I first approached the problem using recursion because the pattern reminded me of the Fibonacci sequence. But as soon as I tried it with larger values of n, I ran into stack overflow issues. So I switched to a loop-based solution that just kept track of the last two values in the sequence. This worked much better and avoided crashing, but it still took some time to run for larger inputs since it processed one step at a time.

To improve that, I realized I could actually calculate two values in each loop, one for the next odd index and one for the next even index, since each value depends on the two before it. This cut the number of loop steps in half, and made the program run significantly faster while still keeping the code simple and efficient.

There may be even faster ways to solve this, and I'd love to explore those methods further in the future.