**Solve any LeetCode problem.**

1. **Understanding the Problem:**
   * **Identify Input Requirements:**
     + Determine the specific inputs required, including data types and potential size constraints.
     + Understand the format of the input data and how it will be provided (e.g., as arrays, strings, integers).
     + Consider any limitations on input size or data format.
   * **Define Output Expectations:**
     + Determine the exact format and content of the output expected from the solution.
     + Understand any constraints or limitations on the output, such as data type or size restrictions.
   * **Analyze Complexity:**
     + Consider how the size of the input data could impact space and time complexity.
     + Evaluate any potential edge cases or special scenarios that may affect complexity analysis.
2. **Devising a Plan:**
   * **Select Appropriate Algorithms and Data Structures:**
     + Research and identify suitable algorithms or data structures for solving the problem efficiently.
     + Evaluate the advantages and disadvantages of different approaches, considering space and time complexity.
   * **Assess Space and Time Complexity:**
     + Analyze the space and time complexity of each chosen algorithm or data structure.
     + Consider trade-offs between different approaches, such as ease of implementation versus efficiency.
   * **Explore Optimization Techniques:**
     + Identify opportunities for optimization to improve space or time efficiency.
     + Implement techniques to minimize unnecessary space usage and enhance time efficiency where possible.
   * **Write out your steps as comments.**
   * **Draw figures, if your mental space isn't large enough.**
3. **Carrying Out the Plan:**
   * **Implement the Solution:**
     + Write code to implement the chosen solution using the selected algorithms and data structures.
     + Ensure efficient utilization of space and time resources during implementation.
   * **Analyze Performance:**
     + Perform a detailed analysis of the space and time complexity of the implemented solution.
     + Test the solution with various inputs to validate correctness and assess performance under different conditions.
4. **Looking Back:**
   * **Evaluate Solution Efficiency:**
     + Assess the space and time complexity of the implemented solution against desired performance criteria.
     + Reflect on the effectiveness of the chosen algorithms and data structures in achieving efficiency goals.
   * **Identify Optimization Opportunities:**
     + Consider areas for further optimization to enhance space or time efficiency, if necessary.
     + Reflect on lessons learned from the problem-solving process, including insights into complexity analysis.
5. **Generalizing:**
   * **Apply Knowledge to Other Scenarios:**
     + Use insights gained from space and time complexity analysis to solve similar problems.
     + Practice analyzing the complexity of solutions to develop a deeper understanding and intuition.
   * **Explore Additional Resources:**
     + Continue learning about algorithms and data structures through additional resources and tutorials.
     + Build proficiency in complexity analysis by applying concepts to various problem-solving scenarios.