Программа, решающая игру Letter Boxed.

Вам предлагается написать программу, которая решает следующий паззл.

Вокруг квадратного поля расположены 12 различных букв, по три на каждой стороне. Будем пока считать, что мы работаем с английским алфавитом.
Ваша задача — обойти все буквы таким образом, чтобы последовательные буквы образовали несколько слов из заданного словаря. Буквы можно посещать больше одного раза. Можно использовать словари для игры в `wordle` — этот или этот.

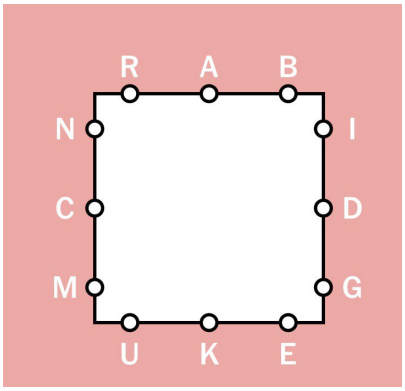
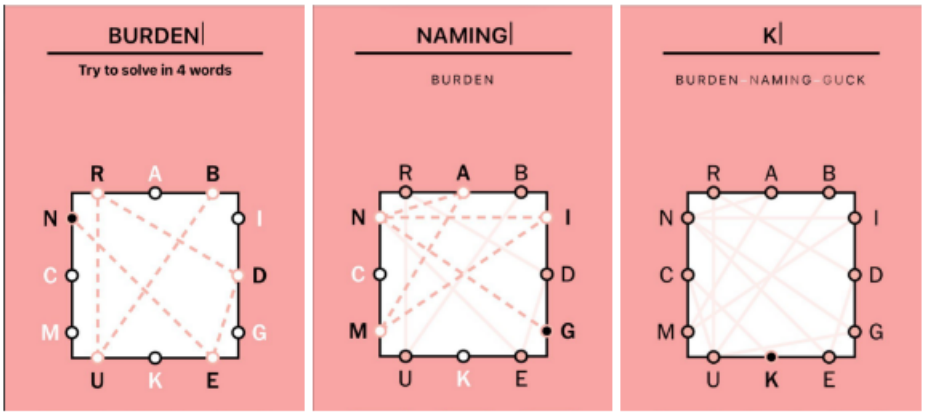


Рис. 1: Пример начального расположения букв

На обход букв и образуемые слова накладываются следующие ограничения:

- каждое слово, начиная со второго, должно начинаться с последней буквы предыдущего;
- все слова должны содержать не менее 3 букв;
- соседние буквы одного слова не должны находиться на одной стороне квадрата (см. картинку)

Вот пример решения приведённого паззла: слова `BURDEN` — `NAMING` — `GUCK`



Ваша задача — написать программу, которая по описанию поля (массив из четырёх 3-буквенных строк) и файлу со словарём выведет все решения паззла.
По правилам игры обычно накладывается ограничение на количество получившихся в итоге слов, вам же предлагается вывести **все** решения. Для удобства их стоит отсортировать по количеству входящих в решение слов.
Если программа работает слишком долго, возможно, стоит ограничить выдачу. Например, учитывать только более длинные слова (заменить ограничение на длину,

равное 3, на большее) или учитывать только решения из небольшого количества слов. Кроме того, на размер ответа существенно влияют буквы — менее частотные порождают существенно меньше слов.
Например, общее количество решений (учитывая порядок слов) в примере, приведённом в этом файле, равно 12164254 (и работает около 5 минут). А если ограничиться словами не короче 4 букв и последовательностями из не более, чем 5 слов, то количество вариантов равно 10475. На их вычисление уходит меньше 10 секунд.

Для удобства проверки стоит предусмотреть следующий формат получения и хранения результата:

- если программа требует ввести исходное поле, то она должна принимать строку вида `RAB IDG UKE NCM` (в любом регистре);
- если программа работает с входной строкой, инициализированной в коде, это место (инициализация) и название переменной должны быть разумными — изменение;
  Например, такой способ — ужасный:
  `x = [['R', 'A', 'B'], ['I', 'D', 'G'], ['U', 'K', 'E'], ['N', 'C', 'M']]`
  Такой тоже неудобный:
  `field = ['RAB', 'IDG', 'UKE', 'NCM']`
  А такой — вполне приемлемый:
  `field = 'RAB IDG UKE NCM'`
  Потом вы, конечно, можете преобразовать эту строку в удобный тип данных.
- при помощи **одной** константы устанавливать минимальную длину слова;
- при помощи **одной** константы устанавливать максимальное количество слов в одном решении (последовательности);
- если ваша программа допускает добавлять в последовательность слова, не «открывающие» новых букв, надо давать возможность при помощи **одной** константы разрешать/запрещать использование таких слов;
- отсортировать все последовательности по алфавиту.
- последовательности выводить в файл, разделяя слова пробелами;

Полезные функции, которые стоит реализовать в своей программе:

- функция, фильтрующая слова из полного набора: оставляет лишь те, которые реализуемы на данном поле.
  На вход принимает поле, заданное массивом 3-буквенных строк и массив слов, возвращает массив слов.
- функция, определяющая является ли данный набор слов закончившим игру, т.е. содержит ли все буквы поля?
  Принимает на вход текущий список слов и поле, заданное массивом 3-буквенных строк, возвращает `True`/`False`.