Space Complexity of Each Step 2 Solution

1) Sum Zero

The space complexity of the `addToZero` function is O(n) because it uses a `Set` to store the unique elements from the input array. In the worst-case scenario, the `Set` could store all the elements from the input array, resulting in linear space usage relative to the size of the input array.

2) Unique Characters

The space complexity of the `hasUniqueChars` function is O(k), where k is the number of unique characters in the input word. It uses a `Set` to store the unique characters encountered while traversing the input word. In the best-case scenario where all characters in the word are unique, the `Set` will store all characters, leading to linear space usage. However, in the worst-case scenario where all characters in the word are the same, the `Set` will store only one character, leading to constant space usage.

3) Pangram Sentence

The space complexity of the `isPangram` function is O(1). The function uses a `Set` of fixed size (26 elements) to keep track of the letters of the alphabet. Regardless of the length of the input sentence, the `Set` will always store a fixed number of elements, resulting in constant space usage.

4) Longest Word

The space complexity of the `findLongestWord` function is O(1). The function only uses a few simple variables (e.g., `longestWordLength`) to keep track of the longest word's length. It does not use any data structures that grow with the input size, so the space usage remains constant regardless of the size of the input array.

Overall, the functions in Step 2 have varying space complexities, with some being constant space (O(1)) and others being linear space (O(n) or O(k)).