Professional Development Skills Session-2

--Patterns
--2 pointer Approach
--Practice programs

- Logic can be developed by doing programs on different “patterns”

```
class Main {
  static void pattern1(int N)
{
   // This is the outer loop which will loop for the rows.
   for (int i = 0; i < N; i++)
   {
      // This is the inner loop which here, loops for the columns
      // as we have to print a rectangular pattern.
      for (int j = 0; j < N; j++)
      {
         System.out.print("* ");
      }

      // As soon as N stars are printed, we move to the
      // next row and give a line break otherwise all stars
      // would get printed in 1 line.
      System.out.println();
   }
}

   public static void main(String[] args) {

      // Here, we have taken the value of N as 5.
      // We can also take input from the user.
      int N = 5;
      pattern1(N);
   }
}
```

* ** *** **** *****	class Main { static void pattern2(int N){ for (int i = 0; i < N; i++){ for (int j = 0; j <= i; j++){ System.out.print("* "); } System.out.println(); } } public static void main(String[] args) { int N = 5; pattern2(N); } }
1 12 123 1234 12345	class Main { static void pattern3(int N){ for (int i = 1; i < N+1; i++){ for (int j = 1; j <= i; j++){ System.out.print(j); } System.out.println(); } } public static void main(String[] args) { int N = 5; pattern2(N); } }
1 22 333 4444 55555	class Main { static void pattern4(int N){ for (int i = 1; i < N+1; i++){ for (int j = 1; j <= i; j++){ System.out.print(i); } System.out.println(); } } public static void main(String[] args) { int N = 5; pattern2(N); } }

`` ***** **** *** ** * ```	``` class Main { static void pattern5(int N){ for (int i = 0; i < N; i++){ for (int j = N; j > i; j--){ System.out.print("* "); } System.out.println(); } } public static void main(String[] args) { int N = 5; pattern2(N); } } ```
`` 12345 1234 123 12 1 ```	``` class Main { static void pattern6(int N){ for (int i = 0; i < N; i++){ for (int j = N; j > i; j--){ System.out.print((N+1)-j); } System.out.println(); } } public static void main(String[] args) { int N = 5; pattern2(N); } } ```
`` * *** ***** ******* ********* ```	In the first row (i=0) there are 4 spaces, 1 star, then again 4 spaces. In the second row (i=1) there are 3 spaces, 3 stars, then again 3 spaces **so there are N-i-1 spaces, 2*i+1 stars, and then again N-i-1 spaces for each row where i is the row index.** We thus simply run 3 inner loops first for printing the spaces, then the stars, and then the spaces again. ``` class Main { static void pattern7(int N){ for (int i = 0; i < N; i++) { for (int j =0; j<N-i-1; j++){ System.out.print(" "); } for(int j=0;j< 2*i+1;j++){ System.out.print("*"); } for (int j =0; j<N-i-1; j++){ System.out.print(" "); ```

	} System.out.println(); } } public static void main(String[] args) { int N = 5; pattern7(N); } }
********* ******* ***** *** *	In the first row (i=0) there are 0 spaces, 9 stars, then again 0 spaces. In the second row (i=1) there is 1 space, 7 stars, then again 1 space **so there are i spaces, 2*N - (2*i+1) stars, and then again i space for each row where i is the row index.** We thus simply run 3 inner loops, first for printing the spaces, then the stars, and then the spaces again.

Two Pointers Technique

Two pointers is really an easy and effective technique which is typically used for searching pairs in a sorted array.

Given a sorted array A (sorted in ascending order), having N integers, find if there exists any pair of elements (A[i], A[j]) such that their sum is equal to X.

We take two pointers, one representing the first element and other representing the last element of the array, and then we add the values kept at both the pointers. If their sum is smaller than X then we shift the left pointer to right or if their sum is greater than X then we shift the right pointer to left, in order to get closer to the sum. We keep moving the pointers until we get the sum as X.

The idea of this technique is to begin with two corners of the given array. We use two index variables left and right to traverse from both corners.

