1. How do you check if two strings are a rotation of each other? import java.io.\*;

import java.util.\*; class Rotation {

static boolean checkString(String s1, String s2,int indexFound, int Size)

{

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

if (s1.charAt(i)

!= s2.charAt((indexFound + i) % Size)) return false;

}

return true;

}

public static void main(String args[])

{

String s1 = "abcd"; String s2 = "cdab";

if (s1.length() != s2.length()) {

System.out.println("s2 is not a rotation on s1");

}

else {

ArrayList<Integer> indexes = new ArrayList<Integer>(); int Size = s1.length();

char firstChar = s1.charAt(0); for (int i = 0; i < Size; i++) {

if (s2.charAt(i) == firstChar) { indexes.add(i);

}

}

boolean isRotation = false; for (int idx : indexes) {

isRotation = checkString(s1, s2, idx, Size); if (isRotation)

break;

}

if (isRotation)

System.out.println("Strings are rotations of each other");

else

System.out.println("Strings are not rotations of each other");

}

}

}

1. How do you check if a given string is a palindrome?

public class Palindrome

{

public static void main(String[] args) { String string = "Sivapriya";

boolean flag = true;

string = string.toLowerCase();

for(int i = 0; i < string.length()/2; i++){ if(string.charAt(i) != string.charAt(string.length()-i-1)){

flag = false; break;

}

}

if(flag)

System.out.println("Given string is palindrome"); else

System.out.println("Given string is not a palindrome");

}

}

1. How is a binary search tree implemented?

public Node search(Node root, int key)

{

if (root==null || root.key==key) return root;

if (root.key < key)

return search(root.right, key); return search(root.left, key);

}

1. How do you perform preorder traversal in a given binary tree? class Node {

int data;

Node left, right;

Node(int d)

{

data = d;

left = right = null;

}

}

class Index {

int index = 0;

}

class BinaryTree {

Index index = new Index();

Node constructTreeUtil(int pre[], Index preIndex,int low, int high, int size)

{

if (preIndex.index >= size || low > high) { return null;

}

Node root = new Node(pre[preIndex.index]); preIndex.index = preIndex.index + 1;

if (low == high) {

return root;

}

int i;

for (i = low; i <= high; ++i) {

if (pre[i] > root.data) { break;

}

}

root.left = constructTreeUtil(

pre, preIndex, preIndex.index, i - 1, size); root.right = constructTreeUtil(pre, preIndex, i,high, size); return root;

}

Node constructTree(int pre[], int size)

{

return constructTreeUtil(pre, index, 0, size - 1,size);

}

void printInorder(Node node)

{

if (node == null) {

return;

}

printInorder(node.left); System.out.print(node.data + " "); printInorder(node.right);

}

public static void main(String[] args)

{

BinaryTree tree = new BinaryTree();

int pre[] = new int[] { 10, 5, 1, 7, 40, 50 }; int size = pre.length;

Node root = tree.constructTree(pre, size); System.out.println("Inorder traversal of the constructed tree is "); tree.printInorder(root);

}

}

1. How do you traverse a given binary tree in preorder without recursion? import java.util.Stack;

class Node { int data;

Node left, right;

Node(int item)

{

data = item;

left = right = null;

}

}

class BinaryTree { Node root;

void iterativePreorder()

{

iterativePreorder(root);

}

void iterativePreorder(Node node)

{

if (node == null) {

return;

}

Stack<Node> nodeStack = new Stack<Node>(); nodeStack.push(root);

while (nodeStack.empty() == false) {

Node mynode = nodeStack.peek();

System.out.print(mynode.data + " "); nodeStack.pop();

if (mynode.right != null) {

nodeStack.push(mynode.right);

}

if (mynode.left != null) {

nodeStack.push(mynode.left);

}

}

}

public static void main(String args[])

{

BinaryTree tree = new BinaryTree(); tree.root = new Node(10); tree.root.left = new Node(8); tree.root.right = new Node(2); tree.root.left.left = new Node(3); tree.root.left.right = new Node(5); tree.root.right.left = new Node(2); tree.iterativePreorder();

}

}

1. How do you perform an inorder traversal in a given binary tree? class Node {

int key;

