DESIGN ANALYSIS AND ALGORITHM

PRACTICAL NO 2 & 2B

027_Abhishek_Ojha

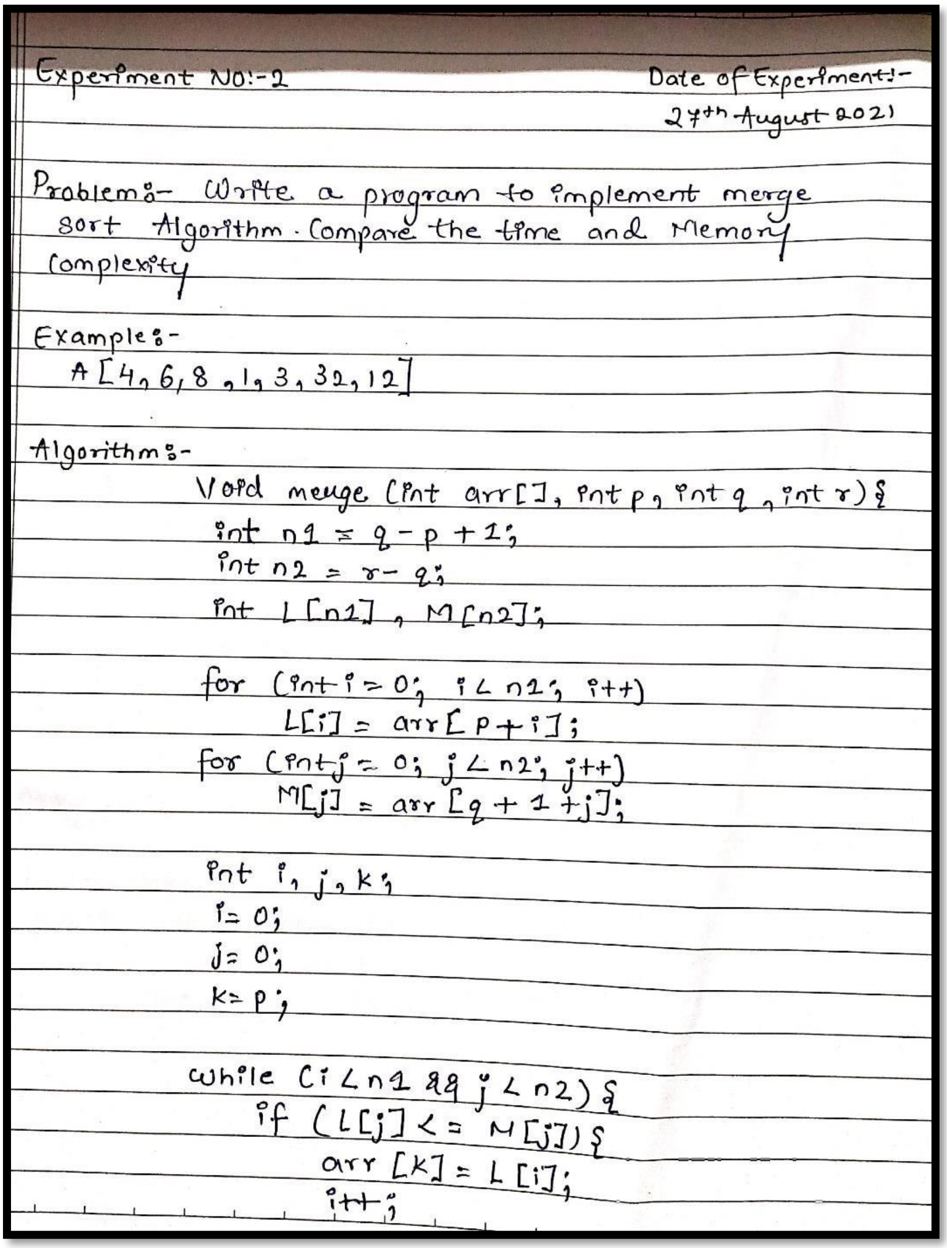
Experiment No -2 **Date of Experiment : 27 August 2021**

Program : - Write a program to implement Merge Sort Algorithm. Compare the time and Memory Complexity

Example :-

Input :- A[4, 6, 8, 1, 3, 32, 12]

Algorithm :-



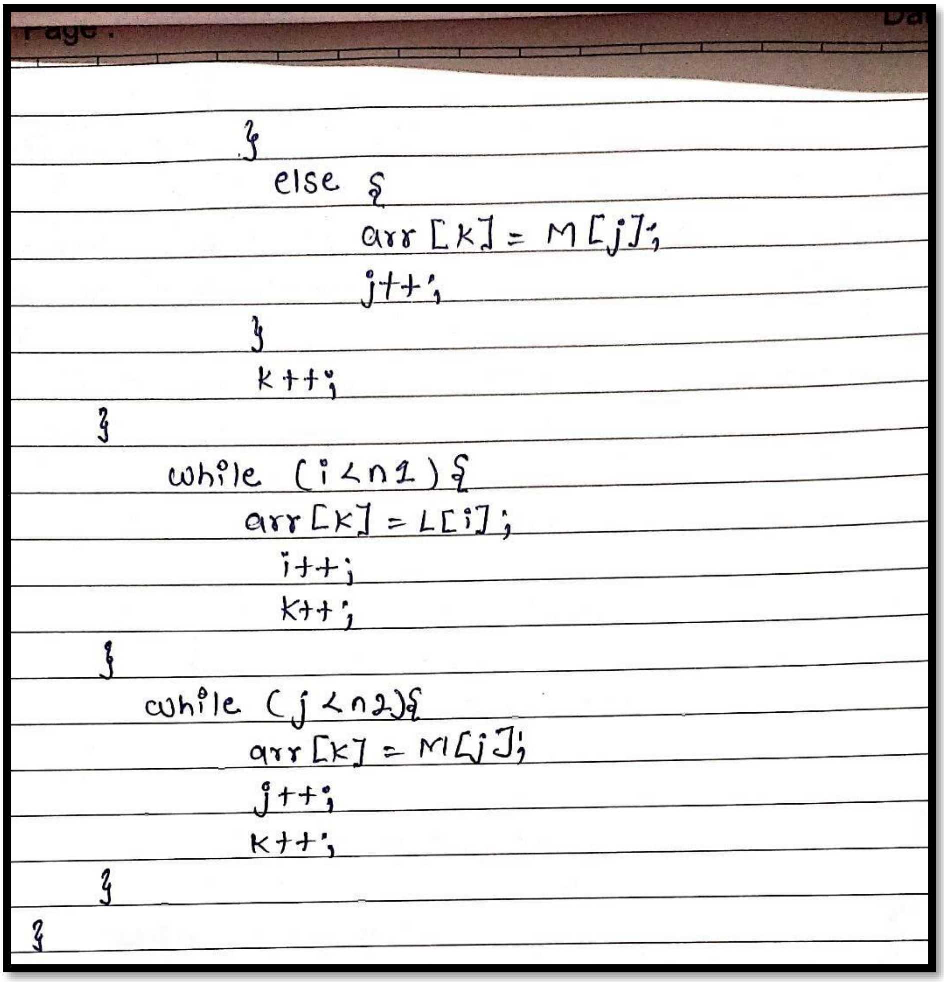
Experiment No:-2 Date of Experiment:-
27th August 2021

