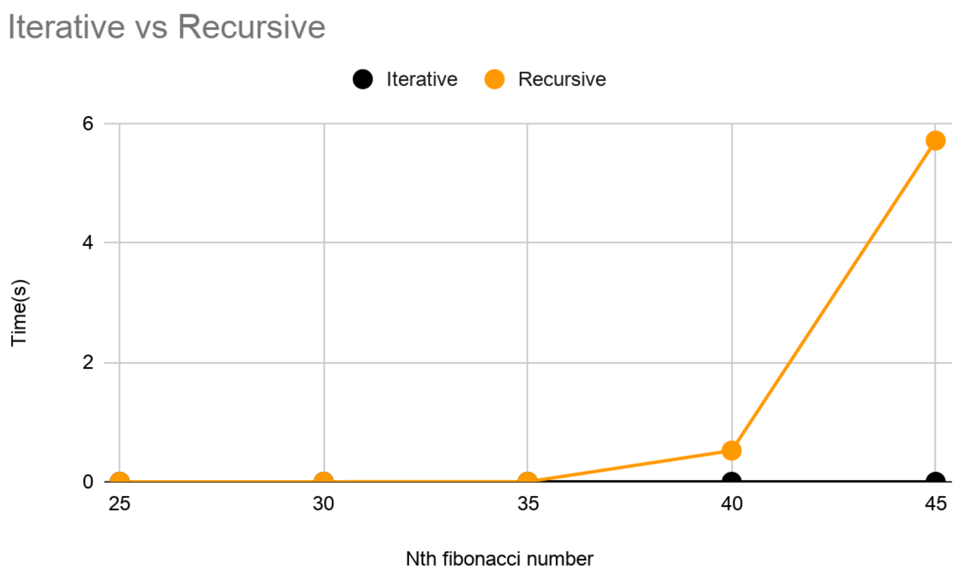
Assignment 3 Analysis

Fibonacci analysis:
(Note in StopWatch i changed System.currentTimeMillis() to System.nanoTime() as it is more accurate then converted said time into seconds)

Fibonacci recursive implementation vs iterative implementation:

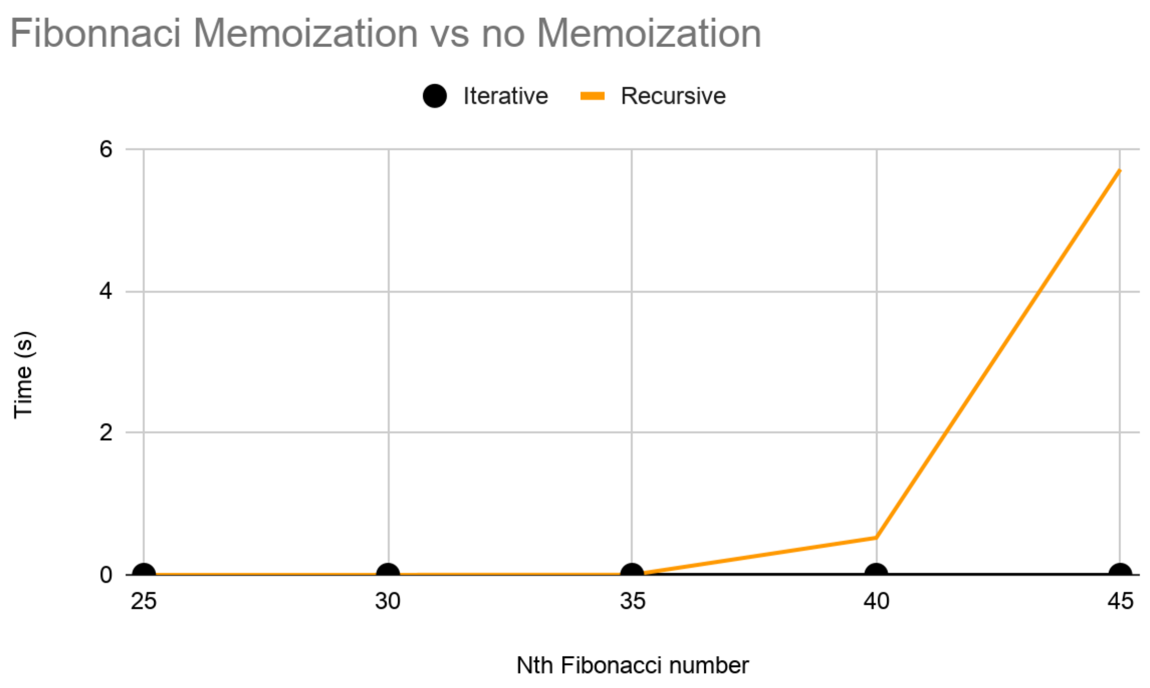
Nth fibonacci number	Fibonacci recursive(Sec)	Fibonacci Iterative (Sec)
10	0.0000256	0.0000035
20	0.0002245	0.0000046
30	0.004692	0.0000043
40	0.52196	0.0000037
45	5.781	0.0000065



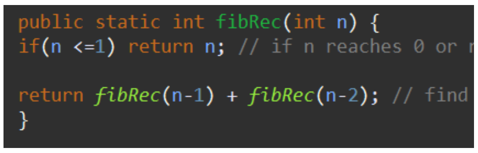
We can see that the 2 are barely even comparable as the recursive approach is O(2^n) as each recursive call branches out like a tree which can create more branches.
While the iterative approach is O(n) as it only runs the loop n times.

