Name:Sudheep S Dept:CSE Reg No:22CS171 Date:18.11.24

1.Bubble Sort

class Main {

public static void main(String[] args) {

int i, j, temp;

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

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

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

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

temp = arr[j];

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

arr[j + 1] = temp;

}

}

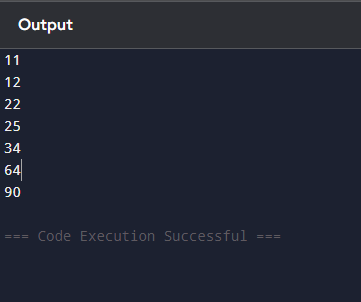
}

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

System.out.println(arr[i]);

}

}

}  
  
  
  
Time Complexity:O(n^2)

Space Complexity:O(1)  
  
2.Quick Sort  
  
import java.util.Stack;

class Main {

public static void main(String[] args) {

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

int n = arr.length;

int low = 0;

int high = n - 1;

Stack<int[]> stack = new Stack<>();

stack.push(new int[]{low, high});

while (!stack.isEmpty()) {

int[] range = stack.pop();

low = range[0];

high = range[1];

if (low < high) {

int pivot = arr[high];

int i = low - 1;

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

if (arr[j] < pivot) {

i++;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

int pi = i + 1;

stack.push(new int[]{low, pi - 1});

stack.push(new int[]{pi + 1, high});

}

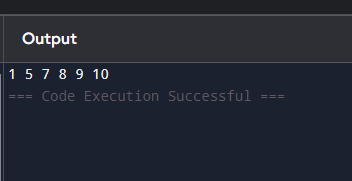
}

for (int val : arr) {

System.out.print(val + " ");

}

}

}  
  


3.Non Repeating Character

class Main {

public static void main(String[] args) {

String s = "racecar";

int n = s.length();

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

boolean found = false;

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

if (i != j && s.charAt(i) == s.charAt(j)) {

found = true;

break;

}

}

if (!found) {

System.out.println(s.charAt(i));

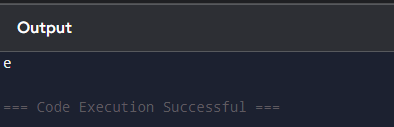
return;

}

}

System.out.println('$');

}

}  
  
Time Complexity:O(n^2)

Space Complexity:O(1)

4.Edit Distance

public class Main {

public static int editDist(String s1, String s2) {

int m = s1.length();

int n = s2.length();

int[] prev = new int[n + 1];

int[] curr = new int[n + 1];

for (int j = 0; j <= n; j++)

prev[j] = j;

for (int i = 1; i <= m; i++) {

curr[0] = i;

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

if (s1.charAt(i - 1) == s2.charAt(j - 1))

curr[j] = prev[j - 1];

else

curr[j] = 1 + Math.min(curr[j - 1], Math.min(prev[j], prev[j - 1]));

}

int[] temp = prev;

prev = curr;

curr = temp;

}

return prev[n];

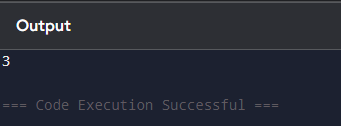
}

public static void main(String[] args) {

String s1 = "GEEXSFRGEEKKS", s2 = "GEEKSFORGEEKS";

System.out.println(editDist(s1, s2));

}

}  


Time Complexity: O(m x n)

Space Complexity: O(n)

5.K largest Element

import java.util.\*;

public class Main {

public static void main(String[] args) {

int[] arr = {1, 23, 12, 9, 30, 2, 50};

int k = 3;

PriorityQueue<Integer> minHeap = new PriorityQueue<>(k);

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

minHeap.add(arr[i]);

}

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

if (arr[i] > minHeap.peek()) {

minHeap.poll();

minHeap.add(arr[i]);

}

}

List<Integer> res = new ArrayList<>();

while (!minHeap.isEmpty()) {

res.add(minHeap.poll());

}

Collections.reverse(res);

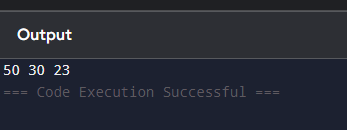
for (int ele : res) {

System.out.print(ele + " ");

}

}

}



Time Complexity: O(n \* log(k))

Space Complexity: O(k)

6.Form the largest number

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static String largestNumber(String[] arr)

{

Comparator<String> myCompare

= (X, Y) -> (X + Y).compareTo(Y + X);

Arrays.sort(arr, myCompare.reversed());

if (arr[0].equals("0")) {

return "0";

}

StringBuilder result = new StringBuilder();

for (String num : arr) {

result.append(num);

}

return result.toString();

}

public static void main(String[] args)

{

String[] arr1 = { "3", "30", "34", "5", "9" };

System.out.println(

largestNumber(arr1));

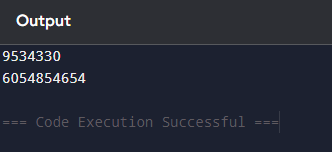
String[] arr2 = { "54", "546", "548", "60" };

System.out.println(

largestNumber(arr2));

}

}



Time Complexity: O(n log n)

Space Complexity: O(1)