Lexicographical Numbers LeetCode

Given an integer n, return all the numbers in the range [1, n] sorted in lexicographical order.

Example: N = 21

Output: [1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 3, 4, 5, 6, 7, 8, 9]

Approach 1: Function to generate lexical order using Trie

Generate the lexicographical order of numbers from 1 to n using a Trie-based approach.

Explanation:

- **TrieNode Class:**
  - Represents a node in the trie with children nodes and an end-of-word flag.
- **Trie Class:**
  - Manages the trie and provides methods to insert words and get lexical order.
- **insert Method:**
  - Inserts each number from 1 to n into the trie as a string.
- **getLexicalOrder Method:**
  - Recursively explores the trie to obtain the lexicographical order.
  - Appends each valid number to the result vector.
- **lexicalOrderTrieApproach Function:**
  - Creates a Trie, inserts numbers, and gets the lexicographical order using the Trie.

Time Complexity:

- **Insertion: O(N * L), where N is the range, and L is the average length of a number string.**
- **Search (getLexicalOrder): O(N * L), where N is the range, and L is the average length of a number string.**

Space Complexity:

- **Trie Storage: O(N * L), where N is the range, and L is the average length of a number string.**
- **Result Vector: O(N).**

Approach 2: Function to get lexical order using combination of recursion and iteration approach

Generate the lexicographical order of numbers from 1 to n using a combination of recursion and iteration.

Explanation:

- **generateLexicalOrder Function:**
    - Recursively generates the lexicographical order of numbers.
    - Uses iteration to explore all valid digits for each position.
- **lexicalOrder Function:**
    - Starts the recursion from digits 1 to 9 to avoid leading zeros.
    - Calls the recursive function for each starting digit.

Time Complexity:

- **Recursion and Iteration: O(N), where N is the range.**

Space Complexity:

- **Result Vector: O(N), where N is the range.**

Conclusion:

- Both approaches provide the correct lexicographical order.
- **Approach 2 is simpler and more efficient in terms of both time and space complexity.**