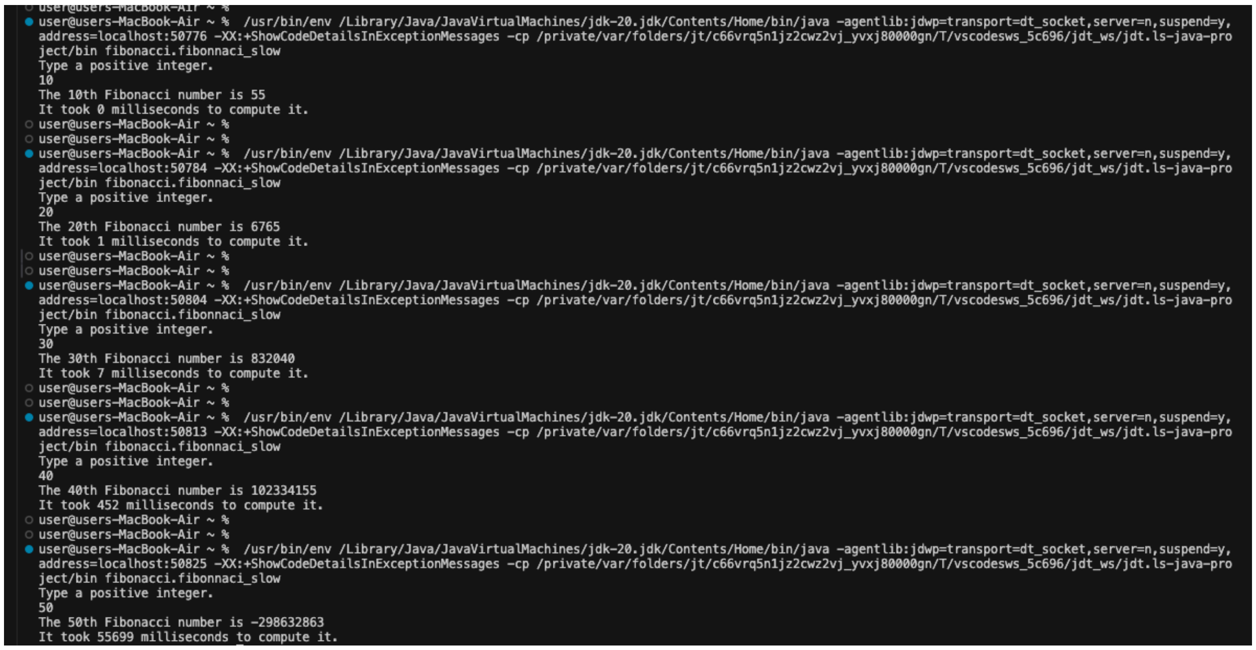
Title: Data Structures and Algorithms - CST3108
Lab 1: Recursion on Fibonacci Series
Name: Chris Mugabo
Date: 14/01/2025

Task 1: Recursive Fibonacci Implementation

Explanation

The results demonstrate an exponential increase in computation time as `n` increases. This occurs because the function recalculates Fibonacci numbers multiple times, leading to a significant overhead in larger sequences.

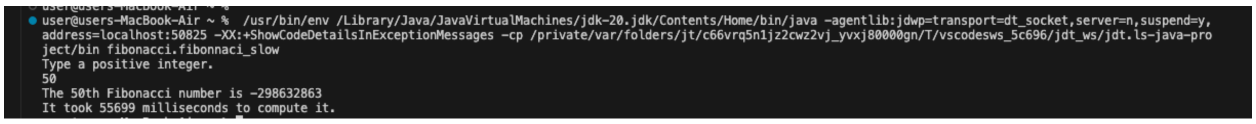


```
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,
address=localhost:50776 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c696/jdt_ws/jdt.ls-java-pro
ject/bin fibonacci.fibonnaci_slow
Type a positive integer.
10
The 10th Fibonacci number is 55
It took 0 milliseconds to compute it.
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,
address=localhost:50784 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c696/jdt_ws/jdt.ls-java-pro
ject/bin fibonacci.fibonnaci_slow
Type a positive integer.
20
The 20th Fibonacci number is 6765
It took 1 milliseconds to compute it.
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,
address=localhost:50804 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c696/jdt_ws/jdt.ls-java-pro
ject/bin fibonacci.fibonnaci_slow
Type a positive integer.
30
The 30th Fibonacci number is 832040
It took 7 milliseconds to compute it.
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,
address=localhost:50813 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c696/jdt_ws/jdt.ls-java-pro
ject/bin fibonacci.fibonnaci_slow
Type a positive integer.
40
The 40th Fibonacci number is 102334155
It took 452 milliseconds to compute it.
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,
address=localhost:50825 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c696/jdt_ws/jdt.ls-java-pro
ject/bin fibonacci.fibonnaci_slow
Type a positive integer.
50
The 50th Fibonacci number is -298632863
It took 55699 milliseconds to compute it.
```