Problem:- Write a program to implement merge sort Algorithm. Compare the time and Memory complexity

Example:-
A [4, 6, 8, 1, 3, 32, 12]

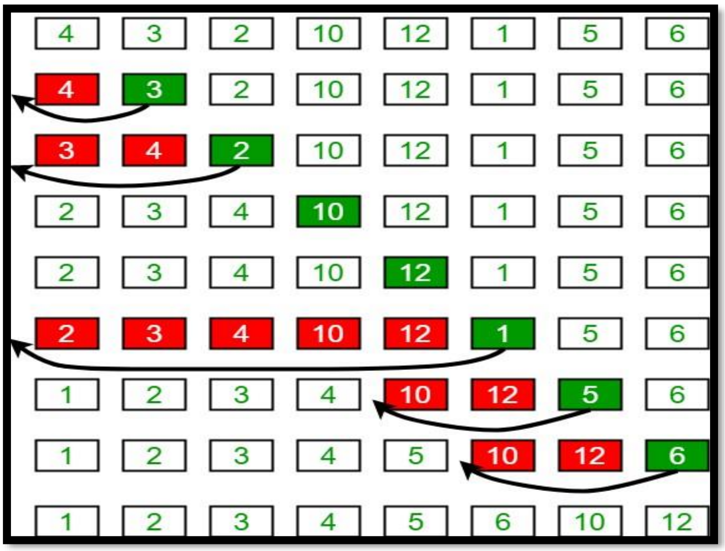
Algorithm:-

```
Void meuge (int arr[], int p, int q, int r) {
  int n1 = q - p + 1;
  int n2 = r - q;
  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
        L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
        M[j] = arr[q + 1 + j];

  int i, j, k;
  i = 0;
  j = 0;
  k = p;

  while (i < n1 && j < n2) {
      if (L[j] <= M[j]) {
          arr [k] = L [i];
          i++;
```

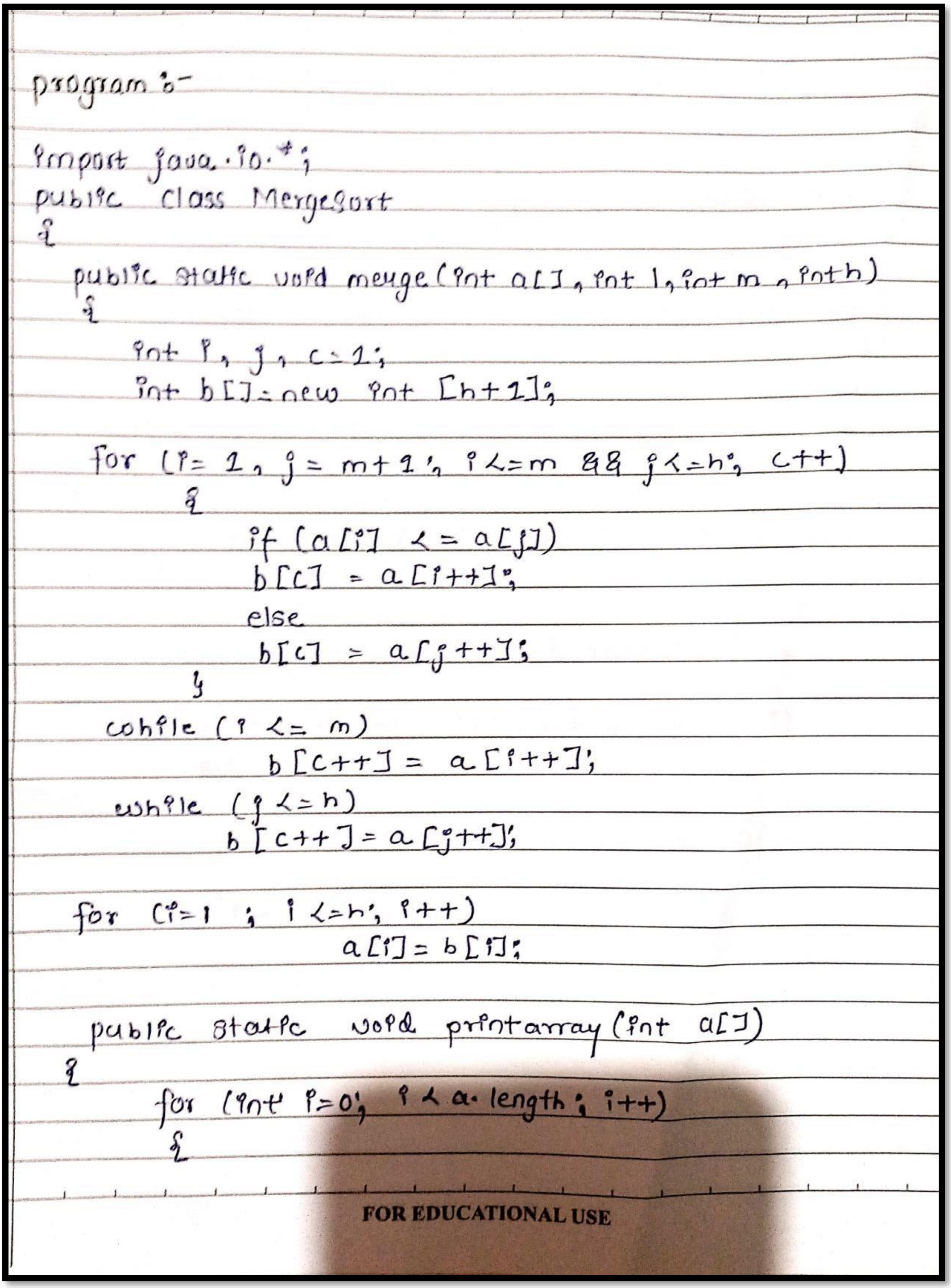


```
        }
          else {
              arr[k] = M[j];
              j++;
        }
          k++;
    }
       while (i<n1) {
           arr[k] = L[i];
            i++;
            k++;
    }
     while (j<n2){
           arr[k] = M[j];
           j++;
           k++;
    }
}
```