Initialize: left = 0, right = n – 1

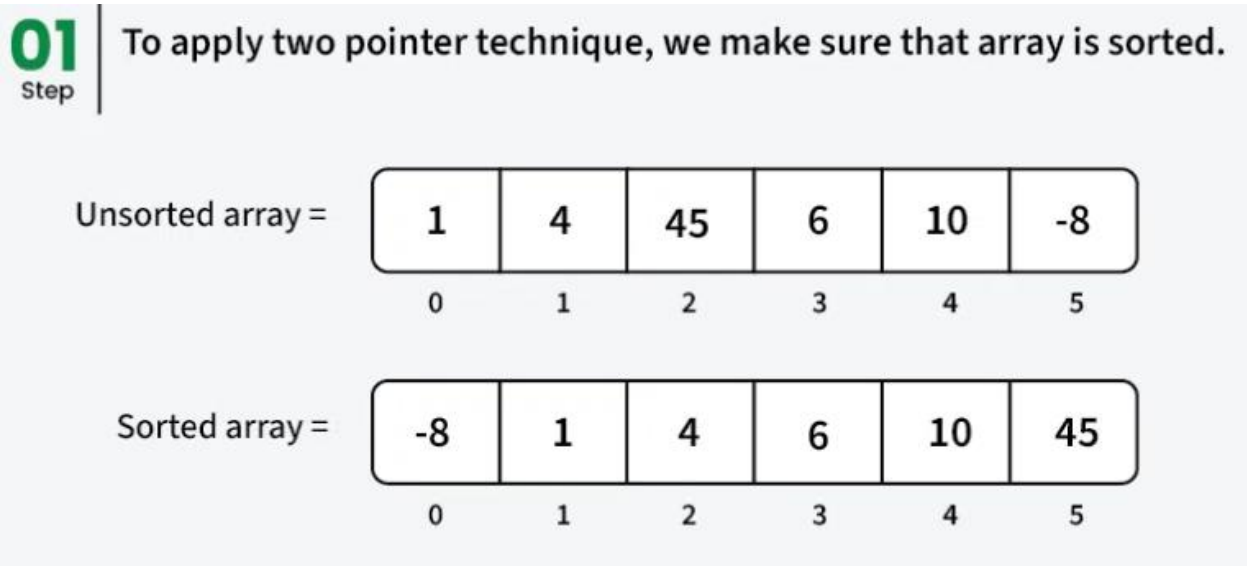
Run a loop while left < right, do the following inside the loop

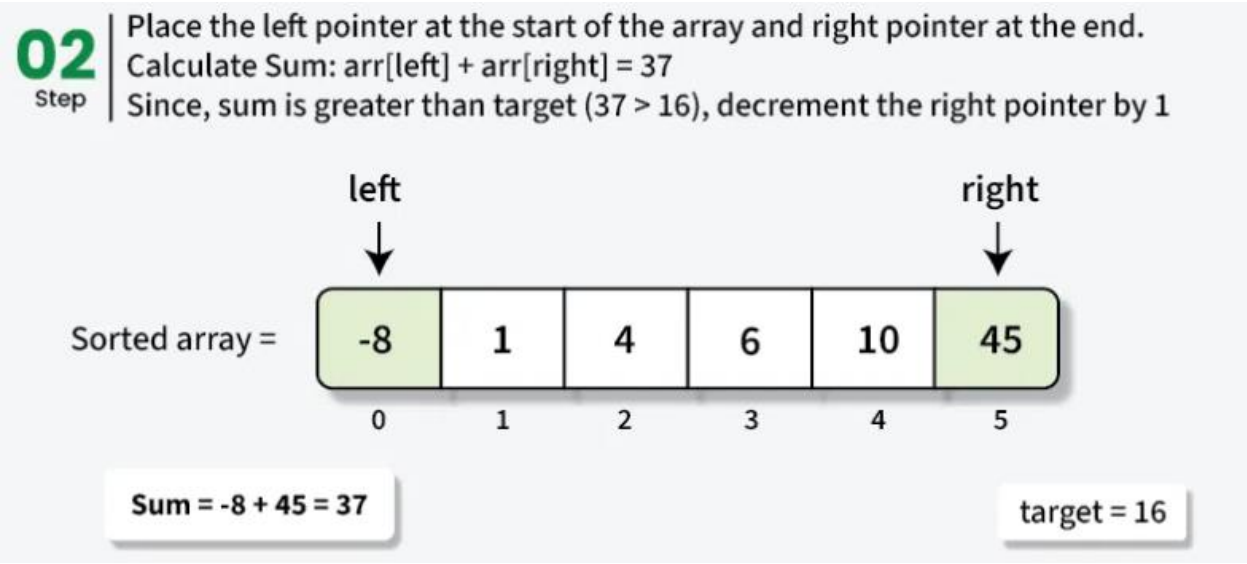
Compute current sum, sum = arr[left] + arr[right]

If the sum equals the target, we've found the pair.

