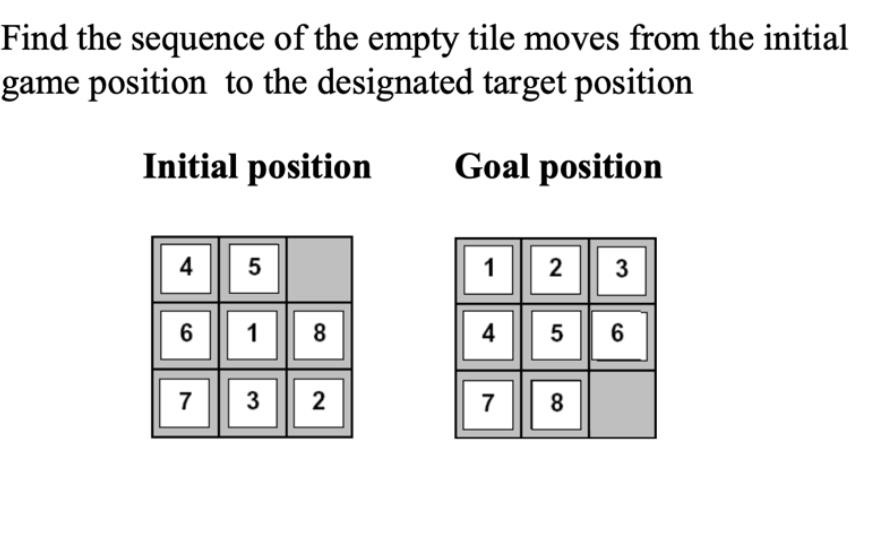
**ARTIFICIAL INTELLIGENCE**

**LAB ASSIGNMENT – 2**

**NAME : PRATHAPANI SATWIKA**

**REG.NO. : 20BCD7160**



**CODE :**

import java.util.\*;

class Main

{

public int dimension = 3;

int[] row = { 1, 0, -1, 0 };

int[] col = { 0, -1, 0, 1 };

public int calculateCost(int[][] initial, int[][] goal)

{

int count = 0;

int n = initial.length;

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

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

{

if (initial[i][j] != 0 && initial[i][j] != goal[i][j])

{

count++;

}

}

}

return count;

}

public void printMatrix(int[][] matrix)

{

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

{

for (int j = 0; j < matrix.length; j++)

{

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

}

System.out.println();

}

}

public boolean isSafe(int x, int y)

{

return (x >= 0 && x < dimension && y >= 0 && y <dimension);

}

public void printPath(Node root)

{

if (root == null)

{

return;

}

printPath(root.parent); printMatrix(root.matrix);

System.out.println();

}

public boolean isSolvable(int[][] matrix)

{

int count = 0;

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

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

for (int j = 0; j < matrix.length; j++)

{

array.add(matrix[i][j]);

}

}

Integer[] anotherArray = new Integer[array.size()];

array.toArray(anotherArray);

for (int i = 0; i < anotherArray.length - 1; i++)

{

for (int j = i + 1; j < anotherArray.length; j++)

{

if (anotherArray[i] != 0 && anotherArray[j] != 0 && anotherArray[i] > anotherArray[j])

{

count++;

}

}

}

return count % 2 == 0;

}

public void solve(int[][] initial, int[][] goal, int x, int y)

{

PriorityQueue<Node> pq = new PriorityQueue<Node>(1000, (a, b) -> (a.cost + a.level) - (b.cost + b.level));

Node root = new Node(initial, x, y, x, y, 0, null); root.cost = calculateCost(initial, goal); pq.add(root);

while (!pq.isEmpty())

{

Node min = pq.poll();

if (min.cost == 0)

{

printPath(min);

return;

}

for (int i = 0; i < 4; i++)

{

if (isSafe(min.x + row[i], min.y + col[i]))

{

Node child = new Node(min.matrix, min.x, min.y, min.x + row[i], min.y + col[i], min.level + 1, min); child.cost = calculateCost(child.matrix, goal); pq.add(child);

}

}

}

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter number of rows in puzzle:");

int r=sc.nextInt();

System.out.println("Enter number of columns in puzzle:");

int c=sc.nextInt();

int[][] initial = new int[r][c];

int[][] goal =new int[r][c];

System.out.println("Enter your initial puzzle:");

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

{

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

{

initial[i][j]=sc.nextInt();

}

}

System.out.println();

System.out.println("Enter your Goal puzzle:");

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

{

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

{

goal[i][j]=sc.nextInt();

} }

int x = 1, y = 0;

Main puzzle = new Main();

if (puzzle.isSolvable(initial))

{

puzzle.solve(initial, goal, x, y);

}

else {

System.out.println("The given initial is possible to solve");

}

}

}

class Node

{

public Node parent;

public int[][] matrix;

public int x, y;

public int cost;

public int level;

Node(int[][] initial, int x, int y, int x0, int y0, int i, Object object)

{

throw new UnsupportedOperationException("Not supported yet.");

}

public Node(int[][] matrix, int x, int y, int newX, int newY, int level, Node parent)

{

this.parent = parent;

this.matrix = new int[matrix.length][];

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

{

this.matrix[i] = matrix[i].clone();

}

this.matrix[x][y] = this.matrix[x][y] + this.matrix[newX][newY]; this.matrix[newX][newY] = this.matrix[x][y] - this.matrix[newX][newY]; this.matrix[x][y] = this.matrix[x][y] - this.matrix[newX][newY];

this.cost = Integer.MAX\_VALUE;

this.level = level;

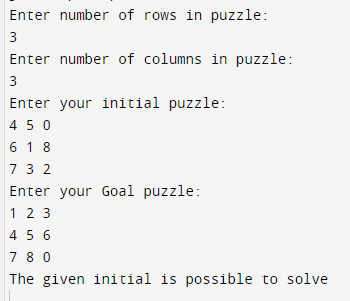
this.x = newX;

this.y = newY;

}

}

**OUTPUT :**

****