DAA LAB ASSIGNMENT

Name:- Sanidhya Varshney
Section:- A(53)
Univ.Roll.No.:- 191500714

Question:- Implement and analyze the time complexity of Binary Search and Heap sort.

Binary Search

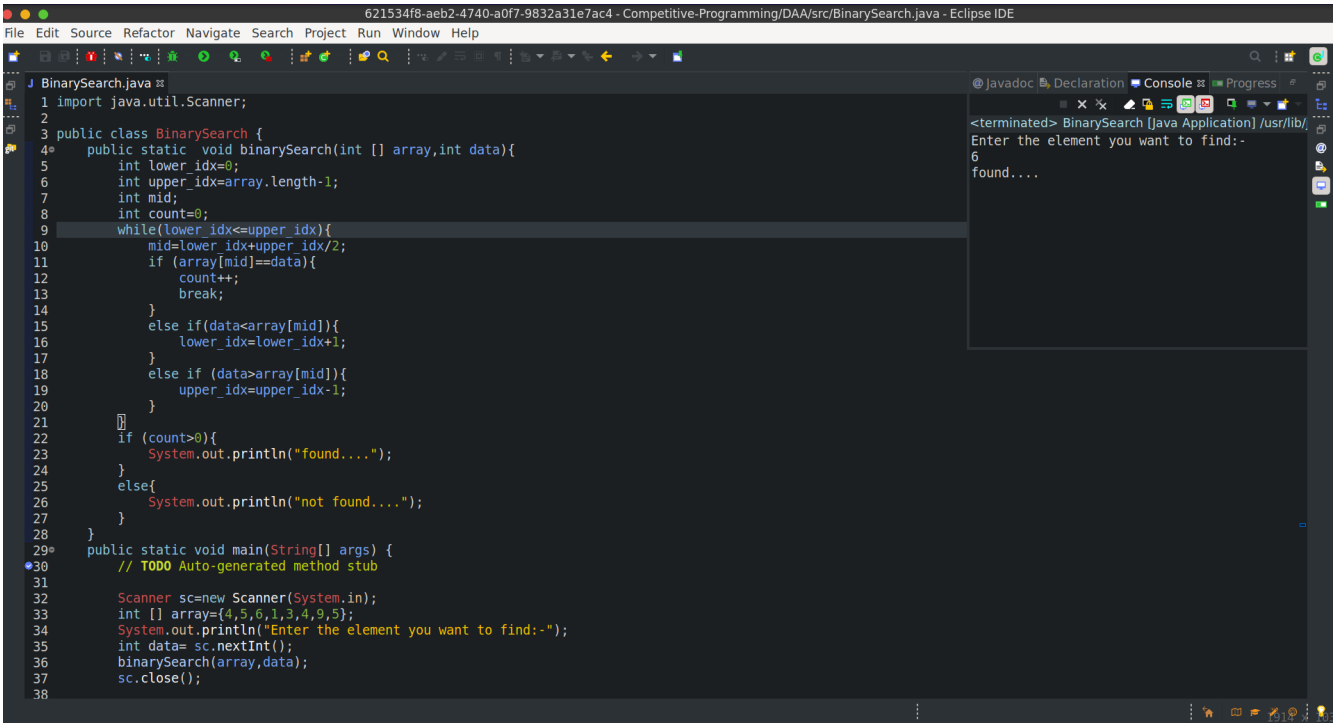
Code:-

```
package DAA.src;

import java.util.Scanner;

public class BinarySearchExample {
    public static  void binarySearch(int [] array,int data){
        int lower_idx=0;
        int upper_idx=array.length-1;
        int mid;
        int count=0;
        while(lower_idx<=upper_idx){
            mid=lower_idx+upper_idx/2;
            if (array[mid]==data){
                count++;
                break;
            }
            else if(data<array[mid]){
                lower_idx=lower_idx+1;
            }
```

```
            else if (data>array[mid]){
                upper_idx=upper_idx-1;
            }
        }
        if (count>0){
            System.out.println("found....");
        }
        else{
            System.out.println("not found");
        }
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int [] array={4,5,6,1,3,4,9,5};
        System.out.println("Enter the element you want to find:-");
        int data= sc.nextInt();
        binarySearch(array,data);
    }
}
```



Time Complexity of binary Search:-

Best case complexity: O(1)

Average case complexity: O(log n)

Worst case complexity: O(log n)

Heap Sort

Code:-

```
public class HeapSort {
	public static void sort(int [] arr) {
		int n=arr.length;
		for (int i = n/2-1; i>=0; i--) {
			heapify(arr,n,i);
		}
		for (int i = n-1; i>0; i--) {
			int temp=arr[0];
			arr[0]=arr[i];
			arr[i]=temp;

			heapify(arr,i,0);
		}
	}
	public static void heapify(int [] arr,int n,int i) {
		int largest=i;
		int left=2*i+1;
		int right=2*i+2;

		if (left<n && arr[left]>arr[largest]) {
			largest=left;
```

