NAME : SAGAR VED BAIRWA
SECTION : 2A
ID : 201951131

INDEX

BUBBLE SORT

ALGORITHAM:

```
Step 1 : Repeat For P = 1 to N – 1          BEGIN
Step 2 :Repeat For J = 1 to N – P          BEGIN
Step 3 :If ( A [ J ] < A [ J – 1 ] )
           Swap ( A [ J ] , A [ J – 1 ] )     BEGIN
            End For
Step 4 : Exit
```

ARRAY IMPILENTATION:

```
#include<stdio.h>

int main()
{
int i,n,temp,j,arr[25];
printf("Enter the number of elements in the Array: ");
scanf("%d",&n);
printf("\nEnter the elements:\n\n");
for(i=0 ; i<n ; i++)
{
printf(" Array[%d] = ",i);
scanf("%d",&arr[i]);
}
for(i=0 ; i<n ; i++)
{
for(j=0 ; j<n-i-1 ; j++)
{
if(arr[j]>arr[j+1])
{
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
printf("\nThe Sorted Array is:\n\n");
for(i=0 ; i<n ; i++)
{
printf(" %4d",arr[i]);
}
}
```

OUTPUT:

```
Enter the number of elements in the Array: 5

Enter the elements:

 Array[0] = 1
 Array[1] = 3
 Array[2] = 5
 Array[3] = 6
 Array[4] = 4

The Sorted Array is:

    1    3    4    5    6
```

LINK LIST IMPLIMENTATION:

INSERTION SORT

ALGORITHAM:

```
Step 1 − If it is the first element, it is already sorted. return 1;
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the
         value to be sorted
Step 5 − Insert the value
Step 6 − Repeat until list is sorted
```

ARRAY IMPLIMENTATION:

```
#include<stdio.h>
int main( )
{
int a[10],i,j,k,n;

printf("How many elements you want to sort?\n");
scanf("%d",&n);
printf("\nEnter the Elements into an array:\n");
for (i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=1;i<n;i++)
{
k=a[i];
for(j= i-1; j>=0 && k<a[j]; j--)
a[j+1]=a[j];
a[j+1]=k;
}
 printf("\n\n Elements after sorting: \n");
for(i=0;i<n;i++)
printf("%d\n", a[i]);
}
```

OUTPUT:

```
How many elements you want to sort?
5

Enter the Elements into an array:
3
2
4
1
5


 Elements after sorting:
1
2
3
4
5
```

LINK LIST IMPLIMENTATION:

SELECTION SORT

ALGORITHAM:

```
Step 1 − Set MIN to location 0
Step 2 − Search the minimum element in the list
Step 3 − Swap with value at location MIN
Step 4 − Increment MIN to point to next element
Step 5 − Repeat until list is sorted
```

ARRAY IMPLIENTATION:

```
#include<stdio.h>
int main( )
{
int i,j,t,n,min,a[10];
printf("\n How many elements you want to sort? ");
scanf("%d",&n);
printf("\n Enter elements for an array:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
{
min=i;
for(j=i+1;j<n;j++)
if(a[j] > a[min])
{
min=j;
}
t=a[i];
a[i]=a[min];
a[min]=t;
} printf("\nAfter sorting the elements are:");
for(i=0;i<n;i++)
printf("%d ",a[i]);

}
```

OUTPUT:

```
How many elements you want to sort? 5

 Enter elements for an array:4
1
3
2
5

After sorting the elements are:5 4 3 2 1
```

LINK LIST IMPLIMENTATION:

MERGE SORT

ALGORITHAM:

```
Step 1 − if it is only one element in the list it is
already sorted, return.
Step 2 − divide the list recursively into two halves
until it can no more be divided.
Step 3 − merge the smaller lists into new list in sorted
order.
```

ARRAY IMPLIMENTATION:

```
#include<stdio.h>
void disp( );
void mergesort(int,int,int);
void msortdiv(int,int);
int a[50],n;
int main( )
{
int i;
printf("\nEnter the n value:");
scanf("%d",&n);
printf("\nEnter elements for an array:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nBefore Sorting the elements are:");
disp( );
msortdiv(0,n-1);
printf("\nAfter Sorting the elements are:");
disp( );
}
void disp( )
{
int i;
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
void mergesort(int low,int mid,int high)
{
int t[50],i,j,k;
i=low;
j=mid+1;
k=low;
while((i<=mid) && (j<=high))
{
if(a[i]>=a[j])
t[k++]=a[j++];
else
t[k++]=a[i++];
}
while(i<=mid)
t[k++]=a[i++];
while(j<=high)
t[k++]=a[j++];
for(i=low;i<=high;i++)
a[i]=t[i];
}
void msortdiv(int low,int high)
{
int mid;
if(low!=high)
{
mid=((low+high)/2);
msortdiv(low,mid);
msortdiv(mid+1,high);
mergesort(low,mid,high);
}
}
```

Online compiler link: **compiler link**

OUTPUT:

```
Enter the n value: 5

Enter elements for an array: 22 44 55 11 13

Before Sorting the elements are: 22 44 55 11 13
After Sorting the elements are: 11 13 22 44 55
```

Time Complexity of merge sort:

Best case: O (n log n)
Average case : O (n log n)
Worst case : O (n log n)

LINK LIST IMPLIMENTATION:

QUICK SORT

ALGORITHAM:

```
Step 1 − Choose the highest index value has pivot
Step 2 − Take two variables to point left and right of the list excluding pivot
Step 3 − left points to the low index
Step 4 − right points to the high
Step 5 − while value at left is less than pivot move right
Step 6 − while value at right is greater than pivot move left
Step 7 − if both step 5 and step 6 does not match swap left and right
Step 8 − if left ≥ right, the point where they met is new pivot
```

ARRAY IMPLIMENTATION:

```
#include<stdio.h>
void quicksort(int[ ],int,int);
int main( )
{
int low, high, n, i, a[10];
printf("\nHow many elements you want to sort ? ");
scanf("%d",&n);
printf("\n Enter elements for an array:");
for(i=0; i<n; i++)
scanf("%d",&a[i]);
low=0;
high=n-1;
quicksort(a,low,high);
printf("\n After Sorting the elements are:");
for(i=0;i<n;i++)
printf("%d ",a[i]);

}
void quicksort(int a[ ],int low,int high)
{
int pivot,t,i,j;
if(low<high)
{
pivot=a[low];
i=low+1;
j=high;
while(1)
{while(pivot>a[i]&&i<=high)
i++;
while(pivot<a[j]&&j>=low)
j--;
if(i<j)
{
t=a[i];
a[i]=a[j];
a[j]=t;
}
else
break;
}
a[low]=a[j];
a[j]=pivot;
quicksort(a,low,j-1);
quicksort(a,j+1,high);
}
}
```

Online compiler link: **compiler link**

OUTPUT:

```
How many elements you want to sort ? 6

 Enter elements for an array:11 13 5 6 7 2

 After Sorting the elements are:2 5 6 7 11 13
```

Time Complexity of Quick sort:

Best case : O (n log n)
Average case : O (n log n)
Worst case : O (n ^{2})

LINK LIST IMPLIMENTATION: