8 Puzzle Solver

Generated by Doxygen 1.8.8

Mon Dec 8 2014 13:50:41

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 SearchAlgortihms.AStar Class Reference

Collaboration diagram for SearchAlgortihms.AStar:

Public Member Functions

- **AStar** ()
- void **astar** (**State** start, **State** goal)
- void **addNode** (**TreeNode** current, **State** goal)

3.1.1 Detailed Description

A∗ algorithm class

Author

Stefan

3.1.2 Constructor & Destructor Documentation

3.1.2.1 SearchAlgortihms.AStar.AStar ( )

A∗ class constructor

3.1.3 Member Function Documentation

3.1.3.1 void SearchAlgortihms.AStar.addNode (TreeNode *current,* State *goal*)

Method to add a new node the the search queue. This method calculates the Manhattan Distance for every child in the current node and only add them to the search queue if the Manhattan Distance is less than the original calculated one.

Parameters

current	The current node

goal	The goal node, use for the Manhattan Distance calculation

Here is the call graph for this function:

Here is the caller graph for this function:

3.1.3.2 void SearchAlgortihms.AStar.astar (State *start*, State *goal*)

A∗ algorithm

Parameters

start	The start State
goal	The goal State

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchAlgortihms/AStar.java

3.2 SearchAlgortihms.BFS Class Reference

Collaboration diagram for SearchAlgortihms.BFS:

Public Member Functions

- **BFS** ()
- void **bfs** (**State** start, **State** goal)

3.2.1 Detailed Description

Class file for the Breadth first search algorithm

Author

stefan

3.2.2 Constructor & Destructor Documentation

3.2.2.1 SearchAlgortihms.BFS.BFS ( )

Constructor of the **BFS** (p. 6) object

3.2.3 Member Function Documentation

3.2.3.1 void SearchAlgortihms.BFS.bfs (State *start*, State *goal*)

Breath-First search method

Parameters

start	The start State of the graph
goal	The goal State of the graph

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchAlgortihms/BFS.java

3.3 SearchAlgortihms.DFS Class Reference

Collaboration diagram for SearchAlgortihms.DFS:

Public Member Functions

- **DFS** ()
- void **dfs** (**State** start, **State** goal)

3.3.1 Detailed Description

Depth-First search algorithm class

Author

stefan

3.3.2 Constructor & Destructor Documentation

3.3.2.1 SearchAlgortihms.DFS.DFS ( )

Constructor of the **DFS** (p. 7) object

3.3.3 Member Function Documentation

3.3.3.1 void SearchAlgortihms.DFS.dfs (State *start*, State *goal*)

Depth-First Search algorithm

Parameters

start	The start state of the graph
goal	The goal state of the graph

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchAlgortihms/DFS.java

3.4 csm6120.FileManager Class Reference

Public Member Functions

- **FileManager** ()
- void **reader** (**State** s, File path)
- void **findInteger** (String s, **State** state)

3.4.1 Detailed Description

This class holds methods to manipulate the input files.

Author

stefan

3.4.2 Constructor & Destructor Documentation

3.4.2.1 csm6120.FileManager.FileManager ( )

Constructor of the **FileManager** (p. 8) class

3.4.3 Member Function Documentation

3.4.3.1 void csm6120.FileManager.findInteger (String *s,* State *state*)

Changes the input line from being Strings to single Integers.

Parameters

s	The String to analyse and change
state	The **State** (p. 14) object to save too

Here is the call graph for this function:

Here is the caller graph for this function:

3.4.3.2 void csm6120.FileManager.reader (State *s,* File *path*)

Read a given file path and calls the **findInteger()** (p. 8) method. This is used to read the input files and read them line for line.

Parameters

s	The **State** (p. 14) object to save too
path	The path of the input file

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/csm6120/FileManager.java

3.5 SearchAlgortihms.GBFS Class Reference

Collaboration diagram for SearchAlgortihms.GBFS:

Public Member Functions

- **GBFS** ()
- void **gbfs** (**State** start, **State** goal)

3.5.1 Detailed Description

Greedy Best-First Search class

Author

stefan

3.5.2 Constructor & Destructor Documentation

3.5.2.1 SearchAlgortihms.GBFS.GBFS ( )

Constructor of the **GBFS** (p. 8) class

3.5.3 Member Function Documentation

3.5.3.1 void SearchAlgortihms.GBFS.gbfs (State *start,* State *goal*)

Greedy Best-First Search algorithm

Parameters

start	The start State of the graph
goal	The goal State of the graph

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchAlgortihms/GBFS.java

3.6 SearchTree.Graph Class Reference

Public Member Functions

- **Graph** ()
- void **nextStep** (**TreeNode** s)
- void **corner** (int tile, **TreeNode** s)
- void **midSection** (int tile, **TreeNode** s)
- void **center** (int tile, **TreeNode** s)

3.6.1 Detailed Description

This class is used to generate the next step in the graph.

