Aliza Siddiqui

asiddiqui1

We can find the Fibonacci number given any x using two different algorithms: One is recursive, and one is iterative. However, the recursive solution is known to be significantly slower than the iterative solution execution wise.

In the iterative solution, we are storing our first two Fibonacci numbers in two variables (in this case olderVal and oldVal) and we use newVal to store our actual Fibonacci number. When we store these values, it saves memory space in the stack. The computation time for the iterative solution is linear.

In the recursive solution, the Fibonacci number for a specific x must be found multiple times before the final Fibonacci number is reached. For example, in order to find the Fibonacci number for x = 6, you must do the Fibonacci of x = 4 and x = 5. The Fibonacci for x = 5 involves finding the Fibonacci for x = 4. Notice this is done twice. The values are not being stored compared to the iterative solution. So, for large numbers, the computation time becomes very large since it is growing exponentially.

x	**Fib(x)**	**Recursive (ns)**	**Iterative (ns)**
10	55	3615800	3700
20	6765	528100	3000
30	832040	6132300	5900
40	102334155	963876700	4200
50	12586269025	60938390900	3800