Recursion vs Iterative for Fibonacci Sequence

Whilst recursion is easier to read from a style and coding perspective (and easier to implement), from a performance and memory approach, iterative would be more efficient.

Recursion requires more memory as it increases with each recursive call, with a very large number, this would pile up eventually and slow the program or system down. This is evident when knowing to calculate the 10th number in the Fibonacci system, a recursive call will need to be made for 9th term and 8th term.

Iteration is efficient by storing terms in variables and calling on them when needed. Much friendlier for memory and performance.

Triangle

public int triangle(int rows) {

// Authenticate whether need recursion or not (check if any rows)

if (rows == 0)

return 0;

// Returns how many blocks in rows so far + current row

else

return rows + triangle(rows - 1);

}

noX

public String noX(String str) {

// Authenticate whether need recursion or not (check length of string)

if (str.length() == 0) {

return "";

}

/\*

Returns character at specific index, if first is x, start recurse

it appends character to resulting string and calls again with remaining

characters

\*/

if (str.charAt(0) == 'x') {

return noX(str.substring(1));

} else {

return str.charAt(0) + noX(str.substring(1));

}

}