Author

Stefan

3.6.2 Constructor & Destructor Documentation

3.6.2.1 SearchTree.Graph.Graph ( )

Constructor for the graph object

3.6.3 Member Function Documentation

3.6.3.1 void SearchTree.Graph.center (int *tile,* TreeNode *s*)

This method generates the next level of the graph if the empty tile(0) is in the center of the puzzle. (Tile 4 in the representation below Saves all possible states to an arrayList.

0 1 2 3 4 5 6 7 8

Parameters

tile	The index of the empty tile
s	The state to base algorithm on

Here is the call graph for this function:

Here is the caller graph for this function:

3.6.3.2 void SearchTree.Graph.corner (int *tile,* TreeNode *s*)

This method is used to generate the next level of the graph when the empty tile is at a corner. Saves all changes to a arrayList of possible states. The tiles where this method is used corresponds with the fields 0, 2, 6 ,and 8 as shown below

0 1 2 3 4 5 6 7 8

Parameters

tile	The index of the empty tile(0)
s	The state to base algorithm on

Here is the call graph for this function:

Here is the caller graph for this function:

3.6.3.3 void SearchTree.Graph.midSection (int *tile,* TreeNode *s*)

This method to generate the next level of the graph when the empty tile(0) is on the midsection of the sides. Saves all possible states to an arrayList. The tiles where this method will be used correspond to the fields 1, 3, 5, and 7 as shown below

0 1 2 3 4 5 6 7 8

Parameters

tile	The index of the empty tile(0)
s	The state to base algorithm on

Here is the call graph for this function:

Here is the caller graph for this function:

3.6.3.4 void SearchTree.Graph.nextStep (TreeNode *s*)

Algorithm to generate the next state in the graph Based on the fact that empty space can only move horizontally and vertically. To make the process easier simple numbers identifiers are assigned to the possible tiles in the puzzle.

These numbers represent indices in the arrayList and are : 0 1 2 3 4 5 6 7 8 This method checks where the empty tile is and calls other methods to switch the tiles.

Parameters

s	The state on which the next step will be based

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchTree/Graph.java

3.7 csm6120.Main Class Reference

Static Public Member Functions

- static void **main** (String[] args)

3.7.1 Detailed Description

This class is the entrance to the program.

Author

stefan

3.7.2 Member Function Documentation

3.7.2.1 static void csm6120.Main.main (String[] *args*) [static]

Main (p. 11) method of the program. This method can be called from the command line with a set of arguments.

javac main theStartFile theGoalFile theAlgorithmToUse

where theStartFile is a text file holding the start **State** (p. 14), theGoalFile holds the goal **State** (p. 14) of the puzzle. TheAlgorithmToUse specifies which algorithm, possibilities are:

bfs - Breadth-First search dfs - Depth-First search gbfs - Greedy Best-First search astar - A$*$ search

Parameters

args	the command line arguments

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/csm6120/Main.java

3.8 SearchAlgortihms.ManhattanDistance Class Reference

Public Member Functions

- **ManhattanDistance** (**State** start, **State** goal)
- int[] **convertToArray** (ArrayList< Integer > i)
- int[][] **convertTo2DArray** (int[] intArray)

- void **setStartArray** (int[][] toSet)
- void **setGoalArray** (int[][] toSet)
- int[][] **findCell** (int[][] array, int index)
- int **findXCoordinate** (int[][] array, int index)
- int **findYCoordinate** (int[][] array, int index)
- int **calcManhattanDistance** (**State** start, **State** goal)

3.8.1 Detailed Description

This class is used to calculate the Manhattan distance for the A∗ algorithm

Author

Stefan

3.8.2 Constructor & Destructor Documentation

3.8.2.1 SearchAlgortihms.ManhattanDistance.ManhattanDistance (State *start,* State *goal*)

Constructor of the **ManhattanDistance** (p. 11) class

Parameters

start	The State to compare to the goal
goal	The goal State to compare too

Here is the call graph for this function:

