Two Pointers: Introduction

Let's go over the Two Pointers pattern, its real-world applications, and some problems we can solve with it.

We'll cover the following

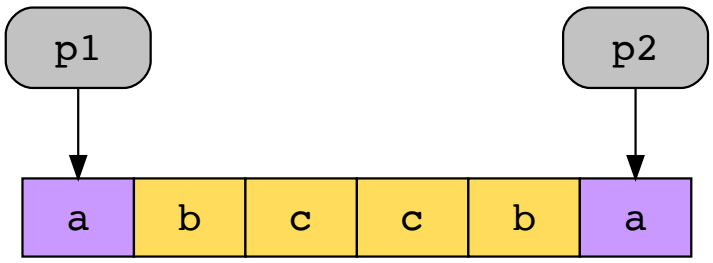
Overview

As the name suggests, the **two pointers** pattern uses two pointers to iterate over an array or list until the conditions of the problem are satisfied. This is useful because it allows us to keep track of the values of two different indexes in a single iteration. Whenever there's a requirement to find two data elements in an array that satisfy a certain condition, the two pointers pattern should be the first strategy to come to mind.

The pointers can be used to iterate the data structure in one or both directions, depending on the problem statement. For example, to identify whether a string is a palindrome, we can use one pointer to iterate the string from the beginning and the other to iterate it from the end. At each step, we can compare the values of the two pointers and see if they meet the palindrome properties.

The naive approach to solving this problem would be using nested loops, with a time complexity of $O(n^2)$. However, by using two pointers moving towards the middle from either end, we exploit the symmetry property of palindromic strings. This allows us to compare the elements in a single loop, making the algorithm more efficient with a time complexity of $O(n)$.

Here's how the pointers will move along the string:



1 of 3

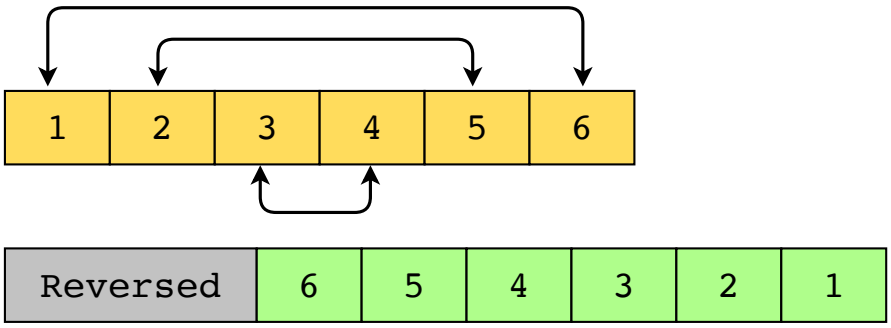
Essentially, the two pointers pattern is an application of the prune-and-search strategy, in which, at every step, we're able to safely prune—that is, eliminate—a set of possible solutions.

Examples

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

Reversing an Array

Traverse the array from the beginning and the end,
and swap the values until we reach the middle.



1 of 2

Does my problem match this pattern?

- Yes, if all of these conditions are fulfilled:
  - The input data can be traversed in a linear fashion, that is, it's in an array, in a linked list, or in a string of characters.
  - The input data is sorted, or else, arranged in a way that is relevant to the problem, such as numeric data sorted in ascending or descending order, or characters arranged symmetrically.
  - We are only considering the *two* elements in the input data that are pointed to by the two pointers rather than the whole set of elements located between the two pointers.

Additionally, problems in this pattern usually involve comparing or swapping values pointed to by two indexes. In less common cases, each index may move along a separate array or string.

- No, if either of these conditions is fulfilled:

- The input data cannot be traversed in a linear fashion, that is, it's neither in an array, nor in a linked list, nor in a string of characters.
- The problem requires an exhaustive search of the solution space, that is, eliminating one solution does not eliminate any others.

Real-world problems

Many problems in the real world use the two pointers pattern. Let's look at some examples.

same route (in reverse) back to the source that the request packet took to the destination. Sometimes, the path may differ due to errors, and we can tolerate at most one diversion router. The path—in terms of the IDs of routers along the way—is recorded in an array present inside the packet. Two pointers can be used to determine whether the same path is followed from the source to the destination and from destination to source to identify the paths with more than one diversion router. So, our topology remains intact and transmission errors are detected.

- **Product suggestions:** While shopping online, when customers view their cart and the current total doesn't qualify for free shipping, we want to show them pairs of products that can be bought together to make the total amount equal to the amount required to be eligible for free delivery. Two pointers can be used to suggest the pairs that add up to the required cost for free shipping.

Strategy time!

Match the problems that can be solved using the two pointers 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

Check if a string is a palindrome.

Find any 3 values in an array that sum up to 825.

Given a string, find its longest substring that includes 7 distinct characters.

Generate a string of n balanced parentheses.

Two Pointers

Some other pattern

Reset | Show Solution | Submit