Joshua Collins

Principles of Software

Section 1

1/22/2024

Question 3: Answering Questions About the Fibonacci Code

1. Why did Fibonacci fail the testThrowsIllegalArgumentException test? What did you have to do to fix it?

   For this case, the testThrowsIllegalArgumentException error was caused by the if statement that determines if integer n is a negative number or not. This issue only occurs if n is equal to zero since zero is a non-negative number. To fix this issue, I changed the line from "if (n <= 0) {" to "if (n < 0) {".

2. Why did Fibonacci fail the testBaseCase test? What (if anything) did you have to do to fix it?

   For this case, the failure of the base case test was due to the if statment returning n if n was less than or equal to two. The base cases for the fibonacci sequence could only be one and zero. To fix this issue, I changed the of statement from "} else if (n <= 2) {" to "} else if (n <= 1) {". This eliminated the issue of two being considered a base case.

3. Why did Fibonacci fail the testInductiveCase test? What (if anything) did you have to do to fix it?

   For this case, the error was cause by an incorrect mathematical equation being used to generate the next term in the fibonacci sequence. To take care of this issue, I changed the return statement from "return getFibTerm(n + 1) - getFibTerm(n - 2);" to "return getFibTerm(n - 1) + getFibTerm(n - 2);". This outputs the correct next term in the fibonacci sequence as the function undoes the recursion.

4. Why did Fibonacci fail the testLargeN test? What (if anything) did you have to do to fix it?

   This error was caused by the same issue that caused the error in the testInductiveCase test. The equation used to generate the next term in the sequence was incorrect, leading to the large number test producing an incorrect output. To take care of this issue, I changed the return statement from "return getFibTerm(n + 1) - getFibTerm(n - 2);" to "return getFibTerm(n - 1) + getFibTerm(n - 2);". This outputs the correct next term in the fibonacci sequence as the function undoes the recursion.

5. What was causing Fibonacci to be so slow on testLargeN test? What did you do to make Fibonacci faster while still preserving the recursive nature of your implementation?

    The long runtime was due to the repetitive nature of the original fibonacci function. The recursion often would recurse down to the base cases and terms already computed. To save time I added a public Hash Map to save all terms smaller than n to. This way, if a term has been calculated already, the function can search for it in the Hash Map. Finding terms in the Hash Map is O(1) and inserting the occasional term is O(log n), which nevertheless cuts down the runtime of the function upon reaching higher values on n. Also, the value of the 60th term outputs is too large for an integer to store, so I changed the Fibonacci function to a long function to allow for more space to store sequence terms.