3.8.3 Member Function Documentation

3.8.3.1 int SearchAlgortihms.ManhattanDistance.calcManhattanDistance (State *start,* State *goal*)

Calculate the Manhattan distance for 2 input states

Parameters

start	The start State for the calculation
goal	The goal State to calculate the distance to

Returns

An integer representing the Manhattan Distance;

Here is the call graph for this function:

Here is the caller graph for this function:

3.8.3.2 int [][] SearchAlgortihms.ManhattanDistance.convertTo2DArray (int[] *intArray*)

Method to convert an 1D integer array to a 2D integer array

Parameters

intArray	The integer array to convert

Returns

An 2D integer array

Here is the caller graph for this function:

3.8.3.3 int [] SearchAlgortihms.ManhattanDistance.convertToArray (ArrayList< Integer > *i*)

Method to convert an ArrayList to an array

Parameters

i	The arrayList to convert

Returns

An integer array

Here is the caller graph for this function:

3.8.3.4 int [][] SearchAlgortihms.ManhattanDistance.findCell (int *array[][]*, int *index*)

Method to find the X and Y coordinates of a given tile in a 2D array

Parameters

array	The 2D array to search in
index	The number/tile to search for

Returns

A 2D array holding the X and Y coordinates

3.8.3.5 int SearchAlgortihms.ManhattanDistance.findXCoordinate (int *array[][]*, int *index*)

Method to find the X coordinates of a given tile in a 2D array

Parameters

array	The 2D array to search through
index	The number/tile to search for

Returns

An integer value representing the X coordinate in a 2d Array

Here is the caller graph for this function:

3.8.3.6 int SearchAlgortihms.ManhattanDistance.findYCoordinate (int *array[][]*, int *index*)

Method to find the Y coordinates of a given tile in a 2D array

Parameters

array	The 2D array to search through
index	The number/tile to search for

Returns

An integer value representing the Y coordinate in a 2d Array

Here is the caller graph for this function:

3.8.3.7 void SearchAlgortihms.ManhattanDistance.setGoalArray (int *toSet[][]*)

Method to set the goalArray

Parameters

toSet	2D array to set too

Here is the caller graph for this function:

3.8.3.8 void SearchAlgortihms.ManhattanDistance.setStartArray (int *toSet[][]*)

Method to set the startArray

Parameters

toSet	2D array to set too

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchAlgortihms/ManhattanDistance.java

3.9 csm6120.State Class Reference

Public Member Functions

- **State** ()
- **State** (**State** s)
- void **addState** (int toAdd)
- void **printArray** ()
- int **returnIndex** (int i)
- void **changeTiles** (int i, int j)
- ArrayList **clone** ()
- boolean **compare** (**State** s)
- String **getStringtoString** ()
- int **compareMatching** (**State** s)
- int **getArraySize** ()
- ArrayList **getStateArray** ()

3.9.1 Detailed Description

This class has methods and variables to hold an input state. This will be used to hold the start and goal state object.

Author

stefan

3.9.2 Constructor & Destructor Documentation

3.9.2.1 csm6120.State.State ( )

Constructor for the **State** (p. 14) object creates an empty arrayList in which the state data will be saved

3.9.2.2 csm6120.State.State (State *s*)

Constructor for the **State** (p. 14) object creates a deep clone of the state object which is specified in the parameter field

Parameters

s	The state to clone

3.9.3 Member Function Documentation

3.9.3.1 void csm6120.State.addState (int *toAdd*)

Method to add an integer to the arrayList

Parameters

toAdd	The integer to add

Here is the caller graph for this function:

3.9.3.2 void csm6120.State.changeTiles (int *i*, int *j*)

Method to exchange to tiles

Parameters

i	Index of the tile to change
j	Index of the Empty tile to change

Here is the caller graph for this function:

3.9.3.3 ArrayList csm6120.State.clone ( )

This method clones the arrayList and returns it

Returns

The cloned arrayList

3.9.3.4 boolean csm6120.State.compare (State *s*)

Method to compare this object to another state object

Parameters

s	The state to compare too

Returns

True if the states are the same, false if not

Here is the caller graph for this function:

3.9.3.5 int csm6120.State.compareMatching (State *s*)

Method to return how many integers in this object compared to another object match

Parameters

s	The state to compare too

Returns

The number of matching ints

3.9.3.6 int csm6120.State.getArraySize ( )