Node left, right; public Node(int item)

{

key = item;

left = right = null;

}

}

class BinaryTree {

Node root;

BinaryTree() { root = null; } void printInorder(Node node)

{

if (node == null)

return; printInorder(node.left); System.out.print(node.key + " "); printInorder(node.right);

}

void printInorder() { printInorder(root); public static void main(String[] args)

{

BinaryTree tree = new BinaryTree(); tree.root = new Node(1); tree.root.left = new Node(2); tree.root.right = new Node(3); tree.root.left.left = new Node(4); tree.root.left.right = new Node(5);

System.out.println("\nInorder traversal of binary tree is "); tree.printInorder();

}

}

1. How do you print all nodes of a given binary tree using inorder traversal without recursion? import java.util.Stack;

class Node

{

int data;

Node left, right; public Node(int item)

{

data = item;

left = right = null;

}

}

class BinaryTree

{

Node root; void inorder()

{

if (root == null)

return;

Stack<Node> s = new Stack<Node>(); Node curr = root;

while (curr != null || s.size() > 0)

{

while (curr != null)

{

s.push(curr); curr = curr.left;

}

curr = s.pop(); System.out.print(curr.data + " "); curr = curr.right;

}

}

public static void main(String args[])

{

BinaryTree tree = new BinaryTree();

tree.root = new Node(1); tree.root.left = new Node(2); tree.root.right = new Node(3); tree.root.left.left = new Node(4); tree.root.left.right = new Node(5); tree.inorder();

}

}

1. How do you implement a postorder traversal algorithm? import java.util.ArrayList;

import java.util.Stack; class Node {

int data;

Node left, right;

Node(int item)

{

data = item; left = right;

}

}

class BinaryTree {

Node root;

ArrayList<Integer> list = new ArrayList<Integer>(); ArrayList<Integer> postOrderIterative(Node node)

{

Stack<Node> S = new Stack<Node>(); if (node == null)

return list; S.push(node);

Node prev = null; while (!S.isEmpty()) {

Node current = S.peek();

if (prev == null || prev.left == current || prev.right == current) { if (current.left != null)

S.push(current.left); else if (current.right != null)

S.push(current.right);

else {

S.pop(); list.add(current.data);

}

}

else if (current.left == prev) {

if (current.right != null)

S.push(current.right);

else {

S.pop(); list.add(current.data);

}

}

else if (current.right == prev) { S.pop(); list.add(current.data);

}

prev = current;

}

return list;

}

public static void main(String args[])

{

BinaryTree tree = new BinaryTree(); tree.root = new Node(1); tree.root.left = new Node(2); tree.root.right = new Node(3); tree.root.left.left = new Node(4); tree.root.left.right = new Node(5); tree.root.right.left = new Node(6); tree.root.right.right = new Node(7);

ArrayList<Integer> mylist = tree.postOrderIterative(tree.root); System.out.println("Post order traversal of binary tree is :"); System.out.println(mylist);

}

}

1. How do you traverse a binary tree in postorder traversal without recursion? How are all leaves of a binary search tree printed?

public void postOrderWithoutRecursion()

{

Stack<TreeNode> nodes = new Stack<>(); nodes.push(root);

while (!nodes.isEmpty())

{

TreeNode current = nodes.peek(); if (current.isLeaf())

{

TreeNode node = nodes.pop(); System.out.printf("%s ", node.data);

}

else

{

if (current.right != null)

{

nodes.push(current.right); current.right = null;

}

if (current.left != null)

{

nodes.push(current.left); current.left = null;

}

}

}

}

1. How do you count the number of leaf nodes in a given binary tree? How do you perform a binary search in a given array?

class Node

{

int data;

Node left, right; public Node(int item)

{

data = item;

left = right = null;

}

}

public class BinaryTree

{

Node root;

int getLeafCount()

{

return getLeafCount(root);

}

int getLeafCount(Node node)

{

if (node == null)

return 0;

if (node.left == null && node.right == null) return 1;

else

return getLeafCount(node.left) + getLeafCount(node.right);

}

public static void main(String args[])

