Documentation

The problem revolves around a guessing game where the task is to identify a number picked within a given range using feedback provided by a pre-defined guess API. This API informs whether the guessed number is too high, too low, or correct. The key to solving this problem efficiently is leveraging the binary search algorithm, which excels in scenarios where the search space can be systematically halved based on feedback. Binary search ensures that we make the minimal guesses required to find the target number within the given range.

The approach begins by setting the search range to [1, n], where n is the upper limit of the range. At each step, the midpoint of the current range is calculated as the guess. The feedback from the guess API is then used to determine whether the target number lies in the left half or the right half of the range. If the feedback indicates the guess is too high, the range is adjusted to the left by updating the upper limit. Conversely, if the feedback suggests the guess is too low, the range is shifted to the right by updating the lower limit.

This process of halving the search range continues iteratively until the correct number is identified. The efficiency of this method lies in the logarithmic reduction of the search space, where each iteration eliminates half of the remaining possibilities. This guarantees that the target number will be found within O(log n) iterations, making the algorithm highly efficient even for large values of n.

The algorithm's space complexity is O(1), as it only uses a few variables to track the search range and the midpoint. This simplicity, combined with its efficiency, makes binary search an ideal solution for this problem. The algorithm systematically converges on the target number without requiring additional memory or complex operations.

A significant observation about this problem is that it is deterministic and guarantees a solution within the defined range. Since the feedback is always accurate and the range is bounded, the algorithm will always terminate successfully. The constraints and properties of the problem ensure that no guess is wasted, and each iteration brings the solution closer.

The use of binary search also highlights the power of divide-and-conquer strategies in solving computational problems. By breaking down the problem into smaller, more manageable subproblems, binary search efficiently narrows the possibilities until the exact solution is found. This approach is not only optimal for this specific problem but also serves as a foundational technique in various applications across computer science.

In conclusion, the problem of guessing a number using the guess API is a textbook example of how binary search can be applied to minimize the number of operations. Its logarithmic time complexity, constant space complexity, and deterministic nature make it an elegant and robust solution. The approach underscores the importance of using feedback effectively and demonstrates how systematic problem-solving techniques can lead to optimal solutions.