



Find two numbers in a sorted array that add upto a target

Session No.: 3

Course Name: Advanced Algorithmic Problem Solving

Course Code: R1UC601B

Instructor Name: Dr. Mohamad Faiz

Duration: 50 mins

Date of Conduction of Class:

Recap of Previous Topic

Split Array in two equal parts

[1-mins]

Brute Force Approach

```
bool canSplitBruteForce(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int prefixSum = 0, suffixSum = 0;

        for (int j = 0; j <= i; j++)
            prefixSum += arr[j];

        for (int j = i + 1; j < n; j++)
            suffixSum += arr[j];

        if (prefixSum == suffixSum)
            return true;
    }
    return false;
}
```



- **Time Complexity:** $O(n^2)$
 (since we re-compute the suffix sum in each iteration).
- **Space Complexity:** $O(1)$
 (only integer variables used).

10

```
bool canSplitOptimized(int arr[], int n) {
    int totalSum = 0;
    for (int i = 0; i < n; i++)
        totalSum += arr[i];
    int prefixSum = 0;
    for (int i = 0; i < n - 1; i++) {
        prefixSum += arr[i];
        if (prefixSum * 2 == totalSum)
            return true;
    }
    return false;
}
```

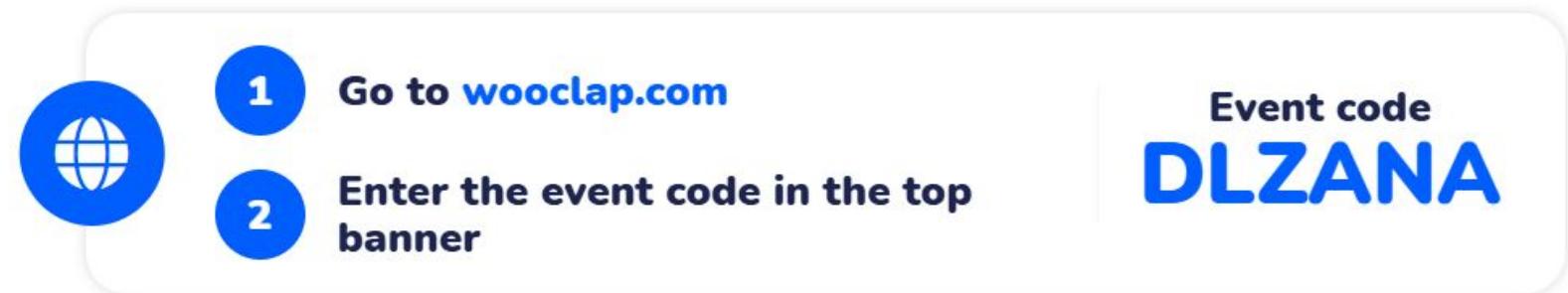
Optimized Approach (Prefix sum)



- **Time Complexity:** $O(n)$
- **Space Complexity:** $O(1)$

Pre-Class Assessment

[2-mins]



[1-mins]



finding two numbers in a sorted array that sum up to a given target.

Learning Outcomes

[2-mins]

By the end of this session, You will be able to:

Explain the two-pointer approach for solving the problem.



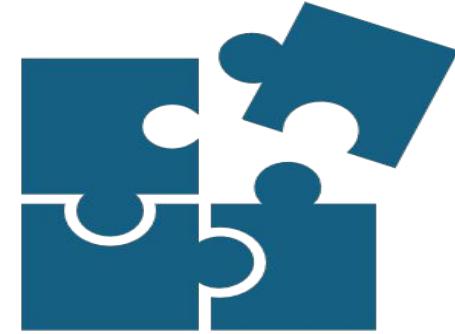
Implement an efficient algorithm to find two numbers that add up to a target in a sorted array.

Session Outline

- Introduce students to the problem of finding two numbers in a sorted array that sum up to a given target.
- Demonstrate the two-pointer approach for solving this problem efficiently.
- Provide hands-on coding practice for implementing this algorithm.

Activity-1 (Think – Pair – Share)

[1-mins]



Discuss in pairs possible methods to find two numbers that add up to a target in a sorted array

Example Input: [1, 2, 3, 4, 6],
Target = 6

Example Output: (2, 4)

Concept and Definition for (LO-1) Find Two Numbers That Add Up to a Target

Brute Force Approach:

Iterate through each element and check if there is another element that sums up to the target.