If the sum is less than the target, move the left pointer to the right to increase the sum.

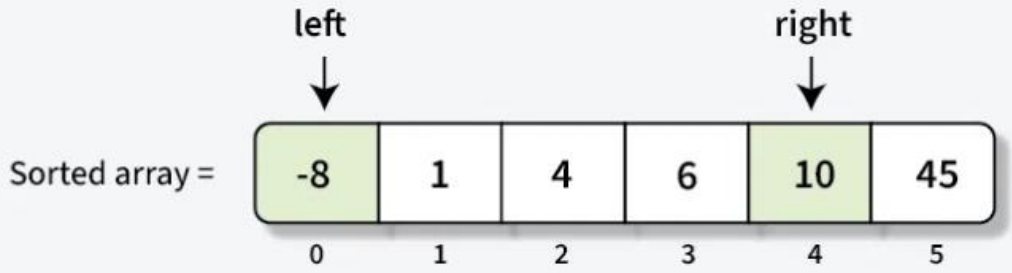
If the sum is greater than the target, move the right pointer to the left to decrease the sum.





03 Step

Calculate Sum: arr[left] + arr[right] = 2
Since sum is smaller than target (2 < 16), increment left pointer by 1

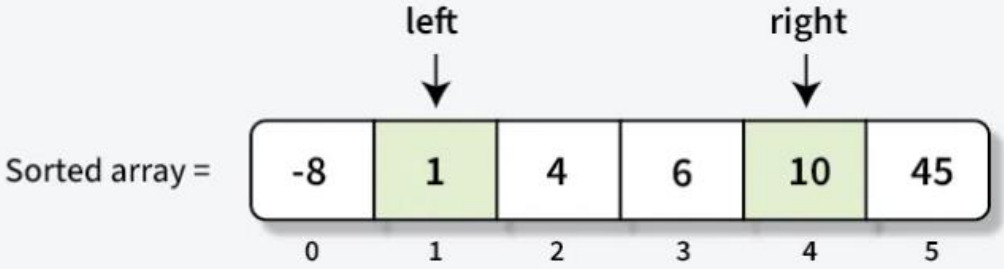


Sum = -8 + 10 = 2

target = 16

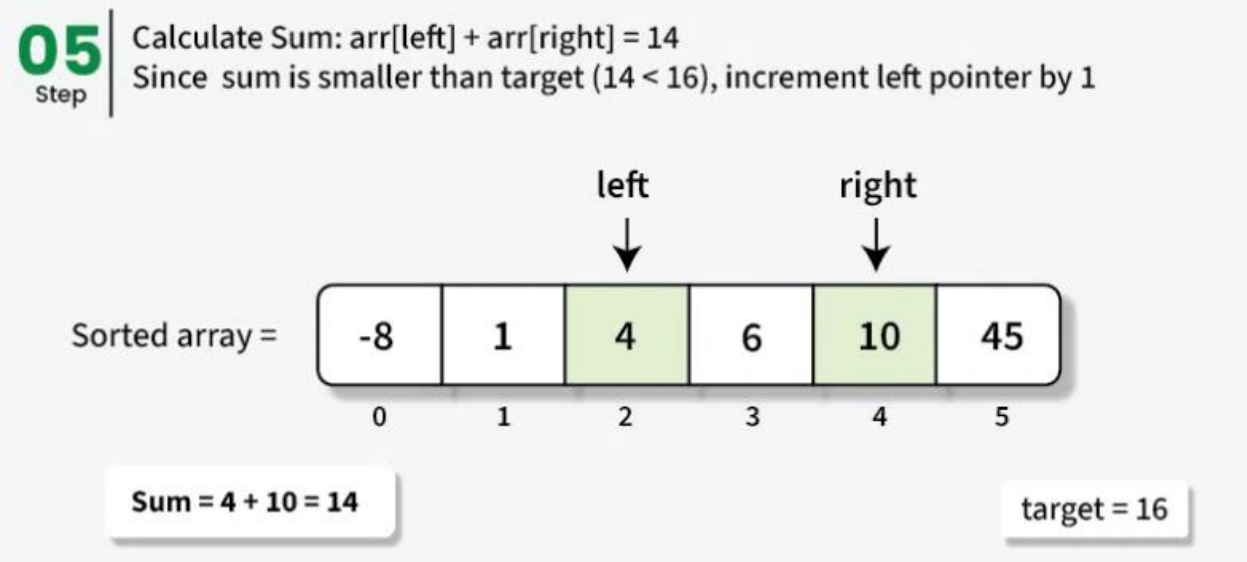
04 Step

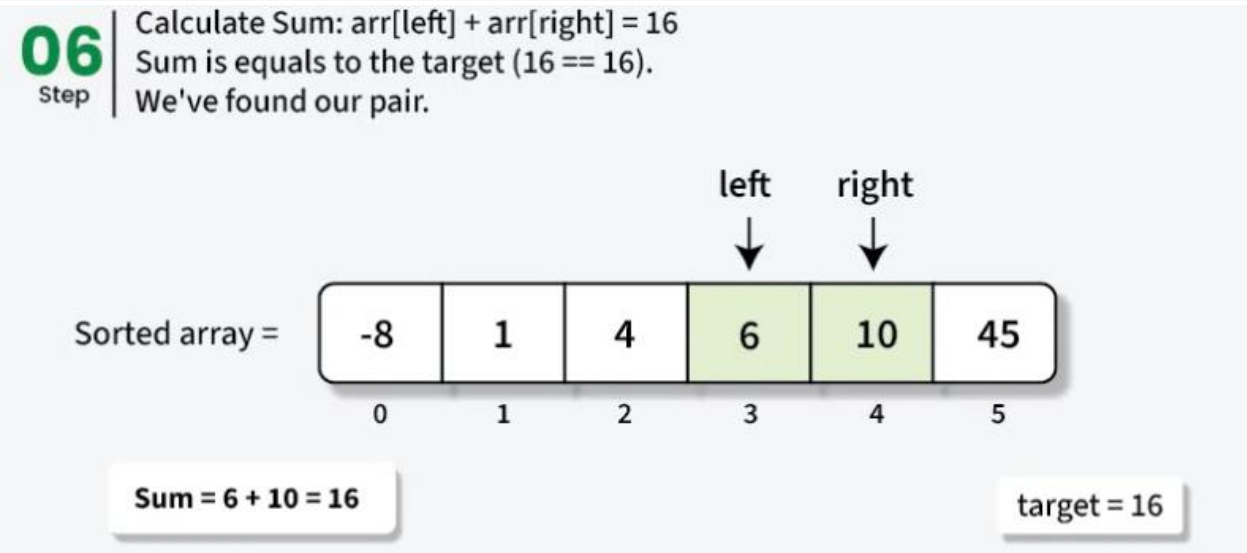
Calculate Sum: arr[left] + arr[right] = 11
Since sum is smaller than target (11 < 16), increment left pointer by 1



Sum = 1 + 10 = 11

target = 16





Navie method	**2 Pointer method**
class Demo { static boolean twoSum(int[] arr, int target){ int n = arr.length; for (int i = 0; i < n; i++) { for (int j = i + 1; j < n; j++) { if (arr[i] + arr[j] == target) { return true; } } } return false; }	import java.util.Arrays; class GfG { static boolean twoSum(int[] arr, int target){ Arrays.sort(arr); int left = 0, right = arr.length - 1; while (left < right) { int sum = arr[left] + arr[right];

```
  public static void main(String[] args){
    int[] arr = { 0, -1, 2, -3, 1 };
