TWO POINTERS

What Are Two Pointers?

- Two pointers are variables or indices that point to positions in an array or a data structure.
- Typically, you have two pointers, often called 'left' and 'right,' that can move towards each other or in opposite directions.
- The two pointers algorithm is typically used for finding solutions in pairs using “two pointers”.
- Each variable moves in a single direction, and they only traverse the array a single time.

Why Use Two Pointers?

- Two pointers are often used to optimize time complexity by avoiding nested loops.
- They are especially helpful in solving problems involving searching, sorting, or finding subarrays/sequences.

Time Complexity

- Therefore, time complexity will be O(N+M) where N and M are the size of array’s A and B.
- Sometimes, two pointers is able to replace binary search, reducing the time complexity, and making the implementation easier.

Basic Operations

- Two common operations with two pointers:
  - Initialization: Set the initial positions of the two pointers.
  - Movement: Move the pointers based on the problem's requirements (e.g., incrementing/decrementing).

Merging two sorted arrays

- We are given two arrays, sorted in ascending order, a and b. We want to combine the elements of these arrays into one big array c, also sorted in ascending order.
- The easiest way to do this is with the following algorithm:
  - collect all the elements into one big array;
  - sort it with any sorting built into your language.
- Such an algorithm will take time O(n·log(n)).

Number of smaller

- Let's apply the same technique to solve a different problem.
- We have two arrays a and b. We want to calculate for each element bj how many such i exist that ai<bj.
- How to solve it? First, let's sort both arrays (if initially they were given unsorted). Now you can use binary search to answer the problem (we covered it in another chapter). For each element of b, you need to find the prefix of array a that is less than this element.

Find Pairs with sum = X

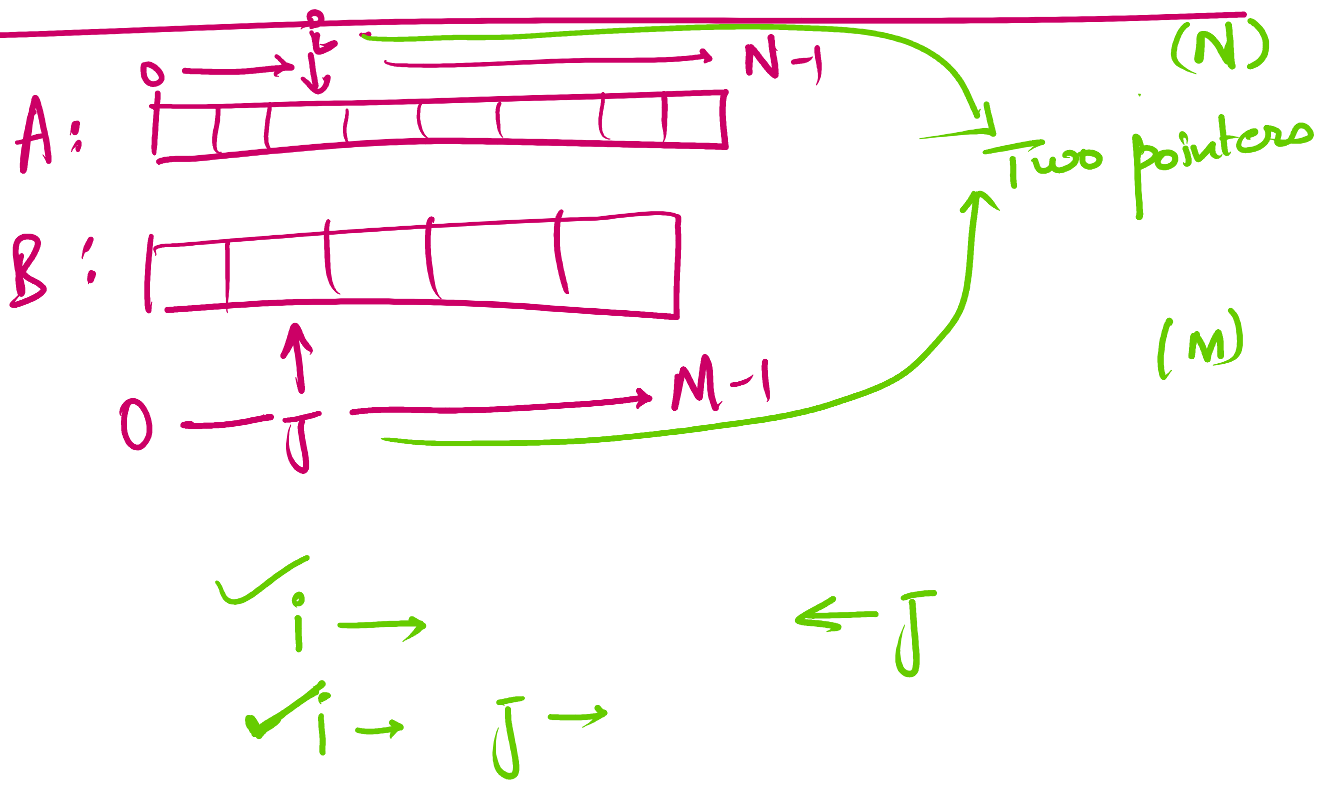
- Given a sorted array A having N unique values find the number of pairs (I, j) such that
  - i < j, and
  - A[i] + A[j] = x
- Brute-force Solution ?
- 2 pointer Solution ?

