JS

function find\_word\_concatenation(str, words) {

if (words.length === 0 || words[0].length === 0) {

return [];

}

wordFrequency = {};

words.forEach((word) => {

if (!(word in wordFrequency)) {

wordFrequency[word] = 0;

}

wordFrequency[word] += 1;

});

const resultIndices = [],

wordsCount = words.length;

wordLength = words[0].length;

for (i = 0; i < (str.length - wordsCount \* wordLength) + 1; i++) {

const wordsSeen = {};

for (j = 0; j < wordsCount; j++) {

next\_word\_index = i + j \* wordLength;

// Get the next word from the string

word = str.substring(next\_word\_index, next\_word\_index + wordLength);

if (!(word in wordFrequency)) { // Break if we don't need this word

break;

}

// Add the word to the 'wordsSeen' map

if (!(word in wordsSeen)) {

wordsSeen[word] = 0;

}

wordsSeen[word] += 1;

// no need to process further if the word has higher frequency than required

if (wordsSeen[word] > (wordFrequency[word] || 0)) {

break;

}

if (j + 1 === wordsCount) { // Store index if we have found all the words

resultIndices.push(i);

}

}

}

return resultIndices;

}

console.log(find\_word\_concatenation('catfoxcat', ['cat', 'fox']));

console.log(find\_word\_concatenation('catcatfoxfox', ['cat', 'fox']));

C++

using namespace std;

#include <iostream>

#include <string>

#include <unordered\_map>

#include <vector>

class WordConcatenation {

public:

static vector<int> findWordConcatenation(const string &str, const vector<string> &words) {

unordered\_map<string, int> wordFrequencyMap;

for (auto word : words) {

wordFrequencyMap[word]++;

}

vector<int> resultIndices;

int wordsCount = words.size(), wordLength = words[0].length();

for (int i = 0; i <= str.length() - wordsCount \* wordLength; i++) {

unordered\_map<string, int> wordsSeen;

for (int j = 0; j < wordsCount; j++) {

int nextWordIndex = i + j \* wordLength;

// get the next word from the string

string word = str.substr(nextWordIndex, wordLength);

if (wordFrequencyMap.find(word) ==

wordFrequencyMap.end()) { // break if we don't need this word

break;

}

wordsSeen[word]++; // add the word to the 'wordsSeen' map

// no need to process further if the word has higher frequency than required

if (wordsSeen[word] > wordFrequencyMap[word]) {

break;

}

if (j + 1 == wordsCount) { // store index if we have found all the words

resultIndices.push\_back(i);

}

}

}

return resultIndices;

}

};

int main(int argc, char \*argv[]) {

vector<int> result =

WordConcatenation::findWordConcatenation("catfoxcat", vector<string>{"cat", "fox"});

for (auto num : result) {

cout << num << " ";

}

cout << endl;

result = WordConcatenation::findWordConcatenation("catcatfoxfox", vector<string>{"cat", "fox"});

for (auto num : result) {

cout << num << " ";

}

cout << endl;

}

Python 3:

def find\_word\_concatenation(str1, words):

if len(words) == 0 or len(words[0]) == 0:

return []

word\_frequency = {}

for word in words:

if word not in word\_frequency:

word\_frequency[word] = 0

word\_frequency[word] += 1

result\_indices = []

words\_count = len(words)

word\_length = len(words[0])

for i in range((len(str1) - words\_count \* word\_length)+1):

words\_seen = {}

for j in range(0, words\_count):

next\_word\_index = i + j \* word\_length

# Get the next word from the string

word = str1[next\_word\_index: next\_word\_index + word\_length]

if word not in word\_frequency: # Break if we don't need this word

break

# Add the word to the 'words\_seen' map

if word not in words\_seen:

words\_seen[word] = 0

words\_seen[word] += 1

# No need to process further if the word has higher frequency than required

if words\_seen[word] > word\_frequency.get(word, 0):

break

if j + 1 == words\_count: # Store index if we have found all the words

result\_indices.append(i)

return result\_indices

def main():

print(find\_word\_concatenation("catfoxcat", ["cat", "fox"]))

print(find\_word\_concatenation("catcatfoxfox", ["cat", "fox"]))

main()

Java:

import java.util.\*;

class WordConcatenation {

public static List<Integer> findWordConcatenation(String str, String[] words) {

Map<String, Integer> wordFrequencyMap = new HashMap<>();

for (String word : words)

wordFrequencyMap.put(word, wordFrequencyMap.getOrDefault(word, 0) + 1);

List<Integer> resultIndices = new ArrayList<Integer>();

int wordsCount = words.length, wordLength = words[0].length();

for (int i = 0; i <= str.length() - wordsCount \* wordLength; i++) {

Map<String, Integer> wordsSeen = new HashMap<>();

for (int j = 0; j < wordsCount; j++) {

int nextWordIndex = i + j \* wordLength;

// get the next word from the string

String word = str.substring(nextWordIndex, nextWordIndex + wordLength);

if (!wordFrequencyMap.containsKey(word)) // break if we don't need this word

break;

wordsSeen.put(word, wordsSeen.getOrDefault(word, 0) + 1); // add the word to the 'wordsSeen' map

// no need to process further if the word has higher frequency than required

if (wordsSeen.get(word) > wordFrequencyMap.getOrDefault(word, 0))

break;

if (j + 1 == wordsCount) // store index if we have found all the words

resultIndices.add(i);

}

}

return resultIndices;

}

public static void main(String[] args) {

List<Integer> result = WordConcatenation.findWordConcatenation("catfoxcat", new String[] { "cat", "fox" });

System.out.println(result);

result = WordConcatenation.findWordConcatenation("catcatfoxfox", new String[] { "cat", "fox" });

System.out.println(result);

}

}