Knowing What to Track: Introduction

Let's go over the Knowing What to Track pattern, its real-world applications, and some problems we can solve with it.

We'll cover the following

Overview

The **knowing what to track** pattern is mostly used to find the frequency count of the data. It's able to avoid the $O(n^2)$ time by simplifying the problem and identifying certain key numeric properties in the given data.

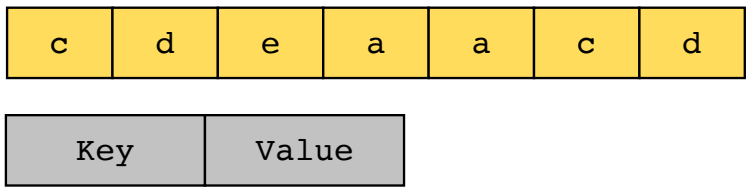
This pattern is often used with an array and strings, and helps solve problems that require counting data frequencies or returning boolean values (TRUE/FALSE). More often than not, the problems that can be solved using this pattern include comparison and checks, such as the examples given below:

- Check if the values of the first array have corresponding squared values in the second array
- Check if two strings are anagrams
- Check if the player wins the game

Hashmaps can store the frequency count of the given data while allowing search and retrieval of data in constant $O(1)$ time.

There might be a chance to solve some problems without using hash maps. We can use separate arrays or variables to store the frequency counts in those cases.

Let's look at the following illustration to understand how we can use hash maps to keep the counts of given data:



1 of 8

Examples

The following examples illustrate some problems that can be solved with this approach:

Contains duplicate II

Check if there are two distinct indexes, i and j, in the array, nums, such that nums[i] == nums[j] and abs(i - j) < = k.

nums	3	2	1	3	2	1

k = 4

Key	Value
0	3
1	2
2	1
3	3
4	2
5	1

Are there two distinct indexes? Yes.

Does my problem match this pattern?

Yes, it does if the problem seems hard with a naive approach of at least $O(n^2)$ time complexity, if not worse. Yet, on careful inspection, we realize that there are certain key characteristics of the input data that, if tracked, allow us to solve the problem far more efficiently.

Generally speaking, this approach is worth trying when we are required to choose from a fixed set of possibilities: Yes / No, True / False, Valid / Invalid, Player 1 / Player 2.

Key characteristics could include the frequency of characters in a string, the pattern of generation of a sequence of permutations, or the implications of a given move in a game like tic-tac-toe or chess.

No, if either of these conditions is fulfilled:

- We are unable to identify key characteristics that simplify the solution.
- The problem is a variant of a problem that falls in this pattern, except that, instead of requiring a True / False response, we're required to locate a particular element. For example, in the case of the palindrome permutation problem, our efficient solution only responds in a True / False sense, but if we had been required to

is insufficient.

Real-world problems

Many problems in the real world use the knowing what to track pattern. Let's look at some examples.

- **Video streaming:** To enhance the video streaming user experience, revamp the continue-watching bar that will return the most frequently watched show.
- **E-commerce:** Show product recommendations with items that are frequently viewed together.

Strategy time!

Match the problems that can be solved using the knowing what to track pattern.

> **Note:** Select a problem in the left-hand column by clicking it, and then click one of the two options in the right-hand column.

Match The Answer

Select an option from the left-hand side

Given a string, count the substrings that have all of the vowels.

Given a half-filled board, fill the empty cells to solve the Sudoku puzzle.

Given a string, check if any of its permutations can form a valid palindrome.

Check if a given string is a valid palindrome.

Knowing What to Track

Some other pattern

Reset | Show Solution | Submit