The benefit of calling `square` during `expmod` is that every time that `exp` is even, you prevent a duplication of steps. This is because each (`expmod base (/ exp 2) m`) needs to only be evaluated once, and then the final value, which will just be a primitive number, can be squared. The result is a logarithmic procedure that requires an additional step (division in the `even?` case) as the input size doubles. Louis Reasoner's implementation would instead result in tree recursion every time that `exp` is even, duplicating the amount of work at each step. The more times `exp` is even, the more times that the remaining work will be duplicated, and thus the total number of steps will grow in proportion to the input size.

Specifically, what was before $\Theta(\log n)$ has been transformed into $\Theta\left(2^{\log_2 n}\right)$, because the equivalent `expmod` calls in the `even?` case will result in two identical branches at each `even?` level, and the number of those levels is how many times n will be divided by 2 ($\log_2 n$). $\Theta\left(2^{\log_2 n}\right)$ is equivalent to $\Theta(n)$ and thus a $\Theta(\log n)$ process has been transformed into a $\Theta(n)$ process.