|  |  |
| --- | --- |
| Assignment 2 Solutions in Java | Ishaan mishra    1024030346 |

# Question 1

class BinarySearch {

public static int binarySearch(int arr[], int x) {

int low = 0, high = arr.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] < x)

low = mid + 1;

else

high = mid - 1;

}

return -1;

}

public static void main(String args[]) {

int arr[] = {2, 3, 4, 10, 40};

int result = binarySearch(arr, 10);

if (result == -1)

System.out.println("Element not found");

else

System.out.println("Element found at index: " + result);

}

}

# Question 2

class BubbleSort {

static 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;

}

}

}

}

public static void main(String args[]) {

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

bubbleSort(arr);

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

for (int num : arr)

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

}

}

# Question 3

// Question 3: Find Missing Number in a Sorted Array

class MissingNumber {

// Linear Time Approach

static int findMissingLinear(int arr[], int n) {

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

boolean found = false;

for (int j : arr) {

if (j == i) {

found = true;

break;

}

}

if (!found)

return i;

}

return -1;

}

// Binary Search Approach

static int findMissingBinary(int arr[], int n) {

int low = 0, high = n - 2;

while (low <= high) {

int mid = (low + high) / 2;

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

high = mid - 1;

else

low = mid + 1;

}

return low + 1;

}

public static void main(String args[]) {

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

int n = 5;

System.out.println("Missing Number (Linear): " + findMissingLinear(arr, n));

System.out.println("Missing Number (Binary): " + findMissingBinary(arr, n));

}

}

# Question 4

// Question 4: String Related Programs  
import java.util.\*;

class StringPrograms {

public static void main(String args[]) {

// (a) Concatenate

String s1 = "Hello";

String s2 = "World";

System.out.println("Concatenated: " + s1 + s2);

// (b) Reverse

String s = "Hello";

String rev = new StringBuilder(s).reverse().toString();

System.out.println("Reversed: " + rev);

// (c) Delete vowels

String str = "Hello World";

System.out.println("Without vowels: " + str.replaceAll("[aeiouAEIOU]", ""));

// (d) Sort strings

String arr[] = {"banana", "apple", "cherry"};

Arrays.sort(arr);

System.out.println("Sorted strings: " + Arrays.toString(arr));

// (e) Uppercase to lowercase

String upper = "HELLO";

System.out.println("Lowercase: " + upper.toLowerCase());

}

}

# Question 5

// Question 5: Efficient Storage of Special Matrices  
/\*  
Diagonal Matrix – store only n diagonal elements in 1D array.  
Tri-diagonal Matrix – store 3n-2 elements.  
Lower Triangular Matrix – store n(n+1)/2 elements.  
Upper Triangular Matrix – store n(n+1)/2 elements.  
Symmetric Matrix – store n(n+1)/2 elements (A[i][j] = A[j][i]).  
Implementation depends on index mapping formulas.  
\*/  
class SpecialMatrices {  
 public static void main(String args[]) {  
 System.out.println("Special matrix storage formulas implemented conceptually.");  
 }  
}

# Question 6

// Question 6: Sparse Matrix Operations using Triplet Representation  
class SparseMatrixSimple {

import java.util.\*;

class SparseMatrixTriplet {

// Convert 2D matrix to Triplet

static int[][] toTriplet(int mat[][]) {

int rows = mat.length, cols = mat[0].length, count = 0;

// Count non-zeros

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

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

if (mat[i][j] != 0) count++;

int triplet[][] = new int[count + 1][3];

triplet[0][0] = rows; // total rows

triplet[0][1] = cols; // total cols

triplet[0][2] = count; // non-zero count

int k = 1;

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

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

if (mat[i][j] != 0) {

triplet[k][0] = i;

triplet[k][1] = j;

triplet[k][2] = mat[i][j];

k++;

}

}

}

return triplet;

}

// Transpose

static int[][] transpose(int t[][]) {

int rows = t[0][1], cols = t[0][0], count = t[0][2];

int trans[][] = new int[count + 1][3];

trans[0][0] = rows;

trans[0][1] = cols;

trans[0][2] = count;

int k = 1;

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

trans[k][0] = t[i][1]; // swap row <-> col

trans[k][1] = t[i][0];

trans[k][2] = t[i][2];

k++;

