```
from collections import deque

# Step 2: Function to check one-letter difference
def is_one_letter_apart(word1, word2):
    diff_count = sum(c1 != c2 for c1, c2 in zip(word1, word2))
    return diff_count == 1

# Step 3: BFS algorithm
def shortest_transformation_length(startWord, endWord, wordList):
    # Step 1: Check if the target word is in the dictionary
    if endWord not in wordList:
        return 0

    # Initialize a queue with the starting word and length 1
    queue = deque([(startWord, 1)])

    # Step 3 continued: BFS loop
    while queue:
        current_word, length = queue.popleft()

        # Check if we reached the target word
        if current_word == endWord:
            return length

        # Explore words one letter apart
        for word in wordList:
            if is_one_letter_apart(current_word, word):
                queue.append((word, length + 1))
                wordList.remove(word)  # Remove the word to avoid revisiting

    # Step 4: If no transformation sequence is found
    return 0

# Example usage:
startWord = "hit"
endWord = "cog"
wordList = ["hot", "dot", "dog", "lot", "log", "cog"]

result = shortest_transformation_length(startWord, endWord, wordList)
print(result)
```

```
5
```