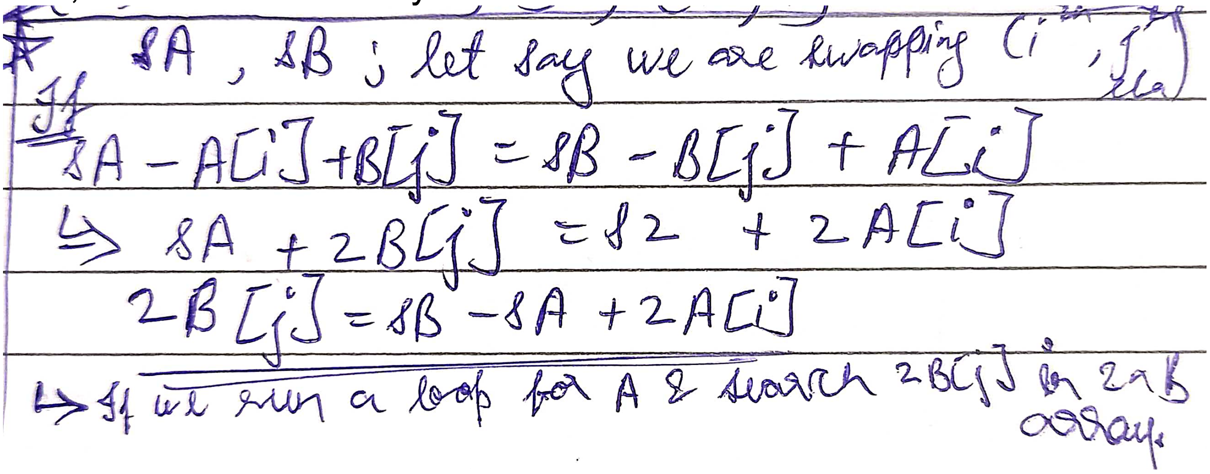
```
// Swapping pairs make sum equal

// Given two arrays of integers A[] and B[] of size n and m, the task is to check if a pair of values (one value from each array) exists such that swapping the elements of the pair will make the sum of two arrays equal.

// Expected Time Complexity: O(mlogm+nlogn).
// Expected Auxiliary Space: O(1).
```

Idea: sA and sB are sum of all elements in array A[] and array B[] respectively. Here, if we are considering ith element of A and jth element of B, then it should satisfy a condition…



sA , sB ; let say we are swapping (i^{th}, j^{th} ele)

If

$sA - A[i] + B[j] = sB - B[j] + A[i]$

$\Rightarrow sA + 2B[j] = s2 + 2A[i]$

$2B[j] = sB - sA + 2A[i]$

⇒ If we run a loop for A & search 2B[j] in 2*B array.

If we make i constant for a moment, we find that for a unique element in A, there exists a unique value in 2*B[] which will satisfy our condition. We can make all elements in B[] twice and then, while traversing A[] binary search for the element **sB – sA + 2A[i]** in **2*B[]**.

Note: we will **not find (sB – sA + 2A[i]) / 2 in B[]** array because the division of integers may cause some inaccuracies in the result.

```
int findSwapValues(int A[], int n, int B[], int m) // Optimized: O(mlogm+nlogn)
{
    // sort it so that we can use binary search
    sort(B, B + m); // nlog(n)
    int sA = 0;
    int sB = 0;
    // sum of elements in A
        -----
    // sum of elements in B and multiplying its each element by 2.
```

```
       ----

    for (int i = 0; i < n; i++)
    {
        if (binary_search(B, B + m, sB - sA + 2 * A[i]))
            return 1;
    }

    return -1;
}
```