Ask a Question

Introduction to Knowing What to Track

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...

About the pattern

Imagine you work for a big online shop like Amazon, and you need to suggest products to users based on their purchase history. There is a massive database with millions of users and the items they’ve bought. To find out which products are popular among each user, a count for each product purchased is required. Doing it manually with traditional methods would be highly inefficient and time-consuming. Therefore, you would need to set up a system that automatically keeps track of how often each product is purchased by each user. It’s like having a digital tally counter for every product and every user. This way, when it’s time to make recommendations, you can do it efficiently because you already have a clear picture of which products each user prefers.

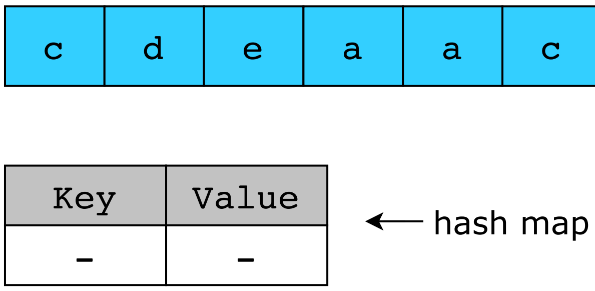
This is exactly what the knowing what to track pattern does. It involves counting the occurrences of elements in a given data structure, mostly an array or a string, and using this frequency information to solve the problem efficiently. The pattern can be divided into two main phases:

- **Counting phase:** This is to iterate through the elements of the data structure and count the frequency of each element. We can use different data structures to achieve this, such as hash maps, dictionaries, arrays, or even simple variables.
- **Utilization phase:** Once the frequencies are calculated, we can use this information to solve the specific problem at hand. This could involve any task that benefits from knowing the frequency of elements, such as:
    - Find the most frequent element.
    - Identify elements that occur only once.
    - Check if two arrays are permutations of each other.
    - Check if the player wins the game.

Let’s analyze how we can use the two most common data structures—hash map and array—for frequency counting. The **hash map** stores elements as keys and their frequencies as values. While iterating over the data structure, for each element encountered, we update its frequency as follows:

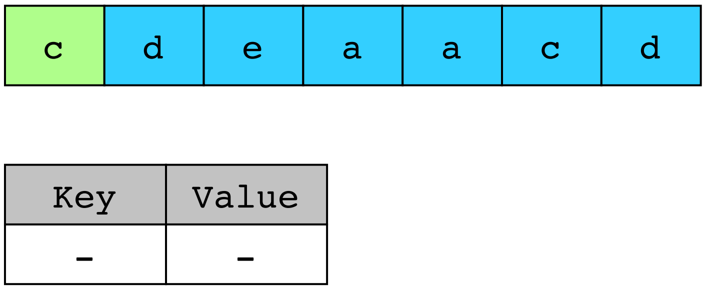
- If the element is already a key in the hash table, its frequency value is incremented by 1.
- Otherwise, the element is added to the hash table with a frequency of 1.

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



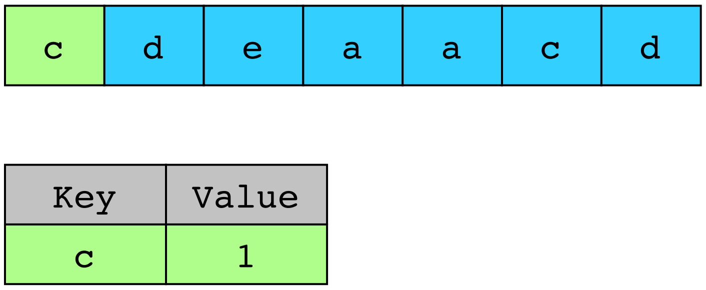
Let's count the frequency of each element in the given input using hash map.

