Problem 0

Debug the code and "step into" the function for fib(5) and recursive calls on the function stack

- fib(5)
  - fib(4)
    - fib(3)
      - fib(2)
        - fib(1) -> returns 1
        - fib(0) -> returns 0
      - fib(1) -> returns 1
    - fib(2)
      - fib(1) -> returns 1
      - fib(0) -> returns 0
  - fib(3)
    - fib(2)
      - fib(1) -> returns 1
      - fib(0) -> returns 0
    - fib(1) -> returns 1

The final result for fib(5) is 5, just as it should be

2. To prove the time complexity of algorithm

So here time complexity can be proven as $O(2^n)$

→ Each call to fib(n) results in two more calls, fib(n-1) and fib(n-2)
→ This forms binary tree height of tree is n
→ so at each level of tree no. of calls doubles so approx 2^n

$O(2^n)$

3. Improving implementation

☆ Bottom-Up approach
→ uses simple loop to calculate fibonacci numbers storing only the last two computed value

☆ Memoization (Top-Down)

Stores result of previously computed fibonacci number in array. prevent redundant calculation