**Approach:**

1. **Initialization**: Start with step = 0, which represents the current position or depth in a file system.
2. **Processing Each Log**:
   * Iterate through each log in the logs array.
   * If the log is "../", decrement step by 1, but only if step is greater than 0 (ensures we don't move above the root).
   * If the log is not "./", increment step by 1 (handles all other valid operations, such as moving into a directory).
3. **Final Output**: After processing all logs, step will represent the final depth or position in the file system.

**Time Complexity:**

* **O(n)**, where n is the number of logs in the logs array. This is because the program iterates through each log exactly once, performing constant-time operations (O(1)) on each log.

**Space Complexity:**

* **O(1)**. The space used by the program is constant and independent of the input size n. The primary space usage is for the integer variable step and the temporary string variable log used in the loop. No additional space scales with the input size.

**Explanation:**

* **Time Complexity**: Linear O(n), where n is the number of logs. This efficiency arises because each log is processed exactly once in a single loop iteration.
* **Space Complexity**: Constant O(1). The space used remains constant regardless of the input size n. This is because the program only uses a fixed amount of space for variables (step and log), and it does not depend on the size of logs.