Meta Interview 2025-04-07

Preparation 2025-04-02

Introduction to Two Pointers

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 positions 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.

[Images]

1 / 5

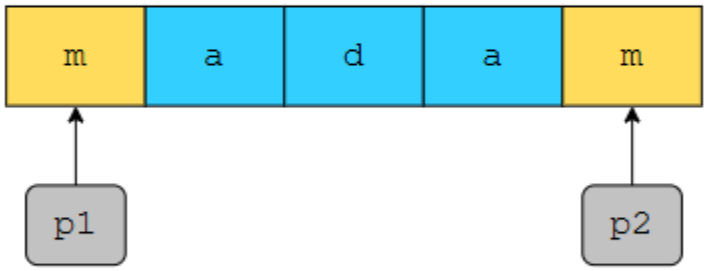
The naive approach to solving this problem would be using nested loops, with a time complexity of O(n2)$O(n2)$. However, by using two pointers moving toward the middle from either end, we exploit the symmetry property of palindromic strings. This allows us to compare the elements in a single loop, making the algorithm more efficient with a time complexity of O(n)$O(n)$.

Detecting a palindrome

Given a string, determine whether it's a palindrome or not.

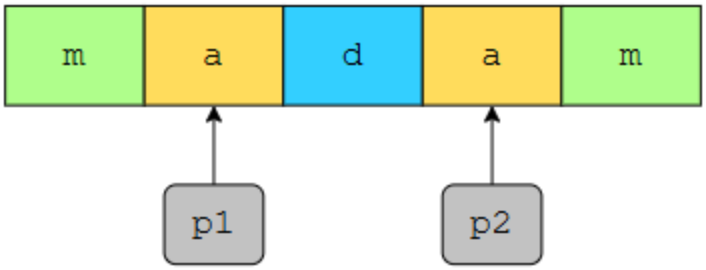
Detecting a palindrome

We initialize two pointers at the start and end of the string.
In each iteration, we compare the characters at both pointers, and if they match, we move the pointers inward.



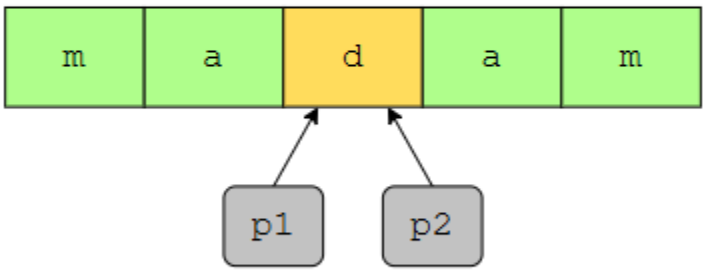
Detecting a palindrome

The two characters matched, so we moved the pointers inward i.e., we moved **p1** forward and **p2** backward.



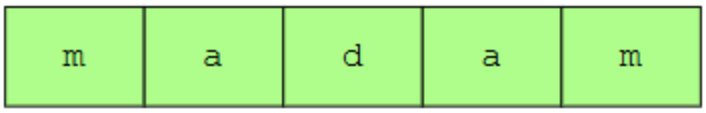
Detecting a palindrome

The two characters matched again, so we moved the pointers inward.



Detecting a palindrome

Because the pointers have met, we conclude that the given string is a palindrome.



Deep Dive

Examples

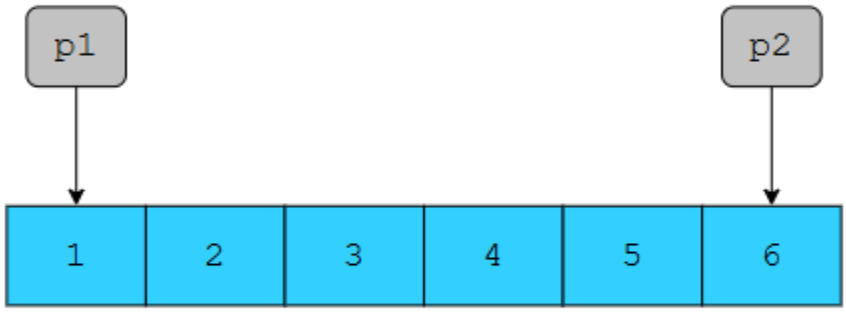
The following examples illustrate some problems that can be solved with this approach:

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

This is the input array that we need to reverse in place.