THANK YOU !

Two Pointers

- Theory
  - why
  - How
  - Time complexity
  - Operations
  - Classicals Problems

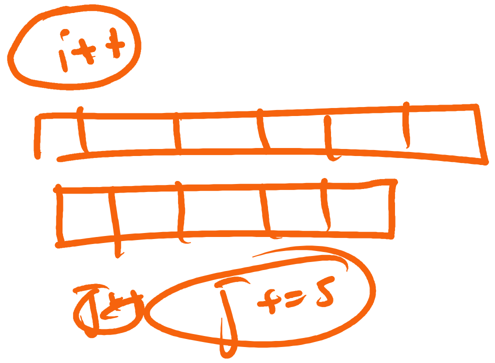
- Solve Problems



Operations :-

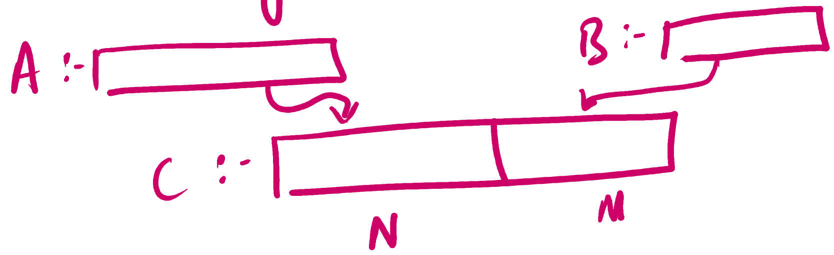
① Initialization

② Movement



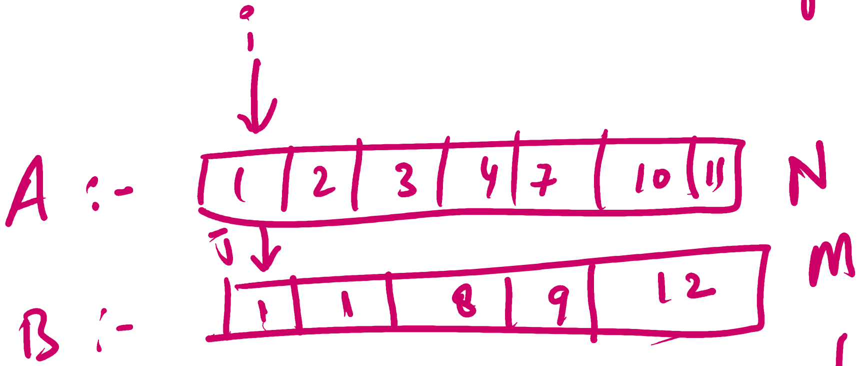
Merging

Using BS



N+m

(N+M) log(N+M)