{

BinaryTree tree = new BinaryTree(); tree.root = new Node(1); tree.root.left = new Node(2); tree.root.right = new Node(3); tree.root.left.left = new Node(4); tree.root.left.right = new Node(5);

System.out.println("The leaf count of binary tree is : "+ tree.getLeafCount());

}

}

1. How is a bubble sort algorithm implemented? class BubbleSort {

void bubbleSort(int arr[])

{

int n = arr.length;

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

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;

}

}

void printArray(int arr[])

{

int n = arr.length;

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

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

}

public static void main(String args[])

{

BubbleSort ob = new BubbleSort(); int arr[] = { 64, 34, 25, 12, 22, 11, 90 };

ob.bubbleSort(arr); System.out.println("Sorted array"); ob.printArray(arr);

}

}

1. How is an iterative quicksort algorithm implemented? How do you implement an insertion sort algorithm?

import java.util.\*; class QuickSort {

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

{

int pivot = arr[high]; int i = (low - 1);

for (int j = low; j <= high - 1; 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; return i + 1;

static void qSort(int arr[], int low, int high)

{

if (low < high) {

int pi = partition(arr, low, high); qSort(arr, low, pi - 1); qSort(arr, pi + 1, high);

}

}

public static void main(String args[])

{

int n = 5;

int arr[] = { 4, 2, 6, 9, 2 };

qSort(arr, 0, n - 1);

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

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

}

}

}

1. How is a merge sort algorithm implemented? How do you implement a bucket sort algorithm? import java.util.\*;

import java.util.Collections;

class BucketSort{

static void bucketSort(float arr[], int n)

{

if (n <= 0)

return; @SuppressWarnings("unchecked") Vector<Float>[] buckets = new Vector[n];

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

buckets[i] = new Vector<Float>();

}

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

float idx = arr[i] \* n; buckets[(int)idx].add(arr[i]);

}

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

Collections.sort(buckets[i]);

}

int index = 0;

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

for (int j = 0; j < buckets[i].size(); j++) { arr[index++] = buckets[i].get(j);

}

}

}

public static void main(String args[])

{

float arr[] = { (float)0.897, (float)0.565,(float)0.656, (float)0.1234,(float)0.665,

(float)0.3434 };

int n = arr.length;

bucketSort(arr, n); System.out.println("Sorted array is "); for (float el : arr) {

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

}

}

}

class MergeSort {

void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1; int n2 = r - m;

int L[] = new int[n1]; int R[] = new int[n2]; for (int i = 0; i < n1; ++i)

L[i] = arr[l + i]; for (int j = 0; j < n2; ++j)

R[j] = arr[m + 1 + j] int i = 0, j = 0;

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i]; i++;

}

else {

arr[k] = R[j]; j++;

} k++;

}

while (i < n1) {

arr[k] = L[i]; i++;

k++;

}

while (j < n2) {

arr[k] = R[j]; j++;

k++;

}

}

void sort(int arr[], int l, int r)

{

if (l < r) {

int m = l + (r - l) / 2; sort(arr, l, m);

sort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

static void printArray(int arr[])

{

int n = arr.length;

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

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

}

public static void main(String args[])

{

int arr[] = { 12, 11, 13, 5, 6, 7 };

System.out.println("Given Array");

printArray(arr);

MergeSort ob = new MergeSort(); ob.sort(arr, 0, arr.length - 1); System.out.println("\nSorted array"); printArray(arr);

}

}

1. How do you implement a counting sort algorithm? How is a radix sort algorithm implemented? import java.io.\*;

import java.util.\*; class Radix {

static int getMax(int arr[], int n)

{

int mx = arr[0];

for (int i = 1; i < n; i++)

if (arr[i] > mx)

mx = arr[i];

return mx;

}

static void countSort(int arr[], int n, int exp)

{

int output[] = new int[n]; int i;

int count[] = new int[10]; Arrays.fill(count, 0);

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

count[(arr[i] / exp) % 10]++; for (i = 1; i < 10; i++)

count[i] += count[i - 1]; for (i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

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

arr[i] = output[i];

}

static void radixsort(int arr[], int n)

{

int m = getMax(arr, n);

for (int exp = 1; m / exp > 0; exp \*= 10) countSort(arr, n, exp);