However to combat this we can implement memoization to remove branches for integers we have already calculated.

Nth fibonacci number	Fibonacci recursive(Sec)	Fibonacci Memoization
10	0.0000256	0.0001569
20	0.0002245	0.0001484
30	0.004692	0.000151
40	0.52196	0.0001654
50	5.781	0.0001523

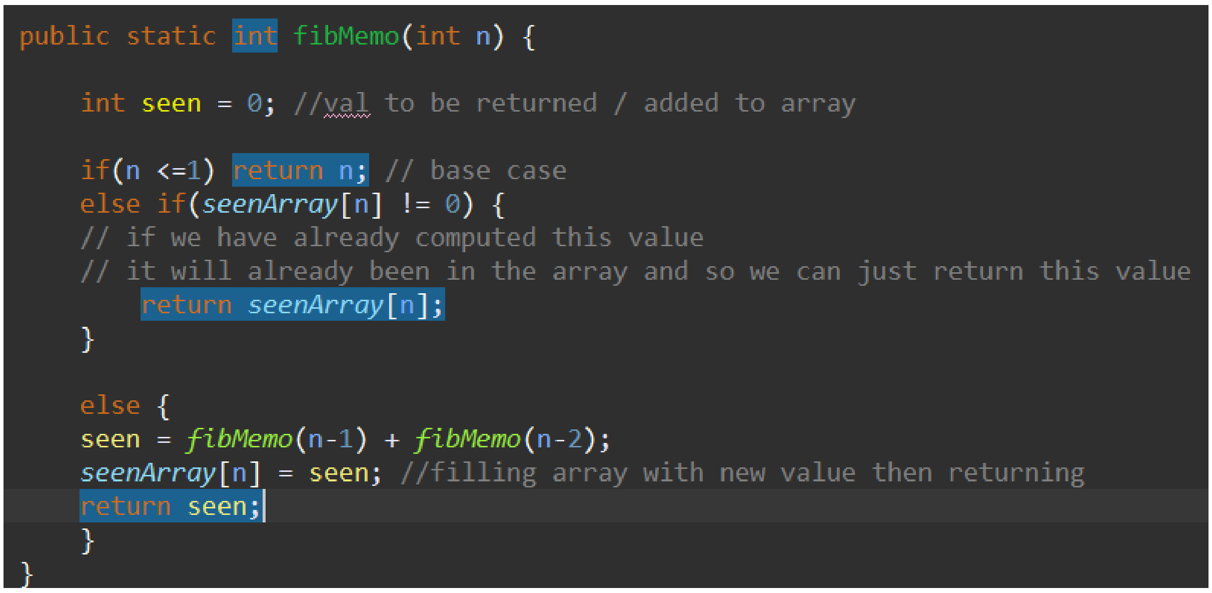


Memoization reduces the time complexity down to O(n) however we lose some of the neatness of the recursive implementation which may have been the reason for implementing fibonacci recursively as opposed to iteratively.



```
public static int fibRec(int n) {
if(n <=1) return n; // if n reaches 0 or
return fibRec(n-1) + fibRec(n-2); // find
}
```

vs

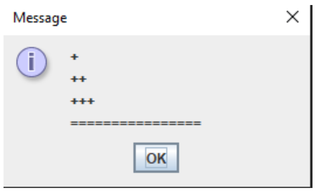


```
public static int fibMemo(int n) {

    int seen = 0; //val to be returned / added to array

    if(n <=1) return n; // base case
    else if(seenArray[n] != 0) {
    // if we have already computed this value
    // it will already been in the array and so we can just return this value
        return seenArray[n];
    }

    else {
    seen = fibMemo(n-1) + fibMemo(n-2);
    seenArray[n] = seen; //filling array with new value then returning
    return seen;
    }
}
```

Hanoi analysis:

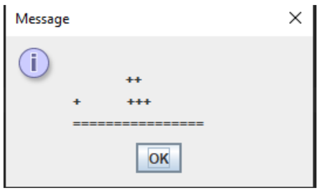
For hanoi I implemented 2 versions the first version is the basic version used to analyse the time complexity of the algorithm and then the second version is a basic game implementation of hanoi using Jpanel messages.The program runs the algorithm showing the steps on a 3 high stack

Demo:



