**SYED NADEEM G  
SDE Practice Day-6**

(18/11/2024)

1. **Bubble Sort**

**Code:**

import java.util.\*;

public class Practice {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

int n = sc.nextInt();

int[] arr = new int[n];

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

arr[i] = sc.nextInt();

}

*bubbleSort*(arr);

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

System.***out***.print(arr[i]+" ");

}

}

public static void bubbleSort(int arr[]) {

int n = arr.length;

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

boolean swapped = false;

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

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

int temp = arr[j];

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

arr[j+1] = temp;

swapped = true;

}

}

if(!swapped){

break;

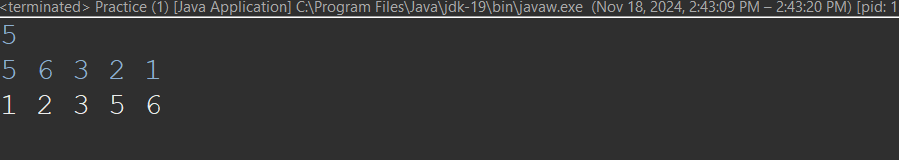
}

}

}

}

**Output:**



Time Complexity: O(n^2)

1. **Quick Sort**

**Code:**

import java.util.Scanner;

public class Practice {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

// Input array size

System.***out***.print("Enter the number of elements in the array: ");

int n = scanner.nextInt();

// Input array elements

int[] arr = new int[n];

System.***out***.println("Enter the elements of the array:");

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

arr[i] = scanner.nextInt();

}

// Perform quick sort

*quickSort*(arr, 0, n - 1);

// Output sorted array

System.***out***.println("Sorted array:");

for (int num : arr) {

System.***out***.print(num + " ");

}

}

// Function to sort an array using quick sort algorithm.

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

if (low < high) {

int pivotIndex = *partition*(arr, low, high);

*quickSort*(arr, low, pivotIndex - 1);

*quickSort*(arr, pivotIndex + 1, high);

}

}

static int partition(int arr[], int low, int high) {

int pivot = arr[high];

int indexForPlacingSmallerNumbers = low - 1;

for (int j = low; j < high; j++) {

if (arr[j] < pivot) {

indexForPlacingSmallerNumbers++;

*swap*(arr, indexForPlacingSmallerNumbers, j);

}

}

*swap*(arr, indexForPlacingSmallerNumbers + 1, high);

return indexForPlacingSmallerNumbers + 1;

}

static void swap(int[] arr, int i, int j) {

int temp = arr[i];

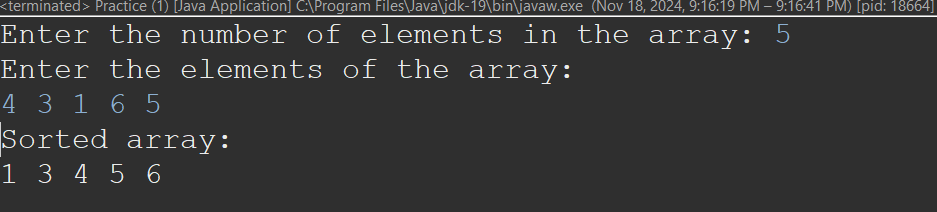
arr[i] = arr[j];

arr[j] = temp;

}

}

**Output:**



Time Complexity: O(n log n)

Space Complexity: O(1)

1. **Non-Repeating Characters**

**Code:**

import java.util.Scanner;

public class Practice {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

// Input string

System.***out***.print("Enter a string: ");

String s = scanner.nextLine();

// Find the first non-repeating character

char result = *nonRepeatingChar*(s);

// Output the result

if (result != '$') {

System.***out***.println("The first non-repeating character is: " + result);

} else {

System.***out***.println("No non-repeating character found.");

}

}

// Function to find the first non-repeating character

static char nonRepeatingChar(String s) {

int[] freq = new int[26]; // Array to store frequency of each character (a-z)

// Count frequency of each character

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

freq[c - 'a']++;

}

// Find the first character with frequency 1

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

if (freq[c - 'a'] == 1) {

return c;

}

}

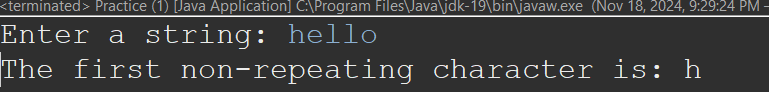
// Return '$' if no non-repeating character is found

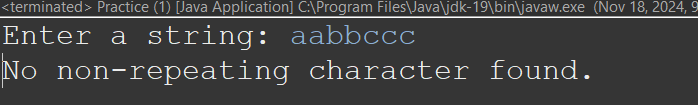
return '$';

}

}

**Output:**





Time Complexity: O(n)

1. **Kth Largest Number**

**Code:**

import java.util.\*;

class Practice {

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

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

PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> (b - a));

for (int i : arr) {

pq.add(i);

}

while (!pq.isEmpty() && k-- > 0) {

ls.add(pq.poll());

}

return ls;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] arr = new int[n];

System.***out***.println("Enter the elements of the array:");

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

arr[i] = scanner.nextInt();

}

System.***out***.print("Enter the value of k: ");

int k = scanner.nextInt();

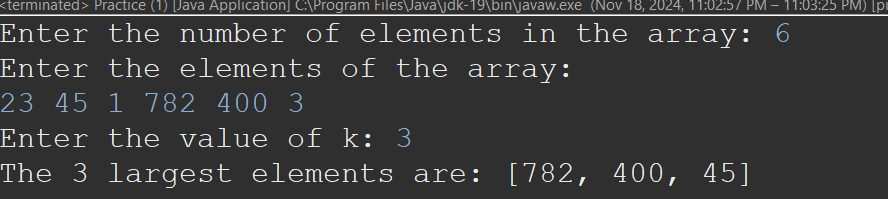
List<Integer> result = *kLargest*(arr, k);

System.***out***.println("The " + k + " largest elements are: " + result);

scanner.close();

}

}

**Output:** 

Time Complexity: O(n)

1. **Form Largest Number**

**Code:**

import java.util.Arrays;

import java.util.Scanner;

class Practice {

String printLargest(int[] arr) {

int n = arr.length;

String[] res = new String[n];

int ind = 0;

for(int i:arr) res[ind++] = i + "";

Arrays.*sort*(res, (X, Y) -> (Y + X).compareTo(X + Y));

if(res[0].equals("0")) return "0";

StringBuilder sb = new StringBuilder();

for(String s:res) sb.append(s);

return sb.toString();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter the number of elements: ");

int n = sc.nextInt();

int[] arr = new int[n];

System.***out***.println("Enter the elements:");

for(int i = 0; i < n; i++) arr[i] = sc.nextInt();

Practice solution = new Practice();

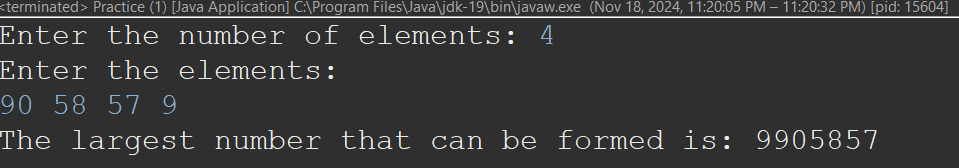
String result = solution.printLargest(arr);

System.***out***.println("The largest number that can be formed is: " + result);

sc.close();

}

}

Ouput: 

Time Complexity: O(n)