Important questions:

**Artificial Intelligence**

**1. Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.**

import java.util.\*;

public class GraphTraversal {

static void dfs(List<Integer>[] graph, int node, boolean[] visited) {

visited[node] = true;

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

for (int neighbor : graph[node]) {

if (!visited[neighbor]) {

dfs(graph, neighbor, visited);

}

}

}

static void bfs(List<Integer>[] graph, int start) {

Queue<Integer> queue = new LinkedList<>();

boolean[] visited = new boolean[graph.length];

queue.offer(start);

visited[start] = true;

while (!queue.isEmpty()) {

int node = queue.poll();

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

for (int neighbor : graph[node]) {

if (!visited[neighbor]) {

queue.offer(neighbor);

visited[neighbor] = true;

}

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

List<Integer>[] graph = new List[n];

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

graph[i] = new ArrayList<>();

}

System.out.println("Enter the edges (node1 node2), enter -1 to stop:");

while (true) {

int node1 = scanner.nextInt();

if (node1 == -1) break;

int node2 = scanner.nextInt();

graph[node1].add(node2);

}

System.out.println("DFS traversal:");

boolean[] visitedDFS = new boolean[n];

dfs(graph, 0, visitedDFS); // Start DFS from node 0

System.out.println("\nBFS traversal:");

bfs(graph, 0); // Start BFS from node 0

scanner.close();

}

}

/\* Output

Enter the number of nodes: 6

Enter the edges (node1 node2), enter -1 to stop:

0 1

0 2

1 3

1 4

2 4

3 5

4 5

-1

DFS traversal:

0 1 3 5 4 2

BFS traversal:

0 1 2 3 4 5

\*/

**2. Implement A star Algorithm for any game search problem.**

import java.io.\*;

import java.util.\*;

class AStar {

public static int N1 = 3;

public static class Node {

// storing the parent node of the current node

// will help in the tracing path when the answer is found

Node parent1;

int m[][] = new int[N1][N1];// storing the matrix

int x1, y1;// storing the blank tile coordinates

int cost1;// storing the number of misplaced tiles

int level1;// storing the number of moves so far

}

// Method for printing N1 x N1 matrix

public static void printMatrix(int m[][]) {

for (int i1 = 0; i1 < N1; i1++) {

for (int j1 = 0; j1 < N1; j1++) {

System.out.print(m[i1][j1] + " ");

}

System.out.println("");

}

}

// Method for allocating a new node

public static Node newNode(int m[][], int x1, int y1,

int newX1, int newY1, int level1,

Node parent1) {

Node node1 = new Node();

node1.parent1 = parent1;// setting pointer from the path to the root

// copying data from the parent node to the current node

node1.m = new int[N1][N1];

for (int i1 = 0; i1 < N1; i1++) {

for (int j1 = 0; j1 < N1; j1++) {

node1.m[i1][j1] = m[i1][j1];

}

}

// moving tile by 1 position

int temp1 = node1.m[x1][y1];

node1.m[x1][y1] = node1.m[newX1][newY1];

node1.m[newX1][newY1] = temp1;

node1.cost1 = Integer.MAX\_VALUE;// setting number of misplaced tiles

node1.level1 = level1;// setting number of moves so far

// updating new blank tile coordinates

node1.x1 = newX1;

node1.y1 = newY1;

return node1;

}

// bottom value, left value, top value, the right value

public static int row1[] = { 1, 0, -1, 0 };

public static int col1[] = { 0, -1, 0, 1 };

// Method for calculating the number of misplaced tiles

// that is the number of non-blank tiles not in their goal position

public static int calculateCost(int initialM[][], int finalM[][]) {

int count1 = 0;

for (int i1 = 0; i1 < N1; i1++)

for (int j1 = 0; j1 < N1; j1++)

if (initialM[i1][j1] != 0 && initialM[i1][j1] != finalM[i1][j1])

count1++;

return count1;

}

// method for checking if (x1, y1) is a valid matrix coordinate or not

public static int isSafe(int x1, int y1) {

return (x1 >= 0 && x1 < N1 && y1 >= 0 && y1 < N1) ? 1 : 0;

}

// printing path from a root node to the destination node

public static void printPath(Node root1) {

if (root1 == null) {

return;

}

printPath(root1.parent1);

printMatrix(root1.m);

System.out.println("");