Fig :

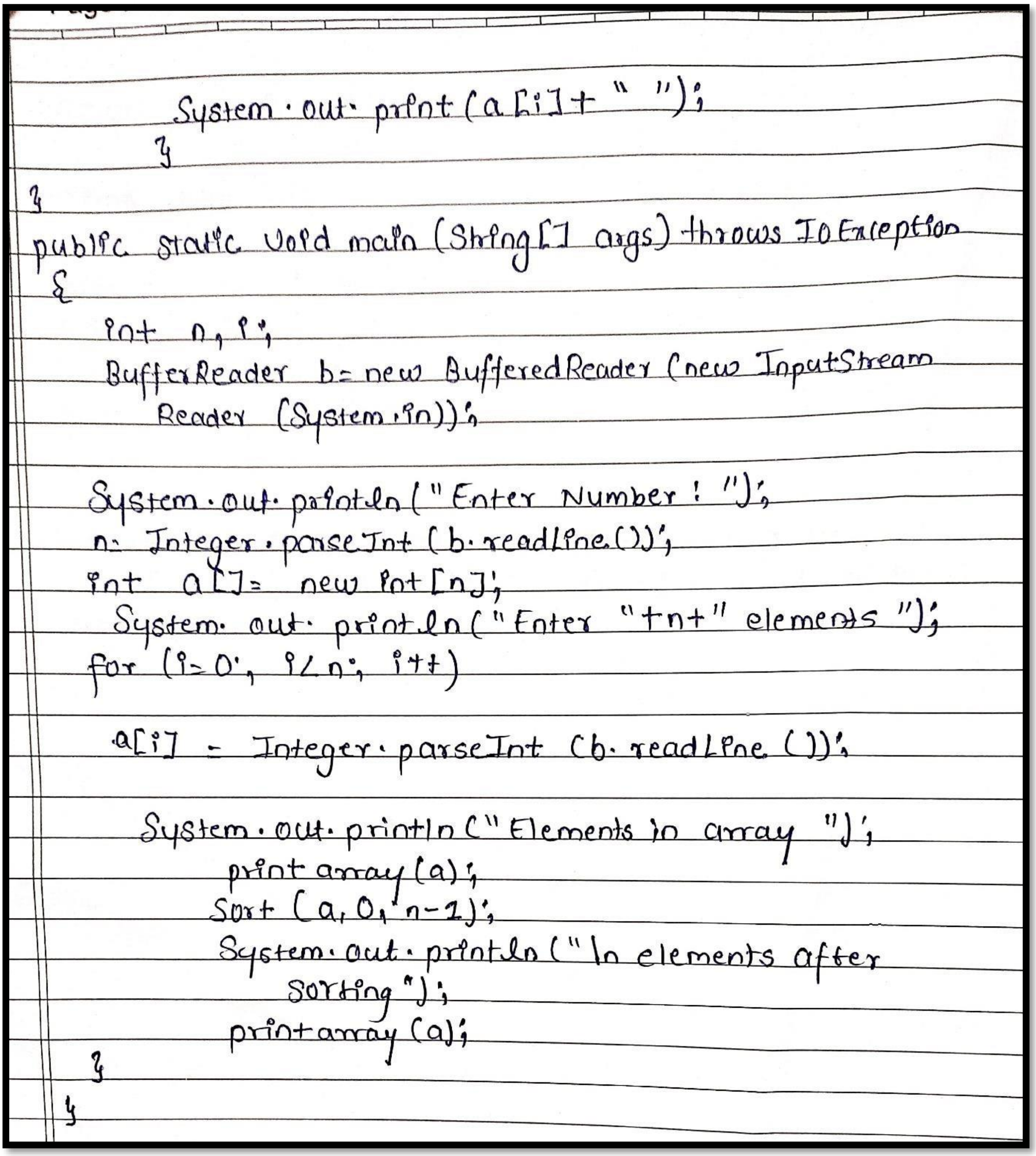


Program:-

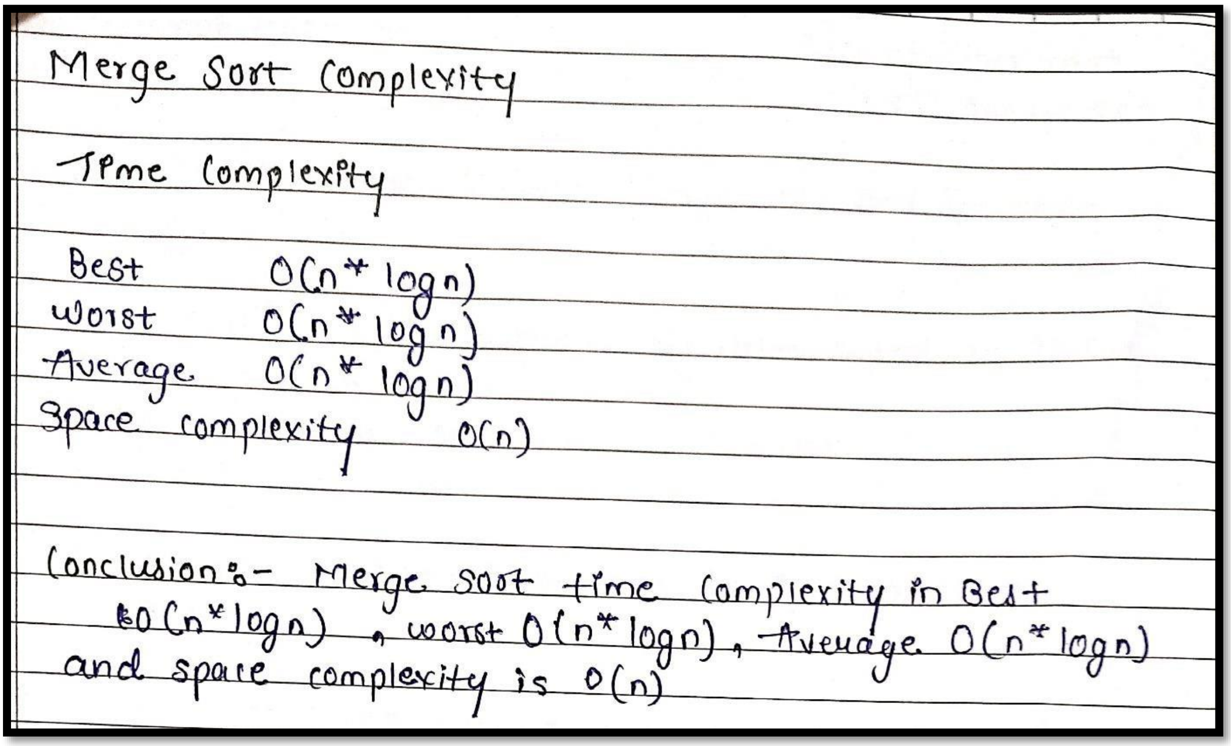


```
program :-

import java.io.*;
public class Mergesort
{
    public static void merge(int a[], int l, int m, int h)
    {
         int i, j, c=1;
         int b[]=new int [h+1];

      for (i= 1, 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=1  ; i<=h; i++)
                            a[i] = b[i];

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

FOR EDUCATIONAL USE



```
        System.out.print(a[i]+" ");
      }
}
public static void main (String[] args) throws IOException
 {
   int n, i;
   BufferReader b= new BufferedReader (new InputStream
        Reader (System.in));

  System.out.println("Enter Number! ");
  n= Integer.parseInt (b.readLine());
  int a[]= new int[n];
   System. out. println("Enter "+n+" elements ");
  for (i=0; i<n; i++)

   a[i] = Integer.parseInt (b.readLine ());