}

static void print(int arr[], int n)

{

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

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

}

public static void main(String[] args)

{

int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };

int n = arr.length; radixsort(arr, n); print(arr, n);

}

}

class CountingSort {

void sort(char arr[])

{

int n = arr.length;

int count[] = new int[256]; for (int i = 0; i < 256; ++i)

count[i] = 0;

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

++count[arr[i]]; for (int i = 1; i <= 255; ++i)

count[i] += count[i - 1]; for (int i = n - 1; i >= 0; i--) {

output[count[arr[i]] - 1] = arr[i];

--count[arr[i]];

}

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

arr[i] = output[i];

}

public static void main(String args[])

{

CountingSort ob = new CountingSort();

char arr[] = { 'g', 'e', 'e', 'k', 's', 'f', 'o','r', 'g', 'e', 'e', 'k', 's' }; ob.sort(arr);

System.out.print("Sorted character array is "); for (int i = 0; i < arr.length; ++i)

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

}

}

1. How do you swap two numbers without using the third variable? How do you check if two rectangles overlap with each other?

class Rectangle { static class Point {

int x, y;

}

static boolean doOverlap(Point l1, Point r1, Point l2, Point r2) {

if (l1.x == r1.x || l1.y == r1.y || r2.x == l2.x || l2.y == r2.y) return false;

if (l1.x > r2.x || l2.x > r1.x) { return false;

}

if (r1.y > l2.y || r2.y > l1.y) { return false;

}

return true;

}

public static void main(String[] args) {

Point l1 = new Point(),r1 = new Point(),

l2 = new Point(),r2 = new Point(); l1.x=0;l1.y=10; r1.x=10;r1.y=0;

l2.x=5;l2.y=5; r2.x=15;r2.y=0;

if (doOverlap(l1, r1, l2, r2)) { System.out.println("Rectangles Overlap");

} else {

System.out.println("Rectangles Don't Overlap");

}

}

}

1. How do you design a vending machine?
2. How can you find the first non-repeated character in a word? import java.io.\*;

class Charater{

static final int NO\_OF\_CHARS = 256;

static char count[] = new char[NO\_OF\_CHARS]; static void getCharCountArray(String str)

{

for (int i = 0; i < str.length(); i++) count[str.charAt(i)]++;

}

static int firstNonRepeating(String str)

{

getCharCountArray(str); int index = -1, i;

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

if (count[str.charAt(i)] == 1) { index = i;

break;

}

}

return index;

}

public static void main(String[] args)

{

String str = "geeksforgeeks";

int index = firstNonRepeating(str); System.out.println(

index == -1? "Either all characters are repeating or string "+ "is empty": "First non-repeating character is "+ str.charAt(index));

}

}

1. How can you remove duplicates from arrays? class Main {

static int removeDuplicates(int arr[], int n)

{

if (n == 0 || n == 1)

return n;

int[] temp = new int[n]; int j = 0;

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

if (arr[i] != arr[i + 1])

temp[j++] = arr[i]; temp[j++] = arr[n - 1];

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

arr[i] = temp[i];

return j;

}

public static void main(String[] args)

{

int arr[] = { 1, 2, 2, 3, 4, 4, 4, 5, 5 };

int n = arr.length;

n = removeDuplicates(arr, n); for (int i = 0; i < n; i++)

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

}

}

1. How can we check if a number is a prime number? import java.lang.\*;

import java.util.\*; class Prime{

static boolean isPrime(int n)

{

if (n <= 1)

return false; Check if number is 2 else if (n == 2)

return true; else if (n % 2 == 0)

return false;

for (int i = 3; i <= Math.sqrt(n); i += 2) { if (n % i == 0)

return false;

}

return true;

}

public static void main(String[] args)

{

if (isPrime(19))

System.out.println("true");

else

System.out.println("false");

}

}

1. How can you check if strings contain only digits? class Digits{

public static boolean onlyDigits(String str, int n)

{

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

if (str.charAt(i) < '0'

|| str.charAt(i) > '9') { return false;

}

}

return true;

}

public static void main(String args[])

{

String str = "1a234"; int len = str.length();

System.out.println(onlyDigits(str, len));

}

}