}

// Comparing instances to be used to order the heap

public static class comp implements Comparator<Node> {

// @Overriding

public int compare(Node lhs1, Node rhs1) {

return (lhs1.cost1 + lhs1.level1) > (rhs1.cost1 + rhs1.level1) ? 1 : -1;

}

}

// Method for solving N1\*N1 - 1 puzzle algorithm using

// Branch and Bound method. x1 and y1 are blank tile coordinates

// in starting state

public static void solve(int initialM[][], int x1,

int y1, int finalM[][]) {

// Creating a priority queue for storing the live nodes of the search tree

PriorityQueue<Node> pq1 = new PriorityQueue<>(new comp());

// creating a root node and calculating its cost

Node root1 = newNode(initialM, x1, y1, x1, y1, 0, null);

root1.cost1 = calculateCost(initialM, finalM);

// Adding root1 to the list of the live nodes;

pq1.add(root1);

// Finding a live node with the least cost,

// and adding its children to the list of

// live nodes and

// finally deleting it from the list.

while (!pq1.isEmpty()) {

Node min1 = pq1.peek();// Finding a live node with the least estimated cost

pq1.poll();// The found node has been deleted from the list of live nodes

// if minimum is an answer node

if (min1.cost1 == 0) {

printPath(min1);// printing the path from the root to the destination;

return;

}

// doing for each child of minimum

// maximum 4 children for a node

for (int i1 = 0; i1 < 4; i1++) {

if (isSafe(min1.x1 + row1[i1], min1.y1 + col1[i1]) > 0) {

// creating a child node and calculating

// its cost

Node child1 = newNode(min1.m, min1.x1, min1.y1, min1.x1 + row1[i1], min1.y1 + col1[i1],

min1.level1 + 1, min1);

child1.cost1 = calculateCost(child1.m, finalM);

// Adding a child node to the list of the live nodes

pq1.add(child1);

}

}

}

}

// main code

public static void main(String args[]) {

// first configuration

// Value 0 is used for the null space

int initialM[][] = {

{ 0, 1, 2 },

{ 4, 5, 8 },

{ 6, 7, 3 }

};

// Solvable last configuration

// Value 0 is used for the null space

int finalM[][] = {

{ 0, 3, 7 },

{ 4, 2, 5 },

{ 6, 8, 1 }

};

// Blank tile coordinates in the first

// configuration

int x1 = 1, y1 = 2;

solve(initialM, x1, y1, finalM);

}

}

// Output:

// 0 1 2

// 4 5 8

// 6 7 3

// 0 1 2

// 4 5 3

// 6 7 8

// 0 1 2

// 4 5 3

// 6 8 7

// 0 1 2

// 4 8 3

// 6 5 7

// 0 8 2

// 4 1 3

// 6 5 7

// 0 2 8

// 4 1 3

// 6 5 7

// 0 2 3

// 4 1 8

// 6 5 7

// 0 2 3

// 4 1 7

// 6 5 8

// 0 2 3

// 4 1 7

// 6 8 5

// 0 2 3

// 4 8 7

// 6 1 5

// 0 8 3

// 4 2 7

// 6 1 5

// 0 3 8

// 4 2 7

// 6 1 5

// 0 3 7

// 4 2 8

// 6 1 5

// 0 3 7

// 4 2 5

// 6 1 8

// 0 3 7

// 4 2 5

// 6 8 1

**3. Implement Greedy search algorithm for any of the following application:**

**I. Selection Sort**

public class SelectionSort {

public static void selectionSort(int[] arr) {

int n = arr.length;

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

int minIndex = i;

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

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

int temp = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

}

}

public static void main(String[] args) {

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

selectionSort(arr);

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

for (int i : arr) {

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

}

}

}

**V. Prim's Minimal Spanning Tree Algorithm**

import java.util.\*;

public class PrimsAlgorithm {

static class Edge {

int source, dest, weight;

Edge(int source, int dest, int weight) {

this.source = source;

this.dest = dest;

this.weight = weight;

}

}

static List<Edge> primMST(List<List<Edge>> graph) {

int V = graph.size();

boolean[] visited = new boolean[V];

List<Edge> MST = new ArrayList<>();

PriorityQueue<Edge> minHeap = new PriorityQueue<>(Comparator.comparingInt(edge -> edge.weight));

minHeap.addAll(graph.get(0));

visited[0] = true;

while (!minHeap.isEmpty()) {

Edge edge = minHeap.poll();

if (visited[edge.dest]) continue;

visited[edge.dest] = true;

