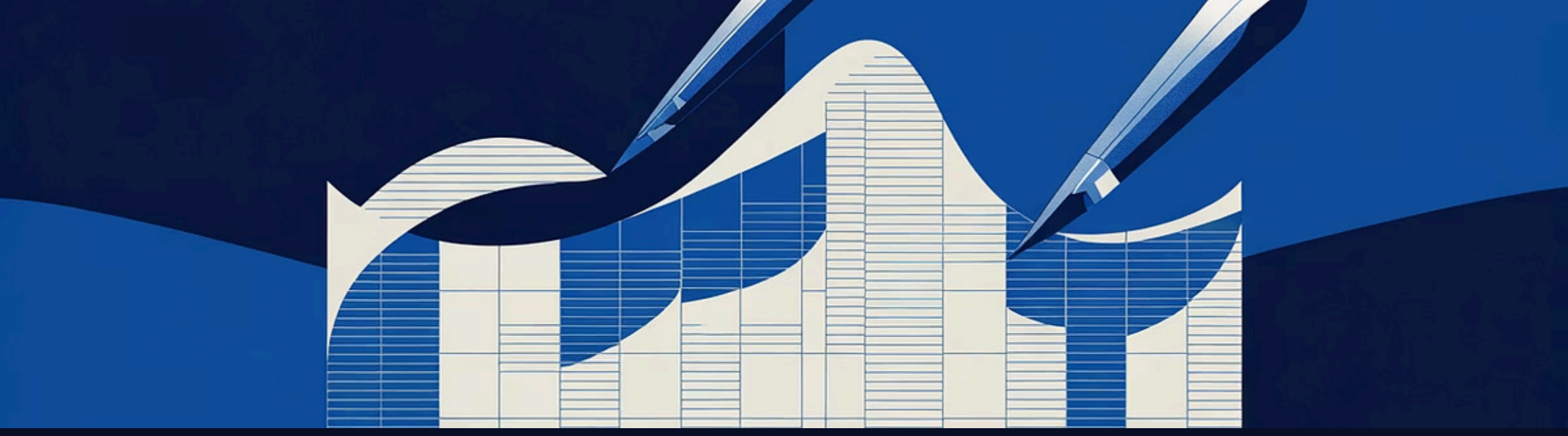
TWO POINTERS PATTERN



What is Two Pointers?

Two Pointers

A technique where two variables (pointers) traverse a data structure (like an array, string, or linked list) simultaneously from different positions.

Efficiency

Reduces time complexity from $O(n^2)$ to $O(n)$.

In-Place Operation

Uses $O(1)$ space, modifying data without extra memory.

