Why does your program take as long as it does?

The *fib(int k)* function takes a long time as it recalculates values many times.
For example: calling *fib(6)* will calculate *fib(5)* and *fib(4)*. However *fib(5)* will calculate *fib(4)* as well. This continues with each further call. *fib(4)* will calculate *fib(3)* and *fib(2)* - but *fib(3)* also calculates *fib(2)*. The number of values that are recalculated grows enormously as *k* increases.

In comparison my *fib2(int k)* function only calculates each value once.

The below image shows what each *fib(6)* calculates. You can clearly see how many times some values are recalculated.

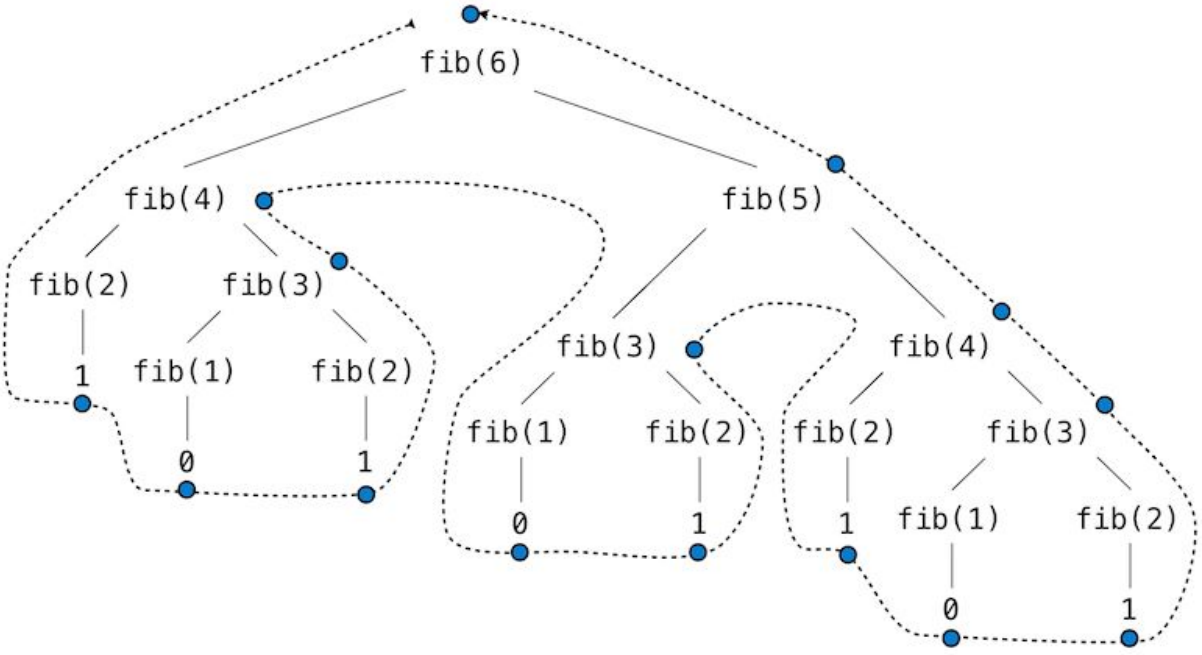

Image source: http://composingprograms.com/pages/28-efficiency.html

What is the order of the algorithm (in k)?

I timed *fib(k)* with k=15-32 and plotted the results. In the below graph you can clearly see the order is exponential.

As each function call makes 2 more function calls (except *fib(0)* and *fib(1)* which do not make any further calls) the upper limit of the order is O(2^k).

With further analysis it can be shown that the order of the function is ~O(1.6^k).
Say fib(k) takes time T(k)
fib(k) = fib(k-1) + fib(k-2)
Therefore T(k) = T(k-1) + T(k-2)

We can see here that the time taken will grow as fib(k) does, therefore the order is the fibonacci sequence itself, ~O(1.6^k) as the fibonacci sequence grows at approximately the rate of the golden ratio.

In comparison the *fib2(int k)* function has linear (O(k)) growth, as the number of operations performed increases linearly with *k*