     System.out.println("Elements in array ");
        print array(a);
        Sort (a,0,n-1);
        System.out.println("\n elements after
             sorting");
        printarray (a);
  }
 }
```



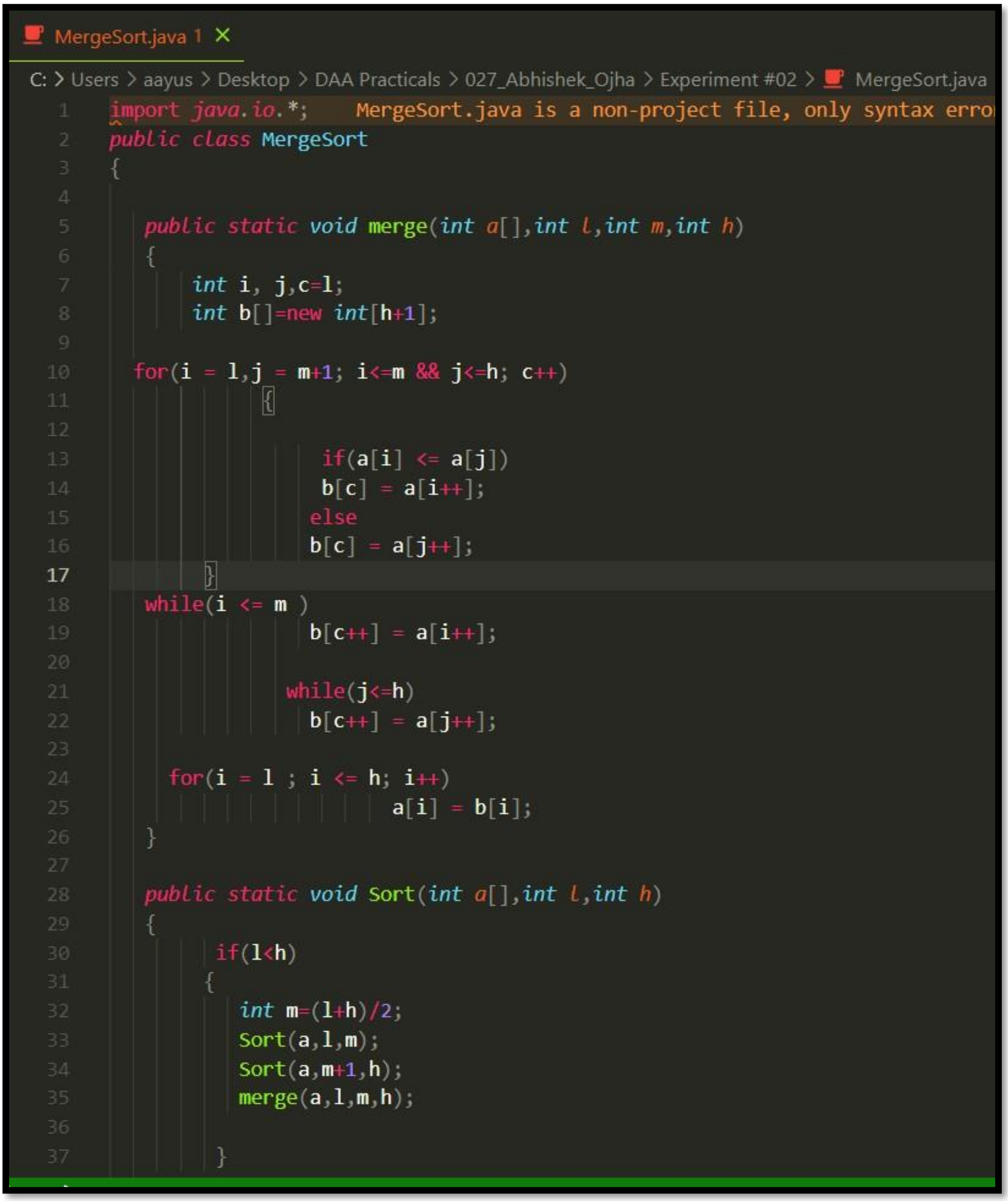
Merge Sort Complexity

Time Complexity

Best	$O(n * \log n)$
worst	$O(n * \log n)$
Average	$O(n * \log n)$
Space complexity	$O(n)$

