Fibonacci Series:

In Fibonacci series the numbers are built up on basis of previous numbers.

Example:

First number in sequence is 0.

Second number in sequence is 1.

Third number is Second number + First number = 0 + 1 = 1.

Fourth Number is Third number + Second number = 1 + 1 = 2.

Fifth Number is Fourth number + Third number = 2 + 1 = 3.

First 10 numbers in series are 0 1 1 2 3 5 8 13 21 34 55 89

Now we will attempt this problem using recursion and iteration.

As the recurrence relation above tells us

If n = 0 then return 0

If n = 1 then return 1

If n = 0 then return Fn-1 + Fn-2

Code for recursion is as follows:

**public** **static** **int** fiboRec(**int** limit){

**if** (limit == 0)

**return** 0;

**if** (limit == 1)

**return** 1;

**int** series = *fiboRec*(limit - 1) + *fiboRec*(limit - 2);

**return** series;

}

Above is the recursion tree. In tree we see that several values are computed again. Example take F(3). It is computed by F(5) and also by F(4). For small values this solution can work but we give big values then this solution will not work. Let us run some bench marks for recursion solution.

i = 0: result is 0: time taken is 0 milliseconds

i = 5: result is 5: time taken is 0 milliseconds

i = 10: result is 55: time taken is 0 milliseconds

i = 15: result is 610: time taken is 0 milliseconds

i = 20: result is 6765: time taken is 0 milliseconds

i = 25: result is 75025: time taken is 1 milliseconds

i = 30: result is 832040: time taken is 5 milliseconds

i = 35: result is 9227465: time taken is 53 milliseconds

i = 40: result is 102334155: time taken is 639 milliseconds

i = 45: result is 1134903170: time taken is 6795 milliseconds

So we can conclude that time take to execute the recursive solution it too much high. The reason is that the recursion tree becomes too much dense and several values are computed again and again.

Now we look at iterative solution.

**public** **static** **int** fiboIterative(**int** limit) {

**int** count = 0;

**int** a = 0, b = 1;

**for** (**int** i = 0; i < limit - 1; i++) {

count = a + b;

a = b;

b = count;

}

**return** count;

}

For the iterative method we just add the previous 2 numbers and built next two number by copying them into variables.

Let us run some bench marks for iterative solution.

i = 0: result is 0: time taken is 0 milliseconds

i = 5: result is 5: time taken is 0 milliseconds

i = 10: result is 55: time taken is 0 milliseconds

i = 15: result is 610: time taken is 0 milliseconds

i = 20: result is 6765: time taken is 0 milliseconds

i = 25: result is 75025: time taken is 0 milliseconds

i = 30: result is 832040: time taken is 0 milliseconds

i = 35: result is 9227465: time taken is 0 milliseconds

i = 40: result is 102334155: time taken is 0 milliseconds

i = 45: result is 1134903170: time taken is 0 milliseconds

From above we can definitely say that our iterative solution works too much faster than recursive solution. Hence iteration is better in Fibonacci series calculation.