1 of 10



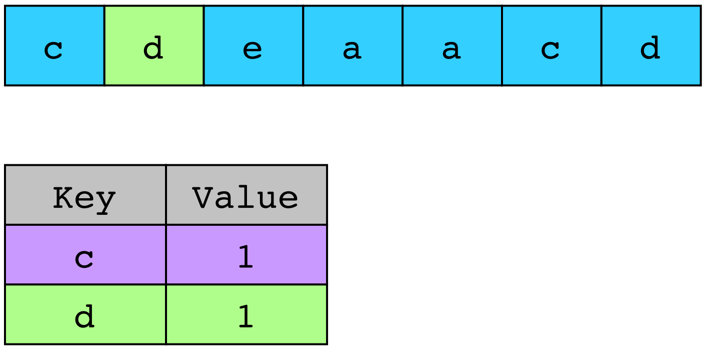
Because **c** is not already in the hash map, let's add a new entry and update its frequency to 1.

2 of 10



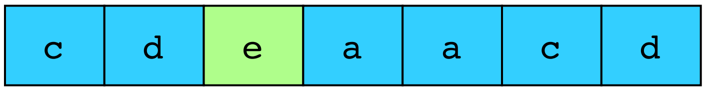
The frequency of the first element, **c**, has been updated.

3 of 10



Because **d** is not already in the hash map, let's add a new entry and update its frequency to 1.

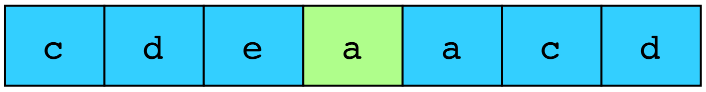
4 of 10



Key	Value
c	1
d	1
e	1

Because **e** is not already in the hash map, let's add a new entry and update its frequency to 1.

5 of 10



Key	Value
c	1
d	1
e	1
a	1

Because **a** is not already in the hash map, let's add a new entry and update its frequency to 1.

6 of 10

c	d	e	a	a	c	d

Key	Value
c	1
d	1
e	1
a	2

Note that **a** is already present in the hash map, so let's just update its frequency to 2.

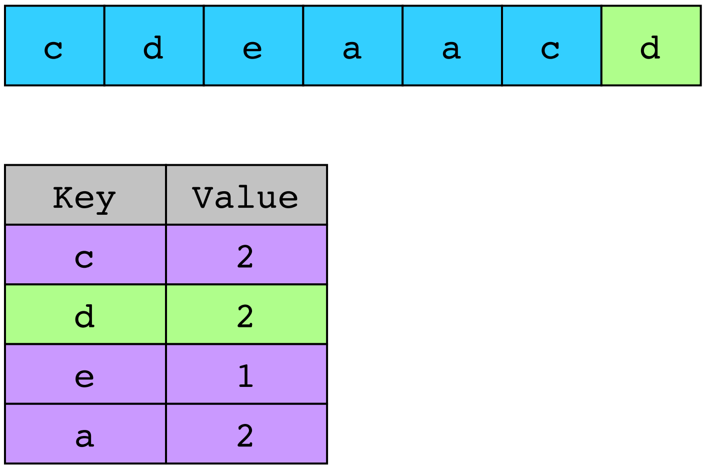
7 of 10

c	d	e	a	a	c	d

Key	Value
c	2
d	1
e	1
a	2

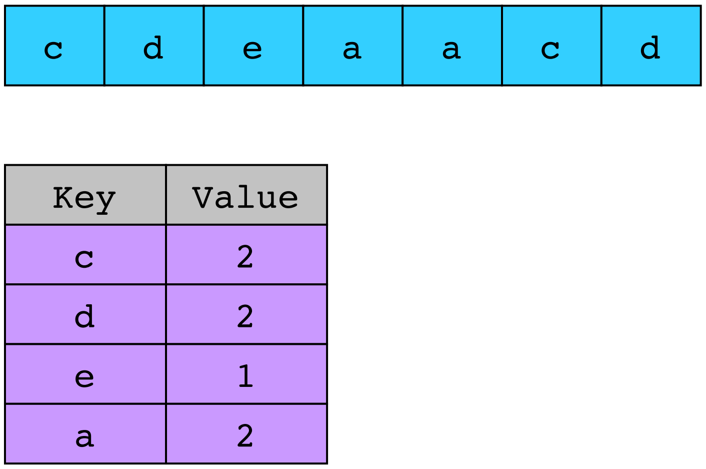
Because **c** is already present in the hash map, so let's just update its frequency to 2.