    int target = -2;
    if (twoSum(arr, target))
      System.out.println("true");
    else
      System.out.println("false");
  }
}
```

Complexity:- O(n^2) Time and O(1) Space

```
    if (sum == target)
      return true;
    else if (sum < target)
      left++; // Move left pointer to the right
    else
      right--; // Move right pointer to the left
  }
  return false;
 }

 public static void main(String[] args){
  int[] arr = { 0, -1, 2, -3, 1 };
  int target = -2;

  if (twoSum(arr, target)) {
    System.out.println("true");
  }
  else {
    System.out.println("false");
  }
 }
}
```

Complexity:- O(n) time and O(1) space

Practice Programs

```
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

```
A B C D E F G
A B C D E F
A B C D E
A B C D
A B C
A B
A
A B
A B C
A B C D
A B C D E
A B C D E F
A B C D E F G
```

```
A
A B
A B C
A B C D
A B C D E
A B C D E F
A B C D E F G
A B C D E F
A B C D E
A B C D
A B C
A B
A
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

```
A
A B
A B C
A B C D
A B C D E
A B C D E F
A B C D E F G
A B C D E F
A B C D E
A B C D
A B C
A B
A
```

```
A
A B
A B C
A B C D
A B C D E
A B C D E F
A B C D E F G
A B C D E F
A B C D E
A B C D
A B C
A B
A
```