Documentation

To solve the Combination Sum III problem, you must find all valid combinations of k distinct numbers that sum up to a given target n, where the numbers are restricted to the range [1, 9]. Each number can be used once in any combination, and no duplicate combinations are allowed. The goal is to return all combinations that satisfy these criteria, and the result can be in any order.

The best approach to solving this problem is using backtracking, which allows you to systematically explore all possible combinations while pruning invalid ones early. Backtracking works by incrementally building combinations, adding numbers from the given range, and checking if they meet the conditions. If the sum of the current combination exceeds n, or if the number of elements exceeds k, the function stops and backtracks, removing the last number added to the combination and trying other possibilities.

The process starts by considering each number from 1 to 9 and adding it to an empty combination. Once a number is added, the algorithm recursively explores further combinations by only considering numbers that come after the current one to avoid duplicates. If a valid combination is found—meaning it has exactly k numbers, and their sum equals n —it is added to the result list. If the current combination's sum or length exceeds the given constraints, the function discards that path and backtracks by removing the last number and trying the next possible number.

Edge cases are important in this problem. For instance, if k is too large relative to n, it becomes impossible to find any valid combinations because the sum of the smallest k numbers will exceed n. Similarly, if n is too small to be formed by adding k distinct numbers from the range [1, 9], no valid combinations exist.

The time complexity of the solution is approximately exponential, specifically ($O(2^n)$), because the algorithm explores all subsets of numbers from the range [1, 9] . However, due to the small range of numbers (only nine), the approach is efficient enough for the input size. The space complexity is linear in terms of the depth of the recursion, which corresponds to the size of the current combination, making it ($O(k)$).

In summary, the backtracking approach efficiently navigates through possible number combinations while maintaining constraints on both the number of elements and their sum, ensuring no duplicate combinations are included in the result.