Problem: Reverse an Array

Level: Easy

Given an array, reverse the order of its elements.

For example, [3,5,2,5,2,3,9] → [9,3,2,5,2,5,3]

Questions to Clarify:
Q. Should the output be a new array or the existing array modified?
A. Modify the existing array.

Solution:
Keep two pointers, one at each end of the array. Swap their values and move both pointers inwards.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a
few lines quickly to plan out your solution. Your actual solution should be in
a real language and use good syntax.)

start = 0, end = a.length - 1
while start < end
     swap a[start] and a[end]
     increase start, decrease end
```

Test Cases:
Edge Cases: empty array, null array
Base Cases: single element
Regular Case: 2 elements, 3 elements, odd elements, even elements

Time Complexity: O(n) aka linear time

Space Complexity: O(1) aka constant space

```
public static void reverse(int[] a) {
    if (a == null)
        return;

    int start = 0, end = a.length - 1;
    while (start < end) {
        Utils.swap(a, start, end);
        start++;
        end--;
    }
}
```

Problem: Two Sum

Level: Easy

2 Sum Problem: Given a sorted array of integers, find two numbers in the array that sum to a given integer target.

For example, if a = [1,2,3,5,6,7] and target = 11, the answer will be 5 and 6.

Questions to Clarify:
Q. How do return the output?
A. Return it as a pair of numbers.

Q. What to return if there is no result?
A. Return null.

Q. Can the array have duplicates?
A. Yes

Q. If there are more than one pair that qualify, which pair should I return?
For example, if a=[2,3,4,5] and target=7, the answer could be either {2,5} or {3,4}
A. Return whichever you like. As long as you return a correct pair, it's fine.

Solution:
Keep two pointers, one at each end of the array (*start* and *end*). If the sum of the two pointers is greater than *target*, we want to decrease the sum, so we decrease *end*. If the sum is less than *target*, we increase *start* to increase the sum. Do this until we either find the *target* sum or the pointers converge.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a
few lines quickly to plan out your solution. Your actual solution should be in
a real language and use good syntax.)

start = 0, end = a.length - 1
while start < end
     sum = a[start] + a[end]
     if sum less than target, increase start
     if sum greater than target, decrease end
     if sum == target, return a[start] and a[end] as a pair.
```

Test Cases:
Edge Case: empty array, single element
Base Case: 2 elements (with and w/o sum)
Regular Case: 3 elements, 4 elements (with and w/o sum and duplicates)

Time Complexity: O(n)

Space Complexity: O(1)

```
public static Pair<Integer> twoSum(int[] a, int target) {
    if (a == null)
        return null;

    int start = 0, end = a.length - 1;

    while (start < end) {
        int sum = a[start] + a[end];
        if (sum < target)
            start++;
        else if (sum > target)
            end--;
        else
            return new Pair<Integer>(a[start], a[end]);
    }

    return null;
}
```