C :-

$A[i] < B[j]$

↓ include in C

i++

$A[i] > B[j]$

↓ include in C

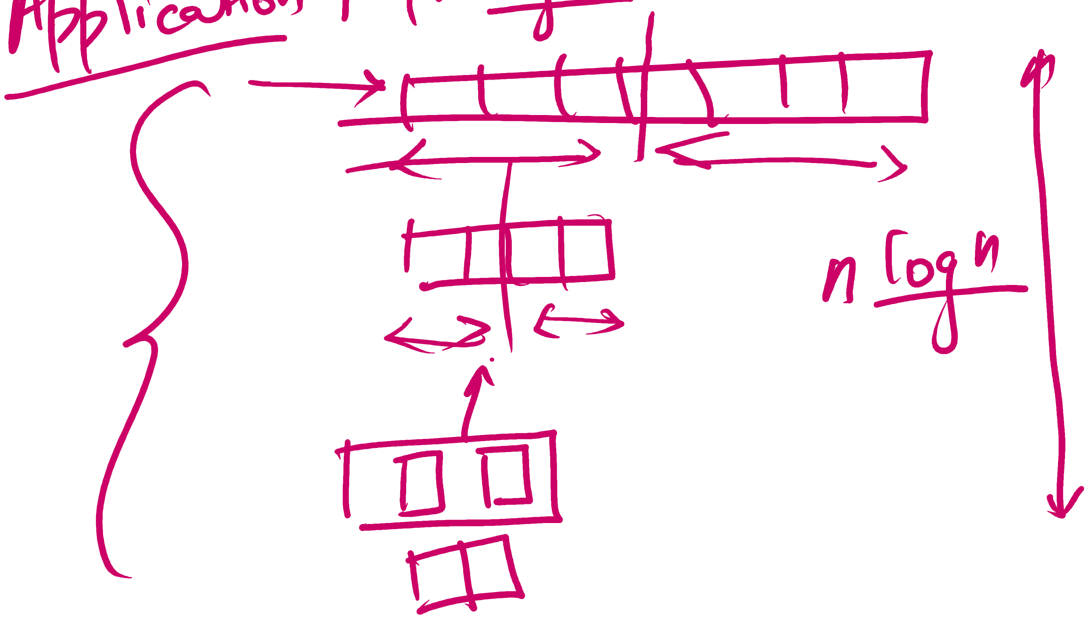
J++

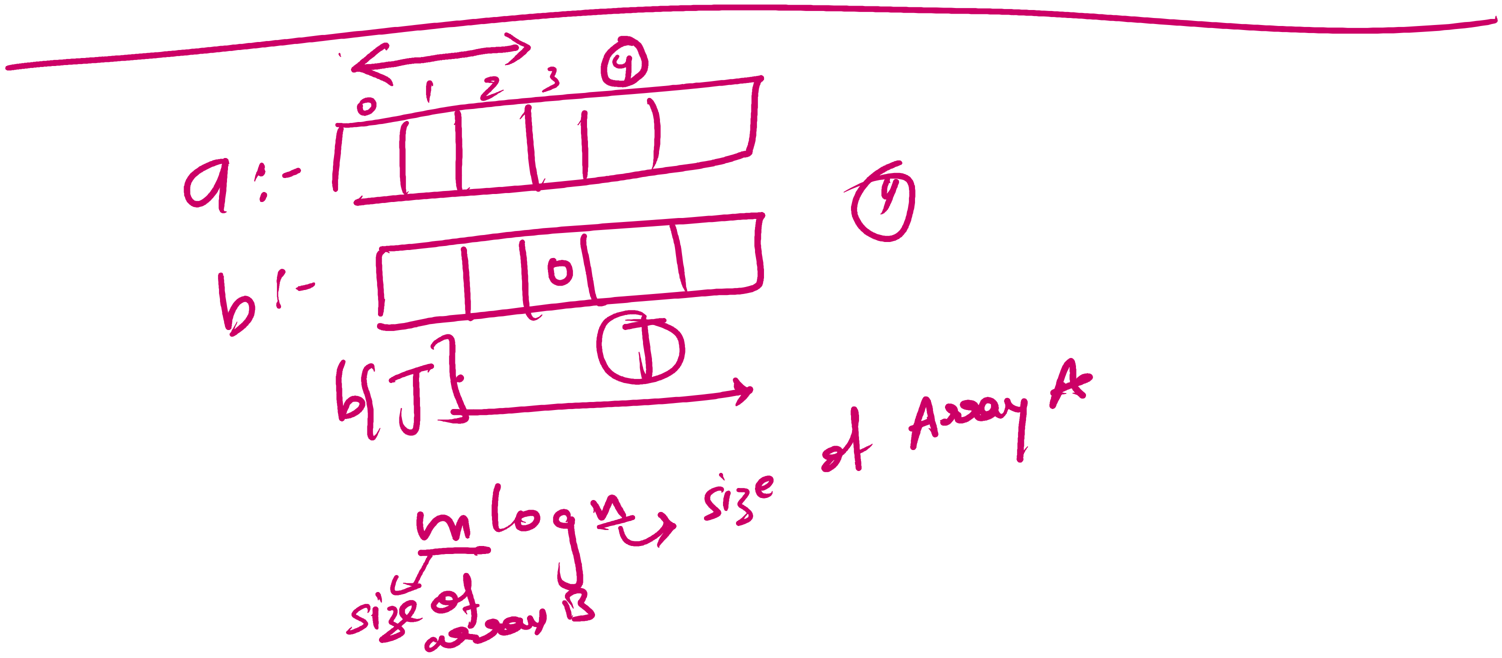
include in | in C
i++ | J++

```
while (i<N && J<M)
{   if (A[i] < B[J]
    {       C ← A[i]
            i++;

    }
    else
    {
            C ← B[J]
            J++;

    }
}
while (i<N)
{   C ← A[i];
    i++ ;
}
while (J<M)
{   C ← B[J] ;
    J++;
}
```

