Interleaving String

Think dynamic programming with a "how much of s1 and s2 can build a prefix of s3" mindset.

Core check:

- If $|s1|+|s2| != |s3|$, answer is false immediately.

2D DP idea (then we'll trim to 1D):

- Let dp[i][j] be true iff the first i chars of s1 and the first j chars of s2 can interleave to form the first i+j chars of s3.
- Base:
  - dp[0][0] = true.
  - First row: dp[0][j] = dp[0][j-1] && s2[j-1] == s3[j-1].
  - First column: dp[i][0] = dp[j-1][0] && s1[i-1] == s3[i-1]
- Transition for general i, j ≥ 1:
  - You can come from top if dp[i-1][j] is true and s1[i-1] == s3[i+j-1].
  - Or from left if dp[i][j-1] is true and s2[j-1] == s3[i+j-1].
  - So: dp[i][j] = (dp[i-1][j] && s1[i-1] == s3[i+j-1]) ||
    (dp[i][j-1] && s2[j-1] == s3[i+j-1]).

The answer is dp[|s1|][|s2|].

Space-optimized (O(|s2|)):

- Keep a 1D array dp[j] representing the current row's dp[i][j].
- Initialization (i = 0 row):
  dp[0] = true and for j = 1 ... |s2|: dp[j] = dp[j-1] && (s2[j-1] == s3[j-1]).
- For each i from 1 ... |s1|:
  - Update the first column:
    dp[0] = dp[0] && (s1[i-1] == s3[i-1]).
  - For each j from 1 ... |s2|:
    - Use the same logic but carefully with overwritten values:
      - "From top" uses the current dp[j] (which still holds the previous row's value before you overwrite it) and checks

s1[i-1] == s3[i+j-1].

- "From left" uses dp[j-1] (already updated for this row) and checks s2[j-1] == s3[i+j-1].
- Combine with OR.

Why this works:

You're deciding for each i+j, whether the last character of the s3-prefix came from s1[i-1] or s2[j-1]. If either source makes the rest consistent, mark true.