8 of 10



Because **d** is already present in the hash map, so let's just update its frequency to 2.

9 of 10



We have traversed the entire array and stored the count of each element in the hash map.

10 of 10

When using an array for frequency counting, we utilize the indexes of th
array to represent the elements, and the corresponding values at those

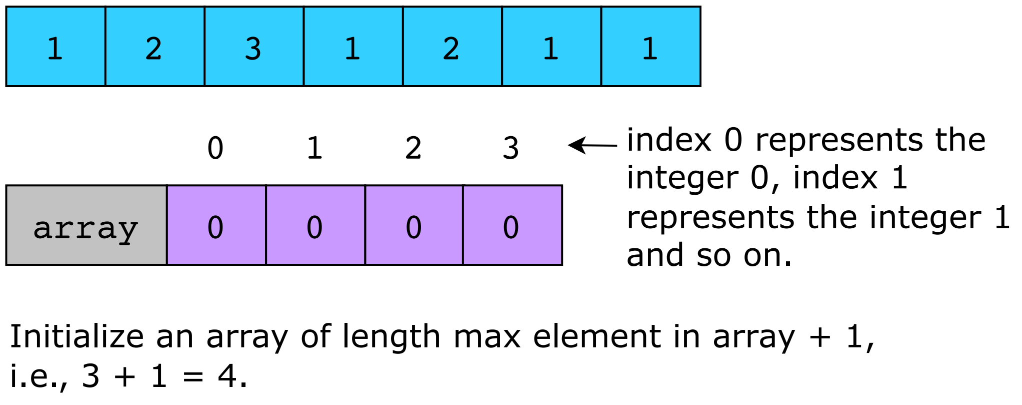
indexes represent the frequencies of those elements. We iterate over each element in our dataset, and for each element encountered, we update its frequency as follows:

- Use the value of the element as an index in the array to access its corresponding frequency counter. Increment the frequency counter by 1.

> **Note:** If we have noninteger elements in the dataset, e.g., alphabets, we can have a mapping of those elements to array indexes. For example, for the dataset $[a, b, c]$, we can assume that a is represented as an index 0, b as index 1, and c as index 2.

Using arrays is efficient when we know the range of elements in advance and when that range is relatively small compared to the size of the dataset.