Abblication I (n $\underline{\log n}$)

Application 1 (n log n)





```
for( int J=0, i=0; J<M; J++)
{
    while(i<N && A[i] < B[J])
        i++
    cout << i << endl;
```

↑ 3

Sum problem:-

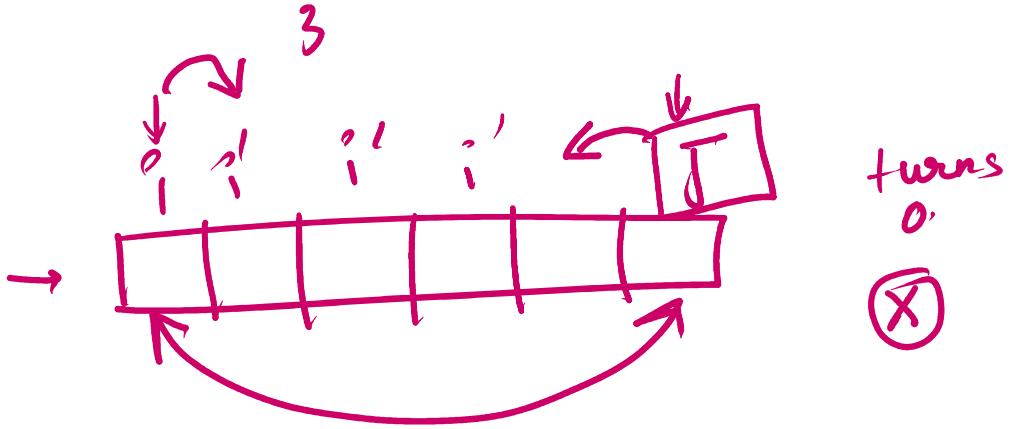
Given an array with N unique values find how many pair exist whose sum = X

(i, j) $(i < j)$

$i == j$
$i > j$

Brute-force:-

```
for(i → 0 — N-1)
{   for(j → i+1 — N-1)
        if(a[i]+a[j]==x)
            ans++
```



① $a[i] + a[j] > x$ ~~X~~
$a[i'] + a[j] == x$

$i' \geq i$

② ... $a[j] < x$

$i' \geq i$

$i' < j$ ① $\rightarrow a[i] + a[j] < x$

$a[i] + a[j'] == x$

$j' < j$
$j' > i$

③ $a[i] + a[j] == x$
{ ans++
i++;
j--;

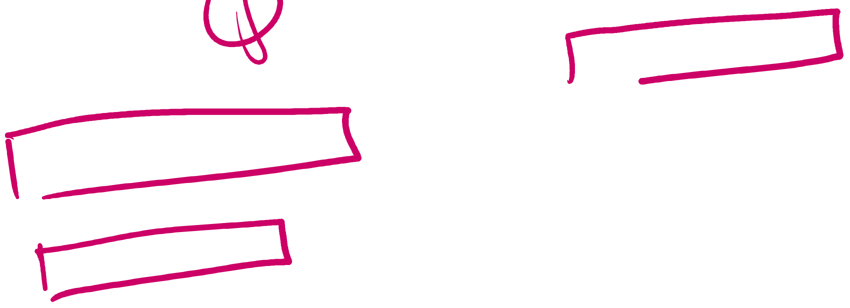
3
~~while (i < j)~~

$i \rightarrow$
$j \rightarrow$

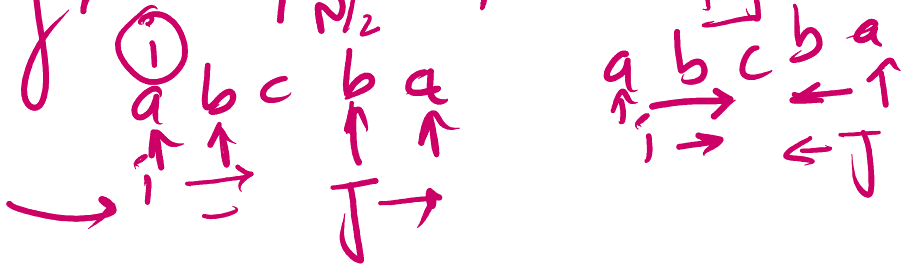
$i \rightarrow \qquad \leftarrow j$

ⓘ $\rightarrow$

ⓙ $\rightarrow$



String/Array is palindrome?



$\rightarrow$ [illegible] $\rightarrow$ [illegible] $\rightarrow$ [illegible]