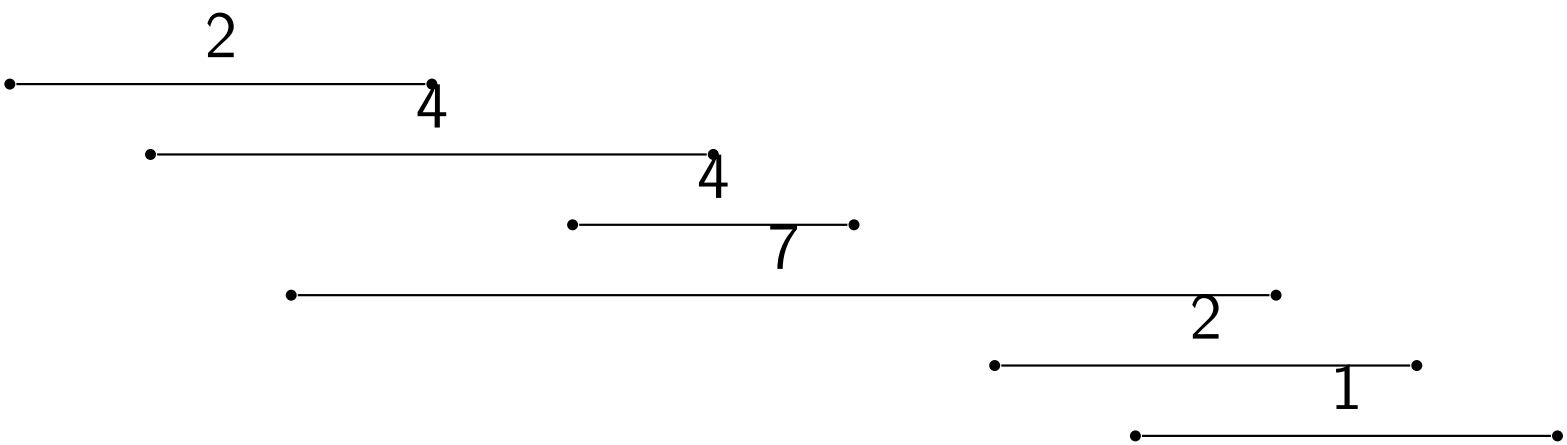
- WIS[i] needs only info from WIS[$0 \ldots i-1$]

- We can write an iterative solution.

declare WIS [0 ... n]
WIS [0] = 0
for i=1 to n
    WIS[i] = max(WIS [i-1],
                 v_i + WIS [p[i]])

Interval Scheduling: Table

i	$p[i]$	$\text{WIS}(\text{p}(i)) + v_i$	$\text{WIS}(i-1)$	$\text{WIS}(i)$
0	N/A	N/A	N/A	0
1	0	0 +2	0	2
2	0	0 +4	2	4
3	1	2 +4	4	6
4	0	0 +7	6	7
5	3	6 +2	7	8
6	3	6 +1	8	8



What classes to take?

- Now we have `WIS[...]` filled in.
- Instead of `return WIS[n]`, output courses.
- Hint: take course n or no?

```
i ← n
While i > 0 :
    if WIS[i] == WIS[i-1] :
        i--
    else :
        output i
        i = p[i]
```

Solving with Dynamic Programming

If asked for a dynamic programming solution:

- Describe *in English* the function
  - Not *how* it works (yet)
  - Yes *what it solves*.
  - Skipping this step = 0 on problem
- Give that function a meaningful variable name.
  - Not "OPT" or "DP" or "table"
  - Not a single letter either.
- Give recursive formulation.