Word Ladder

This problem can be solved efficiently using Breadth-First Search (BFS).

Key Concepts:

1. Graph Construction:
   - Words are nodes.
   - An edge exists between two words if they differ by exactly one letter.
   - Example: hot ↔ dot, hot ↔ lot.
2. Shortest Path Problem:
   - You are finding the shortest path from beginWord to endWord in this implicit graph.
   - BFS is ideal since all edges have equal weight.
3. Constraints to Check:
   - If endWord is not in wordList, the answer is 0.
   - You can transform only one character at a time.
   - You must only use words from wordList except for beginWord.

BFS Strategy:

1. Preprocess:
   - Use a set for wordList for fast lookup.
2. BFS Queue:
   - Each element is a (word, depth) pair.
   - depth represents how many transformations taken so far (starting at 1 with beginWord).
3. At each BFS level:
   - Try changing each character in the word to all 26 letters.
   - If the new word is in the wordList and not visited:
     - Add it to the queue and mark it as visited.
   - If it matches endWord, return depth + 1.
4. If queue gets exhausted: return 0.

Optimization ideas:

- You can preprocess intermediate states like hot, hot to reduce search space.
- Even better: use Bidirectional BFS to cut the search space in half (not required for correctness, but improves performance).

Complexity:

- Time: $O(N * M^2)$, where N is number of words, M is word length.
- Space: $O(N * M)$ for the queue and visited set