Reach Target

n = 6

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

K = 4.

[-1,5]
[1,3] → output

[-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]

↳ using this way
Time Complexity → $O(n^2)$

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

0 1 2 3 4 5
arr = [-1, 1, 2, 3, 4, 5]

Algorithm

1. left = 0

1. left = 0
   right = n-1.

2. arr[left] + a[right] == target
   print(left & right) which is the answer

   left++;
   right--;

3. Again, arr[left] + a[right] > target
   right--;

4. arr[left] + a[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] ...
        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--;
 }
}
```


```java
public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
        int k = sc.nextInt();
        Arrays.sort(arr);
        int left=0;
        int right=n-1;
        while(left<right){
            if(arr[left]+arr[right]==k){
                System.out.println(arr[left]+" "+arr[right]);
                while(arr[left]==arr[left+1]&& left<right){
                    left++;
                }
                while(arr[right]==arr[right-1] && left<right){
                    right--;
                }
                left++;
                right--;
            }else if (arr[left]+arr[right]<k){
                left++;
            }else{
                right--;
            }
        }
    }
}
```

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 -> $O(n^3)$

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

1. Sort the array -> 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) → We can use target sum approach.
-2, 0, 2.

j=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--;
        }
```

```
/* Enter your code here. Read
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int arr[] = new int[n];
for(int i=0;i<n;i++){
    arr[i] = sc.nextInt();
}
Arrays.sort(arr);
for(int i=0;i<n-2;i++){
    if(i>0){
    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]==0){
            System.out.println(arr[i]+" "+arr[left]+" "+arr[right]);
            while(arr[left]==arr[left+1]&& left<right){
                left++;
            }
            while(arr[right]==arr[right-1] && left< right){
                right--;
            }
            left++;
            right--;
        }else if(arr[i]+arr[left]+arr[right]<0){
            left++;
        }else{
            right--;
        }
    }
```