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

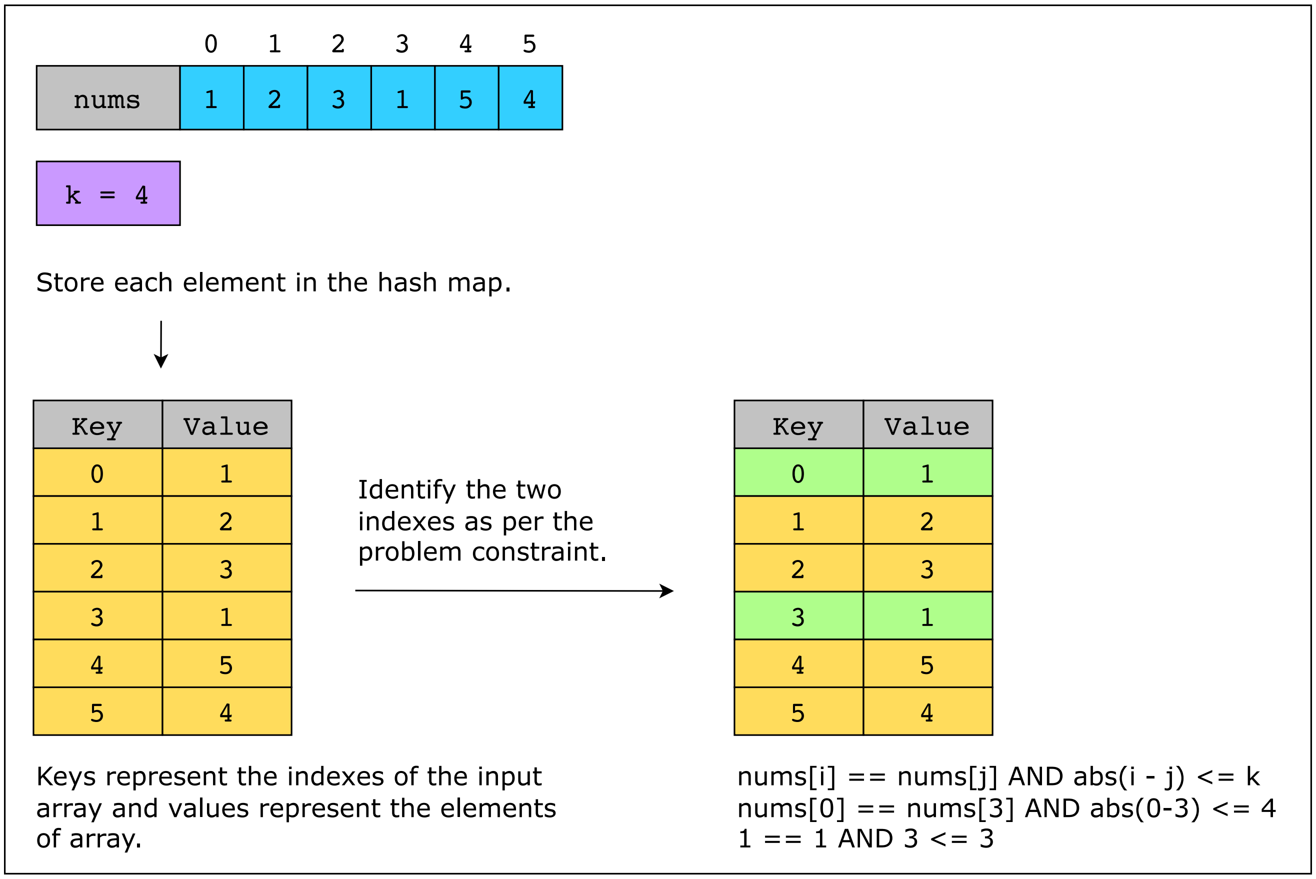

1 2 3 1 2 1 1
0 1 2 3
array 0 0 0 0
index 0 represents the integer 0, index 1 represents the integer 1 and so on.
Initialize an array of length max element in array + 1, i.e., 3 + 1 = 4.

1 of 9

Examples

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

1. **Contains duplicate II:** Given an integer array `nums` and an integer `k`, return TRUE if there are two distinct indexes `i` and `j` in the array such that `nums[i]` $==$ `nums[j]` and `abs(i - j)` $\leq$ `k`.


0 1 2 3 4 5
nums 1 2 3 1 5 4
k = 4
Store each element in the hash map.
Identify the two indexes as per the problem constraint.