Conclusion:- Merge sort time complexity in Best ~~k~~$O(n * \log n)$, worst $O(n * \log n)$, Average $O(n * \log n)$ and space complexity is $O(n)$

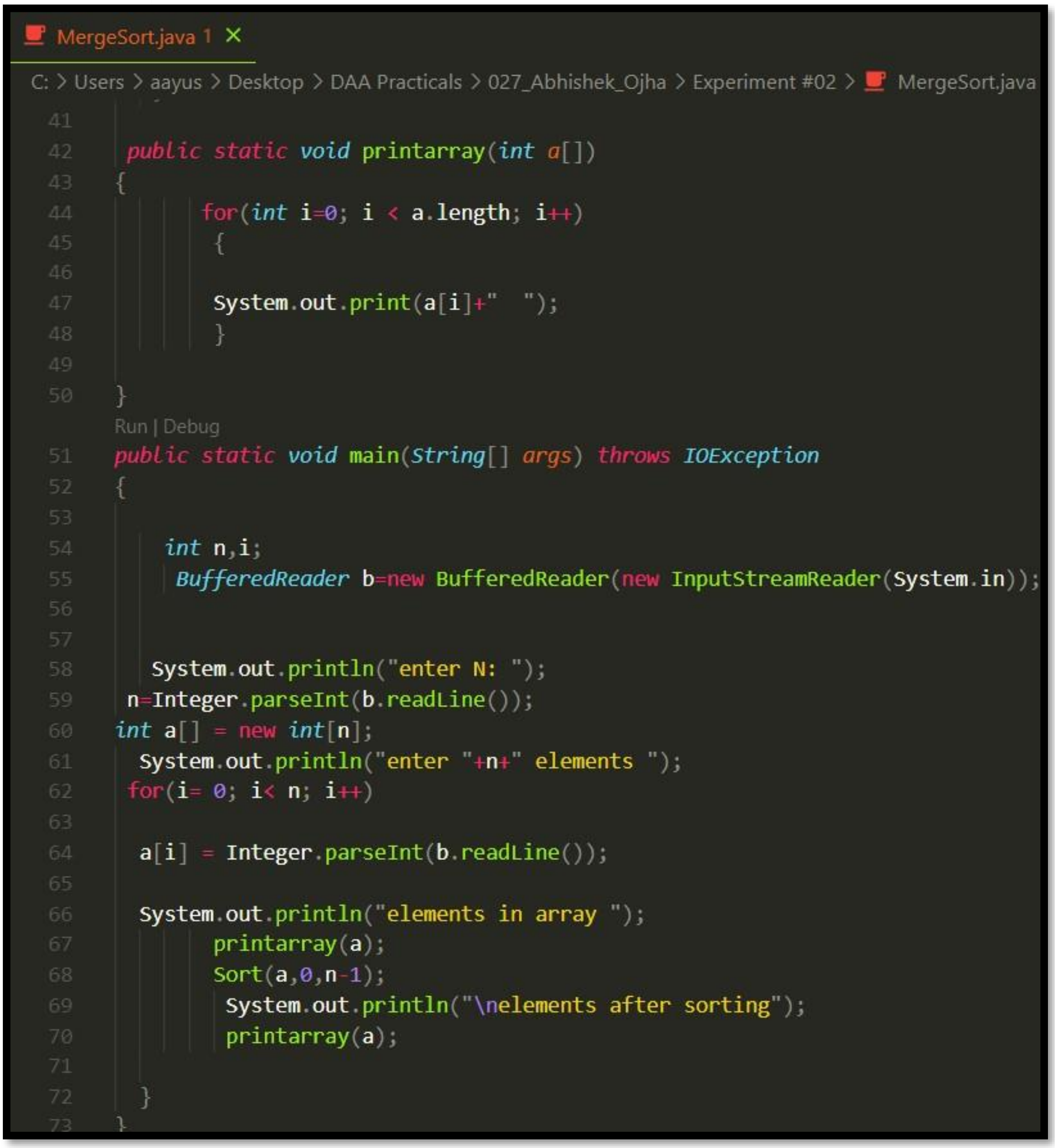
Practical Implementation of Insertion Sort :-



MergeSort.java 1 ×

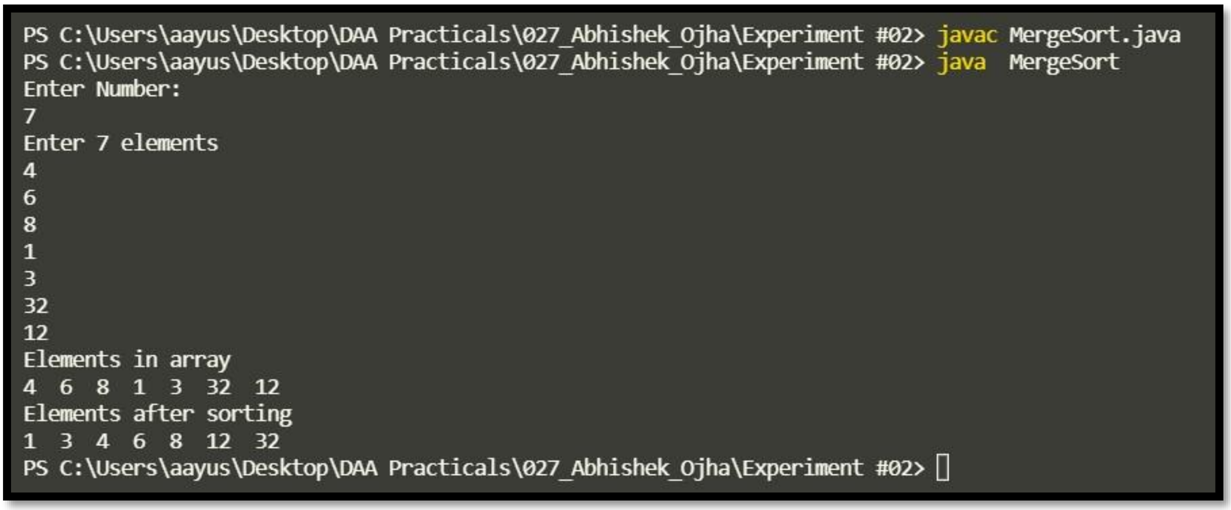
C: > Users > aayus > Desktop > DAA Practicals > 027_Abhishek_Ojha > Experiment #02 > MergeSort.java

