1)Discuss the time complexity of your recursive algorithm

The recursive algorithm calculates the future value by reducing the number of years one by one until it reaches zero. Without memoization, each recursive call would result in a new computation, leading to repeated calculations for the same subproblems. This would result in a time complexity of O(n), where n is the number of years.

However, since memoization is used in the solution, the value for each year is calculated only once and stored. This optimization ensures that each unique subproblem (for each year) is solved once, reducing unnecessary repeated calls. Therefore, the effective time complexity with memoization becomes O(n), as each year is processed only one time and looked up afterward.

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

The recursive solution is optimized using memoization, which stores already computed values in a map (cache). When a value for a certain number of years is needed again, it is retrieved from the cache instead of recalculating it. This avoids exponential growth in function calls and ensures that the computation remains efficient even for large inputs.

Further optimization could include converting the recursive solution to an iterative one. This would eliminate the overhead of recursive calls and reduce the risk of stack overflow for large values of years. Both memoization and iteration help to make the algorithm more scalable and suitable for real-world applications involving financial projections over long periods.