}

return trans;

}

// Addition of two matrices (triplets must have same dimension)

static int[][] add(int A[][], int B[][]) {

if (A[0][0] != B[0][0] || A[0][1] != B[0][1]) {

System.out.println("Addition not possible (different sizes)");

return new int[0][0];

}

int i = 1, j = 1, k = 1;

int C[][] = new int[A[0][2] + B[0][2] + 1][3];

C[0][0] = A[0][0]; C[0][1] = A[0][1];

while (i <= A[0][2] && j <= B[0][2]) {

if (A[i][0] < B[j][0] || (A[i][0] == B[j][0] && A[i][1] < B[j][1])) {

C[k++] = A[i++];

} else if (B[j][0] < A[i][0] || (A[i][0] == B[j][0] && B[j][1] < A[i][1])) {

C[k++] = B[j++];

} else { // same position

int sum = A[i][2] + B[j][2];

if (sum != 0) {

C[k][0] = A[i][0];

C[k][1] = A[i][1];

C[k][2] = sum;

k++;

}

i++; j++;

}

}

while (i <= A[0][2]) C[k++] = A[i++];

while (j <= B[0][2]) C[k++] = B[j++];

C[0][2] = k - 1;

return Arrays.copyOf(C, k);

}

// Multiplication of two matrices

static int[][] multiply(int A[][], int B[][]) {

if (A[0][1] != B[0][0]) {

System.out.println("Multiplication not possible (invalid sizes)");

return new int[0][0];

}

int rows = A[0][0], cols = B[0][1];

Map<String, Integer> map = new HashMap<>();

for (int i = 1; i <= A[0][2]; i++) {

for (int j = 1; j <= B[0][2]; j++) {

if (A[i][1] == B[j][0]) { // col of A == row of B

String key = A[i][0] + "," + B[j][1];

int val = A[i][2] \* B[j][2];

map.put(key, map.getOrDefault(key, 0) + val);

}

}

}

int count = 0;

for (int v : map.values()) if (v != 0) count++;

int C[][] = new int[count + 1][3];

C[0][0] = rows; C[0][1] = cols; C[0][2] = count;

int k = 1;

for (String key : map.keySet()) {

int val = map.get(key);

if (val != 0) {

String parts[] = key.split(",");

C[k][0] = Integer.parseInt(parts[0]);

C[k][1] = Integer.parseInt(parts[1]);

C[k][2] = val;

k++;

}

}

return C;

}

// Print triplet

static void print(int t[][]) {

System.out.println("Row Col Val");

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

System.out.println(t[i][0] + " " + t[i][1] + " " + t[i][2]);

}

}

public static void main(String args[]) {

int A[][] = {

{0, 0, 5},

{0, 2, 8},

{1, 1, 3}

};

int B[][] = {

{0, 1, 6},

{1, 0, 4},

{2, 2, 7}

};

int TA[][] = toTriplet(A);

int TB[][] = toTriplet(B);

System.out.println("Matrix A (Triplet):");

print(TA);

System.out.println("\nTranspose of A:");

print(transpose(TA));

System.out.println("\nA + B:");

print(add(TA, TB));

System.out.println("\nA x B:");

print(multiply(TA, TB));

}

}

# Question 7

// Question 7: Count Inversions in Array  
class Inversions {  
 static int countInversions(int arr[]) {  
 int count = 0;  
 int n = arr.length;  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 if (arr[i] > arr[j])  
 count++;  
 }  
 }  
 return count;  
 }  
  
 public static void main(String args[]) {  
 int arr[] = {1, 20, 6, 4, 5};  
 System.out.println("Number of inversions: " + countInversions(arr));  
 }  
}

# Question 8

// Question 8: Count Distinct Elements in Array

import java.util.\*;  
class DistinctElements {  
 static int countDistinct(int arr[]) {  
 Set<Integer> set = new HashSet<>();  
 for (int num : arr)  
 set.add(num);  
 return set.size();  
 }  
  
 public static void main(String args[]) {  
 int arr[] = {1, 2, 3, 2, 1};  
 System.out.println("Total distinct elements: " + countDistinct(arr));  
 }  
}