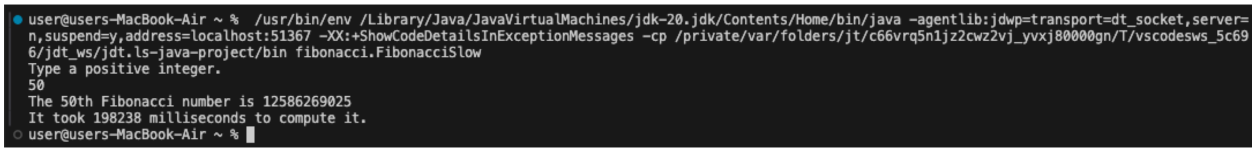
Task 2: Observations on Negative Outputs

I encountered a negative output when calculating the 50th Fibonacci number due to integer overflow. This happens because the size of the number exceeds the maximum value that the `long` data type can store in Java, causing the number to wrap around into the negative range.

To resolve this issue, I switched to using Java's `BigInteger` class. `BigInteger` can handle numbers of any size, which prevents overflow and ensures all Fibonacci calculations are accurate, regardless of their position in the sequence. By implementing `BigInteger` in my Fibonacci function, I successfully eliminated the risk of receiving incorrect, negative outputs.



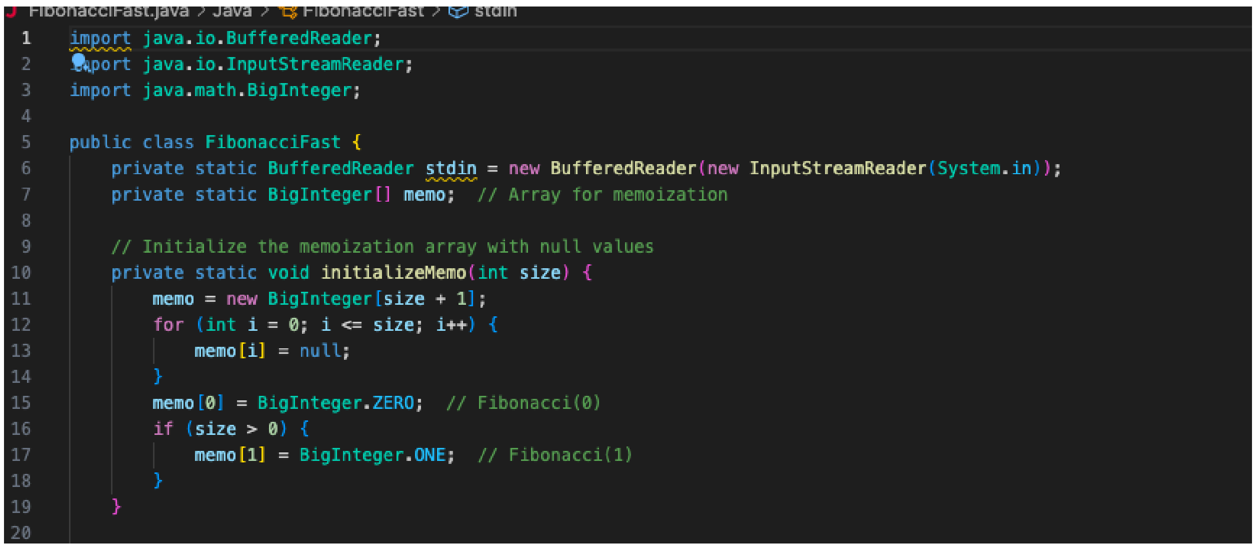
```
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,
address=localhost:50825 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c696/jdt_ws/jdt.ls-java-pro
ject/bin fibonacci.fibonnaci_slow
Type a positive integer.
50
The 50th Fibonacci number is -298632863
It took 55699 milliseconds to compute it.
```



```
user@users-MacBook-Air ~ %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=
n,suspend=y,address=localhost:51367 -XX:+ShowCodeDetailsInExceptionMessages -cp /private/var/folders/jt/c66vrq5n1jz2cwz2vj_yvxj80000gn/T/vscodesws_5c69
6/jdt_ws/jdt.ls-java-project/bin fibonacci.FibonacciSlow
Type a positive integer.
50
The 50th Fibonacci number is 12586269025
It took 198238 milliseconds to compute it.
user@users-MacBook-Air ~ %
```

