Lecture 26: Setup Work
11:00 AM, Apr 6, 2020

Contents

1 A Motivating Problem: Fibonacci Numbers 1

2 Prep Question 2

Motivating Question

How can we improve run-time efficiency by removing redundant computations?

1 A Motivating Problem: Fibonacci Numbers

The Fibonacci sequence is a well-known sequence of numbers in mathematics in which each number is the sum of the two numbers before it (the sequence starts with 0 and 1 as its base cases). Specifically, the sequence looks like: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Computing the $n^{th}$ Fibonacci number is a standard recursive problem in computer science (this version is in Java):

```java
public int fib(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fib(n-1) + fib(n-2);
}
```

What calls get made if we use this definition to compute $\text{fib}(4)$?

![Diagram of recursive calls for fib(4)]

Notice that we repeat several calls. Since there are no assignment operations in this code, $\text{fib}$ returns the same output every time it receives the same input. Thus, there is potential for a lot of
wasted computation here. We could make our implementation more time-efficient by storing the results of \( \text{fib} \) each time it is called, then re-using the stored result if we call the function with the same inputs again later. When computing \( \text{fib}(4) \), for example, each of the shaded calls below could get looked up, rather than recomputed.

\[
\begin{array}{c}
\text{fib}(4) \\
\downarrow \\
\text{fib}(3) \\
\downarrow \\
\text{fib}(2) \quad \text{fib}(2) \\
\downarrow \quad \downarrow \\
\text{fib}(1) \quad \text{fib}(1) \\
\downarrow \quad \downarrow \\
\text{fib}(0) \quad \text{fib}(0)
\end{array}
\]

2 Prep Question

How might you go about modifying the fibonacci code above to avoid recomputing results that you had computed before? **We’re not asking you to write the code.** Rather, think about how you could track which computations you’ve already done and how you could access and reuse the results. Think about which data structures you would use and how you would work those into the original program.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS18 document by filling out the anonymous feedback form: [https://cs.brown.edu/courses/cs018/feedback](https://cs.brown.edu/courses/cs018/feedback)