2 SUM
METHOD 0 USING 2 NESTED loops
METHOD 1 USING 2 POINTER
METHOD 2 USING MAP

```java
public static int[] twoSum(int[] A, int target) {
    // Your code here
    int i = 0;
    int j = A.length - 1;

    while (i < j) {
        int sum = A[i] + A[j];
        if (sum == target) {
            int[] ans = new int[2];
            ans[0] = i + 1;
            ans[1] = j + 1;
            return ans ;

        } else if (sum < target) {
            i++;
        } else {
            j--;
        }
    }
    return new int[]{-1, -1};
}
```

Method 1

```java
public int[] twoSum(int[] nums, int T) {

    Map<Integer,Integer>m = new HashMap<>();

    for(int i = 0 ; i< nums.length ; i++){
        int find = T - nums[i] ;

        if(m.containsKey(find)){
            return new int[]{m.get(find),i};
        }

        m.put(nums[i],i);
    }
    return new int[]{} ;
}
```

mehtod 2

3 sum problem

```java
public List<List<Integer>> threeSum(int[] nums) {
    List<List<Integer>>ans = new ArrayList<>() ;
    Arrays.sort(nums);

    for(int i = 0 ; i<nums.length ; i++){
        if(i>0 && nums[i] == nums[i-1]) continue ;

        int j = i+1 ;
        int k = nums.length - 1 ;
        while(j<k){
            int sum = nums[i] + nums[j] + nums[k];
            if(sum > 0) k--;
            else if(sum < 0) j++ ;
            else{
                // sum == 0
                // List<Integer>temp = Arrays.asList(nums[i],nums[j],nums[k]);
                List<Integer>temp = new ArrayList<>();
                temp.add(nums[i]);
                temp.add(nums[j]);
                temp.add(nums[k]);
                ans.add(temp);
                j++;
                k--;
                while(j<k && nums[j] == nums[j-1]) j++;
                while(j<k && nums[k]== nums[k+1]) k--;
            }
        }
    }
    return ans ;
}
```


4 sum problem

```java
public List<List<Integer>> fourSum(int[] nums, int target) {
    int n = nums.length; // size of the array
    List<List<Integer>> ans = new ArrayList<>();
    Arrays.sort(nums);

    for (int i = 0; i < n; i++) {
        if (i > 0 && nums[i] == nums[i - 1]) continue;

        for (int j = i + 1; j < n; j++) {
            if (j > i + 1 && nums[j] == nums[j - 1]) continue;

            int k = j + 1;
            int l = n - 1;
            while (k < l) {
                long sum = nums[i];
                sum += nums[j];
                sum += nums[k];
                sum += nums[l];
                if (sum == target) {
                    List<Integer> temp = new ArrayList<>();
                    temp.add(nums[i]);
                    temp.add(nums[j]);
                    temp.add(nums[k]);
                    temp.add(nums[l]);
                    ans.add(temp);
                    k++;
                    l--;

                    while (k < l && nums[k] == nums[k - 1]) k++;
                    while (k < l && nums[l] == nums[l + 1]) l--;
                }
                else if (sum < target) k++;
                else l--;
            }
        }
    }
    return ans;
}
```