Staircase:

1. Time Complexity: The time complexity of this program is primarily determined by the nested loops. The outer loop runs `n` times, where `n` is the size of the staircase. For each iteration of the outer loop, the inner loop also runs `n` times. Therefore, the total number of iterations is `n \* n`, which is n^2.

Time Complexity: **O(n^2)**

2. Space Complexity: The function only uses a fixed number of integer variables (`i`, `j`, and `n`), regardless of the size of the staircase. No additional data structures that grow with the input size are used.

Space Complexity: O(1)

Alternating Characters

Recursive Definition: The recursive approach in your program breaks down the problem into smaller problems, each representing a state of the string starting from a specific index. By incrementing the index and comparing adjacent characters, it calculates the minimum number of deletions required. The recursion unfolds back up, accumulating the total deletions needed for the entire string.

Time Complexity: O(n)

The primary operation in the function is the comparison between adjacent characters. This comparison is performed once for each character in the string (except the first character). Therefore, if the length of the string is `n`, the time complexity is O(n) as each character is processed once.

Space Complexity: O(n)

The space complexity is primarily determined by the depth of the recursion. In the worst case, the recursive function `count` is called for every character in the string. Thus, the maximum depth of the recursive call stack will be `n` (where `n` is the length of the string). This leads to a space complexity of O(n). The algorithm does not use any additional data structures that grow with the input size, so the only significant space usage comes from the recursive call stack.