```
import java.io.*;     MergeSort.java is a non-project file, only syntax erro
public class MergeSort
{

   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);

          }
```


MergeSort.java 1 ×
C: > Users > aayus > Desktop > DAA Practicals > 027_Abhishek_Ojha > Experiment #02 > MergeSort.java
Run | Debug

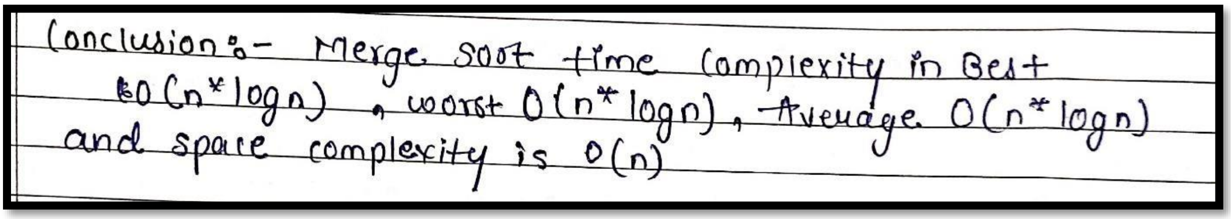
```
 public static void printarray(int a[])
{
        for(int i=0; i < a.length; i++)
         {

         System.out.print(a[i]+"  ");
         }

}
public static void main(String[] args) throws IOException
{

    int n,i;
     BufferedReader b=new BufferedReader(new InputStreamReader(System.in));


   System.out.println("enter N: ");
 n=Integer.parseInt(b.readLine());
int a[] = new int[n];
  System.out.println("enter "+n+" elements ");
 for(i= 0; i< n; i++)

  a[i] = Integer.parseInt(b.readLine());

  System.out.println("elements in array ");
        printarray(a);
        Sort(a,0,n-1);
         System.out.println("\nelements after sorting");
         printarray(a);

  }
}
```

Output:



