

Data Structures and Algorithms

CSE2001

Assignment - 2

Yashwanth Reddy

19BCE7362

Date- 17th July 2021

Problem : Write an algorithm to find the duplicate number on a given integer array. Implement the algorithm with JAVA. Attach the screenshot of the code and execution result here. Compute the time and space complexities of your algorithm.

Space Complexity : $(4n + 12)$ bytes

Time Complexity : $O(2N)$

Algorithm :

- Declare and initialize an array
- Loop Through the Array
- Outer loop will iterate through the array and selects the element
- Inner loop will compare the selected element with array
- If matched the the numbers will be printed

Code

```
import java.util.*;

public class Duplicate{

    public static void main(String[] args)
    {
        int n,i,j;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of elements in the array:");
        n = sc.nextInt();
        int a[] = new int[n];
        System.out.println("Enter "+ n +" elements ");
        for( i=0; i < n; i++)
        {
            a[i] = sc.nextInt();
        }

        System.out.println("Duplicate Numbers in this array: ");

        for(i = 0; i < a.length; i++) {
            for(j = i + 1; j < a.length; j++) {
                if(a[i] == a[j])
                    System.out.println(a[j]);
            }
        }
    }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

C:\Users\yashw\Desktop\Summer\Assignment>java Duplicate
Enter number of elements in the array:12
Enter 12 elements
2 4 6 8 12 16 2 22 16 26 6 4
Duplicate Numbers in this array:
2
4
6
16

C:\Users\yashw\Desktop\Summer\Assignment>
```

Problem : Write a program to implement a circular linked list and its operations

```
class CircularLinkedList {

    static class Node {
        int data;
        Node next;
    };

    static Node addToEmpty(Node last, int data) {
        if (last != null)
            return last;

        Node newNode = new Node();

        newNode.data = data;
```

```
last = newNode;

newNode.next = last;

return last;
}

static Node addFront(Node last, int data) {
    if (last == null)
        return addToEmpty(last, data);

    Node newNode = new Node();

    newNode.data = data;

    newNode.next = last.next;

    last.next = newNode;

    return last;
}

static Node addEnd(Node last, int data) {
    if (last == null)
        return addToEmpty(last, data);

    Node newNode = new Node();
    newNode.data = data;
    newNode.next = last.next;
    last.next = newNode;
}
```

```

        last = newNode;

        return last;
    }

    static Node addAfter(Node last, int data, int item) {
        if (last == null)
            return null;

        Node newNode, p;
        p = last.next;
        do {
            if (p.data == item) {
                newNode = new Node();
                newNode.data = data;

                newNode.next = p.next;

                p.next = newNode;

                if (p == last)
                    last = newNode;
                return last;
            }
            p = p.next;
        } while (p != last.next);

        System.out.println(item + "The given node is not present in the list");
        return last;
    }
}

```

```
static Node deleteNode(Node last, int key) {

    if (last == null)
        return null;

    if (last.data == key && last.next == last) {
        last = null;
        return last;
    }

    Node temp = last, d = new Node();

    if (last.data == key) {
        while (temp.next != last) {
            temp = temp.next;
        }

        temp.next = last.next;
        last = temp.next;
    }

    while (temp.next != last && temp.next.data != key) {
        temp = temp.next;
    }

    if (temp.next.data == key) {
        d = temp.next;
        temp.next = d.next;
    }
}
```

```
    return last;
}

static void traverse(Node last) {
    Node p;

    if (last == null) {
        System.out.println("List is empty.");
        return;
    }

    p = last.next;

    do {
        System.out.print(p.data + " ");
        p = p.next;
    } while (p != last.next);

}

public static void main(String[] args) {
    Node last = null;

    last = addToEmpty(last, 6);
    last = addEnd(last, 8);
    last = addFront(last, 2);

    last = addAfter(last, 10, 2);

    traverse(last);
}
```

```
        deleteNode(last, 8);  
        traverse(last);  
    }  
}
```

Output

```
C:\Users\yashw\Desktop\Summer\Assignment>java CircularLinkedList  
2 10 6 8 2 10 6  
C:\Users\yashw\Desktop\Summer\Assignment>_
```

Problem : Write an algorithm to prepare a list of a student's score. You may input as many student's scores as you wish. Then sort out the scores and prepare a sorted list. Write the JAVA code, and execute, attach the screenshot of the code and results obtained after execution. Compute the complexity of your algorithm.

Space Complexity : $O(n)$

Time Complexity : $O(n \log n)$

Algorithm :

- if it is only one element in the list it is already sorted, return.
- divide the list recursively into two halves until it can no more be divided.
- merge the smaller lists into new list in sorted order.

Code

```
import java.util.*;

public class MSort
{

    public static void merge(int a[],int l,int m,int h)
    {
        int i,j,c=l;
        int b[]=new int[h+1];

        for(i = l,j = m+1; i<=m && j<=h; c++)
        {
            if(a[i] <= a[j])
                b[c] = a[i++];
            else
                b[c] = a[j++];
        }
        while(i <= m )
            b[c++] = a[i++];

        while(j<=h)
            b[c++] = a[j++];

        for(i = l ; i <= h; i++)
            a[i] = b[i];
    }
}
```

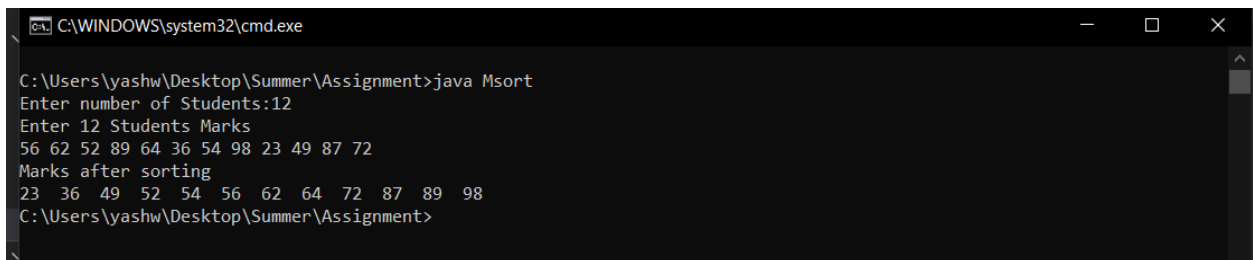
```
public static void Sort(int a[],int l,int h)
{
    if(l<h)
    {
        int m=(l+h)/2;
        Sort(a,l,m);
        Sort(a,m+1,h);
        merge(a,l,m,h);
    }
}

public static void printarray(int a[])
{
    for(int i=0; i < a.length; i++)
    {
        System.out.print(a[i]+" ");
    }
}

public static void main(String[] args)
{
    int n, res,i;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter number of Students:");
    n = s.nextInt();
    int a[] = new int[n];
    System.out.println("Enter "+ n +" Students Marks ");
```

```
for( i=0; i < n; i++)  
{  
    a[i] = s.nextInt();  
}  
  
Sort(a,0,n-1);  
System.out.println( "Marks after sorting");  
printarray(a);  
}  
}
```

Output



```
C:\WINDOWS\system32\cmd.exe  
C:\Users\yashw\Desktop\Summer\Assignment>java Msort  
Enter number of Students:12  
Enter 12 Students Marks  
56 62 52 89 64 36 54 98 23 49 87 72  
Marks after sorting  
23 36 49 52 54 56 62 64 72 87 89 98  
C:\Users\yashw\Desktop\Summer\Assignment>
```