Types of Two Pointers

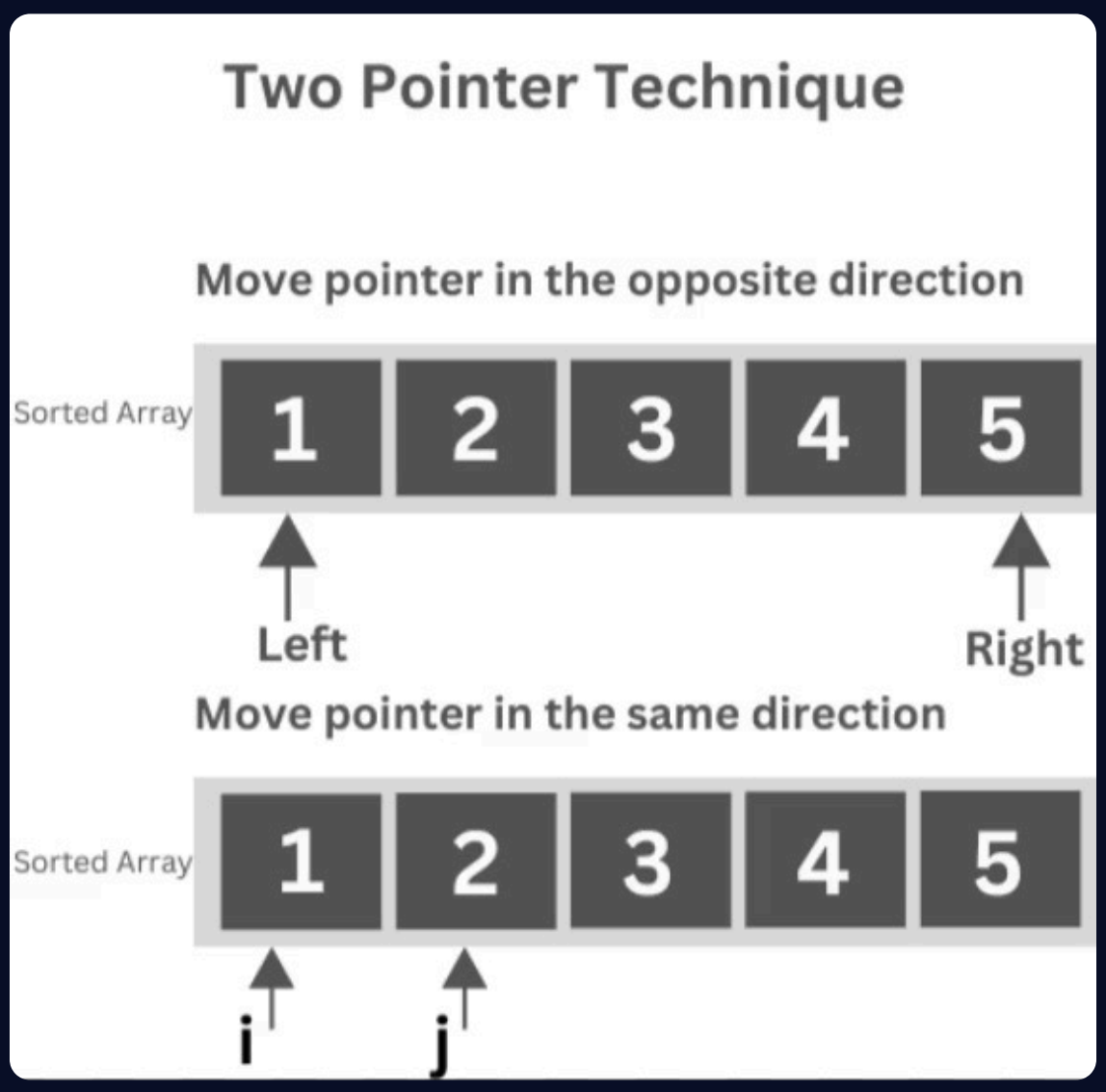
Not all pointers move the same way. Two primary patterns exist:

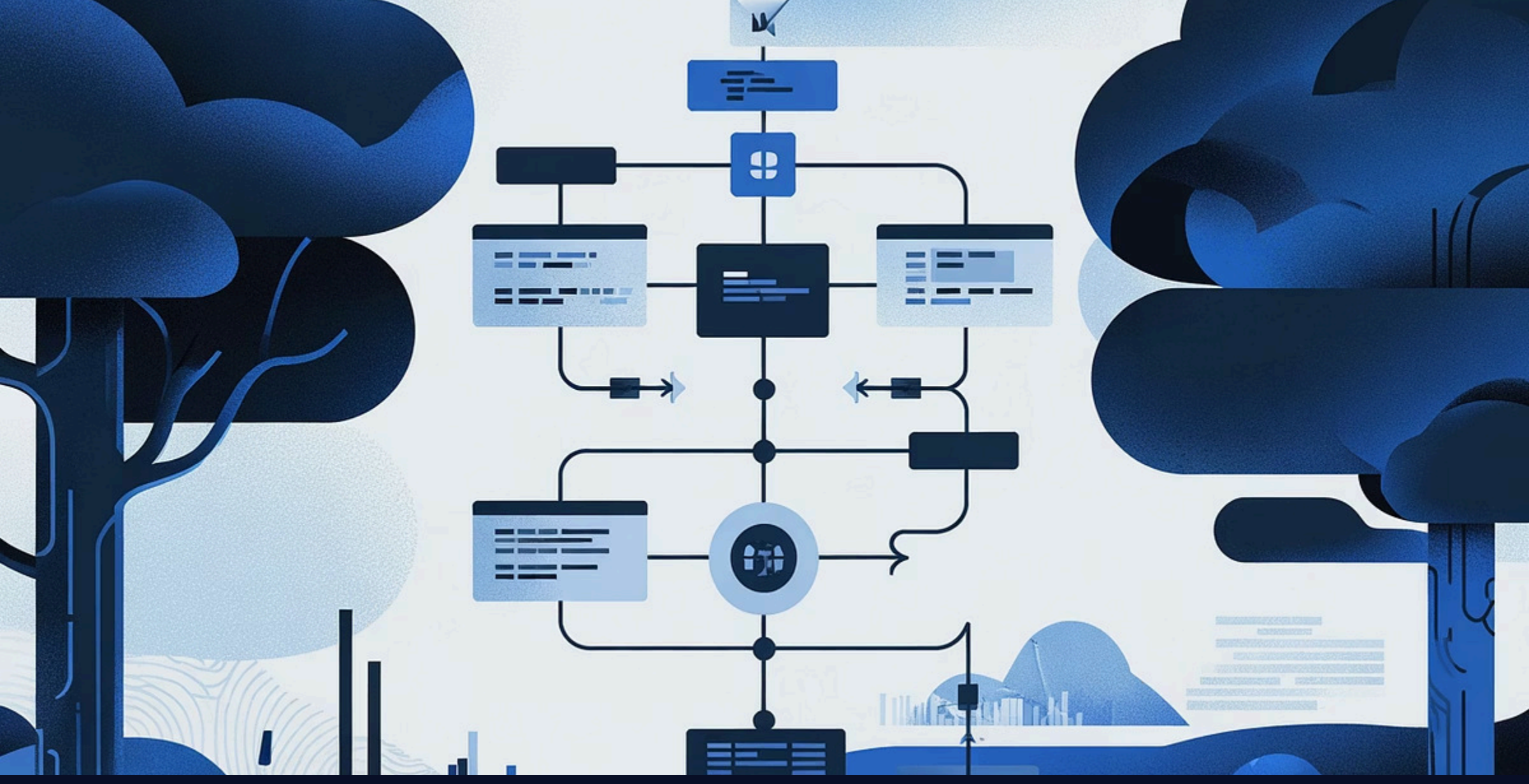
Same-Direction

Both move forward at different speeds to detect cycles or find middle elements.

Opposite-Direction

Pointers start at ends and move toward each other.





When Should You Think of Two Pointers?

1. Is input an array or string?
2. Is data sorted or sortable?
3. Comparing pairs, duplicates, or ranges?
4. Can indices replace nested loops?

Pattern Recognition Trick

Look for these keywords to instantly trigger the pattern:

"sorted array"

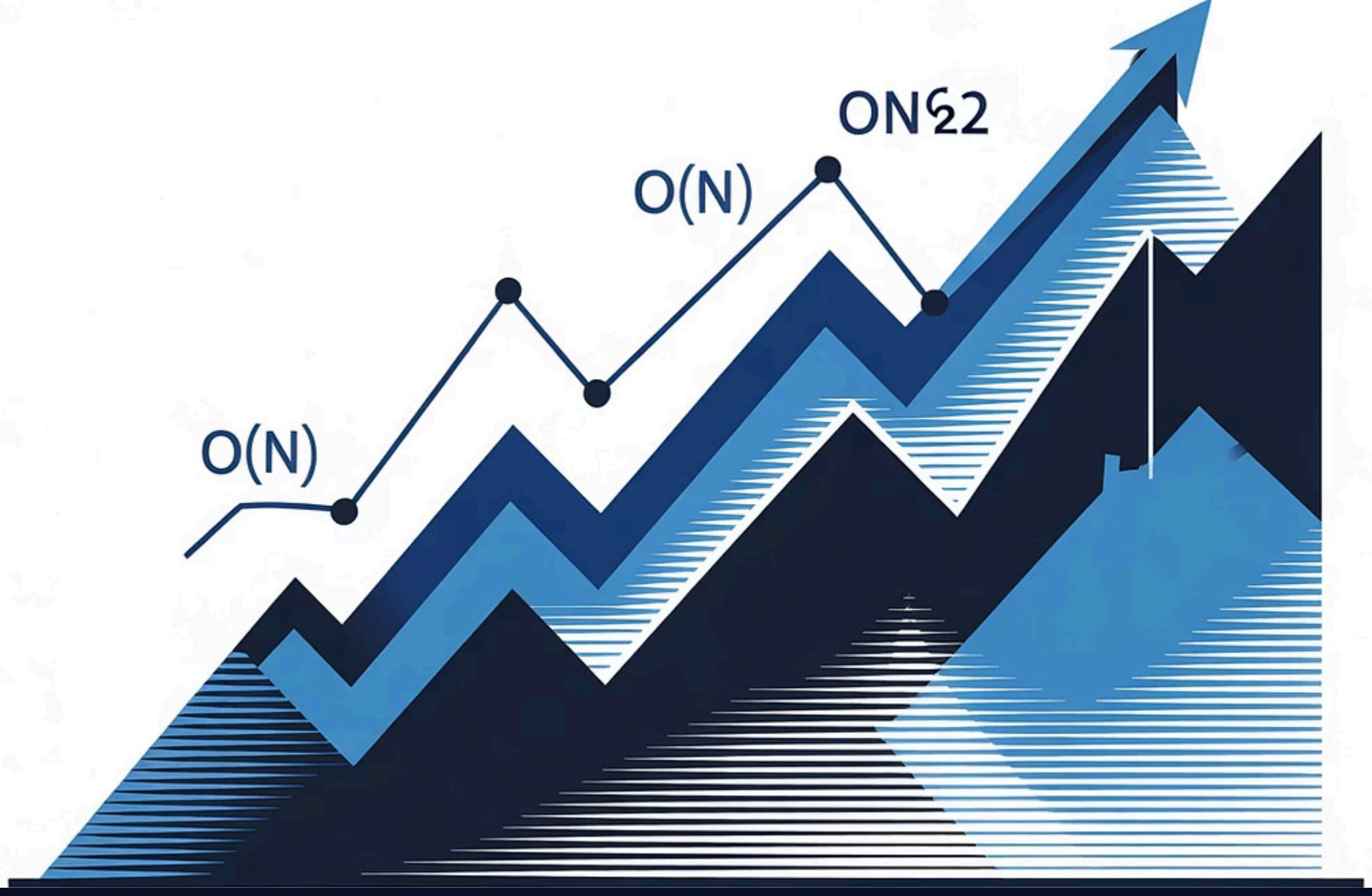
"find pair/triplet"

