Word Ladder LeetCode

A **transformation sequence** from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s_1 -> s_2 -> ... -> s_k such that:

- Every adjacent pair of words differs by a single letter.
- Every s_i for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
- s_k == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return *the **number of words** in the **shortest transformation sequence** from* beginWord *to* endWord*, or* 0 *if no such sequence exists.*

Example:

beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]

Output: 5

Explanation: One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> cog", which is 5 words long.

Approach 1: Function to find the shortest transformation sequence length from startWord to endWord

- **Explanation:**
    - The **ladderLength** function uses BFS to find the shortest transformation sequence length.
    - It creates an unordered set (**s**) for efficient word lookup and enqueues the start word with steps 1 into a queue.
    - During BFS traversal, it generates all possible one-letter transformations of the current word.
    - If a transformed word is in the wordList, it erases the word from the set, enqueues it with updated steps, and continues BFS.
    - The traversal continues until the endWord is found, and the function returns the number of steps.
    - If no transformation sequence is found, it returns 0.
- **Time Complexity:**
    - **The time complexity depends on the length of the wordList and the average length of the words.**

- **In the worst case, it may need to generate all possible transformations for each word, leading to O(L * N), where L is the length of the words and N is the size of the wordList.**
- **Each transformation operation takes O(L) time.**

- **Space Complexity:**
  - **The space complexity is O(N) for the unordered set (s) since it stores the words from the wordList.**
  - **The space complexity is O(W) for the queue, where W is the maximum width of the BFS traversal. In the worst case, it can be as large as the number of words reachable in one step from the start word.**
  - **The Space complexity is O(N + W)**