**Algorithm Steps**

1. **Sort the Array**:
   * First, sort the array of numbers. Sorting takes O(nlog⁡n)O(n \log n)O(nlogn) time.
2. **Initialize Two Pointers**:
   * Set one pointer (left) to the start of the array (index 0) and the other pointer (right) to the end of the array (index n−1n-1n−1).
3. **Iterate with Two Pointers**:
   * While left is less than right:
     + Calculate the sum of the elements at the two pointers: sum = arr[left] + arr[right].
     + If sum is equal to xxx, then the two elements that sum to xxx are found, and we return true.
     + If sum is less than xxx, increment the left pointer (move right) to increase the sum.
     + If sum is greater than xxx, decrement the right pointer (move left) to decrease the sum.
4. **Termination**:
   * If the loop ends without finding such a pair, return false.

**Pseudocode:  
function hasPairWithSum(arr, n, x):**

**// Step 1: Sort the array**

**sort(arr)**

**// Step 2: Initialize two pointers**

**left = 0**

**right = n - 1**

**// Step 3: Iterate with two pointers**

**while left < right:**

**sum = arr[left] + arr[right]**

**if sum == x:**

**return true // Found the pair**

**if sum < x:**

**left++ // Move left pointer to increase sum**

**else:**

**right-- // Move right pointer to decrease sum**

**// Step 4: No pair found**

**return false**

**Time Complexity**

* **Sorting the array takes O(nlog⁡n)O(n \log n)O(nlogn).**
* **The two-pointer iteration runs in O(n)O(n)O(n).**
* **Overall, the time complexity of the algorithm is O(nlog⁡n)O(n \log n)O(nlogn).**

**Example**

**Let's say we have an array of numbers: [10,15,2,7][10, 15, 2, 7][10,15,2,7] and we want to find if there are two numbers that sum to x=17x = 17x=17.**

1. **Sort the Array: [2,7,10,15][2, 7, 10, 15][2,7,10,15]**
2. **Initialize Pointers: left at 2, right at 15.**
3. **Iterate:**
   * **sum = 2 + 15 = 17 → Found the pair (2, 15).**
4. **Return True.**