Basanta:

Overall code looks good. A few enhancements could make the code even more effective and readable. For instance, adding edge case checks for empty or non-standard grids at the start of getSolution() could streamline handling special cases. Passing the visited state as a parameter in _dfs() could also help avoid unintended overlaps in recursive calls. Adding brief comments, especially within _dfs, would improve readability, and incorporating docstrings for each method would make the code easier for others to understand.

The debug print statements for found words are useful, but including an option to toggle these on or off as needed for debugging might be helpful. Expanding the tests to include cases like single row or column grids or grids with repeating letters could further strengthen your approach. Overall, it's a solid, well-thought-out solution with excellent use of helper functions.

Rijan:

It sounds like the code isn't passing tests for 3x3 and 4x4 grids. Could this be due to boundary handling or a dictionary lookup issue? Maybe adding some specific tests within the code to check smaller grids could help pinpoint where it's breaking. Would that help make debugging more straightforward?

Naming Consistency

The method names setGrid and setDictionary work fine, but in Python, it's generally more common to see snake_case for methods instead of camelCase. Would renaming them to set_grid and set_dictionary feel a bit more Pythonic and consistent with the style guide?

Visited Tracking

Right now, visited is set at the class level, so I wonder if it could cause issues if getSolution is called multiple times on the same instance. Would resetting visited inside getSolution before each search help ensure each search starts fresh?

Debug Output

The print(f"Found valid word: {current_word}") line is super useful for debugging, but it might clutter the output during regular use. What do you think about adding a debug flag to the class? That way, you could control whether or not this output shows up.

Efficiency Thoughts

The current DFS search will stop if a word gets too long, which is good! But I wonder if we could make it even more efficient by checking if each prefix is valid as we build up the word. Maybe a Trie could be handy for this? Or, alternatively, could we use a helper function to check if any prefix matches a start of words in the dictionary?
Grid Validation

Finally, you're already handling empty grids and dictionaries well. Just to cover edge cases, do you think adding a quick check for non-rectangular grids might make it more foolproof?
What's Working Well
The backtracking with visited is working great; it's impressive how you've managed to make it concise and effective.
Converting the dictionary into a set for lookups is definitely a good call for efficiency.
The setGrid and setDictionary methods make the code flexible for different grid setups, which is a thoughtful touch.