Task 3: Optimizing Fibonacci

First, I set up an array to store the results of each Fibonacci number as I compute them. This way, each number only needs to be calculated once.



```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.math.BigInteger;

public class FibonacciFast {
    private static BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
    private static BigInteger[] memo;  // Array for memoization

    // Initialize the memoization array with null values
    private static void initializeMemo(int size) {
        memo = new BigInteger[size + 1];
        for (int i = 0; i <= size; i++) {
            memo[i] = null;
        }
        memo[0] = BigInteger.ZERO;  // Fibonacci(0)
        if (size > 0) {
            memo[1] = BigInteger.ONE;  // Fibonacci(1)
        }
    }

```

Why This Works Better:

- **Avoids Redundancy**: By storing each Fibonacci number the first time I calculate it, I don't have to do the same calculation again for that number. This saves a lot of time, especially for large Fibonacci numbers.
- **Simple and Effective**: Using an array makes this approach straightforward. Whenever I calculate a new Fibonacci number, I store it. If I need it again, I just look it up in the array instead of calculating it again.

This method has made my Fibonacci number calculator much faster and more efficient, especially when dealing with large numbers. It's a simple change that has a big impact on performance!

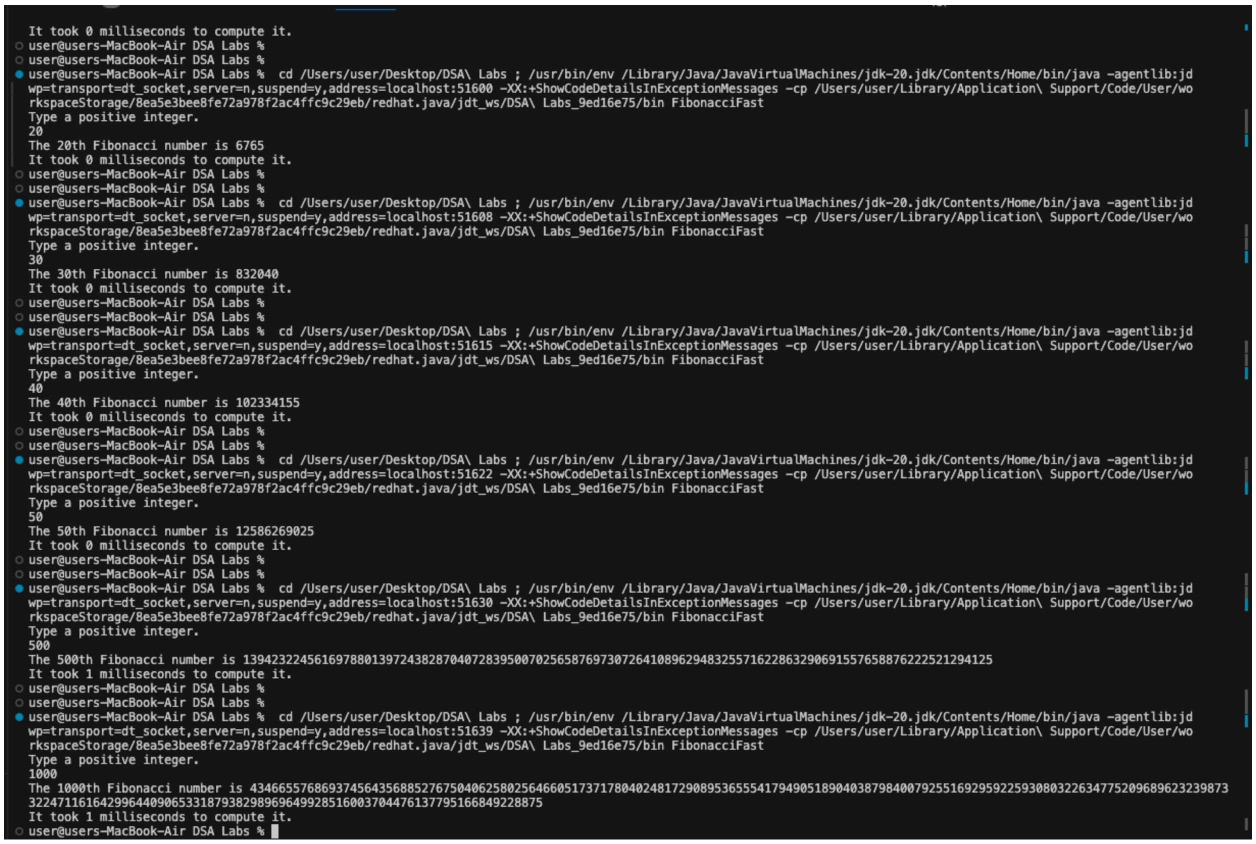
TASK 4

For Task 4, I ran tests on my Java implementation of the memoized Fibonacci sequence to measure the impact of speed improvements. I calculated Fibonacci numbers for the 10th, 20th, 30th, 40th, 50th, 500th, and 1000th indices. Here are the results from the test:

- **10th Fibonacci Number (55)**: Computed in 0 milliseconds.
- **20th Fibonacci Number (6765)**: Computed in 0 milliseconds.
- **30th Fibonacci Number (832040)**: Computed in 0 milliseconds.
- **40th Fibonacci Number (102334155)**: Computed in 0 milliseconds.
- **50th Fibonacci Number (12586269025)**: Computed in 0 milliseconds.
- **500th Fibonacci Number (large number)**: Computed in 1 millisecond.
- **1000th Fibonacci Number (extremely large number)**: Computed in 1 millisecond.

