Concordia University

COMP 352 – Data Structures and Algorithms

Fall 2024

Programming Assignment 1

Dr. Aiman Hanna

Submitted by
Geon Kim (40264507)

Friday, October 4th, 2024

a)

Linear Recursion

```
FUNCTION linearOdd(n)
   IF n <= 3 THEN
      RETURN [1, 1, 1]
   ELSE
      temp = linearOdd(n - 1)

      answer = [temp[0] + temp[1] + temp[2], temp[0], temp[1]]

      RETURN answer
   END IF
END FUNCTION
```

Exponential Recursion

```
FUNCTION oddonacciExponential(n)
        IF n == 0 THEN
                RETURN 1
        ELSE if n == 1 THEN
                RETURN 1
        ELSE IF n == 2 THEN
                RETURN 1
        ELSE n == 3
                RETURN 1
        END IF
        RETURN oddonacciExponential(n - 1) + oddonacciExponential(n - 2) +
        oddonacciExponential(n -3)
END FUNCTION
```

b)

Analysis of Algorithm Complexities

1. **Exponential Complexity in the First Algorithm (ExponentialOddonacci):**

   - **Time Complexity: $O(3^n)$**
   - Each of these three recursive calls will again split into three more recursive calls, creating a tree-like
   - For large values of n, this approach becomes very inefficient due to repeated calculations of the same subproblems.

2. **Linear Complexity in the Second Algorithm (LinearOddonacci):**

   - **Time Complexity: O(n)**
   - The algorithm only loops through the sequence once and computes each Oddonacci number in constant time.
   - Each step only involves updating an array of three values, so the number of operations grows linearly with n.

Bottleneck Resolution: The second algorithm uses an array to store the last three computed Oddonacci values, resolving the redundant calculations of the exponential algorithm. This approach avoids recomputation by reusing stored values, allowing the next number to be calculated in constant time.

c)

Do any of the previous two algorithms use tail recursion? Why or why not?

- **No**, neither the exponential nor the linear algorithms use tail recursion.
- In both algorithms, after the recursive call, additional operations are performed. Tail recursion requires the the recursive call is the last operation in the function.

Can a tail-recursive version of the Oddonacci calculator be designed?

- **Yes**, a tail-recursive version can be designed.

```java
public class TailRecursiveOddonacci {

    // Tail-recursive helper method
    private static long tailOdd(int n, long a, long b, long c) {  2 usages
        if (n == 0) return a; // Return the first Oddonacci number
        if (n == 1) return b; // Return the second Oddonacci number
        if (n == 2) return c; // Return the third Oddonacci number
        // Recursive call, updating values
        return tailOdd( n: n - 1, b, c,  c: a + b + c);
    }

    // Public method to start the calculation
    public static long oddonacciTail(int n) {  1 usage
        return tailOdd(n,  a: 1,  b: 1,  c: 1); // Start with the first three Oddonacci numbers
    }

    public static void main(String[] args) {
        for (int i = 0; i <= 20; i++) {
            System.out.println("Oddonacci(" + i + "): " + oddonacciTail(i));
        }
    }
}
```