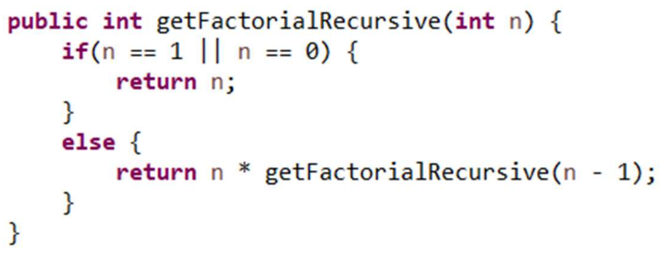
Tutorial Course 1 : Iterative And Recursive

Akash SELVARATNAM

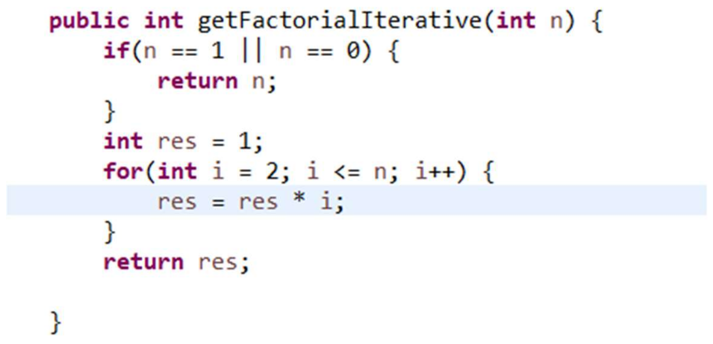
Part 1 : Factorial

1. Write a recursive solution



```java
public int getFactorialRecursive(int n) {
    if(n == 1 || n == 0) {
        return n;
    }
    else {
        return n * getFactorialRecursive(n - 1);
    }
}
```

2. Implement an iterative solution via letter swapping



```java
public int getFactorialIterative(int n) {
    if(n == 1 || n == 0) {
        return n;
    }
    int res = 1;
    for(int i = 2; i <= n; i++) {
        res = res * i;
    }
    return res;

}
```

3. Compare the efficiency of both solutions and plot the complexity of both solutions.

For the recursive solution:

The complexity of the best case is O(1) and the complexity of the worst case is O(n)

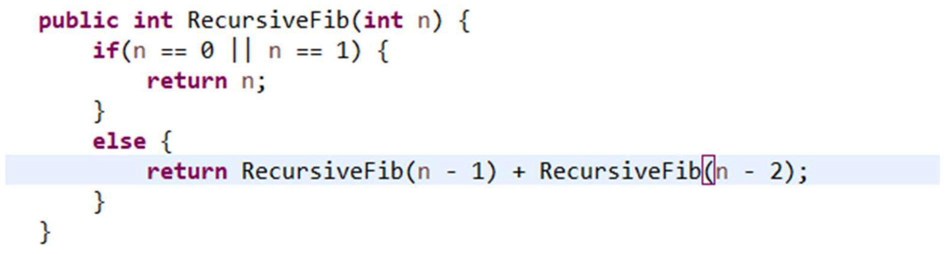
For the iterative solution:

The complexity of the best case is O(1) and the complexity of the worst case is O(n)

I think for each solution, the efficiency is the same.

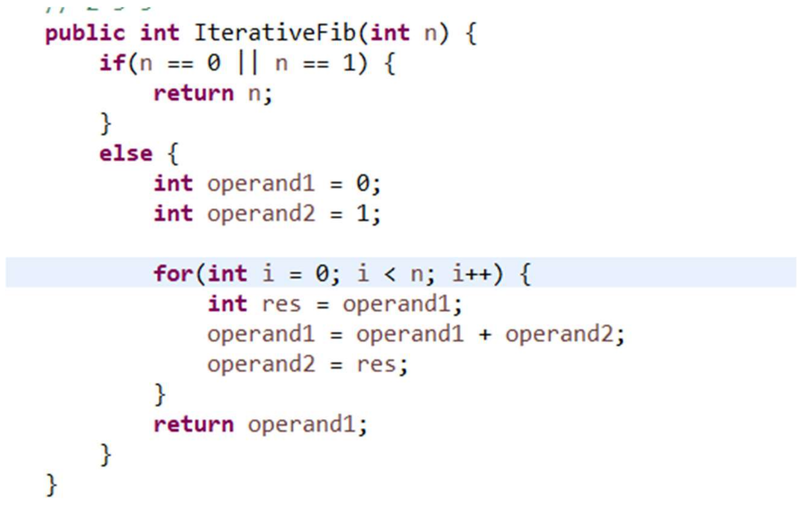
Part 2 : Fibonacci

1. Create a method called RecursiveFib(n), which given a number n, computes and returns it’s Fibonacci sequence using a recursive algorithm



