## Assignment 2

**Question:**

**What is time complexity and space complexity?**

In my understanding:

time complexity means how many times would an algorithm run based on the size of input;

space complexity is the total spaces or memories that one algorithm used based on the size of input, normally it includes space used by input and extra or temporary spaces used by an algorithm.

**What is the time complexity and space complexity of the code you wrote (question 3) ?**

**Problem 1**

Assume the space that variables/allocation/function call/data type take k spaces

```
	public static void main(String[] args) {              //s1(n)=k+nk
		int arr[]= {2,7,11,15};                          //t1(n)=1,s2(n)=nk
		int target=9;                                     //t2(n)=1,s3(n)=k
		System.out.println("Target Number is:"+target+",Array is:");
		printArray(arr);                                 //t3(n)=T3(n),s4(n)=S3(n)
		//int arr2[]=indices(arr,target);                //t4(n)=T2(n)',s5(n)=S2(n)'
		int arr2[]=indices2(arr,target);                 //t4(n)= T2(n),s5(n)= S2(n)
		if (arr2[1]!=arr2[0]){                           //t5(n)=1
			System.out.println("The indices are:");
			printArray(arr2);                            //t6(n)=1
		}else {
			System.out.println("Not find!");
		}

	}
```

For the part above, the time complexity:

$T_1(n)=4+T_3(n)+T_2(n)/T_2(n)'$

And the space complexity:

$S_1(n)=2k(n+1)+S_3(n)+S_2(n)/ S_2(n)'$

```
	private static int[] indices2(int[] arr, int target) {     //s1(n)=k+2nk
		HashMap<Integer,Integer> h=new HashMap();              //s2(n)=nk
		int temp[]=new int[2];                                 //s3(n)=nk
		for(int i=0;i<arr.length;i++) {                       //t1(n)=n,s4(n)=k
			if(/*h.get(target - arr[i]) != null*/h.containsKey(target-
arr[i])){                                                      //t2(n)= n *1
				temp[0]=h.get(target-arr[i]);                  //t3(n)= 1
				temp[1]=i;                                     //t4(n)= 1
				break;
			}else {
				h.put(arr[i],i);                               //t5(n)= n *1
			}
```

```
		}
		return temp;
	}
```

This part the time complexity:

$T_2(n)=3n+2 \approx O(n)$

And the space complexity:

$S_2(n)=k(4n+1) \approx O(n)$

```
	private static int[] indices(int[] arr, int target) {   //s1(n)=k+2nk
		int temp[]=new int[2];                               //s2(n)=nk
		for(int i=0;i<arr.length;i++) {                  //t1(n)=n,s3(n)=k
			for(int j=i+1;j<arr.length;j++) {        //t2(n)=(n)*(n-1),s4(n)=nk
				if(arr[i]+arr[j]==target) {      //t3(n)=n*(n-1)(1)
					temp[0]=i;               //t4(n)=1
					temp[1]=j;               //t5(n)=1
					break;
				}
			}
		}
		return temp;
	}
```

In this part, the time complexity:

$T_2(n)' = 2n^2+2 \approx O(n^2)$

And the space complexity:

$S_2(n)' = 4nk+2k \approx O(n)$

```
	private static void printArray(int[] arr) {        //s1(n)=k+nk
		System.out.print("[");
		for(int i=0;i<arr.length-1;i++) {            //t1(n)=n-1,s2(n)=k
			System.out.print(arr[i]+", ");
		}
		System.out.println(arr[arr.length-1]+"]");
	}
```

In this part, the time complexity is:

$T_3(n)=n-1 \approx O(n)$

And the space complexity is:

$S_3(n)=k(n+2) \approx O(n)$

So, for whole program, the time complexity is:

When using method indicies2

$T(n)=4+O(n)+O(n) \approx O(n)$

When using method indicies

$T(n)=4+O(n)+O(n^2) \approx O(n^2)$

And the space complexity is:

No matter using method indices2 or indices:

$S(n)=2k(n+1)+O(n)+O(n) \approx O(n)$

**Problem 2**

```
public class NumIsands {

	private static int numIsands(int[][] target) {
		int row=target.length;
		int col=target[0].length;
		if(target==null||row==0||col==0) {
			return 0;
		}else {
			int count=0;
			for(int i=0;i<row;i++) {
				for(int j=0;j<col;j++) {
					if(target[i][j]==1) {
						count++;
						union(target,row,col,i,j);
					}
				}
			}
		return count;
		}
	}
```

In this part:

$T_1(n)=2mn+n+5+T_2(n)$

$S_1(n)=k(n+6)+S_2(n)$

```
	private static void union(int[][] target, int row, int col, int i, int j) {
		if(i<0||i>row||j<0||j>col||target[i][j]!=1) {
			return;
		}
		target[i][j]=-1;
		union(target,row,col,i-1,j);
		union(target,row,col,i+1,j);
		union(target,row,col,i,j-1);
		union(target,row,col,i,j+1);
	}
```

In this part:

$T_2(n) \approx O(n)$

$S_2(n) \approx O(n)$

```
	private static void PrintArray(int[][] target) {
		for(int i=0;i<target.length;i++) {
			for(int j=0;j<target[0].length;j++) {
				System.out.print(target[i][j]+" ");
			}
			System.out.println();
		}
	}
```

In this part:

$T_3(n) \approx O(mn)$

$S_3(n) \approx O(n)$

```
public static void main(String[] args) {                          //s₁(n)=k+nk
        int[][] target= {{1,1,1,1,0},{1,1,0,1,0},{1,1,0,0,0},{0,0,0,0,0}};
                                                                    //s₂(n)=k,t₁(n)=1
        PrintArray(target);                                 //s₃(n)=S₃(n),t₂(n)= T₃(n)
        int num=numIsands(target);                  //s₄(n)=k+S₁(n),t₃(n)=1+T₁(n)
        System.out.println("Number of Island:"+num);
    }

}
```

So, the total time complexity and space complexity for this problem would be:

T(n)=2+2*row*col+n+5+O(mn) +O(n) ≈O(mn)

S(n)=k(n+3)+k(n+6)+O(n)+ O(n) ≈O(n)