Bubble sort in C:-

#Bubble sort has best time complexity of O(n) which is possible if the list is already sorted.

#It has a worst case time complexity of O(n2).

#Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.   
In computer graphics, it is popular for its capability to detect a tiny error (like a swap of just two elements) in almost-sorted arrays and fix it with just linear   
complexity (2n).

#include <stdio.h>

int main() {

int a, i;

printf("Enter the number of elements in the array: ");

scanf("%d", &a);

int arr[a];

for(i=0; i<a; i++){

printf("%d element of array: ",i);

scanf("%d", &arr[i]);

}

for(i=0; i<a-1; i++){

for(int j=0; j<a-i-1; j++){

if(arr[j]>arr[j+1]){

int swap=arr[j];

arr[j]=arr[j+1];

arr[j+1]=swap;

}

}

}

for(i=0; i<a; i++){

printf("%d ", arr[i]);

}

return 0;

Insertion sort in C:-

#Insertion sort is inefficient against larger umber of datasets and displays worst case time complexity of O(n2).

#It’s best case time complexity is O(n) and space complexity of O(1).

#Insertion sort is based on iterating over the existing elements while taking input and placing them where they are ought to be.

#include<stdio.h>

int main(){

int i, j, a, b;

printf("Enter the number of elements: ");

scanf("%d", &a);

int arr[a];

for(i=0; i<a; i++){

printf("Enter the %dth element: ", i);

scanf("%d", &arr[i]);

}

for(i=1; i<=a-1; i++){

for(j=i; j>0 && arr[j-1]> arr[j]; j--){

b=arr[j];

arr[j]= arr[j-1];

arr[j-1]=b;

}

}

for(i=0; i<a; i++){

printf("%d ", arr[i]);

}

return 0;

}

Selection sort:-

#The best case, average case and worst case time complexity of selection sort is the same, i.e. O(n2).

#It has a space complexity of O(1).

#include <stdio.h>

int main()

{

int a, i, j, position, temp;

printf("Enter number of elements\n");

scanf("%d", &a);

int arr[a];

for (i = 0; i< a; i++){

printf("Enter the %dth element: ", i);

scanf("%d", &arr[i]);

}

for (i = 0; i<(a-1); i++){

position=i;

for (j = i+1; j<a; j++){

if (arr[position] > arr[j]){

position = j;

}

}

if (position != i){

temp = arr[i];

arr[i] = arr[position];

arr[position] = temp;

}

}

for (i= 0; i < a; i++){

printf("%d ", arr[i]);

}

return 0;

}

Merge Sort:-

#Merge sort has the worst, average and best time complexity of O(n(logn)). It’s space complexity depends on the number of comparisons it has to make.

#It is better than other types of sorts like selection, insertion and bubble.

#It works by splitting the array into 1-element-containing array and then merging it, sorting it while merging it.

#include <stdio.h>

void merge\_sort(int i, int j, int a[], int temp[]) {

if (j <= i) {

return;

}

int mid = (i + j) / 2;

merge\_sort(i, mid, a, temp);

merge\_sort(mid + 1, j, a, temp);

int left = i;

int right = mid + 1;

int k;

for (k = i; k <= j; k++) {

if (left == mid + 1) {

temp[k] = a[right];

right++;

}

else if (right == j + 1) {

temp[k] = a[left];

left++;

}

else if (a[left] < a[right]) {

temp[k] = a[left];

left++;

}

else{

temp[k] = a[right];

right++;

}

}

for (k = i; k <= j; k++) {

a[k] = temp[k];

}

}

int main() {

int temp[100], n, i, d, swap;

printf("Enter number of elements in the array: ");

scanf("%d", &n);

int a[n];

for (i = 0; i < n; i++){

printf("Enter the %d element: ", i);

scanf("%d", &a[i]);

}

merge\_sort(0, n - 1, a, temp);

for (i = 0; i < n; i++){

printf("%d\n", a[i]);

}

return 0;

}

Sorting codes in python:

Bubble sort:-

def bubble(list):

for i in range(len(list) - 1, 0, -1):

no\_swap = True

for j in range(0, i):

if list[j + 1] < list[j]:

list[j], list[j + 1] = list[j + 1], list[j]

no\_swap = False

if no\_swap:

return

list = input('Enter the list of numbers: ').split()

list = [int(x) for x in list]

bubble(list)

print('Sorted list: ', end='')

print(list)

Insertion Sort in Python:-

for i in a:

j = a.index(i)

while j>0:

if a[j-1] > a[j]:

a[j-1],a[j] = a[j],a[j-1]

else:

break

j = j-1

print (a)

#Where a is the array you want to sort.

Selection sort:-

def selection\_sort(array):

length = len(array)

for i in range(length-1):

minIndex = i

for j in range(i+1, length):

if array[j]<array[minIndex]:

minIndex = j

array[i], array[minIndex] = array[minIndex], array[i]

return array

array =

#enter the array you want to sort

print(selection\_sort(array))

Merge sort in python:-

def merge(a1,a2):

c=[]

x=0

y=0

while(x<len(a1) and y<len(a2)):

if(a1[x]<a2[y]):

c.append(a1[x])

x+=1

else:

c.append(a2[y])

y+=1

while(x<len(a1)):

c.append(a1[x])

x+=1

while(y<len(a2)):

c.append(a2[y])

y+=1

return c

def mergesort(array):

if(len(array)==1):

return array

mid=(len(array))//2

a1=mergesort(array[:mid])

a2=mergesort(array[mid:])

return merge(a1,a2)

array=[2, 4, 1, 8, 6, 8]#enter the array you want to sort

print(mergesort(array))