**Program:**

import java.util.\*;

public class AStar {

public static final int DIAGONAL\_COST = 14;

public static final int V\_H\_COST = 10;

static class Cell {

int heuristicCost = 0;

int finalCost = 0;

int i, j;

Cell parent;

Cell(int i, int j) {

this.i = i;

this.j = j;

}

@Override

public String toString() {

return "[" + this.i + ", " + this.j + "]";

}

}

static Cell [][] grid = new Cell[5][5];

static PriorityQueue<Cell> open;

static boolean closed[][];

static int startI, startJ;

static int endI, endJ;

public static void setBlocked(int i, int j) {

grid[i][j] = null;

}

public static void setStartCell(int i, int j) {

startI = i;

startJ = j;

}

public static void setEndCell(int i, int j) {

endI = i;

endJ = j;

}

static void checkAndUpdateCost(Cell current, Cell t, int cost) {

if (t == null || closed[t.i][t.j])return;

int t\_final\_cost = t.heuristicCost + cost;

boolean inOpen = open.contains(t);

if (!inOpen || t\_final\_cost < t.finalCost) {

t.finalCost = t\_final\_cost;

t.parent = current;

if (!inOpen)open.add(t);

}

}

public static void AStar() {

open.add(grid[startI][startJ]);

Cell current;

while (true) {

current = open.poll();

if (current == null)break;

closed[current.i][current.j] = true;

if (current.equals(grid[endI][endJ])) {

return;

}

Cell t;

if (current.i - 1 >= 0) {

t = grid[current.i - 1][current.j];

checkAndUpdateCost(current, t, current.finalCost + V\_H\_COST);

if (current.j - 1 >= 0) {

t = grid[current.i - 1][current.j - 1];

checkAndUpdateCost(current, t, current.finalCost + DIAGONAL\_COST);

}

if (current.j + 1 < grid[0].length) {

t = grid[current.i - 1][current.j + 1];

checkAndUpdateCost(current, t, current.finalCost + DIAGONAL\_COST);

}

}

if (current.j - 1 >= 0) {

t = grid[current.i][current.j - 1];

checkAndUpdateCost(current, t, current.finalCost + V\_H\_COST);

}

if (current.j + 1 < grid[0].length) {

t = grid[current.i][current.j + 1];

checkAndUpdateCost(current, t, current.finalCost + V\_H\_COST);

}

if (current.i + 1 < grid.length) {

t = grid[current.i + 1][current.j];

checkAndUpdateCost(current, t, current.finalCost + V\_H\_COST);

if (current.j - 1 >= 0) {

t = grid[current.i + 1][current.j - 1];

checkAndUpdateCost(current, t, current.finalCost + DIAGONAL\_COST);

}

if (current.j + 1 < grid[0].length) {

t = grid[current.i + 1][current.j + 1];

checkAndUpdateCost(current, t, current.finalCost + DIAGONAL\_COST);

}

}

}

}

public static void test(int tCase, int x, int y, int si, int sj, int ei, int ej, int[][] blocked) {

System.out.println("\n\nTest Case #" + tCase);

grid = new Cell[x][y];

closed = new boolean[x][y];

open = new PriorityQueue<>((Object o1, Object o2) -> {

Cell c1 = (Cell)o1;

Cell c2 = (Cell)o2;

return c1.finalCost < c2.finalCost ? -1 :

c1.finalCost > c2.finalCost ? 1 : 0;

});

setStartCell(si, sj);

setEndCell(ei, ej);

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

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

grid[i][j] = new Cell(i, j);

grid[i][j].heuristicCost = Math.abs(i - endI) + Math.abs(j - endJ);

}

}

grid[si][sj].finalCost = 0;

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

setBlocked(blocked[i][0], blocked[i][1]);

}

System.out.println("Grid: ");

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

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

if (i == si && j == sj)System.out.print("SO ");

else if (i == ei && j == ej)System.out.print("DE ");

else if (grid[i][j] != null)System.out.printf("%-3d ", 0);

else System.out.print("BL ");

}

System.out.println();

}

System.out.println();

AStar();

System.out.println("\nScores for cells: ");

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

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

if (grid[i][j] != null)System.out.printf("%-3d ", grid[i][j].finalCost);

else System.out.print("BL ");

}

System.out.println();

}

System.out.println();

if (closed[endI][endJ]) {

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

Cell current = grid[endI][endJ];

System.out.print(current);

while (current.parent != null) {

System.out.print(" -> " + current.parent);

current = current.parent;

}

System.out.println();

} else System.out.println("No possible path");

}

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

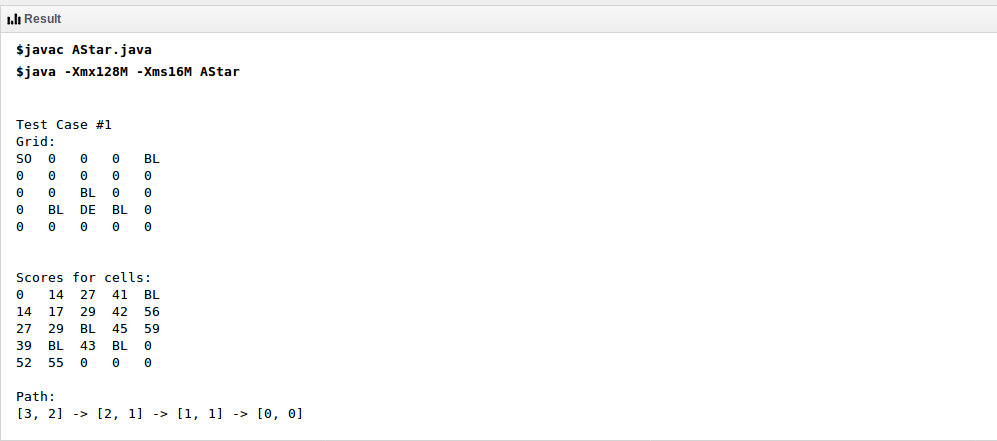
test(1, 5, 5, 0, 0, 3, 2, new int[][] {{0, 4}, {2, 2}, {3, 1}, {3, 3}});

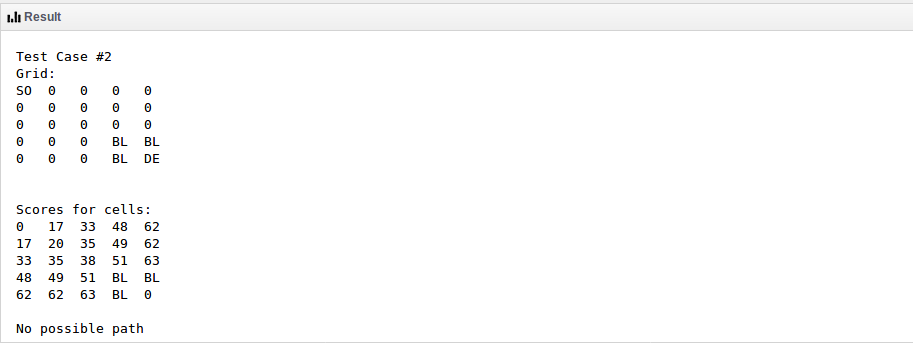
test(2, 5, 5, 0, 0, 4, 4, new int[][] {{3, 4}, {3, 3}, {4, 3}});

}

}

**Output:**

****

****