Artificial Intelligence Mini-Project Documentation

Gardener & Thirsty Plant

(Shortest - route problem)

Submitted by

Team - Good-Learners

Lokesh kirad (18MCMC22) & Korbha Nagesh (18MCMC61)

Under Supervision of:

Dr. PSVS Sai Prasad Sir

CA554 - Introduction to AI

Master of computer application(MCA)

School Of Computer & Information Science

University Of Hyderabad

May 10 , 2020

Rules of The Game:

1. Create Garden by clicking on the button "Create Garden".
2. Select cell by selecting Combo box list options, To place position for Gardener & Thirsty Plant.
3. After Generating a map or Create Garden, you can remove Or make a wall by selecting any cell with given list options.
4. Start your search to find the shortest way to reach destination plant (cell) by the Clicking the button "Start Search".
5.you can reset & Clear Garden/Map by clicking on buttons "RESET " or "Clear ALL".

Environment Setup:

In order to run this GUI application, java (JDK) version "1.8.0_231" & Java SE Runtime Environment (JRE) Should be installed On your System (Having Windows 10 OS ,Intel(R)Core i5 Processor, 4 GB RAM,1 TB Hard disk, Keyboard & Mouse, Color Display.)
To Develop this java application, I Used 'Eclipse IDE for Java Developers-Version: 2019-06 (4.12.0)' But we can simply run this application by Compile and Run Gardener.java file on Your Command Prompt (CLI).
There is no need to install anything else apart JDK & JRE (mentioned above) which consists of all environment setup to run this JAVA application.

Code Explanation:

Graphical user interface :
a) **"Create Garden "** Button (Event handler) used for creating a map (named Canvas)or garden.

- Canvas is the object of Map class which is responsible for managing the cell (two-dimensional array)& this class also Containing the method **paintComponent(Graphics g)** which is used to **fillRect() , drawRect(), and setColor()** on cells when needed to update the map.

- Cells: Each cell of Canvas Map (two-dimensional array) Is maintained by Class Node (user-defined data type for each cell) which consist of the properties of cell & and their behaviors (operational method) functions.

- Node Class consists mainly of **X, Y & Lastx,Lasty** for index Position of each cell, and their supporting method to set & get methods to fetch and set current value.

- it calls the method named **generateMap()** to create a map by generating a random number to paint cell (black) for wall .

b) **"Reset "** Button (Event handler) remove only explored (CYAN) & shortest path (yellow) cells On a map named Canvas.

- it calls method named **resetMap()** & **Update()** to apply changes on map when event is occur

c) **"Clear ALL "** Button (Event handler) clear all the map (Canvas).

- it calls the method **clearMap()** to clear all maps by paint all cells of the map as white.

d) **"Select cell for (Combo-box list -options) "** used to set the cells position for *Gardener ,Thirsty plant & make wall ,empty wall* using **paintComponent()** Method.

- it is called by Map class constructor to (calling **paintComponent()**) update map in order to set the position for *gardener, plant & wall or empty.*

c) **"Start Search"** Button (Event handler) is responsible to call Djikstra-algorithm-function **Djikstra()** to find the distance between Strat and Last cell.

- it calls **startSearch()** to call a Djikstra-algorithm function named **Dijkstra()** in order to get the shortest path between Gardener cell & plant cell.

Backend : -

- Here we used java Swing/AWT event handling function call required method.

- **Djikstra shortest path algorithm** is used here on two-dimensional array to get the shortest path between gardener & thirsty plant

- we are maintaining to **ArrayList** of Node(class) data type: **explored** and method named **exploreNeighbors()** which fill the explore list.

- **exploreNeighbors(Node,int) :** where **exploreNeighbors()** is used as a function in order to find a valid cell and their related neighbors which are linked to other (exact neighbor cell) cells if they are valid nodes then that are nodes are added to **explore list.**

- **explore:** is maintaining the cells position and their corresponding nearest connection **(x:y::lastx:lasty)** & **hops** value.it would help us to get **backtrack** the way.

- **backtrack(int , int , int):** it works on the explore list which contains a cell having cell index & their connection ,once we reach to the last cells (thirsty plant cell) this method is called.

- **Hops :** hops are the variables of node class which used to count the length of shortest path and also used to reach back at start cells (Gardener).

- **pause() & delay() :** is used for time-sleep to show flow of program in better way.

Division of modules among team members:

- **All back-end part** including function or block of codes for Djikstra() ,exploreNeighbors() & explore list ,backtrack() ,pause() & delay() , Node & Map class is implementaed by **Lokesh kirad(18MCMC22)**

- **All GUI-part** including event-handling method, position settings of swing / Awt component and Map & Node class, a method like resetMap(), clearMap() is implemented by **Korbha Nagesh (18MCM61).**