"merge/rearrange"

"detect cycle"

"check palindrome"

"remove duplicates in-place"



Time Complexity Advantage

Why do interviewers love this? The math doesn't lie:

O(n)

Two Pointers

Execution time stays linear and predictable.

O(1)

Space

Constant space by modifying data in-place.

LeetCode Problems :

125. Valid Palindrome

```java
class Solution {
public boolean isPalindrome(String s) {

int i = 0;
int j = s.length()-1;

while(i < j){
    char left = s.charAt(i);
    char right = s.charAt(j);

    if(!Character.isLetterOrDigit(left)){
        i++;
        continue;
    }

    if(!Character.isLetterOrDigit(right)){
        j--;
        continue;
    }

        if(Character.toLowerCase(left) != Character.toLowerCase(right)){
            return false;
        }

        i++;
        j--;
    }
    return true;
  }
}

T.C- O(n)
S.C- O(1)
```

344. Reverse String

```
class Solution {
public void reverseString(char[] s) {
int i = 0;
int j = s.length-1;


    while(i<j){
        char temp = s[i];
        s[i] = s[j];
        s[j] = temp;

        i++;
        j--;
    }
  }
}

T.C- O(n)
S.C- O(1)
```

977. Square of Sorted Array

```
class Solution {
public int[] sortedSquares(int[] nums) {
int[] res = new int[nums.length];

    int i = 0;
    int j = nums.length-1;

    int k = nums.length-1;

    while (i <= j){
        if(Math.abs(nums[i]) < Math.abs(nums[j])){
            res[k] = nums[j] * nums[j];
            j--;
        }
        else{
            res[k] = nums[i] * nums[i];
            i++;
        }
        k--;
    }
    return res;
}
}


T.C- O(n)
S.C- O(n)
```

680. Valid Palindrome II

```java
class Solution {
    public boolean palindromeHelper(int i , int j, String s){
        while(i < j){
            if(s.charAt(i) != s.charAt(j)){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
    public boolean validPalindrome(String s) {

        int i = 0;
        int j = s.length()-1;

        while(i < j){
            char left = s.charAt(i);
            char right = s.charAt(j);

            if(left != right){

                return palindromeHelper(i+1, j, s) || palindromeHelper(i, j-1, s);

            }else{
                i++;
                j--;
            }
        }
        return true;
    }
}

T.C- O(n)
S.C- O(1)
```

Valid Word Abbreviation

```java
class Solution {
    public boolean validWordAbbreviation(String word, String abbr) {
        int i = 0;
        int j = 0;

        while (i < word.length() && j < abbr.length()) {
            char w_c = word.charAt(i);
            char a_c = abbr.charAt(j);

            if (Character.isDigit(a_c)) {
                if (a_c == '0') {
                    return false;
                }

                int curr = 0;
                while (j < abbr.length() && Character.isDigit(abbr.charAt(j))) {
                    curr = curr * 10 + (abbr.charAt(j) - '0');
                    j = j + 1;
                }

                i = i + curr;
            } else {
                if (w_c != a_c) {
                    return false;
                }

                i = i + 1;
                j = j + 1;
            }
        }

        return i == word.length() && j == abbr.length();
    }
}
T.C- O(m+n)
S.C- O(1)
```