Practical 1

Aim: 1. Implement and analyze the algorithms given below.

1.1 Factorial of a given number (Iterative and Recursive)

1.2 Fibonacci Series (Iterative and Recursive)

1.3 Linear Search (Iterative and Recursive)

```java
class IttFact {
    public static void main(String[] args) {
        int num = 10;
        int loop = 0;

        int fact = 1;
        loop++; //Init of fact
        loop++; //Init of i
        for(int i = 1;i <= num;i++) {
            fact *= i;
            loop++; //Math operation
            loop++; //Condition checking
            loop++; //Increment
        }
        loop++; //Compensating for last condition check

        System.out.println("Fact = " + fact);
        System.out.println("Number of loops used: " + loop);
    }
}
```

```java
class RecFact {
    static int loop = 0;
    public static void main(String[] args) {
        for(int i = 1;i < 10001;i++) {
            System.out.println(i + "  " + loop_test(i));
            loop = 0;
        }
    }
    public static int loop_test(int num) {
        int fact = factorial(num);
        loop++;// for fact init;

        return loop;
    }

    public static int factorial (int n) {
        loop++;//for function being called
        loop++; //for condition checking
        if (n == 0) {
            loop++; //for return statment
            return 1;
        }
        loop++; //for return statment
        loop++; //for function call
        loop++; //for math opeartion
        return n * factorial(n - 1);
    }
}
```

