CS 1083

Assignment #7

Author: Yulong Wang

Id: 3713596

1. Source Code:

a. Seq.java:

```java
import java.lang.reflect.Array;
import java.util.ArrayList;

/**
Seq.java
A utility class that provide methods to compute elements of the
recursive sequence.
*/
public class Seq {
    private static ArrayList<Integer> memory;

    /**
     * Recursively computes seq(n)
     *
     * @param n Non-negative integer.
     * @return int Element n in the recursive sequence.
     */
    public static int seqR(int n) {
        if (n == 0) {
            return 1;
        } else if (n == 1) {
            return 3;
        } else {
            return seqR(n - 1) - seqR(n - 2) * 2;
        }
    }

    /**
     * Recursively computes seq(n) using memory.
     *
     * @param n Non-negative integer.
     * @return int Element n in the recursive sequence.
     */
    public static int seqM(int n) {
        // if array not initialized, create an arraylist and add init value.
        if (memory == null) {
            memory = new ArrayList<>();
            memory.add(1);
            memory.add(3);
        }
        if (n+1 > memory.size()) {
            memory.add(seqM(n - 1) - seqM(n - 2) * 2);
        }
        return memory.get(n);
    }

    /**
```

```
     * Iteratively computes seq(n) with memory.
     *
     * @param n Non-negative integer.
     * @return int Element n in the recursive sequence.
     */
    public static int seqI(int n) {
        //init an array and add init value.
        int[] cache = new int[n+1];
        if(n>-1){
            cache[0] = 1;
        }
        if(n>0){
            cache[1] = 3;
        }
        for(int i=2;i<=n;i++){
            cache[i] = cache[i-1] - cache[i-2]*2;
        }
        return cache[n];
    }
}
```

b. TestSeq.java

```
/**
TestSeq.java
A simple driver that uses the Seq class to compute the
nth element of the sequence.
*/

import java.text.NumberFormat;
import java.util.Scanner;

public class TestSeq{

    public static void main(String[] args) {
        int n, seqRec;

        Scanner scan = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        n = scan.nextInt();

        seqRec = Seq.seqR(n);
        System.out.println("seqR(" + n + ") is: " + seqRec);
        seqRec = Seq.seqM(n);
        System.out.println("seqM(" + n + ") is: " + seqRec);
        seqRec = Seq.seqI(n);
        System.out.println("seqI(" + n + ") is: " + seqRec);


        NumberFormat form = NumberFormat.getInstance();
        form.setMaximumFractionDigits(7);
        form.setMinimumFractionDigits(7);
```

```
        System.out.println("Execution Times in Milliseconds (ms)");
        System.out.println("n\tRecursive\tSeq(n)\tMemoization" +
                "\tSeq(n)\tItertive\tSeq(n)");
        for (int i = 15; i <= 35; i += 10) {
            long start = System.nanoTime();
            int seqA = Seq.seqR(i);
            long end = System.nanoTime();
            double time = (double) (end - start) / 1000000;
            System.out.print(i + "\t" + form.format(time) + "\t" +
String.format("%-5s",seqA));

            start = System.nanoTime();
            int seqB = Seq.seqM(i);
            end = System.nanoTime();
            time = (double) (end - start) / 1000000;
            System.out.print("\t" + form.format(time) + "\t" +
String.format("%-5s",seqB));

            start = System.nanoTime();
            int seqC = Seq.seqI(i);
            end = System.nanoTime();
            time = (double) (end - start) / 1000000;
            System.out.print("\t" + form.format(time) + "\t" +
String.format("%-5s",seqC) + "\n");
        }
    }
}
```

2. Sample Output

```
Enter a positive integer: 0
seqR(0) is: 1
seqM(0) is: 1
seqI(0) is: 1
Execution Times in Milliseconds (ms)
n    Recursive    Seq(n)    Memoization    Seq(n)    Itertive    Seq(n)
15    0.4817540    -85      0.0163730    -85      0.0013400    -85
25    0.9705760    -7917    0.0153280    -7917    0.0014180    -7917
35    46.6120940    -364229    0.0086080    -364229    0.0016090    -364229

Enter a positive integer: 1
seqR(1) is: 3
seqM(1) is: 3
seqI(1) is: 3
Execution Times in Milliseconds (ms)
n    Recursive    Seq(n)    Memoization    Seq(n)    Itertive    Seq(n)
15    0.6211220    -85      0.0256300    -85      0.0017560    -85
25    1.3639030    -7917    0.0185990    -7917    0.0019660    -7917
35    53.1591750    -364229    0.0132520    -364229    0.0022070    -364229
```

```
Enter a positive integer: 2
seqR(2) is: 1
seqM(2) is: 1
seqI(2) is: 1
Execution Times in Milliseconds (ms)
n    Recursive    Seq(n)    Memoization    Seq(n)    Itertive    Seq(n)
15    0.4626670    -85      0.0227460    -85      0.0016670    -85
25    1.4986440    -7917    0.0218540    -7917    0.0020240    -7917
35    52.2144860    -364229    0.0122980    -364229    0.0019910    -364229
```