Fibornacci Iterative

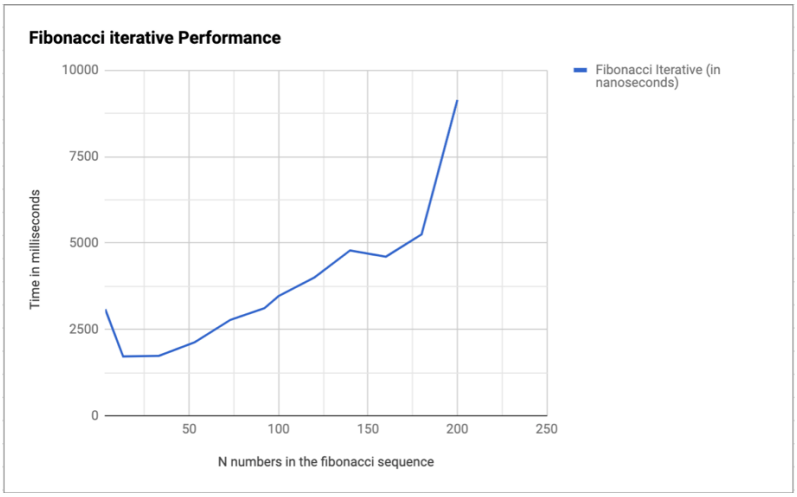
```
static int fibonacciIterative(int n){
    if (n<=1)
        return 1;

    int fib = 1;
    int prevFib =  1;

    for (int i = 2; i < n; i++) {
        int temp = fib;
        fib = fib + prevFib;
        prevFib = temp;
    }
    return fib;
}
```

This first implementation of computing N Fibonacci numbers iterates through each Nth number of the sequence through a for loop. This then means the algorithm has a linear time complexity O(n).

Table representation of acquired times

N	Fibonacci Iterative (in nanoseconds)
3	3088.00
13	1723.00
33	1738.00
53	2133.00
73	2781.00
92	3116.00
100	3468.00
120	4005.00
140	4785.00
160	4605.00
180	5251.00
200	9137.00

Graphical Representation of times



We can see that this algorithm for the most part has a linear growth in time.

Fibonacci Recursive

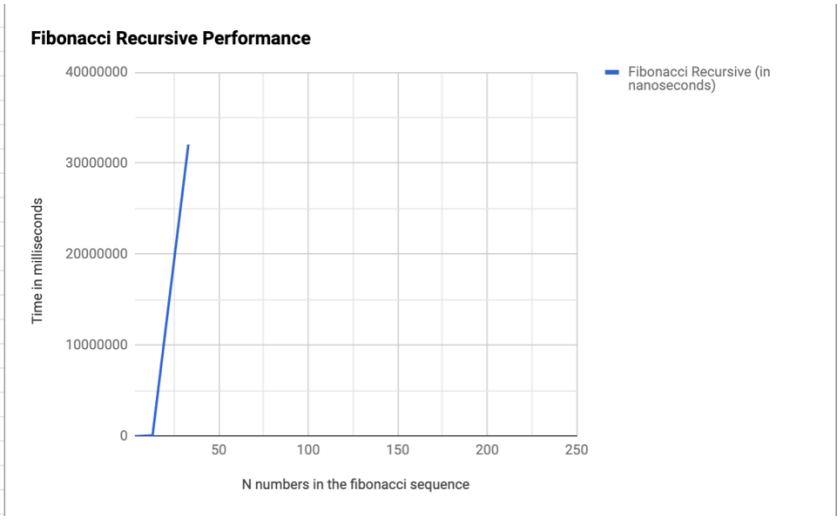
```
static int fibonacciRecursive(int n){
    if(n==0){
        return 0;
    }else
    if(n==1){
        return 1;
    }
    return (fibonacciRecursive( n: n-1) + fibonacciRecursive( n: n-2));
}
```

This implantation uses recursion to compute the first N numbers of the Fibonacci sequence. We will find that the algorithm's elapsed time exponentially grows when computing more than the first 33 numbers of the sequence. Timings couldn't be acquired for values greater than 33.

Table representation of timings

N	Fibonacci Recursive (in nanoseconds)
3	3404.00
13	92161.00
33	32054013.00
53	
73	
92	
100	
120	
140	
160	
180	
200	

Graphical representation of timings



The sharp growth in time proves that this algorithm has an exponential time complexity. This shows that a recursive function is not always the most efficient algorithm.

Tower Of Hanoi algorithm

```
static void towerOfHanoi(int disk, char source, char dest, char aux){
    if(disk == 1){
        System.out.println("Move disk from " + source + " to " + dest);
    }else {
        towerOfHanoi( disk: disk - 1, source, aux, dest);
        System.out.println("Move disk from " + source + " to " + dest);
        towerOfHanoi( disk: disk - 1, aux, dest, source);
    }
}
```

This algorithm is designed to allow a user to enter N number of disks on the puzzle “Tower Of Hanoi”. It will then tell the user how to move all the discs from A -> C step by step.

Table & Graphical representation of timings found

N	Tower of Hanoi (in nanoseconds)
3	3405.00
6	6209.00
9	86871.00
12	390126.00
15	3661873.00
18	1080173.00
21	5769700.00
24	22053765.00
27	341404053.00
30	1097395843.00