MST.add(edge);

minHeap.addAll(graph.get(edge.dest));

}

return MST;

}

public static void main(String[] args) {

int V = 5;

List<List<Edge>> graph = new ArrayList<>();

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

graph.add(new ArrayList<>());

}

// Adding edges to the graph

graph.get(0).add(new Edge(0, 1, 2));

graph.get(0).add(new Edge(0, 3, 6));

graph.get(1).add(new Edge(1, 0, 2));

graph.get(1).add(new Edge(1, 2, 3));

graph.get(1).add(new Edge(1, 3, 8));

graph.get(1).add(new Edge(1, 4, 5));

graph.get(2).add(new Edge(2, 1, 3));

graph.get(2).add(new Edge(2, 4, 7));

graph.get(3).add(new Edge(3, 0, 6));

graph.get(3).add(new Edge(3, 1, 8));

graph.get(3).add(new Edge(3, 4, 9));

graph.get(4).add(new Edge(4, 1, 5));

graph.get(4).add(new Edge(4, 2, 7));

graph.get(4).add(new Edge(4, 3, 9));

List<Edge> MST = primMST(graph);

System.out.println("Edges of Minimum Spanning Tree:");

for (Edge edge : MST) {

System.out.println(edge.source + " - " + edge.dest + ": " + edge.weight);

}

}

}

**4. Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.**

public class NQueens {

private static final int N = 8; // Change N to desired board size

private static boolean isSafe(int[][] board, int row, int col) {

// Check if there is a queen in the same column

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

if (board[i][col] == 1) {

return false;

}

}

// Check upper left diagonal

for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {

if (board[i][j] == 1) {

return false;

}

}

// Check upper right diagonal

for (int i = row, j = col; i >= 0 && j < N; i--, j++) {

if (board[i][j] == 1) {

return false;

}

}

return true;

}

private static boolean solveNQueensUtil(int[][] board, int row) {

if (row == N) {

printSolution(board);

return true;

}

for (int col = 0; col < N; col++) {

if (isSafe(board, row, col)) {

board[row][col] = 1;

if (solveNQueensUtil(board, row + 1)) {

return true;

}

board[row][col] = 0; // backtrack

}

}

return false;

}

private static void printSolution(int[][] board) {

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

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

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

}

System.out.println();

}

System.out.println();

}

public static void main(String[] args) {

int[][] board = new int[N][N];

if (!solveNQueensUtil(board, 0)) {

System.out.println("No solution exists");

}

}

}

/\*

1 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

0 0 0 0 0 0 1 0

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

\*/

**5. Develop an elementary chatbot for any suitable customer interaction application.**

import java.util.Scanner;

public class SimpleChatbot {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Hello! I'm a simple chatbot. How can I help you today?");

System.out.println("Type 'bye' to end the conversation.");

String userInput;

do {

System.out.print("You: ");

userInput = scanner.nextLine();

// Bot's responses based on user input

if (userInput.equalsIgnoreCase("hello")) {

System.out.println("Bot: Hi there!");

} else if (userInput.equalsIgnoreCase("how are you")) {

System.out.println("Bot: I'm just a computer program, but thanks for asking!");

} else if (userInput.equalsIgnoreCase("bye")) {

System.out.println("Bot: Goodbye! Have a great day!");

} else {

System.out.println("Bot: Sorry, I didn't understand that.");

}

} while (!userInput.equalsIgnoreCase("bye"));

// Display farewell message before terminating

System.out.println("Bot: Conversation ended. Goodbye!");

scanner.close();

}

}

**Information Security**

**1. Write a Java/C/C++/Python program that contains a string (char pointer) with a value \Hello World’. The program should AND or and XOR each character in this string with 127 and display the result.**

public class ANDXOR {

public static void main(String[] args) {

// Input string

String input = "Hello World";

// Perform AND operation with 127

String andResult = bitwiseAND(input, 127);

System.out.println("AND Result: " + andResult);

// Perform XOR operation with 127

String xorResult = bitwiseXOR(input, 127);

System.out.println("XOR Result: " + xorResult);

}

// Method to perform bitwise AND operation

public static String bitwiseAND(String input, int value) {

StringBuilder result = new StringBuilder();

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

result.append((char) (c & value));

}

return result.toString();

}

// Method to perform bitwise XOR operation

public static String bitwiseXOR(String input, int value) {

StringBuilder result = new StringBuilder();

