**1)Bubble Sort**

Program:

public class Bubblesort {

public static void main(String[] args) {

int arr[] = { 64, 34, 25, 12, 22, 11, 90 };

Bubblesort obj=new Bubblesort();

System.out.println(Arrays.toString(obj.sort(arr)));

}

int[] sort(int[] arr){

boolean swapped;

for(int i=0;i<arr.length;i++){

swapped=false;

for(int j=0;j<arr.length-1-i;j++){

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

int temp=arr[j];

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

arr[j+1]=temp;

swapped=true;

}

}

if(swapped==false){

break;

}

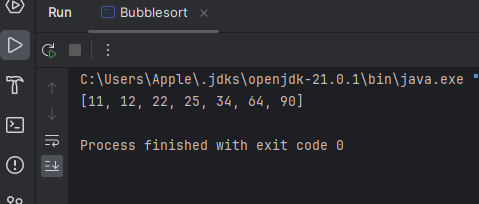
}

return arr;

}

}

Output:



TimeComplexity:O(n\*2)

SpaceComplexity:O(1)

**2)First NonRepeating Character:**

import java.util.HashMap;

public class Nonrepeating {

public static void main(String[] args) {

Nonrepeating obj=new Nonrepeating();

String s = "geeksforgeeks";

System.out.println(obj.nonRepeatingChar(s));

}

static char nonRepeatingChar(String s) {

// Your code here

HashMap<Character,Integer> map=new HashMap<>();

for(char c:s.toCharArray()){

map.put(c,map.getOrDefault(c,0)+1);

}

for(char c:s.toCharArray()){

if(map.get(c)==1){

return c;

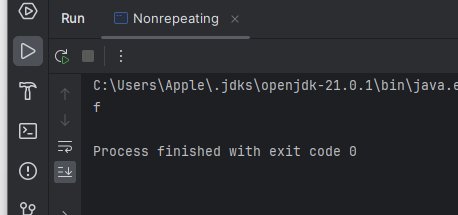
}

}

return '$';

}

}



TimeComplexity:O(n)

SpaceComplexity:O(n)

**3)Quick Sort**

import java.util.Arrays;

public class Quicksort {

public static void main(String[] args) {

Quicksort obj=new Quicksort();

int[] arr = {10, 7, 8, 9, 1, 5};

int low=0;

int high= arr.length-1;

System.out.println(Arrays.toString(obj.quickSort(arr, low, high)));

}

int part(int[] arr,int low,int high){

int pivot=arr[low];

int i=low;

int j=high;

while(i<j){

while(arr[i]<=pivot && i<high ){

i++;

}

while (arr[j] > pivot && j > low) {

j--;

}

if(i<j){

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

int temp=arr[low];

arr[low]=arr[j];

arr[j]=temp;

return j;

}

int[] quickSort(int arr[], int low, int high) {

// code here

if(low<high){

int partition=part(arr,low,high);

quickSort(arr,low,partition-1);

quickSort(arr,partition+1,high);

}

return arr;

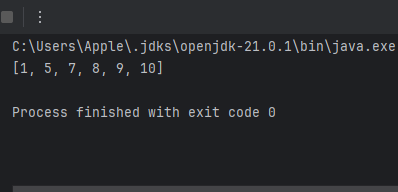
}

}

TimeComplexity:O(n log n)

SpaceComplexity:O(1)

Output:



**4)Kth largest Element**

Program:

import java.util.ArrayList;

import java.util.List;

import java.util.PriorityQueue;

import java.util.\*;

public class KthLargestElement {

public static void main(String[] args) {

KthLargestElement obj=new KthLargestElement();

int[] arr = { 10, 5, 4, 3, 48, 6, 2, 33, 53, 10 };

int K = 4;

System.out.println(obj.kLargest(arr,K));

}

List<Integer> kLargest(int arr[], int k) {

// write code here

ArrayList<Integer> list=new ArrayList<>();

PriorityQueue<Integer> maxheap=new PriorityQueue<>(Collections.reverseOrder());

for(int i:arr){

maxheap.add(i);

}

for(int j=0;j<k;j++){

list.add(maxheap.poll());

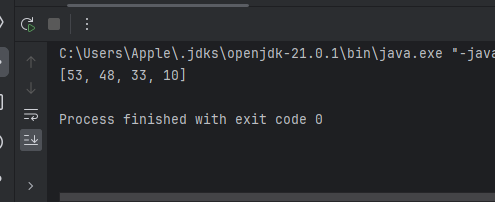
}

return list;

}

}

Output:



TimeComplexity:O(n log n)

SpaceComplexity:O(n)

**5)Form Largest Number**

import java.util.Arrays;

import java.util.Comparator;

public class largestElement {

public static void main(String[] args) {

largestElement obj=new largestElement();

int[] arr={3, 4, 6, 5, 9};

String res=obj.printLargest(arr);

System.out.println(res);

}

String printLargest(int[] arr) {

// code here

String[] str=new String[arr.length];

for(int i=0;i<arr.length;i++){

str[i]=String.valueOf(arr[i]);

}

Comparator<String> mycomp=(x,y)->(y+x).compareTo(x+y);

Arrays.sort(str,mycomp);

if(str[0].equals(0)){

return "0";

}

StringBuilder str1=new StringBuilder();

for(String i:str){

str1.append(i);

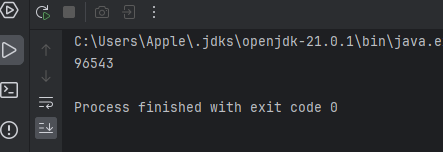
}

return str1.toString();

}

}

Output:



TimeComplexity:O(n log n)

SpaceComplexity:O(n)