```
PS C:\Users\aayus\Desktop\DAA Practicals\027_Abhishek_Ojha\Experiment #02> javac MergeSort.java
PS C:\Users\aayus\Desktop\DAA Practicals\027_Abhishek_Ojha\Experiment #02> java  MergeSort
Enter Number:
7
Enter 7 elements
4
6
8
1
3
32
12
Elements in array
4  6  8  1  3  32  12
Elements after sorting
1  3  4  6  8  12  32
PS C:\Users\aayus\Desktop\DAA Practicals\027_Abhishek_Ojha\Experiment #02> 
```

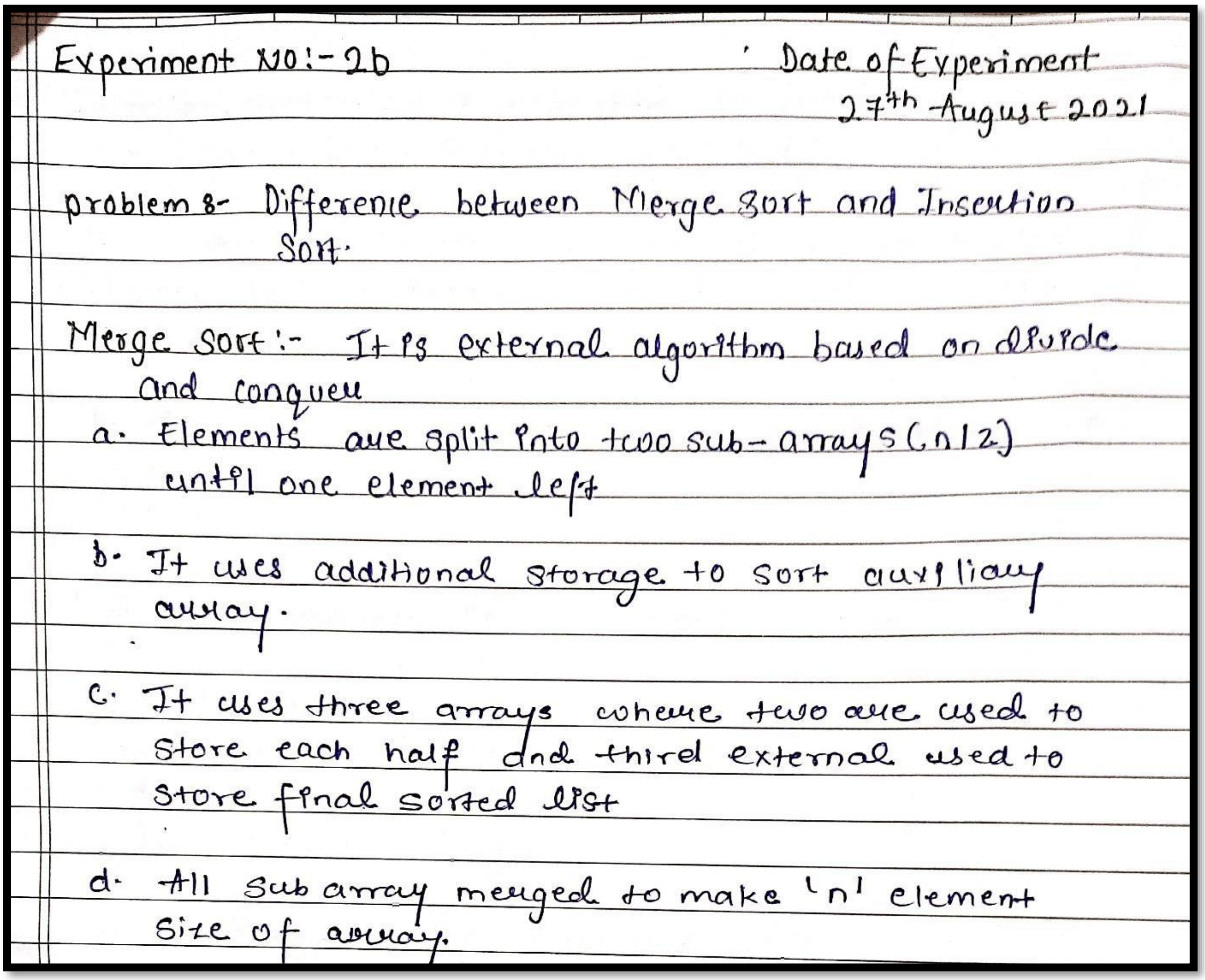
Conclusion :



Conclusion :- Merge sort time complexity in Best O(n*logn) , worst O(n*logn), Average O(n*logn) and space complexity is O(n)

Experiment No -2B **Date of Experiment : 27 August 2021**

Program : Difference between Merge Sort and Insertion Sort



Experiment No:-2b Date of Experiment
27th August 2021

problem :- Differenie between Merge Sort and Insertion Sort.

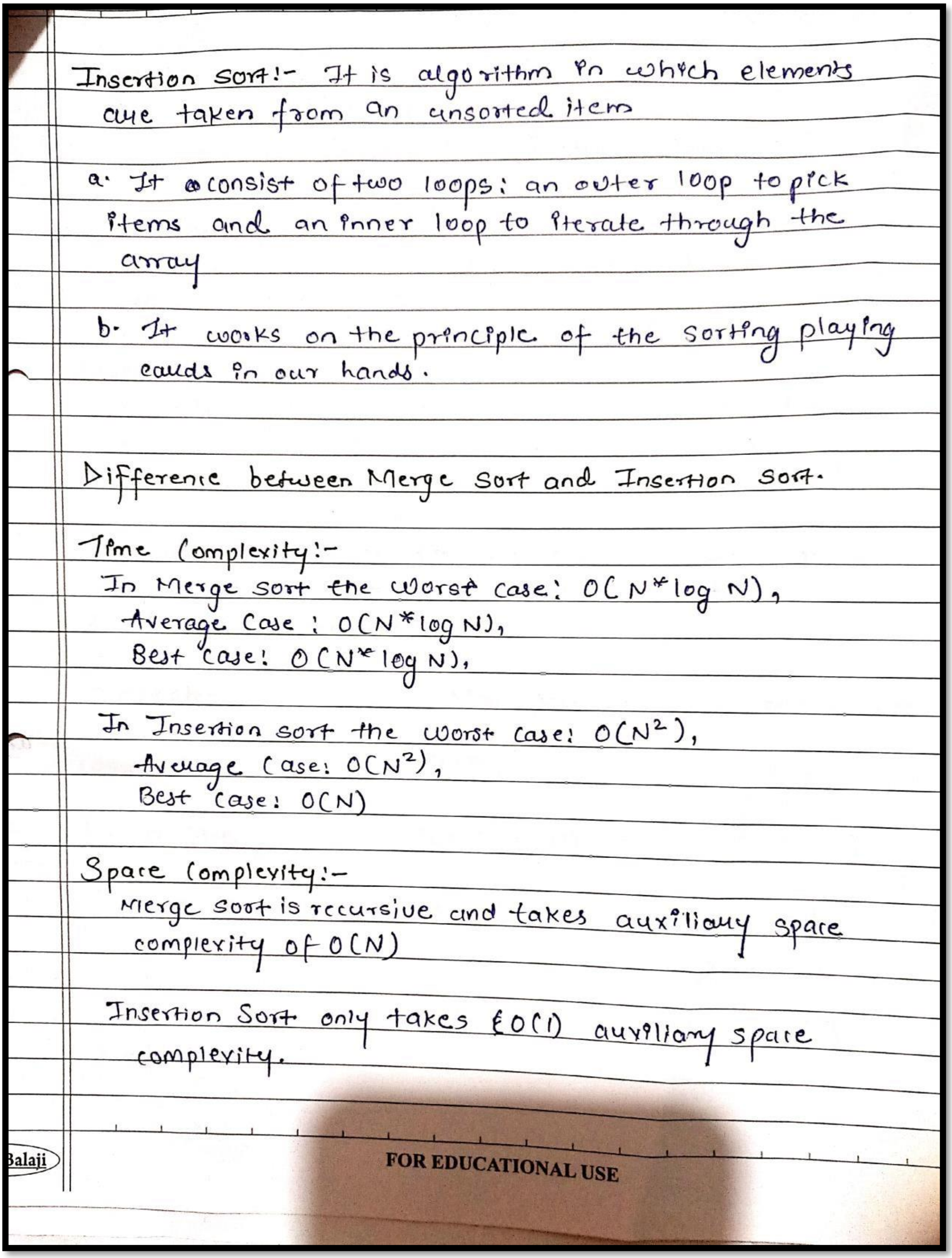
Merge Sort :- It is external algorithm based on divide and conquer

a. Elements are split into two sub-arrays (n/2) until one element left

b. It uses additional storage to sort auxiliary array.

c. It uses three arrays where two are used to store each half and third external used to store final sorted list

d. All sub array merged to make 'n' element size of array.



Insertion Sort:- It is algorithm in which elements are taken from an unsorted item

a. It consist of two loops: an outer loop to pick items and an inner loop to iterate through the array

