JS

function find\_string\_anagrams(str, pattern) {

let windowStart = 0,

matched = 0,

charFrequency = {};

for (i = 0; i < pattern.length; i++) {

const chr = pattern[i];

if (!(chr in charFrequency)) {

charFrequency[chr] = 0;

}

charFrequency[chr] += 1;

}

const resultIndices = [];

// our goal is to match all the characters from the 'charFrequency' with the current window

// try to extend the range [windowStart, windowEnd]

for (windowEnd = 0; windowEnd < str.length; windowEnd++) {

const rightChar = str[windowEnd];

if (rightChar in charFrequency) {

// decrement the frequency of matched character

charFrequency[rightChar] -= 1;

if (charFrequency[rightChar] === 0) {

matched += 1;

}

}

if (matched === Object.keys(charFrequency).length) { // have we found an anagram?

resultIndices.push(windowStart);

}

// shrink the sliding window

if (windowEnd >= pattern.length - 1) {

leftChar = str[windowStart];

windowStart += 1;

if (leftChar in charFrequency) {

if (charFrequency[leftChar] === 0) {

matched -= 1; // before putting the character back, decrement the matched count

}

charFrequency[leftChar] += 1; // put the character back

}

}

}

return resultIndices;

}

console.log(find\_string\_anagrams('ppqp', 'pq'));

console.log(find\_string\_anagrams('abbcabc', 'abc'));

C++:

using namespace std;

#include <iostream>

#include <string>

#include <unordered\_map>

#include <vector>

class StringAnagrams {

public:

static vector<int> findStringAnagrams(const string &str, const string &pattern) {

int windowStart = 0, matched = 0;

unordered\_map<char, int> charFrequencyMap;

for (auto chr : pattern) {

charFrequencyMap[chr]++;

}

vector<int> resultIndices;

// our goal is to match all the characters from the map with the current window

for (int windowEnd = 0; windowEnd < str.length(); windowEnd++) {

char rightChar = str[windowEnd];

// decrement the frequency of the matched character

if (charFrequencyMap.find(rightChar) != charFrequencyMap.end()) {

charFrequencyMap[rightChar]--;

if (charFrequencyMap[rightChar] == 0) {

matched++;

}

}

if (matched == (int)charFrequencyMap.size()) { // have we found an anagram?

resultIndices.push\_back(windowStart);

}

if (windowEnd >= pattern.length() - 1) { // shrink the window

char leftChar = str[windowStart++];

if (charFrequencyMap.find(leftChar) != charFrequencyMap.end()) {

if (charFrequencyMap[leftChar] == 0) {

matched--; // before putting the character back, decrement the matched count

}

// put the character back

charFrequencyMap[leftChar]++;

}

}

}

return resultIndices;

}

};

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

auto result = StringAnagrams::findStringAnagrams("ppqp", "pq");

for (auto num : result) {

cout << num << " ";

}

cout << endl;

result = StringAnagrams::findStringAnagrams("abbcabc", "abc");

for (auto num : result) {

cout << num << " ";

}

cout << endl;

}

Python:

def find\_string\_anagrams(str1, pattern):

window\_start, matched = 0, 0

char\_frequency = {}

for chr in pattern:

if chr not in char\_frequency:

char\_frequency[chr] = 0

char\_frequency[chr] += 1

result\_indices = []

# Our goal is to match all the characters from the 'char\_frequency' with the current window

# try to extend the range [window\_start, window\_end]

for window\_end in range(len(str1)):

right\_char = str1[window\_end]

if right\_char in char\_frequency:

# Decrement the frequency of matched character

char\_frequency[right\_char] -= 1

if char\_frequency[right\_char] == 0:

matched += 1

if matched == len(char\_frequency): # Have we found an anagram?

result\_indices.append(window\_start)

# Shrink the sliding window

if window\_end >= len(pattern) - 1:

left\_char = str1[window\_start]

window\_start += 1

if left\_char in char\_frequency:

if char\_frequency[left\_char] == 0:

matched -= 1 # Before putting the character back, decrement the matched count

char\_frequency[left\_char] += 1 # Put the character back

return result\_indices

def main():

print(find\_string\_anagrams("ppqp", "pq"))

print(find\_string\_anagrams("abbcabc", "abc"))

main()

Java:

import java.util.\*;

class StringAnagrams {

public static List<Integer> findStringAnagrams(String str, String pattern) {

int windowStart = 0, matched = 0;

Map<Character, Integer> charFrequencyMap = new HashMap<>();

for (char chr : pattern.toCharArray())

charFrequencyMap.put(chr, charFrequencyMap.getOrDefault(chr, 0) + 1);

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

// our goal is to match all the characters from the map with the current window

for (int windowEnd = 0; windowEnd < str.length(); windowEnd++) {

char rightChar = str.charAt(windowEnd);

// decrement the frequency of the matched character

if (charFrequencyMap.containsKey(rightChar)) {

charFrequencyMap.put(rightChar, charFrequencyMap.get(rightChar) - 1);

if (charFrequencyMap.get(rightChar) == 0)

matched++;

}

if (matched == charFrequencyMap.size()) // have we found an anagram?

resultIndices.add(windowStart);

if (windowEnd >= pattern.length() - 1) { // shrink the window

char leftChar = str.charAt(windowStart++);

if (charFrequencyMap.containsKey(leftChar)) {

if (charFrequencyMap.get(leftChar) == 0)

matched--; // before putting the character back, decrement the matched count

// put the character back

charFrequencyMap.put(leftChar, charFrequencyMap.get(leftChar) + 1);

}

}

}

return resultIndices;

}

public static void main(String[] args) {

System.out.println(StringAnagrams.findStringAnagrams("ppqp", "pq"));

System.out.println(StringAnagrams.findStringAnagrams("abbcabc", "abc"));

}

}