**Why is it a bad idea to use recursion method to find the Fibonacci of a number?**

The Fibonacci series is a series of numbers in which each number is a sum of the previous two (TechTerms, 2020). Recursion method is a method or algorithm that invokes itself repetitively with different arguments till a condition is met and breaks the loop (TechTerms, 2020). When solving for the Fibonacci of a number using the recursion method, a function is written such that it calls itself repetitively (with a less value each time) until the stated base case is reached. The value returned from each call is stored in memory stacks. Despite the fact that the recursion method offers a very direct approach when writing the function for finding the Fibonacci of a number- in that all you need basically is to define the recursive and base case- it might not always be the best idea to use it for a number of reasons as discussed below:

The recursion method is seemingly seamless when trying to find the Fibonacci of smaller numbers as fewer calls are made. When solving for the Fibonacci of larger numbers, the function would have to make a lot more calls with most of them having the same argument and return value as previous ones. This results in mundane repetition and inappropriate use of space (memory stacks) which greatly slows down the program due to the overhead of maintaining the stacks (Educative, 2020).

Another downside to using the recursion method when solving for the Fibonacci of larger numbers is if the memory stacks get filled before the base case of the function is arrived at, the loop would terminate resulting in a program error. Infinite recursion could occur if the recursion function is not implemented correctly and does not reduce the problem such that it converges at a final step (base case). This could lead to stack overflow and a system crash (Trinket, 2020).

The recursion method approach can be expensive in both processor time and memory space. When using recursion to solve for Fibonacci problems, it is imperative to make use of optimization techniques such as tail call or memorization (saving each return value to avoid future calls) to reduce overhead (Medium, 2018).  Another approach would be to use the iteration method instead of recursion as it is deemed to be more efficient and incur less overhead.

**References**

Educative, 2020. Retrieved from  <https://www.educative.io/courses/recursion-for-coding-interviews-in-cpp/qAx1lwQYDNG#advantages-of-recursion>

Medium, 2018. Retrieved from <https://medium.com/@nothingisfunny/memoization-improving-recursive-solution-for-fibonacci-sequence-problem-c02dab7a74e5>

TechTerms, 2020. Retrieved from <https://techterms.com/definition/recursive_function>

Trinket, 2020. Retrieved from <https://books.trinket.io/thinkjava2/chapter8.html>