1)Discuss the time complexity of your recursive algorithm

The recursive algorithm computes the future value by reducing the number of years incrementally until reaching the base case of zero years. Without memoization, this approach would result in repeated calculations for the same subproblems, leading to an O(n) time complexity where n represents the number of years. However, by implementing memoization, each year's computation is stored and retrieved in constant time, eliminating redundant work. As a result, the algorithm processes each year only once, maintaining an overall time complexity of O(n) while significantly improving efficiency through optimized lookup of previously solved subproblems.

2)Explain how to optimize the recursive solution to avoid excessive computation

The recursive algorithm employs memoization to optimize performance by caching previously computed results in a lookup table. This technique prevents redundant calculations by checking the cache before performing any computation, ensuring each subproblem is solved only once. As a result, the solution avoids the exponential time complexity that would otherwise arise from repeated recursive branching, maintaining an efficient O(n) time complexity where n represents the number of years. For additional optimization, the recursive approach could be converted to an iterative solution, which would eliminate function call overhead and prevent potential stack overflow issues with large inputs. Together, these optimization strategies - memoization and iteration - significantly enhance the algorithm's scalability, making it robust enough for practical financial applications involving long-term projections.