









Introduction to Two Pointers

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



- About the pattern
- Examples
- Does your problem match this pattern?
- Real-world problems
- Strategy time!

About the pattern#

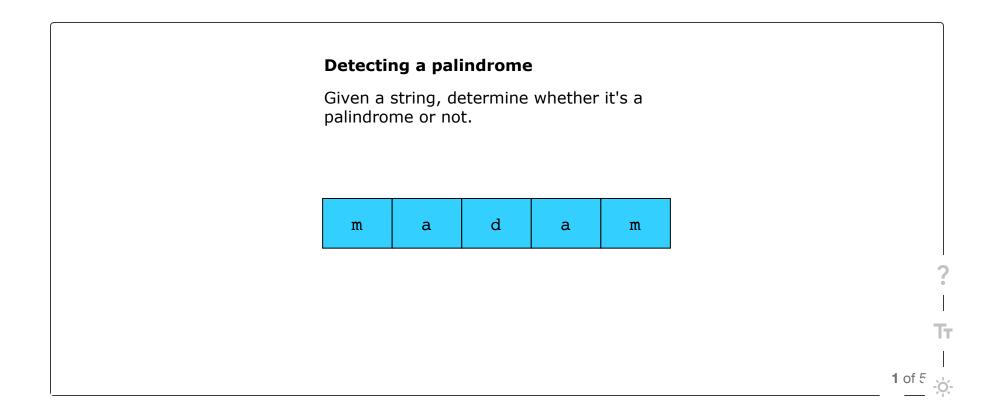


The **two pointers** pattern is a versatile technique used in problem-solving to efficiently traverse or manipulate sequential data structures, such as arrays or linked lists. As the name suggests, it involves maintaining two pointers that traverse the data structure in a coordinated manner, typically starting from different position or



moving in opposite directions. These pointers dynamically adjust based on specific conditions or criteria, allowing for the efficient exploration of the data and enabling solutions with optimal time and space complexity. 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 through 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 toward 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).

Examples

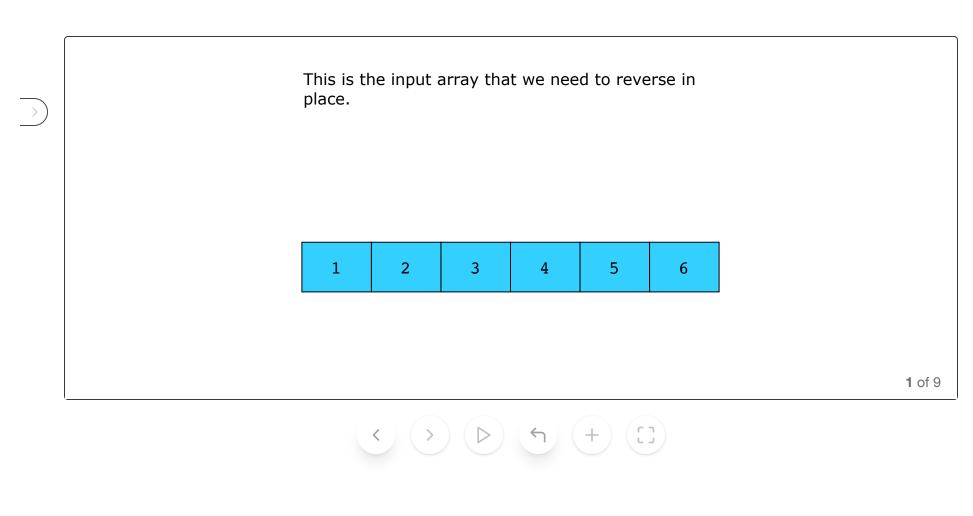
The following examples illustrate some problems that can be solved with this approach:

1. Reversing an array: Given an array of integers, reverse it in place.





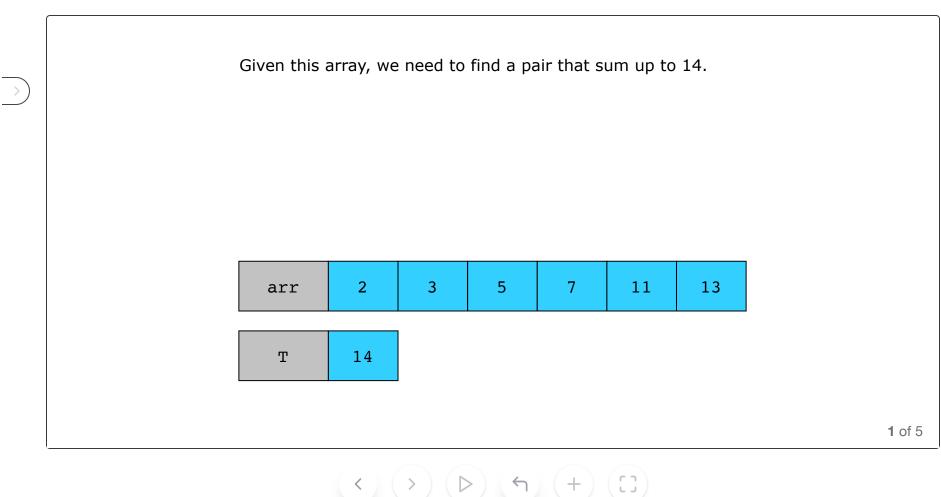




2. **Pair with given sum in a sorted array:** Given a sorted array of integers, find a pair in the array that sums to a number T.









Does your problem match this pattern?

Yes, if all of these conditions are fulfilled:

• Linear data structure: The input data can be traversed in a linear fashion, such as an array, linked list, or string.



- Process pairs: Process data elements at two different positions simultaneously.
- Dynamic pointer movement: Both pointers move independently of each other according to certain conditions or criteria. In addition, both pointers might move along the same or two different data structures.

Real-world problems

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

• **Memory management:** The two pointers pattern is vital in memory allocation and deallocation. The memory pool is initialized with two pointers: the *start* pointer, pointing to the beginning of the available memory block, and the *end* pointer, indicating the end of the block. When a process or data structure requests memory allocation, the *start* pointer is moved forward, designating a new memory block for allocation. Conversely, when memory is released (deallocated), the *start* pointer is shifted backward, marking the deallocated memory as available for future allocations.

Strategy time!









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

Statement Match With

Α

Given an array of integers, move all zeros to the end of the array.

В

Generate a string of n balanced parentheses.

Two Pointers

Some other pattern





