Artificial Intelligence

Asssignment 1 B

Name : Swati Anil Sonone

Div : CS\_D

Roll no. : 53

Problem Statement : AStar

Code :

import java.util.ArrayList;

import java.util.List;

class Node {

    Node parent;

    int[] position;

    int g, h, f;

    public Node(Node parent, int[] position) {

        this.parent = parent;

        this.position = position;

        this.g = 0;

        this.h = 0;

        this.f = 0;

    }

    public boolean equals(Node other) {

        return this.position[0] == other.position[0] && this.position[1] == other.position[1];

    }

}

public class AStar {

    public static List<int[]> astar(int[][] graph, int[] start, int[] end) {

        Node startNode = new Node(null, start);

        startNode.g = startNode.h = startNode.f = 0;

        Node endNode = new Node(null, end);

        endNode.g = endNode.h = endNode.f = 0;

        List<Node> openList = new ArrayList<>();

        List<Node> closedList = new ArrayList<>();

        openList.add(startNode);

        int count=0;

        while (!openList.isEmpty()) {

            Node currentNode = openList.get(0);

            int currentIndex = 0;

            for (int i = 0; i < openList.size(); i++) {

                Node item = openList.get(i);

                if (item.f < currentNode.f) {

                    currentNode = item;

                    currentIndex = i;

                    count+=item.f;

                }

            }

            openList.remove(currentIndex);

            closedList.add(currentNode);

            if (currentNode.equals(endNode)) {

                List<int[]> path = new ArrayList<>();

                // int count = 0;

                Node current = currentNode;

                while (current != null) {

                    path.add(current.position);

                    current = current.parent;

                    // count+=current.f;

                }

                // System.out.println("cost of min path : "+count);

                return reverseList(path);

            }

            List<Node> children = new ArrayList<>();

            int[][] newPositions = { { 0, -1 }, { 0, 1 }, { -1, 0 }, { 1, 0 }, { -1, -1 }, { -1, 1 }, { 1, -1 },

                    { 1, 1 } };

            for (int[] newPosition : newPositions) {

                int[] nodePosition = { currentNode.position[0] + newPosition[0],

                        currentNode.position[1] + newPosition[1] };

                if (nodePosition[0] > graph.length - 1 || nodePosition[0] < 0 || nodePosition[1] > graph[0].length - 1

                        || nodePosition[1] < 0)

                    continue;

                if (graph[nodePosition[0]][nodePosition[1]] != 0)

                    continue;

                Node newNode = new Node(currentNode, nodePosition);

                children.add(newNode);

            }

            for (Node child : children) {

                for (Node closedChild : closedList) {

                    if (child.equals(closedChild))

                        continue;

                }

                child.g = currentNode.g + 1;

                child.h = (int) (Math.pow(child.position[0] - endNode.position[0], 2) +

                        Math.pow(child.position[1] - endNode.position[1], 2));

                child.f = child.g + child.h;

                for (Node openNode : openList) {

                    if (child.equals(openNode) && child.g > openNode.g)

                        continue;

                }

                openList.add(child);

            }

        }

        return new ArrayList<>();

    }

    public static List<int[]> reverseList(List<int[]> list) {

        List<int[]> reversedList = new ArrayList<>();

        for (int i = list.size() - 1; i >= 0; i--) {

            reversedList.add(list.get(i));

        }

        return reversedList;

    }

    public static void main(String[] args) {

        int[][] graph = { { 0, 0, 0, 0, 1, 0 },

                { 0, 0, 0, 0, 1, 0 },

                { 0, 0, 0, 0, 1, 0 },

                { 0, 0, 0, 0, 1, 0 },

                { 0, 1, 1, 0, 1, 0 },

                { 0, 0, 0, 0, 0, 0 } };

        int[] start = { 0, 0 };

        int[] end = { 5, 5 };

        List<int[]> path = astar(graph, start, end);

        System.out.println("Path Coordinates : ");

        for (int[] is : path) {

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

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

            }

            System.out.println();

        }

        // System.out.println("min cost of path ");

    }

}

Output

