1 Introduction to Dynamic Programming

1.1 Fibonacci Numbers

Source code: `w06A-fib.pdf`

- Draw the recursive call tree for `fib(5)` invoked on an instance of the `FibRec` class.
- How many times does `fib(2)` occur in the tree?
- Why is `FibRec.fib()` extremely slow?

- Draw the recursive call tree for `fib(5)` invoked on an instance of the `FibMemo` class.

FibMemo uses an array ("table") to memoize previously computed results. It turns out we can fill the array in directly. Pick a few cells in the following table. For each one, draw an arrow(s) to the cells that its value depends on.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

- Trace the execution of `fib(14)` invoked on an instance of the `FibDP` class.

1.2 Text Segmentation

Recall:

$$Splittable(i) = \begin{cases} \textsc{True} & \text{if } i > n \\ \displaystyle\bigvee_{j=i}^{n} \big(\textsc{IsWord}(i,j) \wedge Splittable(j+1)\big) & \text{otherwise} \end{cases}$$

```
⟨⟨Is the suffix $A[i..n]$ Splittable?⟩⟩
SPLITTABLE($i$):
  if $i > n$
      return TRUE
  for $j \leftarrow i$ to $n$
      if ISWORD($i, j$)
          if SPLITTABLE($j + 1$)
              return TRUE
  return FALSE
```

Pick $i = 5$ (as an example). Which cells of the array $S[0..14]$ does $S[5]$ depend on?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

- Write a dynamic programming implementation of Splittable.

Review Section 3.4 (page 105) in the textbook.