Documentation for findRepeatedDnaSequences Method

Problem Overview:

- This method solves the problem of identifying repeated DNA sequences from a given string s that represents a DNA sequence. DNA sequences are composed of nucleotides, which are abbreviated as 'A', 'C', 'G', and 'T'. The task is to find all 10-letter-long sequences that appear more than once in the DNA sequence.

The string may contain millions of characters, making it crucial to optimize the solution for both time and space efficiency.

Parameters:

- **s (str):** The input string representing a DNA sequence. Each character in the string is a nucleotide, which can be either 'A', 'C', 'G', or 'T'. The length of the string is guaranteed to be between 1 and 100,000.

Return Value:

- **List[str]:** The method returns a list of all 10-letter-long sequences (substrings) that occur more than once in the input string s. The sequences can appear in any order in the output list, and each sequence is returned only once, even if it appears multiple times in the string.

Approach:

1. **Sliding Window:**

- A sliding window of size 10 is used to extract all possible 10-letter-long substrings from the input string. This is done by iterating through the string from the start to the point where only 9 characters are left, extracting each 10-letter substring at each step.

2. Tracking Seen Sequences:

- *Two sets are used to track the substrings:*
  - ***seen:*** This set stores all the unique 10-letter-long sequences encountered during the traversal of the input string.
  - ***repeated:*** This set stores all the sequences that appear more than once.

3. Checking for Repeats:

- For each 10-letter substring, if it has been seen before (i.e., it exists in the seen set), it is added to the repeated set. If the substring is being encountered for the first time, it is added to the seen set.

4. Returning Results:

- Once all substrings have been processed, the repeated set (which contains the sequences that occurred more than once) is converted to a list and returned as the final result.

Edge Cases:

1. Short Strings:

- If the length of the string is less than 10, the function immediately returns an empty list since there cannot be any 10-letter-long sequences in such cases.

2. No Repeated Sequences:

- If there are no repeated 10-letter sequences in the input string, the function will return an empty list.

3. **All Identical Sequences:**
   - If the input string contains the same character repeated multiple times (e.g., "AAAAAAAAAAAAA"), the function will correctly identify the repeated 10-letter sequence and return it.

Complexity Analysis:

Time Complexity: O(n)

- The time complexity is linear with respect to the length of the input string s, where n is the length of the string. This is because we only traverse the string once and use efficient operations for set insertion and membership checks, both of which are O(1) on average.

Space Complexity: O(n)

- The space complexity is also linear with respect to n since we store the substrings in sets (seen and repeated). In the worst case, the size of the seen set could grow up to n 9 elements (for a string of length n), and the repeated set will contain at most the number of repeated sequences.

Example 1:

- **Input:** "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
- **Steps:**
  - Slide a 10-letter window over the string and extract substrings like "AAAAACCCCC", "AAAACCCCCA", etc.
  - Track the occurrences of each substring in the seen set.
  - Identify that "AAAAACCCCC" and "CCCCCAAAAA" appear more than once.
- **Output:** ["AAAAACCCCC", "CCCCCAAAAA"]

Example 2:

- **Input:** "AAAAAAAAAAAAA"
- **Steps:**
  - The 10-letter substring "AAAAAAAAAA" is repeated multiple times.
  - It will be added to the repeated set.
- **Output:** ["AAAAAAAAAA"]

Assumptions:

1. The input string s will only contain valid nucleotide characters ('A', 'C', 'G', 'T').
2. The DNA sequence will always be a contiguous string without spaces or separators.
3. The input string may be very large (up to 100,000 characters), so an efficient solution is required.

Key Points:

- The problem is about identifying repeating substrings of a fixed length (10 characters) in a large string.
- The use of sets ensures efficient tracking and identification of repeated sequences.
- The method handles edge cases like strings shorter than 10 characters and returns correct results for all valid input scenarios.

This approach provides a fast and memory-efficient way to solve the problem, making it suitable for large inputs like DNA sequences of up to 100,000 nucleotides.