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

result.append((char) (c ^ value));

}

return result.toString();

}

}

**2. Write a Java/C/C++/Python program to perform encryption and decryption using the method of Transposition technique.**

import java.util.Scanner;

public class ColumnerTransposition {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input plaintext

System.out.print("Enter the plaintext: ");

String plaintext = scanner.nextLine().toUpperCase();

// Input key (order of columns)

System.out.print("Enter the key (order of columns): ");

String key = scanner.nextLine();

// Encrypt the plaintext

String ciphertext = encrypt(plaintext, key);

System.out.println("Encrypted Text: " + ciphertext);

scanner.close();

}

public static String encrypt(String plaintext, String key) {

// Remove spaces from the key

key = key.replaceAll("\\s", "");

// Calculate the number of columns

int numCols = key.length();

// Calculate the number of rows

int numRows = (int) Math.ceil((double) plaintext.length() / numCols);

// Create a 2D array to represent the grid

char[][] grid = new char[numRows][numCols];

// Fill the grid with the plaintext characters

int index = 0;

for (int col = 0; col < numCols; col++) {

for (int row = 0; row < numRows; row++) {

if (index < plaintext.length()) {

grid[row][col] = plaintext.charAt(index++);

} else {

// If the plaintext is shorter than the grid, fill with 'X'

grid[row][col] = 'X';

}

}

}

// Encrypt the plaintext based on the column order given by the key

StringBuilder ciphertext = new StringBuilder();

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

char columnChar = key.charAt(i);

int columnIndex = columnChar - '1'; // Convert char to index

// Append characters from the specified column to the ciphertext

for (int row = 0; row < numRows; row++) {

ciphertext.append(grid[row][columnIndex]);

}

}

return ciphertext.toString();

}

}

**3. Write a Java/C/C++/Python program to implement DES algorithm.**

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

public class DES {

public static void main(String[] args) throws Exception {

String plainText = "Hello, World!";

String key = "SecretKe"; // 8 characters for DES

SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "DES");

// Encryption

Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());

String encryptedText = Base64.getEncoder().encodeToString(encryptedBytes);

System.out.println("Encrypted Text: " + encryptedText);

// Decryption

cipher.init(Cipher.DECRYPT\_MODE, secretKey);

byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));

String decryptedText = new String(decryptedBytes);

System.out.println("Decrypted Text: " + decryptedText);

}

}

**4. Write a Java/C/C++/Python program to implement AES Algorithm.**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.IvParameterSpec;

import java.security.SecureRandom; // Add this import statement

import java.util.Base64;

public class AES {

public static void main(String[] args) throws Exception {

// Generate a secret key

SecretKey secretKey = generateSecretKey();

// Original message

String originalMessage = "computer";

System.out.println("Original Message: " + originalMessage);

// Encrypt the message

String encryptedMessage = encrypt(originalMessage, secretKey);

System.out.println("Encrypted Message: " + encryptedMessage);

// Decrypt the message

String decryptedMessage = decrypt(encryptedMessage, secretKey);

System.out.println("Decrypted Message: " + decryptedMessage);

}

private static SecretKey generateSecretKey() throws Exception {

KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");

keyGenerator.init(256); // 256-bit AES key size for enhanced security

return keyGenerator.generateKey();

}

private static String encrypt(String plainText, SecretKey secretKey) throws Exception {

Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

// Generate a random Initialization Vector (IV)

byte[] ivBytes = new byte[cipher.getBlockSize()];

SecureRandom random = new SecureRandom();

random.nextBytes(ivBytes);

IvParameterSpec ivParameterSpec = new IvParameterSpec(ivBytes);

cipher.init(Cipher.ENCRYPT\_MODE, secretKey, ivParameterSpec);

byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());

byte[] combinedBytes = new byte[ivBytes.length + encryptedBytes.length];

System.arraycopy(ivBytes, 0, combinedBytes, 0, ivBytes.length);

System.arraycopy(encryptedBytes, 0, combinedBytes, ivBytes.length, encryptedBytes.length);

return Base64.getEncoder().encodeToString(combinedBytes);

}

private static String decrypt(String encryptedText, SecretKey secretKey) throws Exception {

Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

// Extract IV from the combined byte array

byte[] combinedBytes = Base64.getDecoder().decode(encryptedText);

byte[] ivBytes = new byte[cipher.getBlockSize()];

byte[] encryptedBytes = new byte[combinedBytes.length - ivBytes.length];