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

2. Create a method called IterativeFib(n), which given a number n, computes and returns it’s Fibonacci sequence using an iterative algorithm



```java
public int IterativeFib(int n) {
    if(n == 0 || n == 1) {
        return n;
    }
    else {
        int operand1 = 0;
        int operand2 = 1;

        for(int i = 0; i < n; i++) {
            int res = operand1;
            operand1 = operand1 + operand2;
            operand2 = res;
        }
        return operand1;
    }
}
```

3. What is the theoretical complexity for computing Fibonacci

The theoretical complexity for computing Fibonacci is O(n)

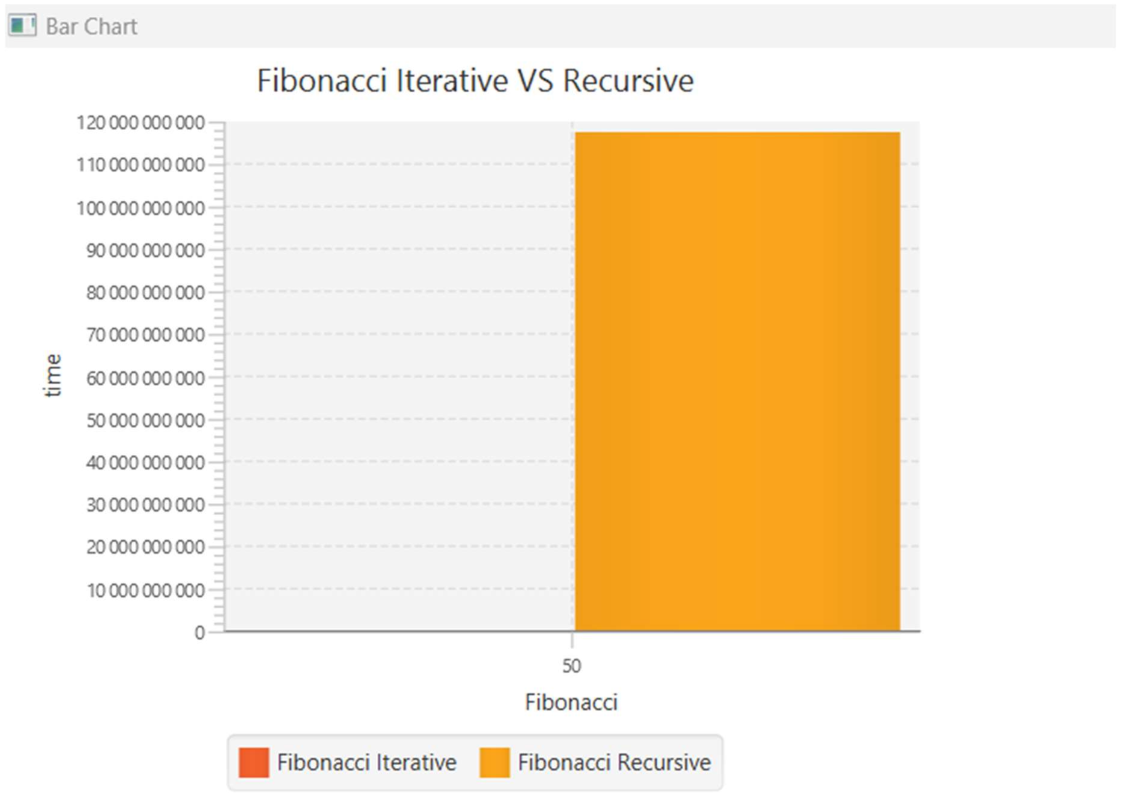
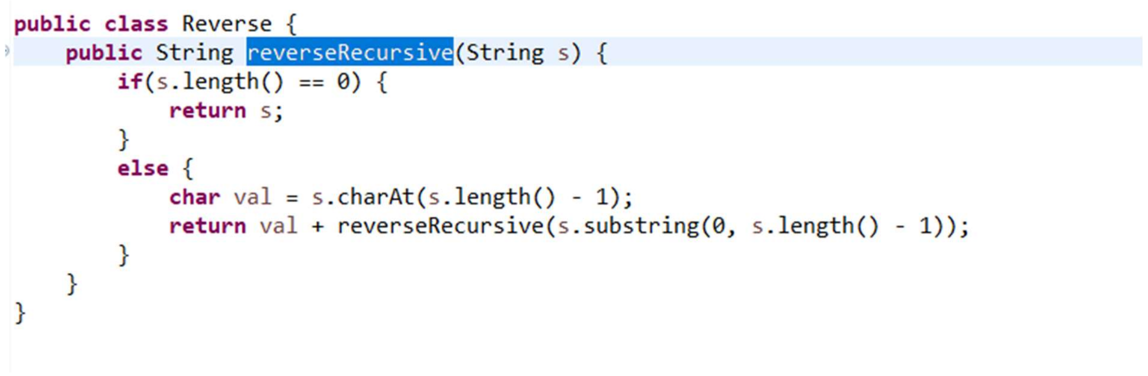


Figure 1 : Fibonacci Iterative VS Recursive Fibonacci(50)

