# Section 6: 100% OPTIONAL Challenges

Slides：<https://cs.slides.com/colt_steele/problem-solving-patterns>

## 35. IMPORTANT NOTE

### Coding Exercise 3: Frequency Counter - sameFrequency

Write a function called sameFrequency. Given two positive integers, find out if the two numbers have the same frequency of digits.

Your solution MUST have the following complexities:

Time: O(N)

sameFrequency(182,281) *//true*

sameFrequency(34,14) *//false*

sameFrequency(3589578, 5879385) *//true*

sameFrequency(22, 222) *// false*

sameFrequency Solution

function sameFrequency(num1, num2) {

let strNum1 = num1.toString();

let strNum2 = num2.toString();

if (strNum1.length !== strNum2.length) return false;

let countNum1 = {};

let countNum2 = {};

for (let i = 0; i < strNum1.length; i++) {

countNum1[strnum1[i]] = (countNum2[strNum1[i]] || 0) + 1;

}

for (let j = 0; j < strNum1.length; j++) {

countNum2[strNum2[j]] = (countNum2[strNum2[j]] || 0) + 1;

}

for (let key in countNum1) {

if (countNum1[key] !== countNum2[key]) return false;

}

return true;

}

### Coding Exercise 4: Frequency Counter / Multiple Pointers - areThereDuplicates

Implement a function called, areThereDuplicates which accepts a variable number of arguments, and checks whether there are any duplicates among the arguments passed in. You can solve this using the frequency counter pattern Or the multiple pointers pattern.

areThereDuplicates Solution(Frequency Counter)

function areThereDuplicates() {

let collection ={}

for (let val in arguments) {

collection[arguments[val]]= (collection[arguments[val]])

}

for (let key in collection) {

if (collection[key] >1) return true

}

return false;

}

areThereDuplicates Solution(Multiple Pointers)