System.arraycopy(combinedBytes, 0, ivBytes, 0, ivBytes.length);

System.arraycopy(combinedBytes, ivBytes.length, encryptedBytes, 0, encryptedBytes.length);

IvParameterSpec ivParameterSpec = new IvParameterSpec(ivBytes);

cipher.init(Cipher.DECRYPT\_MODE, secretKey, ivParameterSpec);

byte[] decryptedBytes = cipher.doFinal(encryptedBytes);

return new String(decryptedBytes);

}

}

**5. Write a Java/C/C++/Python program to implement RSA algorithm.**

import math

# Step 1: Choose two distinct prime numbers p and q

p = 61

q = 53

# Step 2: Compute n

n = p \* q

print("n =", n)

# Step 3: Compute Euler's totient function (phi)

phi = (p - 1) \* (q - 1)

print("phi =", phi)

# Step 4: Choose e (public key)

e = 17 # Typically a small prime number

print("e =", e)

# Step 5: Compute d (private key)

def modinv(a, m):

"""Compute the modular multiplicative inverse of a modulo m."""

g, x, \_ = egcd(a, m)

if g != 1:

raise Exception('Modular inverse does not exist')

return x % m

def egcd(a, b):

"""Extended Euclidean algorithm."""

if a == 0:

return (b, 0, 1)

else:

g, y, x = egcd(b % a, a)

return (g, x - (b // a) \* y, y)

d = modinv(e, phi)

print("d =", d)

# Public and private key pair

public\_key = (e, n)

private\_key = (d, n)

print("Public key:", public\_key)

print("Private key:", private\_key)

# Plain text message

msg = 42

print("Original message:", msg)

# Encryption

C = pow(msg, e, n)

print("Encrypted message:", C)

# Decryption

M = pow(C, d, n)

print("Decrypted message:", M)

**6. Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Document</title>

<script>

function checkvalue() {

var a = document.getElementById("no").value;

if (a <= 1) {

alert("Please give prime number");

return;

false;

}

for (var i = 2; i <= Math.sqrt(a); i++) {

if (a % i === 0) {

alert("Please give prime number");

return false;

}

}

var b = document.getElementById("pr").value;

for (var i = 2; i < a - 1; i++) {

if (Math.pow(b, i) % a === 1) {

alert("Please enter correct primitive root");

return false;

}

}

var XA = parseInt(document.getElementById("xa").value);

var XB = parseInt(document.getElementById("xb").value);

var n = Math.pow(b, XA);

var YA = n % a;

alert("YA is " + YA);

var n1 = Math.pow(b, XB);

var YB = n1 % a;

alert("YB is " + YB);

var n3 = Math.pow(YB, XA);

var k1 = n3 % a;

alert("K1 is " + k1);

var n4 = Math.pow(YA, XB);

var k2 = n4 % a;

alert("K2 is " + k2);

if (k1 == k2) {

alert("K1 == k2");

alert("Key is Successfully Exchange");

}

return true;

}

</script>

</head>

<body>

<form onsubmit="return checkvalue()">

<center>

<h1>Diffie Hellman key Exchange</h1>

Enter the Prime no:

<input type="text" name="no" id="no" />

<br /><br />

Enter Primitive root:

<input type="text" name="pr" id="pr" /> <br /><br />

Select XA:

<input type="text" name="xa" id="xa" /> <br /><br />

Select XB:

<input type="text" name="xb" id="xb" /> <br /><br />

<input type="submit" value="Check all the values" />

</center>

</form>

</body>

</html>

**7. Calculate the message digest of a text using the MD5 algorithm in JAVA.**

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class MD5 {

// Hash function to get the MD5 hash

public static String getMd5Hash(String input) {

try {

// Static getInstance() method is called with hashing MD5

MessageDigest md = MessageDigest.getInstance("MD5");

// Calculating message digest of an input that returns an array of bytes

byte[] messageDigest = md.digest(input.getBytes());

// Converting byte array into signum representation

BigInteger no = new BigInteger(1, messageDigest);

// Converting message digest into hex value

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

} catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

// Driver code

public static void main(String args[]) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the text to calculate MD5 hash: ");

String userInput = scanner.nextLine();

scanner.close();

System.out.println("HashCode Generated for the string is: " + getMd5Hash(userInput));

}

}