

Reach Target

$$n=6$$

$$arr = [-1, 1, 2, 3, 4, 5]$$

$$k=4.$$

$$\begin{bmatrix} -1, 5 \\ 1, 3 \end{bmatrix} \rightarrow \text{output}$$

$$\begin{bmatrix} -1, 1 & 1, 2 & 2, 3 & 3, 4 & 4, 5 \\ -1, 2 & 1, 3 & 2, 4 & 3, 5 \\ -1, 3 & 1, 4 & 2, 5 \\ -1, 4 & 1, 5 \\ -1, 5 \end{bmatrix}$$

↳ Using this way

Time Complexity $\rightarrow O(n^2)$

We can do this in $O(n)$ using two pointer approach.

$$arr = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ [-1, 1, 2, 3, 4, 5] \end{matrix}$$

Algorithm

1. $left = 0$

1. $left = 0$
 $right = n - 1$.
2. $arr[left] + arr[right] == target$
print(left & right) which is the answer
 $left++$;
 $right--$;
3. Again, $arr[left] + arr[right] > target$
 $right--$;
4. $arr[left] + arr[right] < target$
 $left++$;

Code

```
int left = 0;
int right = n - 1;
while (left < right) {
    if (arr[left] + arr[right] == k) {
        S.o.Pln(left + " " + right);
        left++;
        right--;
    } else if (arr[left] + arr[right] > k) {

```

```
} else if (arr[left] + arr[right] > 1)
    right--;
```

```
} else {
    left++;
}
```

```
}
```

Target Sum

arr = [3, 3, 5, 5]

k = 8

Using previous Approach
it will print

3 5

3 5

But, in this we should get this as

3 5

Now, we have to skip duplicate values
in left and right side.

Example 2:

arr = [3, 3, 3, 3, 5, 5, 5, 5]

Output → 3, 5

Code: Arrays.sort(arr);

while (left < right) {

if (arr[left] + arr[right] == k) {

S.o.pln (arr[left] + " " + arr[right]);

while (arr[left] == arr[left+1] & left < right) {

left++;

} while (arr[right] == arr[right-1] & left < right) {

right--;

} left++;

```

    }
    left++;
    right--;
} else if (arr[left] + arr[right] > k) {
    right--;
}
else {
    left++;
}
}

```

Or.

```

while (left < right) {
    if (arr[left] == arr[left+1]) {
        left++;
    } else if (arr[right] == arr[right-1]) {
        right--;
    } else if (arr[left] + arr[right] == k) {
        S.O.P(arr[left] + " " + arr[right]);
        left++; right--;
    } else if (arr[left] + arr[right] < k) {
        left++;
    } else {
        right--;
    }
}
}

```

```

4 public class Solution {
5
6     public static void main(String[] args) {
7         /* Enter your code here. Read input from STDIN. Print output to
8         Scanner sc = new Scanner(System.in);
9         int n = sc.nextInt();
10        int arr[] = new int[n];
11        for(int i=0;i<n;i++){
12            arr[i] = sc.nextInt();
13        }
14        int k = sc.nextInt();
15        Arrays.sort(arr);
16        int left=0;
17        int right=n-1;
18        while(left<right){
19            if(arr[left]+arr[right]==k){
20                System.out.println(arr[left]+" "+arr[right]);
21                while(arr[left]==arr[left+1]&& left<right){
22                    left++;
23                }
24                while(arr[right]==arr[right-1] && left<right){
25                    right--;
26                }
27                left++;
28                right--;
29            }else if (arr[left]+arr[right]<k){
30                left++;
31            }else{
32                right--;
33            }
34        }
35    }
36 }

```

3 Sum

$n = 6$

$arr = [-2, 0, 2, 4, -2, -8]$

Output

$-2, 0, 2$

$-2, 4, -2$

Brute Force.

$-2 \ 0 \ 2$

$-2 \ 0 \ 4$

$-2 \ 0 \ -2$

$-2 \ 0 \ -8$

$-2 \ 2 \ 4$

$-2 \ 2 \ -2$

$-2 \ 2 \ 8$ and so on.

Time Complexity $\rightarrow O(n^3)$

We can reduce T.C. by two pointer approach.

1. Sort the array $\rightarrow \text{Arrays.sort}(arr)$

2. $[-8, -2, -2, 0, 2, 4]$

$i=0$ } 1st value = -8
2nd & 3rd value from $[-2, -2, 0, 2, 4]$
total sum = 0
Sum of remaining two = $0 - (-8)$
= 8

$i=1$ | 1st value = -2
Sum of rest two values = $0 - (-2)$
= 2
 $[-2, 0, 2, 4]$

$-2, 0, 2$ } \rightarrow (2) \rightarrow We can use target sum approach.

$i=2$ | 1st value = -2
Sum of remaining values = $0 - (-2)$
= 2
 $[0, 2, 4]$
 $-2, 0, 2$

Code.

Arrays.sort(arr);

```
for (int i = 0; i < n - 2; i++) {  
    if (arr[i + 1] == arr[i])  
        continue;
```

```
    int left = i + 1;
```

```
    int right = n - 1;
```

```
    while (left < right) {
```

```
        if (arr[i] + arr[left] + arr[right] == k) {
```

```
            S.o.pln(arr[i] + " " + arr[left] + " " + arr[right]);
```

```
            while (arr[left] == arr[left + 1] && left < right)  
                left++;
```

```
            while (arr[right] == arr[right - 1] && left < right)  
                right--;
```

```
        } else if (arr[i] + arr[left] + arr[right] < k) {  
            left++;
```

```
        } else {  
            right--;
```

```
        }
```

}

```

7  /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class
8  Scanner sc = new Scanner(System.in);
9  int n = sc.nextInt();
10 int arr[] = new int[n];
11 for(int i=0;i<n;i++){
12     arr[i] = sc.nextInt();
13 }
14 Arrays.sort(arr);
15 for(int i=0;i<n-2;i++){
16     if(i>0){
17         if(arr[i-1]==arr[i]){
18             continue;
19         }
20     }
21     int left = i+1;
22     int right = n-1;
23     while(left<right){
24         if(arr[i]+arr[left]+arr[right]==0){
25             System.out.println(arr[i]+" "+arr[left]+" "+arr[right]);
26             while(arr[left]==arr[left+1]&& left<right){
27                 left++;
28             }
29             while(arr[right]==arr[right-1] && left< right){
30                 right--;
31             }
32             left++;
33             right--;
34         }else if(arr[i]+arr[left]+arr[right]<0){
35             left++;
36         }else{
37             right--;
38         }
39     }

```