**Bidirectional Linear Search**

**Concept**

* **Standard Linear Search**: Searches sequentially from the start (index 0) to the end (index n-1).
* **Improvement**: Bidirectional Linear Search enhances efficiency by searching from both ends simultaneously (start and end) toward the middle. This can halve the average search time by reducing the number of elements checked, especially when the target’s position is unpredictable.
* **How It Works**: Uses two pointers—one starting at the beginning (left) and one at the end (right)—moving them toward each other until the target is found or the pointers meet.

**Example with Numbers**

Let’s search for the target value **7** in the array: [4, 2, 7, 1, 9, 3].

* **Array**: [4, 2, 7, 1, 9, 3]
* **Target**: 7
* **Initial Setup**:
  + left = 0 (points to 4)
  + right = 5 (points to 3, since array length is 6 and 5 is the last index)
* **Step 1**:
  + Check arr[left] = 4 → Not 7
  + Check arr[right] = 3 → Not 7
  + Update: left = 1, right = 4
* **Step 2**:
  + Check arr[left] = 2 → Not 7
  + Check arr[right] = 9 → Not 7
  + Update: left = 2, right = 3
* **Step 3**:
  + Check arr[left] = 7 → Found 7!
  + Return left = 2 (the index of 7)

**Result**: The target 7 is found at index 2 after 3 steps, faster than checking all 6 elements sequentially from one end.

**Psuedocode**

function BidirectionalLinearSearch(arr, target):

left = 0

right = length(arr) - 1

while left <= right:

if arr[left] == target:

return left

if arr[right] == target:

return right

left = left + 1

right = right - 1

return -1

**The following is the code implementation**

**C++ code Implementation**

#include <iostream>

// Bidirectional Linear Search function for a raw array

int bidirectionalLinearSearch(int arr[], int size, int target) {

int left = 0; // Start from the beginning

int right = size - 1; // Start from the end

while (left <= right) { // Continue until pointers meet or cross

if (arr[left] == target) {

return left; // Target found at left index

}

if (arr[right] == target) {

return right; // Target found at right index

}

left++; // Move left pointer inward

right--; // Move right pointer inward

}

return -1; // Target not found

}

// Example usage

int main() {

int arr[] = {4, 2, 7, 1, 9, 3};

int size = sizeof(arr) / sizeof(arr[0]);

int target = 7;

int result = bidirectionalLinearSearch(arr, size, target);

if (result != -1) {

std::cout << "Target " << target << " found at index " << result << std::endl;

} else {

std::cout << "Target " << target << " not found" << std::endl;

}

return 0;

}