Message

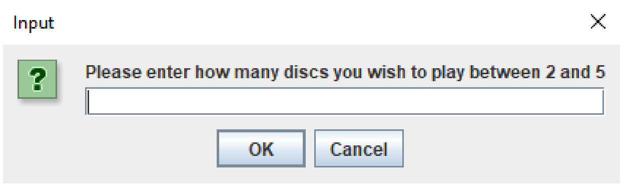
```
+
++
+++
================
```



Message

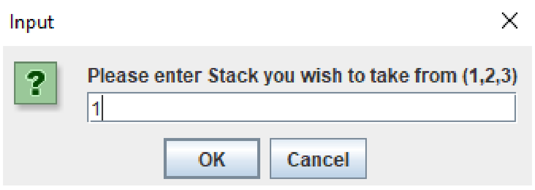
```
        ++
+       +++
================
```

Game:



Input

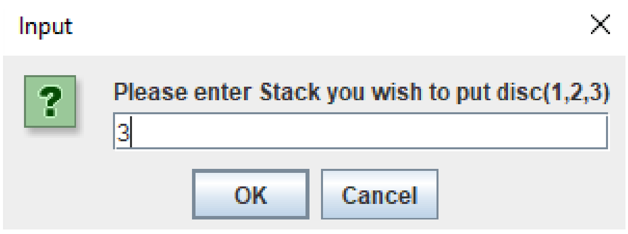
Please enter how many discs you wish to play between 2 and 5



Input

Please enter Stack you wish to take from (1,2,3)

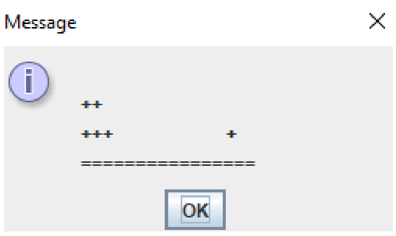
1



Input

Please enter Stack you wish to put disc(1,2,3)

3

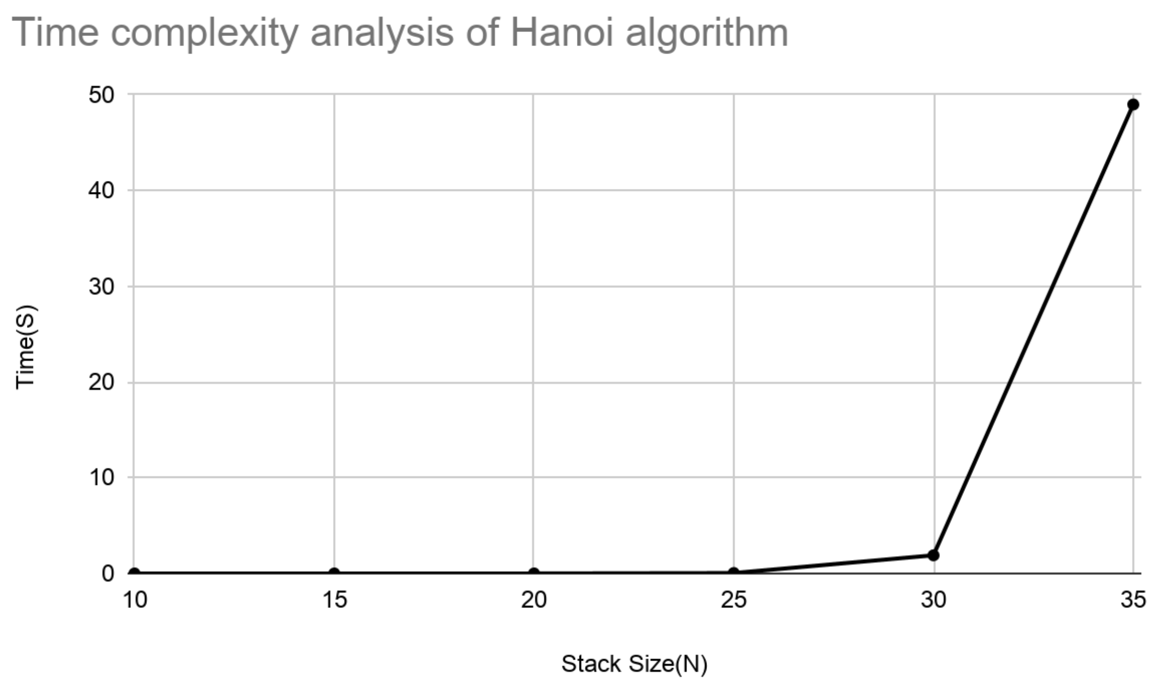


Message

```
++
+++           +
================
```

As the game interferes with time taken to execute the moves I also implemented an analysis version which only prints out steps. I will create a graph on the analysis version.

Stack Size(N)	Time(seconds)
10	0.000089
15	0.0003876
20	0.0022806
25	0.0480903
30	1.8994268
35	48.987321



From both the graph and the algorithm written we can see that the function has an exponential time complexity O(2^n).