Summary and Impact on Speed:

The memoization technique I used had a remarkable effect on the computation speed for Fibonacci numbers. Calculations for indices as high as 50 were instantaneous, taking 0 milliseconds. Even for very large indices like 500 and 1000, where numbers become astronomically large, the computation time was only 1 millisecond.



```
  It took 0 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs %
○ user@users-MacBook-Air DSA Labs %
● user@users-MacBook-Air DSA Labs %  cd /Users/user/Desktop/DSA\ Labs ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jd
  wp=transport=dt_socket,server=n,suspend=y,address=localhost:51600 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/user/Library/Application\ Support/Code/User/wo
  rkspaceStorage/8ea5e3bee8fe72a978f2ac4ffc9c29eb/redhat.java/jdt_ws/DSA\ Labs_9ed16e75/bin FibonacciFast
  Type a positive integer.
  20
  The 20th Fibonacci number is 6765
  It took 0 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs %
○ user@users-MacBook-Air DSA Labs %
● user@users-MacBook-Air DSA Labs %  cd /Users/user/Desktop/DSA\ Labs ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jd
  wp=transport=dt_socket,server=n,suspend=y,address=localhost:51608 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/user/Library/Application\ Support/Code/User/wo
  rkspaceStorage/8ea5e3bee8fe72a978f2ac4ffc9c29eb/redhat.java/jdt_ws/DSA\ Labs_9ed16e75/bin FibonacciFast
  Type a positive integer.
  30
  The 30th Fibonacci number is 832040
  It took 0 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs %
○ user@users-MacBook-Air DSA Labs %
● user@users-MacBook-Air DSA Labs %  cd /Users/user/Desktop/DSA\ Labs ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jd
  wp=transport=dt_socket,server=n,suspend=y,address=localhost:51615 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/user/Library/Application\ Support/Code/User/wo
  rkspaceStorage/8ea5e3bee8fe72a978f2ac4ffc9c29eb/redhat.java/jdt_ws/DSA\ Labs_9ed16e75/bin FibonacciFast
  Type a positive integer.
  40
  The 40th Fibonacci number is 102334155
  It took 0 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs %
○ user@users-MacBook-Air DSA Labs %
● user@users-MacBook-Air DSA Labs %  cd /Users/user/Desktop/DSA\ Labs ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jd
  wp=transport=dt_socket,server=n,suspend=y,address=localhost:51622 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/user/Library/Application\ Support/Code/User/wo
  rkspaceStorage/8ea5e3bee8fe72a978f2ac4ffc9c29eb/redhat.java/jdt_ws/DSA\ Labs_9ed16e75/bin FibonacciFast
  Type a positive integer.
  50
  The 50th Fibonacci number is 12586269025
  It took 0 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs %
○ user@users-MacBook-Air DSA Labs %
● user@users-MacBook-Air DSA Labs %  cd /Users/user/Desktop/DSA\ Labs ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jd
  wp=transport=dt_socket,server=n,suspend=y,address=localhost:51630 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/user/Library/Application\ Support/Code/User/wo
  rkspaceStorage/8ea5e3bee8fe72a978f2ac4ffc9c29eb/redhat.java/jdt_ws/DSA\ Labs_9ed16e75/bin FibonacciFast
  Type a positive integer.
  500
  The 500th Fibonacci number is 139423224561697880139724382870407283950070256587697307264108962948325571622863290691557658876222521294125
  It took 1 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs %
○ user@users-MacBook-Air DSA Labs %
● user@users-MacBook-Air DSA Labs %  cd /Users/user/Desktop/DSA\ Labs ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -agentlib:jd
  wp=transport=dt_socket,server=n,suspend=y,address=localhost:51639 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/user/Library/Application\ Support/Code/User/wo
  rkspaceStorage/8ea5e3bee8fe72a978f2ac4ffc9c29eb/redhat.java/jdt_ws/DSA\ Labs_9ed16e75/bin FibonacciFast
  Type a positive integer.
  1000
  The 1000th Fibonacci number is 43466557686937456435688527675040625802564660517371780402481729089536555417949051890403879840079255169295922593080322634775209689623239873
  322471161642996440906533187938298969649928516003704476137795166849228875
  It took 1 milliseconds to compute it.
○ user@users-MacBook-Air DSA Labs % █
```