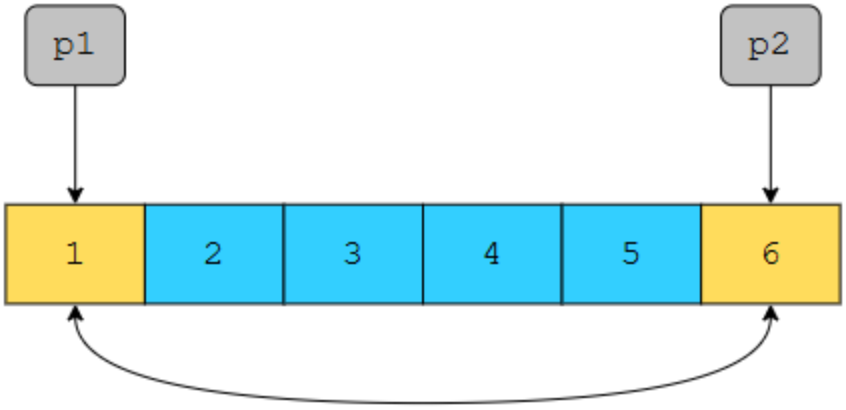
1 / 9

Initialize two pointers, **p1** and **p2**, at the start and end of the array, respectively.

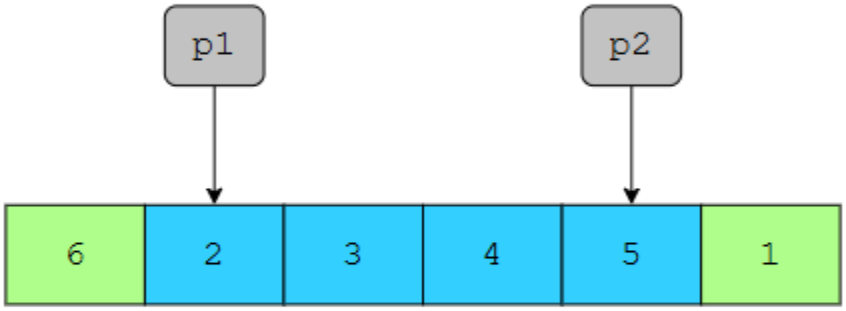


2 / 9

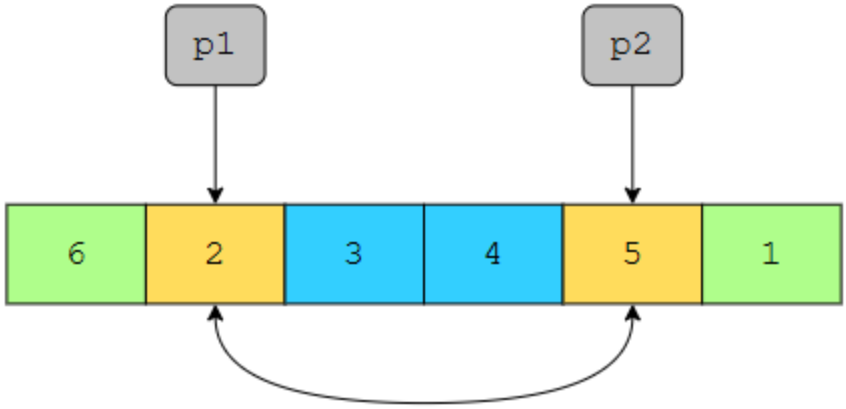
Swap the values of the indexes pointed to by **p1** and **p2**.



