Package FunctionB_ShortestPath

Class AStarAlgorithm

java.lang.Object
FunctionB_ShortestPath.AStarAlgorithm

```
public class AStarAlgorithm
extends Object
```

The AStarAlgorithm class implements the A Star algorithm for finding the shortest path between Tom and Jerry in a maze. It takes the locations of Tom and Jerry, as well as the maze configuration (read by Function A's MazeLoader), and calculates the shortest path using the A Star algorithm. A Star algorithm's heuristic function is using Manhattan Distance which is admissible and will lead to an optimal solution. This class provides methods to change the locations of Tom and Jerry, check if a node has been explored, check if a node is valid, find the neighbors of a given node, generate the shortest path using A Star Algorithm, and determine Tom's next movement based on the calculated path.

See Also:

Node, MazeLoader

Field Summary

Fields

Modifier and Type	Field	Description
int[]	jerryLocation	
final int[][]	maze	
int[]	tomLocation	

Constructor Summary

Constructors

Constructor	Description
AStarAlgorithm(int[] tomLocation, int[] jerryLocation, String maze)	AStarAlgorithm's constructor for constructing the object.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int[]	changeLocation(int[] location, int who)	Changing the location of Tom or Jerry
boolean	checkExplored(List <Node> listOfNode, int[] temp)	Check the node is already explored or not.
boolean	checkValidNode(List <Node> expandedNode, List <Node> frontier, int[] temp)	Check the location is valid to be a neighbor or not
List <Node>	findNeighbor(Node currentNode, List <Node> expandedNode, List <Node> frontier)	Find all neighbor near the current node
List <int[]>	pathGeneratorByAStar()	Generate the shortest path between Tom and Jerry
int[]	tomNextMovement()	Tom's next action

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

tomLocation

```
public int[] tomLocation
```

jerryLocation

```
public int[] jerryLocation
```

maze

```
public final int[][] maze
```

Constructor Details

AStarAlgorithm

```
public AStarAlgorithm(int[] tomLocation,
                      int[] jerryLocation,
                      String  maze)
```

AStarAlgorithm's constructor for constructing the object.

Parameters:

`tomLocation` - Tom's current location

`jerryLocation` - Jerry's current location

`maze` - CSV file name of the maze

Method Details

changeLocation

```
public int[] changeLocation(int[] location,
                            int who)
```

Changing the location of Tom or Jerry

Parameters:

`location` - The latest location

`who` - 0 for changing Tom's location, other for changing Jerry's location

Returns:

The latest location

checkExplored

```
public boolean checkExplored(List <Node> listOfNode,
                             int[] temp)
```

Check the node is already explored or not.

Parameters:

`listOfNode` - The nodes inside the list is explored

`temp` - The current location

Returns:

True if it is already explored, False if not explored

checkValidNode

```
public boolean checkValidNode(List <Node> expandedNode,
                              List <Node> frontier,
                              int[] temp)
```

Check the location is valid to be a neighbor or not

Parameters:

`expandedNode` - A list of nodes which have already expanded

`frontier` - A list of nodes which are candidates to be expanded

`temp` - The current location

Returns:

True if not the node does not explore and within the maze, otherwise False

findNeighbor

```
public List <Node> findNeighbor(Node currentNode,
                                List <Node> expandedNode,
                                List <Node> frontier)
```

Find all neighbor near the current node

Parameters:

`currentNode` - The current node

`expandedNode` - A list of nodes which have already expanded

`frontier` - A list of nodes which are candidates to be expanded

Returns:

A list of valid neighbor with type Node

pathGeneratorByAStar

```
public List <int[]> pathGeneratorByAStar()
```

Generate the shortest path between Tom and Jerry

Returns:

A list of coordinate with type int[] which form the shortest path

tomNextMovement

```
public int[] tomNextMovement()
```

Tom's next action

Returns:

The coordinate of Tom's next movement.