b. It works on the principle of the sorting playing cards in our hands.

Difference between Merge Sort and Insertion Sort.

Time Complexity:-

In Merge sort the worst case: O(N*log N),
Average Case : O(N*log N),
Best case: O(N*log N),

In Insertion sort the worst case: $O(N^2)$,
Average Case: $O(N^2)$,
Best case: O(N)

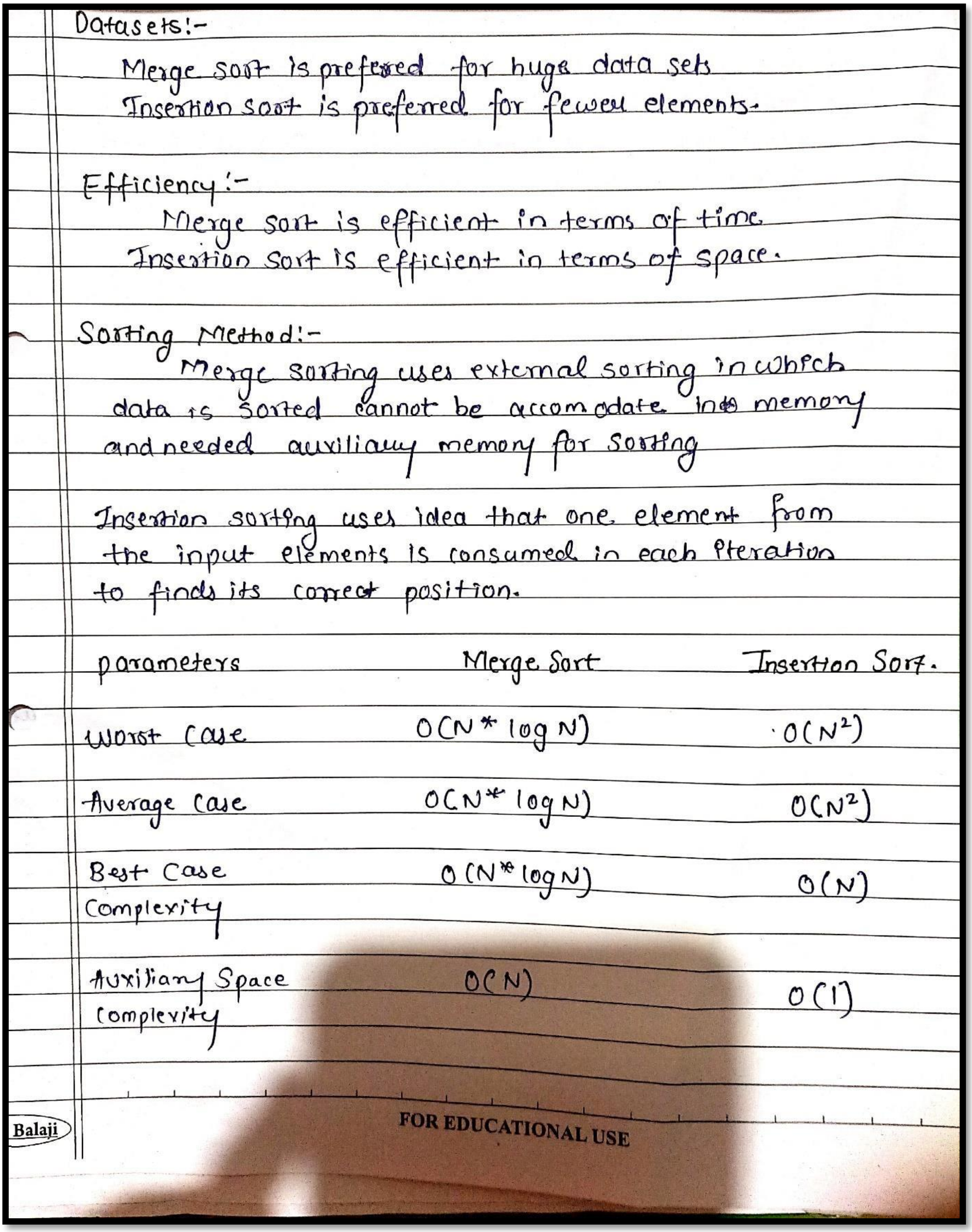
Space Complexity:-

Merge sort is recursive and takes auxiliary space complexity of O(N)

Insertion Sort only takes O(1) auxiliary space complexity.

Balaji

FOR EDUCATIONAL USE



Datasets:-

Merge sort is prefered for huge data sets
Insertion sort is preferred for fewer elements.

Efficiency:-

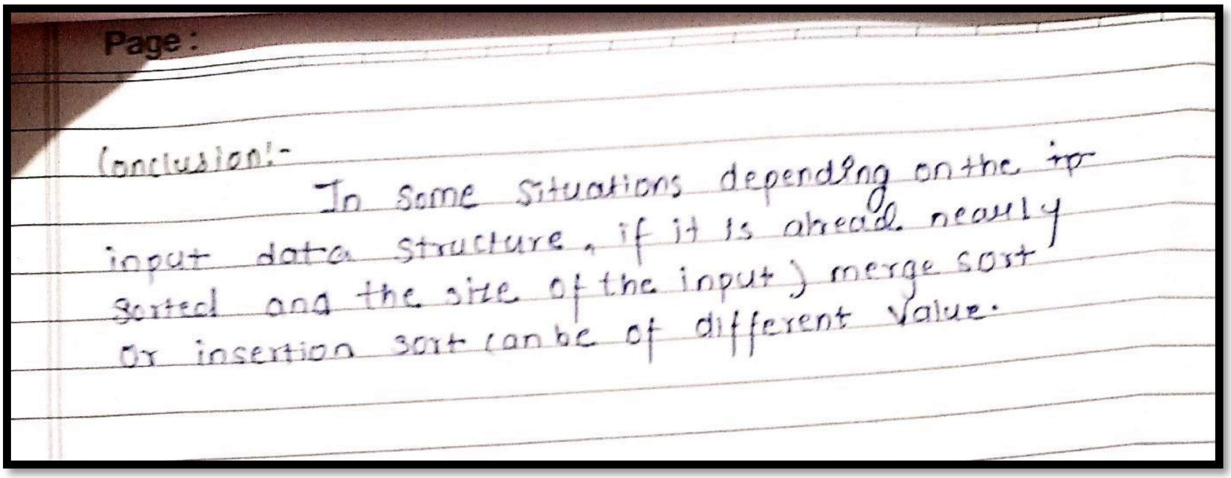
Merge sort is efficient in terms of time
Insertion sort is efficient in terms of space.

Sorting Method:-

Merge sorting uses external sorting in which data is sorted cannot be accomodate into memory and needed auxiliary memory for sorting

Insertion sorting uses idea that one element from the input elements is consumed in each iteration to finds its correct position.

parameters	Merge Sort	Insertion Sort.
Worst case	O(N * log N)	$O(N^2)$
Average case	O(N * log N)	$O(N^2)$
Best Case Complexity	O(N * log N)	O(N)
Auxiliary Space Complexity	O(N)	O(1)



Page :

Conclusion:-

In some situations depending on the ~~ip~~ input data structure, if it is alread nearly sorted and the size of the input) merge sort or insertion sort can be of different value.