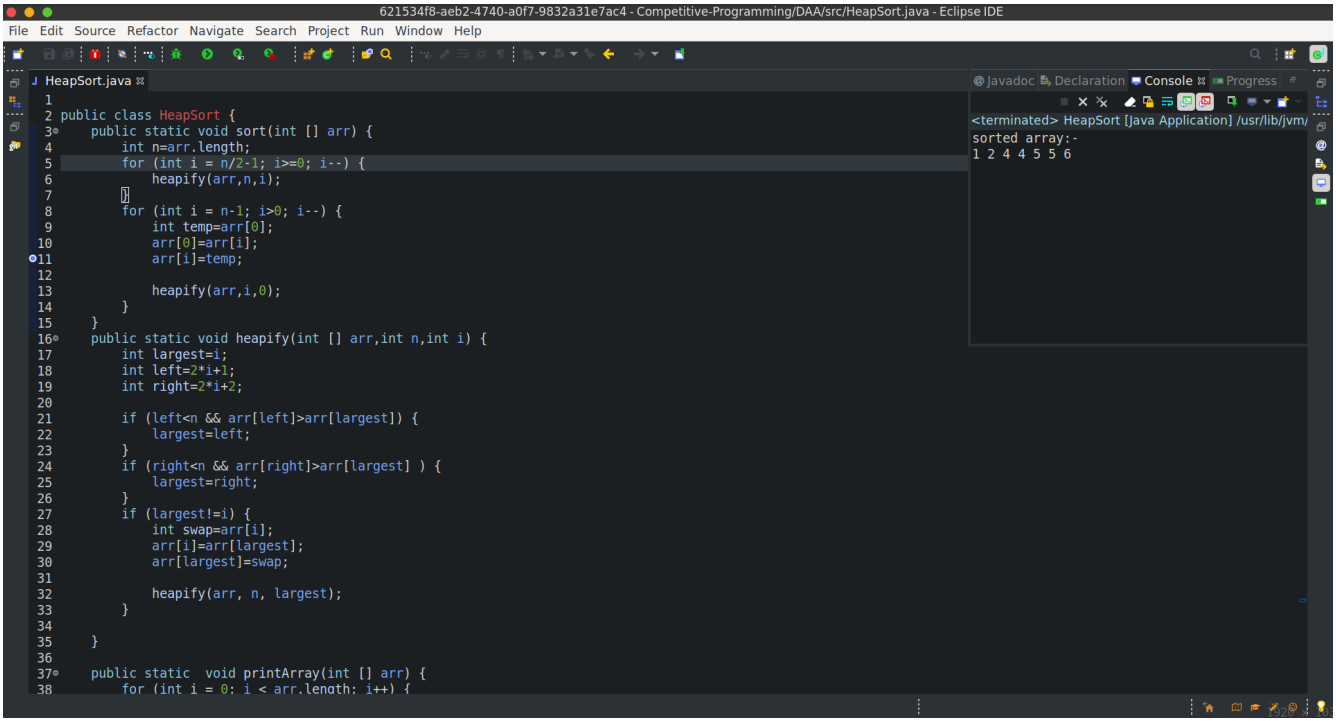
```
            }
            if (right<n && arr[right]>arr[largest] ) {
                  largest=right;
            }
            if (largest!=i) {
                  int swap=arr[i];
                  arr[i]=arr[largest];
                  arr[largest]=swap;

                  heapify(arr, n, largest);
            }

      }

      public static  void printArray(int [] arr) {
            for (int i = 0; i < arr.length; i++) {
                  System.out.print(arr[i]+" ");
            }
            System.out.println();
      }
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            int [] arr= {4,5,6,2,1,4,5};
            sort(arr);
            System.out.println("sorted array:-");
            printArray(arr);
      }

}
```




```java
public class HeapSort {
    public static void sort(int [] arr) {
        int n=arr.length;
        for (int i = n/2-1; i>=0; i--) {
            heapify(arr,n,i);
        }
        for (int i = n-1; i>0; i--) {
            int temp=arr[0];
            arr[0]=arr[i];
            arr[i]=temp;

            heapify(arr,i,0);
        }
    }
    public static void heapify(int [] arr,int n,int i) {
        int largest=i;
        int left=2*i+1;
        int right=2*i+2;

        if (left<n && arr[left]>arr[largest]) {
            largest=left;
        }
        if (right<n && arr[right]>arr[largest] ) {
            largest=right;
        }
        if (largest!=i) {
            int swap=arr[i];
            arr[i]=arr[largest];
            arr[largest]=swap;

            heapify(arr, n, largest);
        }

    }

    public static  void printArray(int [] arr) {
        for (int i = 0; i < arr.length; i++) {
```

Time Complexity:-

1.Best:- O(nlogn)
2.Worst:- O(nlogn)
3.Average:- O(nlogn)