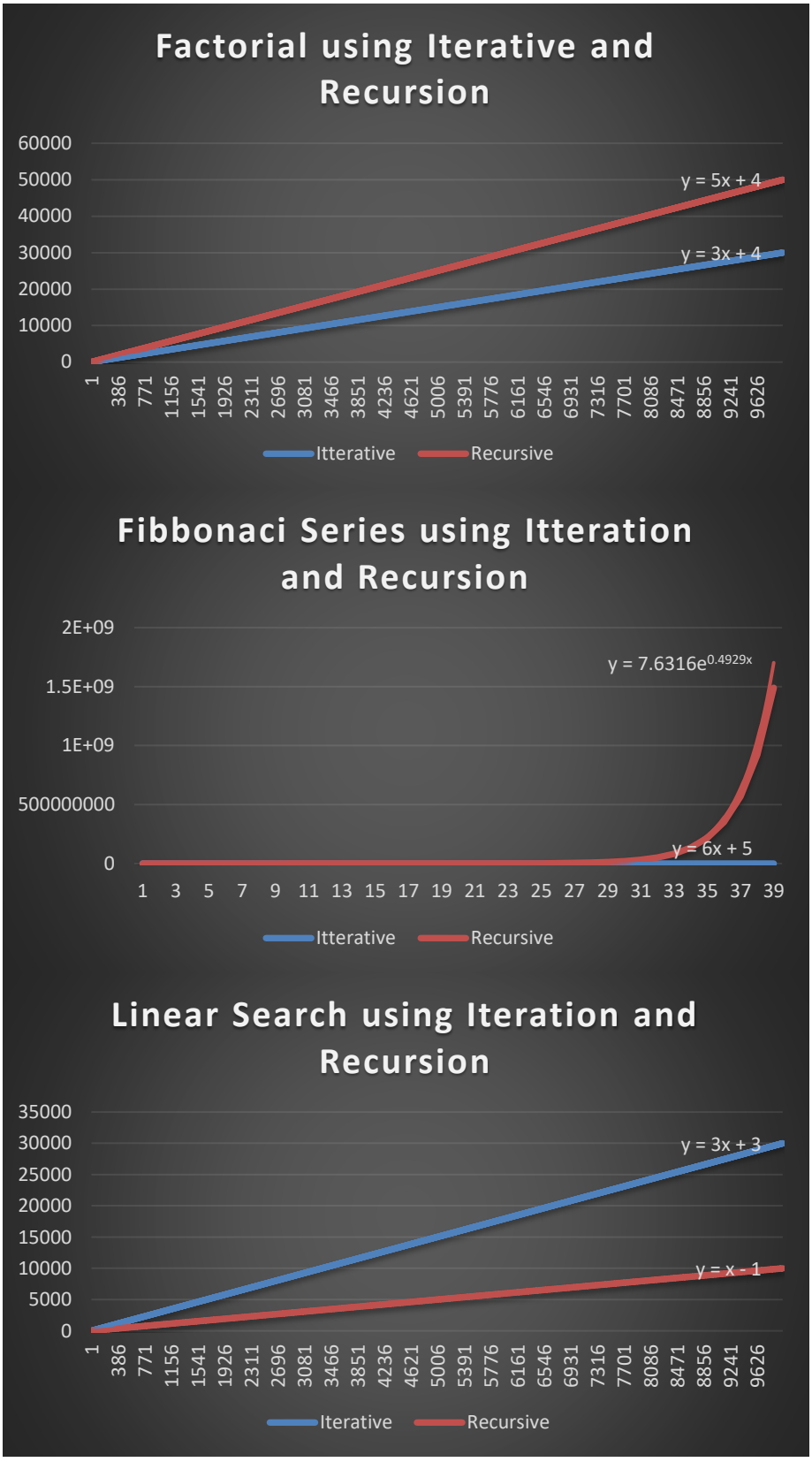
```java
class IttFibo {
    static int loop = 0;
    public static void main(String[] args) {
        for(int i = 1;i < 10001;i++) {
            printFibonacci(i);
            System.out.println(i + " " + loop);
            loop = 0;
        }
    }

    public static void printFibonacci(int n) {
        loop++; //for condition
        if (n <= 0) {
            return;
        }

        loop++; //init a
        loop++; //init b
        long a = 0, b = 1;
        // System.out.print("Fibonacci Series: ");

        loop++; //init i
        for (int i = 0; i < n; i++) {
            loop++; //condition checking
            loop++; //increment of i

            loop++; //declaration of temp
            loop++; //a + b
            long temp = a + b;

            loop++; //a = b;
            a = b;
            loop++; //b = temp;
            b = temp;
        }
        loop++; //conponsation for last condition check of loop
    }
}
```

```java
public class RecFibbo {
    static int loop = 0;
    public static void main(String[] args) {
        for(int i = 1;i <= 10001;i++) {
            fibo(i);
            System.out.println(i + " " + loop);
            loop = 0;
        }
    }

    static int fibo(int n) {
        loop++; //init i
        for (int i = 0; i < n; i++) {
            loop++; //condition checking
            loop++; //increment of i
            fibonacci(i);
        }
        loop++; //Conponsating for last condition check
        return loop;
    }

    public static long fibonacci(int n) {
        loop++; //function call

        loop++; //condition check
        if (n <= 1) {
            loop++; //return
            return n;
        } else {
            loop++; //n-1
            loop++; //n-2
            loop++; //addition
            loop++; //return
            return fibonacci(n - 1) + fibonacci(n - 2);
        }
    }
}
```

```java
class IttLinearSearch {
    static int loop = 0;
    public static void main(String[] args) {
        for(int i = 1;i < 10001;i++) {
            int[] arr = new int[i];
            arr[arr.length - 1] = 10;
            for(int j = 0;j < arr.length - 2;j++) {
                arr[j] = 1;
            }

            System.out.println(i + " " + linear_search(arr, 10));
            loop = 0;
        }
    }

    static int linear_search(int arr[], int search) {
        loop++; //Function Call
        loop++; //Condition Checking
        if (arr.length == 0) {
            return -1;
        }
        loop++; //Condition Checking
        loop++; //arr.length - 1
        if (arr[arr.length - 1] == search) {
            return arr.length - 1;
        }
        return linear_search(arr, search);
    }
}
```

```java
class RecLinearSearch {
    static int loop = 0;

    public static void main(String[] args) {
        for(int i = 1;i < 10001;i++) {
            int[] arr = new int[i];
            arr[arr.length - 1] = 10;
            for(int j = 0;j < arr.length - 2;j++) {
                arr[j] = 1;
            }

            linearsearch(arr, i, 10);

            System.out.println(i + " " + loop);
            System.out.println("found at index: " + linearsearch(arr, i, 10));
            loop = 0;
        }
    }

    static int linearsearch(int arr[], int size, int key) {
        // System.out.println("Loop: " + loop + "\t");
        loop++; //Function call
        loop++; //condition checking
        if(size == 0) {
            loop++; //Return statment
            return -1;
        }
        loop++; //condition checking
        loop++; //size - 1
        if (arr[size - 1] == key) {
            loop++; //size - 1
            loop++; //return
            return size - 1;
        }
            loop++; //size - 1
            loop++; //return
        return linearsearch(arr, size - 1, key);
    }
}
```

Graphs:


Factorial using Iterative and Recursion
60000
50000
40000
30000
20000
10000
0
y = 5x + 4
y = 3x + 4
1 386 771 1156 1541 1926 2311 2696 3081 3466 3851 4236 4621 5006 5391 5776 6161 6546 6931 7316 7701 8086 8471 8856 9241 9626
Itterative
Recursive
Fibbonaci Series using Itteration and Recursion
2E+09
1.5E+09
1E+09
500000000
0
y = 7.6316e0.4929x
y = 6x + 5
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
Itterative
Recursive
Linear Search using Iteration and Recursion
35000
30000
25000
20000
15000
10000
5000
0
y = 3x + 3
y = x - 1
1 386 771 1156 1541 1926 2311 2696 3081 3466 3851 4236 4621 5006 5391 5776 6161 6546 6931 7316 7701 8086 8471 8856 9241 9626
Iterative
Recursive

Conclusion: After the experiment, it is concluded that Iterative is most often better then recursive (often exponentially so) but, there might exist a scenario where the recursive method shall be rather preferred over the iterative method as seen while implementing linear search using the two methods.