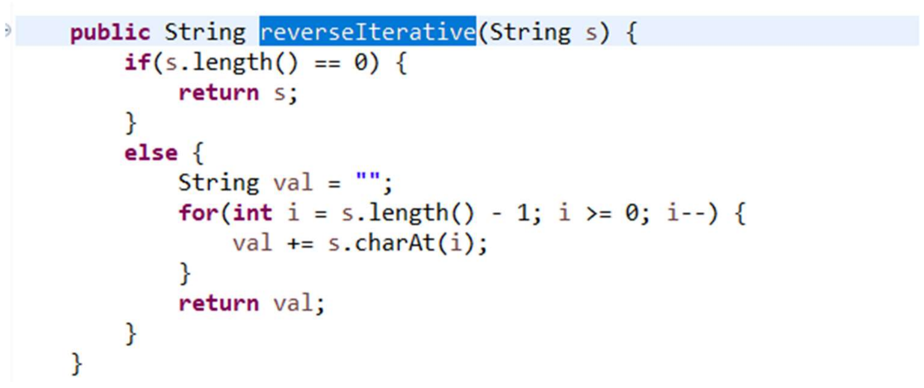
Part 3 : String reversal

4. Write a recursive solution



```java
public class Reverse {
    public String reverseRecursive(String s) {
        if(s.length() == 0) {
            return s;
        }
        else {
            char val = s.charAt(s.length() - 1);
            return val + reverseRecursive(s.substring(0, s.length() - 1));
        }
    }
}
```

5. Implement an iterative solution via letter swapping



```
    public String reverseIterative(String s) {
        if(s.length() == 0) {
            return s;
        }
        else {
            String val = "";
            for(int i = s.length() - 1; i >= 0; i--) {
                val += s.charAt(i);
            }
            return val;
        }
    }
```

6. Compare the efficiency of both solutions and plot the complexity of both solutions, you should consider temporal.

The efficiency of both solutions are 0(n) in the worst case and O(1) in the best case.

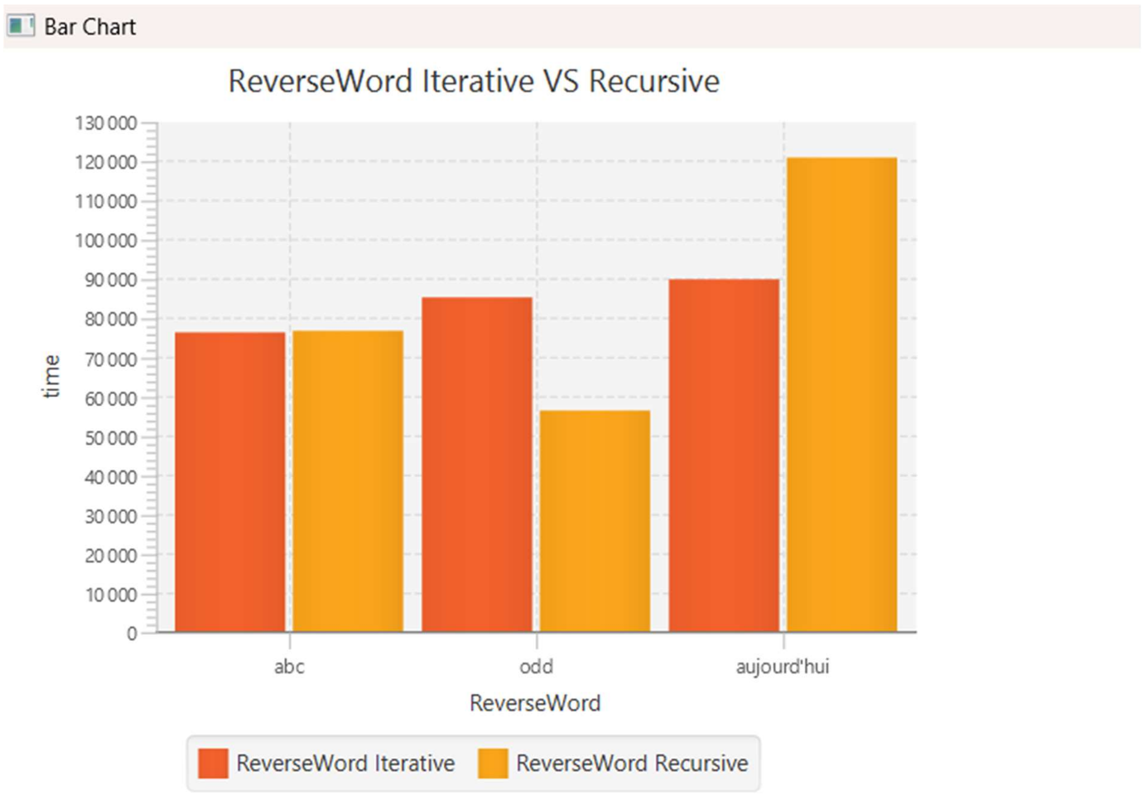


Figure 2 : Recursive VS Iterative ReverseWord