Input:

`arr[] = {1, 2, 3, 4, 6, 8, 9}, target = 10`

Output:

Pair found: (1, 9)

Explanation:

$1+9=10$ (Pair found!)

```
void findPairBruteForce(int arr[], int n, int target)
{
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] + arr[j] == target) {
                cout << "Pair found: (" << arr[i] << ", " <<
                    arr[j] << ")\n";
                return;
            }
        }
    }
    cout << "No pair found\n";
}
```

Time Complexity: $O(n^2)$
Space Complexity: $O(1)$

Concept and Definition for (LO-2) Optimized Approach



Two-Pointer Technique:

Use two pointers: one at the start (left) and one at the end (right).

- If $\text{arr}[\text{left}] + \text{arr}[\text{right}] == \text{target}$, return the pair.
- If $\text{arr}[\text{left}] + \text{arr}[\text{right}] < \text{target}$, move left forward.
- If $\text{arr}[\text{left}] + \text{arr}[\text{right}] > \text{target}$, move right backward.

Time Complexity: $O(n)$ (Single pass)

Activity 2 (Pen Paper): Coding Problem

Problem Statement:

- Given a sorted array and a target sum, find two numbers that add up to the target.
- Example Input: [1, 2, 3, 4, 6], Target = 6
- Example Output: (2, 4)

Optimized Approach (Two Pointer)



```
pair<int, int> findTwoSum(vector<int>& arr, int target)
{
    int left = 0, right = arr.size() - 1;
    while (left < right)
    {
        int sum = arr[left] + arr[right];
        if (sum == target)
            return {arr[left], arr[right]};
        else if (sum < target)
            left++;
        else
            right--;
    }
    return {-1, -1}; // No valid pair found
}
```

- **Time Complexity:** O(n)
- **Space Complexity:** O(1)

```
int main() {  
    vector<int> arr = {1, 2, 3, 4, 6};  
    int target = 6;  
    pair<int, int> result = findTwoSum(arr, target);  
    if (result.first != -1)  
        cout << "Pair found: (" << result.first << ", " <<  
        result.second << ")" << endl;  
    else  
        cout << "No valid pair found." << endl;  
    return 0;  
}
```

Assessment: WooFlash Quiz



Share this link

<https://app.wooflash.com/moodle/DO9M9...>

Copy



Share this code

DO9M9GWI



Copy



Summary

[1-mins]



Concept and Definition for (LO-1) Find Two Numbers That Add Up to a Target



Brute Force Approach:

Iterate through each element and check if there is another element that sums up to the target.

Input:

`arr[] = {1, 2, 3, 4, 6, 8, 9}, target = 10`

Output:

Pair found: (1, 9)

Explanation:

$1+9=10$ (Pair found!)



Concept and Definition for (LO-2) Optimized Approach



Two-Pointer Technique:

Use two pointers: one at the start (left) and one at the end (right).

- If $\text{arr}[\text{left}] + \text{arr}[\text{right}] == \text{target}$, return the pair.
- If $\text{arr}[\text{left}] + \text{arr}[\text{right}] < \text{target}$, move left forward.
- If $\text{arr}[\text{left}] + \text{arr}[\text{right}] > \text{target}$, move right backward.

Time Complexity: $O(n)$ (Single pass)

Learning Outcomes

[2-mins]

Ensure attainment of LO's in alignment to the learning activities:

Explain the two-pointer approach for solving the problem.



Implement an efficient algorithm to find two numbers that add up to a target in a sorted array.

Discussion on the post session activities

Key points:

1. **Brute-Force Approach:**
 - Time Complexity: $O(n^2)$
 - Space Complexity: $O(1)$
2. **Prefix Sum Approach:**
 - Time Complexity: $O(n)$
 - Space Complexity: $O(1)$

Homework Problem:

"Given a sorted array [1, 7, 5, 3, 10] unsorted array and a target sum of 8, find two numbers that add up to the target."

Next Session:

Majority Element: Find the element that appears more than $n/2$ times in an array.



Review and Reflection from students