Lab Tasks week wise

Name: Md. Amirul Islam
ID: 191-15-12123
Section: O-14
Department of CSE

5. Binary Search

```
class Binary SearchExample{
public static void binary Search (int arr[], int first, int last, int key) {
int mid = (first + last)/2;
while (first <=last){
if (arr [mid] <key){
first = mid + 1;
}else if ( arr[mid] ==key){
System.out.println("Element is found at index: " + mid);
break;
}else{
last=mid – 1;
}
mid = (first + last)/2;
}
if (first > last){
System.out.println(“Element is not found!");
}
}
public static void main(String args[]){
int arr[] = {10,20,30,40,50};
int key = 30;
int last=arr.length-1;
binary Search(arr,Q,last,key);
}
}
```

Binary search Algorithm:

Worst case performance O(logn)
Best case performance 0(1)
Average case performance O(log n)

6. Merge Sort

```
public class My Merge Sort
{
void merge(int arr[], int beg, int mid, int end)
{

int l = mid - beg +l;
int r = end- mid;
int LeftArray ( ) = new int [l];
intRightArray ( ) = new int [r];
```

```
for (int i=0,i<l,++i)
LeftArray[i] = arr[beg +l];
for (int j=0;j<r, ++j)
RightArray[j] = arr[mid+l +j];

int i=0,j=0;
int k = beg:
while (i<l&&j<r)
{
if (LeftArray[i]<= RightArray[j])
}
arr[k] = Left Array[i];
i++;
}
else
{
arr[k] = RightArray[j];
j++;
}
k++;
}
while (i<l)
{
arr[k] = LeftArray[i];
i++;
j++;
}
while (j<r)
}
arr[k] = RightArray[i]
j++;
k++;
void sort(int arr[], int beg, int end)
{
if (beg<end)
{
int mid=(beg +end)/2;

sort(arr, beg, mid);
sort(arr, mid+ 1, end);
merge(arr, beg, mil, end);
}
}
public static void main(String args[])
{
intarr[] = {90,23,101,45,65,23,67,89,34,23);
```

```
MyMerge Sort ob = new MyMergeSort();
ob.sort(arr, 0, ar.length-1);
System.out.println("\nSorted array");
for(int i=0; i<arr.length;i++)
{
System.out.println(arr[i]+"");
}
}
}
```

Time complexity of Merge Sort is O(n*Logn)in all the 3 cases

7. Quick Sort

```
public class QuickSort{
public static void main(String[] args) {
int i;
int[] arr=[90,23,101,45,65,23,67,89,34,23);
quickSort(arr, 0,9);
System.out.println("\. The sorted a ray is: \n");
for(i=0;i<10:i++)
System.out.println( arr[i]);
}
public static int partition(int a[], int beg, int end)
{
int left, right, temp, loc, flag:
loc =left = beg:
right = end;
flag = 0;
while(flag != 1)
{
while((a[loc] <= a[right]) && (loc!=right))
right--;
if(loc==right)
flag =l;
elseif (a[loc]>a[right])
{
temp=a[loc];
a[loc] = a[right];
a[right] = temp;
loc=right;
}
if(flag!= 1)
{
while((a[loc] >= a[left]) && (loc!=left))
```

```
left++;
if(loc=left)
flag=l;
elseif(a[loc]<a[left])
{
temp =a[loc];
a[loc] = a[left];
a[left] = temp;
loc = left;
}
}
}
Returnloc;
}
static void quickSort (int a[], int beg, int end)
{
int loc;
if(beg<end)
{
loc = partition(a, beg, end);
quickSort(a, beg, loc-1):
quick Sorula, loc+1, end);
}
}
}
```

Quick Sort Algorithm:

Worst case performance $O(n^2)$
Best case performance O(n)
Average case performance O(n log n)