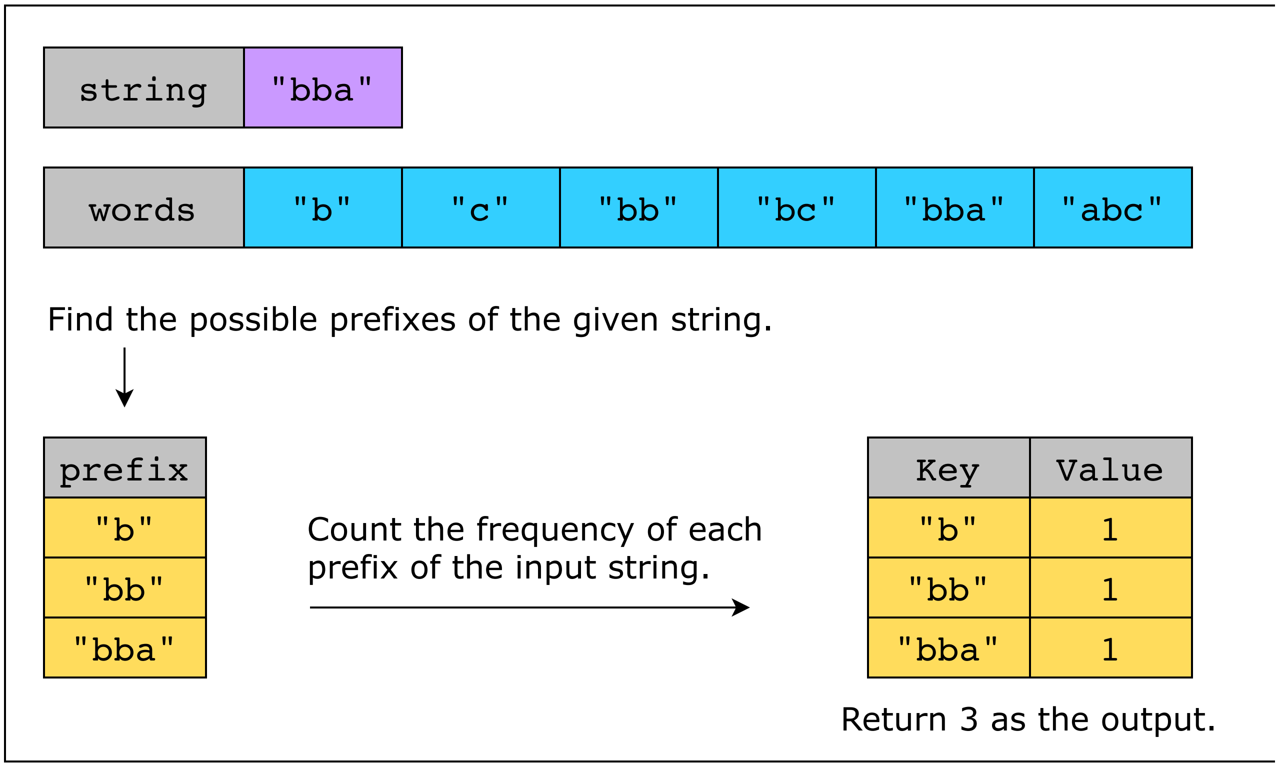
Key	Value
0	1
1	2
2	3
3	1
4	5
5	4

Keys represent the indexes of the input array and values represent the elements of array.

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

nums[i] == nums[j] AND abs(i - j) <= k
nums[0] == nums[3] AND abs(0-3) <= 4
1 == 1 AND 3 <= 3

2. **Count prefixes of a given string:** Given an input string, count the number of prefixes present in it. A prefix of a string is a substring tha occurs at the beginning of the string.



Does your problem match this pattern?

Yes, if either of these conditions are fulfilled:

- **Frequency tracking:** If the problem involves counting the frequencies of elements in a dataset, either individually or in combinations.
- **Pattern recognition:** Look for patterns in the data where certain elements or combinations of elements repeat frequently, indicating a potential use for frequency counting.
- **Fixed set of possibilities:** The problem requires choosing the output from a fixed set of possibilities: Yes/ No, True/False, Valid/Invalid, Player 1/Player 2.

Real-world problems

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

- **DNA sequence analysis**: In genomics, frequency counting is used to analyze the frequency of nucleotides or amino acids in DNA or protein sequences. This information is used for tasks such as identifying genetic variations, predicting protein structure, and studying evolutionary relationships.
- **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.
- **Clickstream analysis**: In web analytics, frequency counting is used to analyze the frequency of user interactions, such as page views, clicks, and conversions. This helps in understanding user behavior, optimizing website design, and improving user experience.

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

Knowing What to Track

the vowels.