Triplet Sum in Array

QUES STATEMENT :

Given an array arr of size N and an integer K. Find if there's a triplet in the array which sums up to the given integer K.

SAMPLE TEST CASE :

```
N = 6, K = 13
arr[] = [1 4 45 6 10 8]
```

OUTPUT : TRUE ({1,4,8})

Expected Time Complexity: $O(N^2)$
Expected Auxiliary Space: O(1)

APPROACH :

1): NAIVE APPROACH :

Looping through array and check
If(arr[i] + arr[j] + arr[k] == sum)
Return true

THIS METHOD WILL REQUIRE THREE NESTED LOOPS

Expected Time Complexity: $O(N^3)$

Expected Auxiliary Space: $O(1)$

2) USING SET

For finding a triplet , we will find a pair of elements from remaining array for every single element of array , We will loop through array once and for each arr[i] , will find another 2 elements for this triplet

CODE :

```
bool find(int arr[],int n, int x,int curr){
   set<int> s;
   // int i =0;
   for(int i =0;i<n;i++){
      if(i!= curr){
         s.insert(arr[i]);
      }
   }

   // finding in set
   for(int i = 0;i<n;i++){
      if(i!= curr){
         bool isin = (s.find(x- arr[i])!=s.end())&&(s.find(x- arr[i])!= s.find(arr[i]));
         if(isin){

            return true ;
         }
      }
   }
   return false;
}
```

```
bool find3Numbers(int arr[], int N, int X)
{
   //Your Code Here
   int flag = 0 ;
   for(int i = 0;i<N;i++){
      flag = find(arr,N,X-arr[i],i);
      if(flag == 1){
         return true;
      }
   }
   return false;
```

Expected Time Complexity: $O(N^2)$
Expected Auxiliary Space: $O(N)$

```
}
```

3) By Sorting the array

Similarly for finding a triplet , we will first find a pair for each element in array
We will sort array and then trasverse it from left and right checking if(arr[left] + arr[right] == sum)

This will reduce the need for using an extra space !!

CODE :

```
bool find(int arr[],int l,int r, int x){
   while(l<r){
      if(arr[l] + arr[r] == x ){
```

```
            return true;
        }
        else if(arr[l] + arr[r] <x){
            l++;
        }
        else{
            r--;
        }

    }
     return false ;
}
bool find3Numbers(int arr[], int N, int X)
{
    //Your Code Here
    sort(arr,arr+N);

    for(int i = 0;i<N;i++){
        if(find(arr,i+1,N-1,X-arr[i])){
            return true;
        }
    }
    return false;
}
```

Expected Time Complexity: $O(N^2)$
Expected Auxiliary Space: $O(1)$