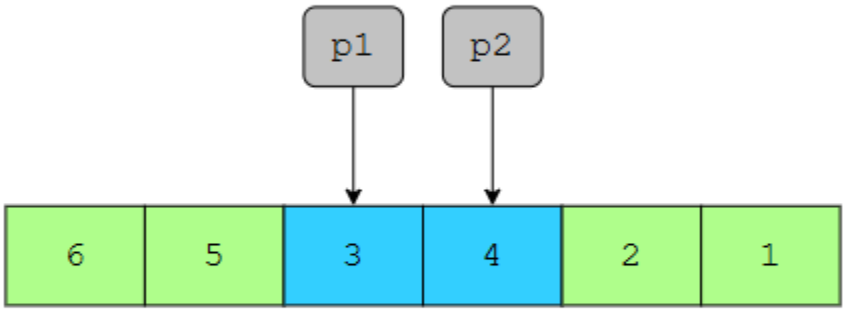
After swapping the values, move the pointers inward, i.e., increment **p1** and decrement **p2**.



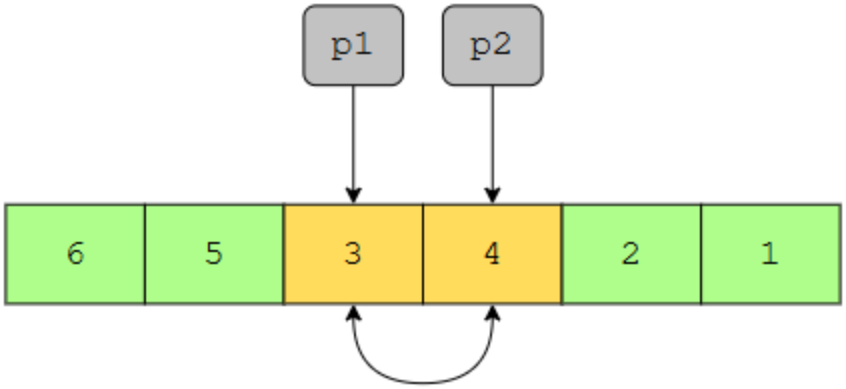
Swap the values of the indexes pointed to by **p1** and **p2**.



After swapping the values, move the pointers inward, i.e., increment **p1** and decrement **p2**.



Swap the values of the indexes pointed to by **p1** and **p2**.



Examples

Examples

Sample example 1

Input

nums	-2	0	2	-2	1	-1

Output

[[-2, 0, 2], [-1, 0, 1]]

Valid triplets that initially appear include:

1. **[-2, 0, 2]** (Valid)
2. **[0, 2, -2]** (Duplicate of the first triplet)
3. **[-1, 0, 1]** (Valid)

Notice that the order of the triplet dose not matter.

Sample example 2

Input

nums	-3	-1	-1	0	1	2	3	3

Output

[[-3, 0, 3], [-3, 1, 2], [-1, -1, 2], [-1, 0, 1]]

Computed triplets include:

1. **[-3, 0, 3]** (Valid)
2. **[-3, 1, 2]** (Valid)
3. **[-1, -1, 2]** (Valid)
4. **[-1, 0, 1]** (Valid)
5. **[-3, 0, 3]** (Duplicate of the first result due to repeated numbers in the input)

Sample example 3

Input

nums	0	0	0	0

Output

[[0, 0, 0]]

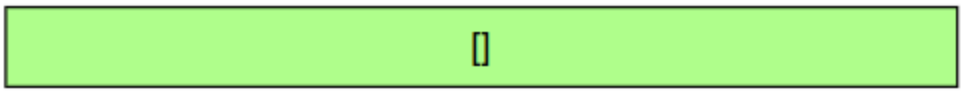
Computed triplets include:

1. **[0, 0, 0]** (Valid)
2. **[0, 0, 0]** (Duplicate of first triplet)

Sample example 4

Input

Output



As all the numbers are positive, it is impossible to find a combination of three numbers that sums to zero.