

GetAhead - Interview Practice 6

Car Plates Vocabulary - Solution

We need to find the shortest word from a vocabulary that includes **all** the letters from a given licence plate. The shorter the word, the better. The licence plates start with two or three letters, then they are followed by 5 characters, from which at most 2 are letters, the rest are digits.

Write a solution that will find the shortest words for 1000 licence plates.

You are given a vocabulary containing all valid words.

- Keep duplicate letters
- Ordering is irrelevant
- Case is irrelevant
- The vocabulary is sorted lexicographically
- The vocabulary contains about 4 million entries

Example:

For the licence plate **RT 123SO** the shortest word would be **SORT**:



for **RC 10014** the shortest word would be **CAR**.



Solution

JAVA

```
// JAVA SOLUTION
import java.util.*;

// This is a solution for the Car Plates Vocabulary problem.

public class ShortestWordInCarPlate {
    private Map<Integer, List<String>> vocabulary_by_size = new TreeMap<Integer,
List<String>>();

    ShortestWordInCarPlate(List<String> vocabulary) {
        for (String word : vocabulary) {
            int word_len = word.length();
            vocabulary_by_size.computeIfAbsent(word_len, k -> new
ArrayList<String>()).add(word);
        }
    }

    public String findShortestWord(String car_plate) {
        // find the letters in the car plate
        List<Character> letters = new ArrayList<>();
        for (int i = 0; i < car_plate.length(); i++) {
            char ch = car_plate.toLowerCase().charAt(i);
            if (Character.isLetter(ch)) {
                letters.add(ch);
            }
        }

        for (Map.Entry<Integer, List<String>> entry : vocabulary_by_size.entrySet())
        {
            // skip vocabulary sizes that are too small
            if (entry.getKey() < letters.size()) {
                continue;
            }

            for (String vocabulary_word : entry.getValue()) {
                // search for a vocabulary word with all the letters from the plate
                boolean is_valid = true;
                Map<Character, Integer> letter_counter = new TreeMap<Character,
Integer>();
                for (int i = 0; i < vocabulary_word.length(); i++) {
                    Character ch = vocabulary_word.charAt(i);
```

```

        int count = letter_counter.getOrDefault(ch, 0);
        letter_counter.put(ch, count + 1);
    }

    for (Character letter : letters) {
        int count = letter_counter.getOrDefault(letter, 0);
        if (count < 1) {
            is_valid = false;
            break;
        }
        letter_counter.put(letter, count - 1);
    }
    if (is_valid) {
        return vocabulary_word;
    }
}

return "";
}

public static void main(String[] args) {
    java.util.ArrayList<String> list = new java.util.ArrayList<String>();
    list.add("sort");
    list.add("car");
    list.add("rest");
    list.add("rust");
    list.add("sir");
    list.add("cast");
    ShortestWordInCarPlate finder = new ShortestWordInCarPlate(list);
    String result = finder.findShortestWord("RT 123 S0");
    System.out.println("Shortest word is " + result);
    result = finder.findShortestWord("RC 10014");
    System.out.println("Shortest word is " + result);
}
}

```

C++

```

// C++ Solution
#include <cassert>
#include <cctype>
#include <map>
#include <string>
#include <unordered_map>

```

```

#include <vector>

class ShortestWordFinder {
public:
    ShortestWordFinder(std::vector<std::string> vocabulary)
        : vocabulary_(vocabulary) {
        PreProcessvocabulary();
    }

    void PreProcessvocabulary() {
        assert(!vocabulary_.empty());

        for (const std::string& word : vocabulary_) {
            words_by_length_[word.size()].push_back(word);
        }
    }

    std::vector<char> ExtractLicensePlateLetters(
        const std::string& license_plate) {
        std::vector<char> license_plate_letters;
        for (char letter : license_plate) {
            char lowercase_letter = std::tolower(letter);
            if (lowercase_letter >= 'a' && lowercase_letter <= 'z')
                license_plate_letters.push_back(lowercase_letter);
        }
        return license_plate_letters;
    }

    std::string GetShortestWordIn(std::string license_plate) {
        if (vocabulary_.empty() || license_plate.empty()) return std::string();

        // Process the chars in |license_plate|.
        std::vector<char> license_plate_letters =
            ExtractLicensePlateLetters(license_plate);

        for (auto word_group : words_by_length_) {
            // Skip words that are too short.
            if (word_group.first < license_plate_letters.size()) continue;

            for (std::string vocabulary_word : word_group.second) {

                // Note the frequency of each letter in the vocabulary word.
                std::unordered_map<char, int> letter_frequencies;
                for (char letter : vocabulary_word)
                    ++letter_frequencies[std::tolower(letter)];

                // Match the frequency of each letter in |license_plate| against

```

```

// that of letters in |vocabulary_word|. We need |vocabulary_word| to
// have at least as many of each letter as there are in
// |license_plate_letters|.

bool has_enough_letters = true;
for (char letter : license_plate_letters) {
    if (--letter_frequencies[letter] < 0) {
        has_enough_letters = false;
        break;
    }
}
if (has_enough_letters) return vocabulary_word;
}
return std::string();
}

private:
std::vector<std::string> vocabulary_;
std::map<int, std::vector<std::string>> words_by_length_;
};

int main(int argc, char** argv) {
    ShortestWordFinder finder1(
        std::vector<std::string>({"step", "steps", "stripe", "stepple"}));
    assert((finder1.GetShortestWordIn("") == std::string()));
    assert((finder1.GetShortestWordIn("1s3 PSt") == std::string("steps")));

    ShortestWordFinder finder2(
        std::vector<std::string>({"looks", "pest", "stew", "show"}));
    assert((finder2.GetShortestWordIn("1s3 456") == std::string("pest")));

    ShortestWordFinder finder3(
        std::vector<std::string>({"SORT", "CAR", "REST", "RUST", "SIR", "CAST"}));
    assert((finder3.GetShortestWordIn("RT 123 SO") == std::string("SORT")));
    assert((finder3.GetShortestWordIn("RC 10014") == std::string("CAR")));

    return 0;
}

```

Python

<http://google3/experimental/users/vterron/get-ahead/2019/car-plate.py>

