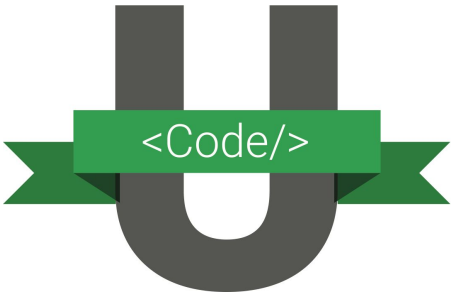


Assignment 3

If anything in the assignment is unclear, you have two options. You can ask for clarifications in the #assignments channel on Slack. It is also great if you can make assumptions: real-world problems are always unclear, and as engineers we want to move on and make progress, even if we need to re-adjust later. If you do make assumptions, please try to identify them and document them as comments in your code.

Word Search

Given a grid of letters and a lexicon (e.g., a representation of a set of words), find all the words from the lexicon that can be formed in the grid.

The rules for forming a word are:

- You can start from any position.
- You can move to one of the 8 adjacent cells (horizontally, vertically, and diagonally).
- You cannot visit the same cell twice in the same word.

Your function receives the grid and the lexicon. You should return the set of all words found.

For example:

A	A	R
T	C	D

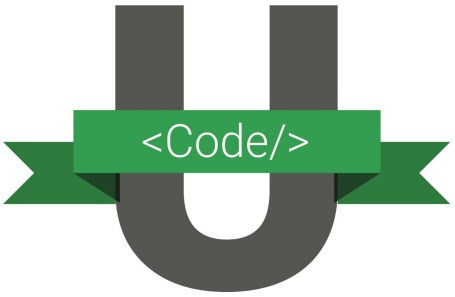
is a visual representation of a 2x3 grid of letters.

The lexicon is made up of words: CAR, CARD, CART, CAT.

The correct output in this case would be: CAR, CARD, CAT.

The lexicon should be represented as an object, class or interface (depending on the language you use) that defines two methods:

- one method called `isWord`, `IsWord` or `is_word` (depending on your naming conventions) that **takes a string and returns whether the string is a word in the lexicon**



- one method called `isPrefix`, `IsPrefix` or `is_prefix` (depending on your naming conventions) that **takes a string and returns whether the string is a prefix of at least one word in the lexicon**

Note: You should provide your own implementation of the lexicon class, e.g., a class that uses an array or list of words and iterates over them to determine whether a string is a word or a prefix.

Note: You need to define a data structure for the grid, which might be something as simple as an array or a list.

Optional Challenges

If you are done with the main assignment, you can pick up any of the optional challenges below and try to solve those as well. If you decide to take one or more optional challenges, make sure these are implemented as separate files from the original challenge. You might want to refactor some of your code so that you can reuse it across the main assignment and the optional challenges. You can submit a separate pull request for the optional challenges or include them with the main assignment.

1. What is the most efficient data structure / implementation of the lexicon?
2. The problem can be reduced to a graph traversal. Can you implement a solution that uses one of the standard graph traversal algorithms?
3. When using one of the standard graph traversal algorithms, you do not need to explicitly generate the graph corresponding to the input grid. How can do you so and what is the advantage?