Method to return the size of the state array

Returns

int value of the state array size

3.9.3.7 ArrayList csm6120.State.getStateArray ( )

Method to return the state ArrayList

Returns

The state ArrayList

Here is the caller graph for this function:

3.9.3.8 String csm6120.State.getStringtoString ( )

Method to return the string representation of the "state" ArrayList

Returns

The toString representation of the "state" ArrayList

Here is the caller graph for this function:

3.9.3.9 void csm6120.State.printArray ( )

Print the ArrayList

Here is the caller graph for this function:

3.9.3.10 int csm6120.State.returnIndex (int *i*)

Method to return the index of a specific item in the ArrayList

Parameters

i	The item to search for

Returns

The position of the item in the ArrayList

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/csm6120/State.java

3.10 SearchAlgortihms.StateComparator Class Reference

Inheritance diagram for SearchAlgortihms.StateComparator:

Collaboration diagram for SearchAlgortihms.StateComparator:

Public Member Functions

- int **compare** (**TreeNode** o1, **TreeNode** o2)

3.10.1 Detailed Description

This class is used to compare to states together. Implements the Comparator interface

Author

Stefan

3.10.2 Member Function Documentation

3.10.2.1 int SearchAlgortihms.StateComparator.compare (TreeNode *o1,* TreeNode *o2*)

Method to compare 2 TreeNode objects for order. Compares 2 objects state string representation and orders them based on their natural ordering i.e. 0 1 2 3 4 5 6 7 8

Parameters

o1	TreeNode object 1 to compare
o2	TreeNode object 2 to compare

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchAlgortihms/StateComparator.java

3.11 SearchTree.TreeNode Class Reference

Public Member Functions

- **TreeNode** (**State** s)
- **TreeNode** (**TreeNode** t)
- void **addChild** (**TreeNode** child)
- void **addSibling** (**TreeNode** sibling)
- **State getState** ()
- **TreeNode getFirstChild** ()
- **TreeNode getFirstSibling** ()
- boolean **siblingIsEmpty** ()
- boolean **childrenIsEmpty** ()
- **TreeNode peekChild** ()

- void **removeFirstChild** ()
- void **setExplored** (boolean b)
- boolean **getExplored** ()
- int **getNumOfChildren** ()

3.11.1 Detailed Description

This class represents a node in the search tree/graph

Author

stefan

3.11.2 Constructor & Destructor Documentation

3.11.2.1 SearchTree.TreeNode.TreeNode (State *s*)

Constructor of the **TreeNode** (p. 17) class. Creates a deep copy of the state which is passed as parameter Initialises the linkedLists.

Parameters

s	The State the node referes too

3.11.2.2 SearchTree.TreeNode.TreeNode (TreeNode *t*)

Constructor of the **TreeNode** (p. 17) class. Creates a deep copy of another **TreeNode** (p. 17) object.

Parameters

t	The **TreeNode** (p. 17) object this instance is a copy off

Here is the call graph for this function:

3.11.3 Member Function Documentation

3.11.3.1 void SearchTree.TreeNode.addChild (TreeNode *child*)

Method to add a child to the linkedList

Parameters

child	The **TreeNode** (p. 17) object to add to the children list

Here is the caller graph for this function:

3.11.3.2 void SearchTree.TreeNode.addSibling (TreeNode *sibling*)

Method to add a sibling to the linkedList of siblings

Parameters

sibling	The **TreeNode** (p. 17) object to add to the siblings list

3.11.3.3 boolean SearchTree.TreeNode.childrenIsEmpty ( )

Method to check if the **TreeNode** (p. 17) object has children. Returns true if the linkedList is empty

Returns

Boolean "True" if the list is empty

Here is the caller graph for this function:

3.11.3.4 boolean SearchTree.TreeNode.getExplored ( )

Method to get the "explored" variable of the object

Returns

The boolean value of "explored"

3.11.3.5 TreeNode SearchTree.TreeNode.getFirstChild ( )

Method to return(poll) and remove the first element of the "Children" linkedList

Returns

The head of the "children" linkedList

Here is the caller graph for this function:

3.11.3.6 TreeNode SearchTree.TreeNode.getFirstSibling ( )

Method to return(poll) and remove the first element of the "siblings" linkedList

Returns

The head of the "siblings" linkedList

3.11.3.7 int SearchTree.TreeNode.getNumOfChildren ( )

Method to return the size of the "children" linkedList

Returns

The size of the List

3.11.3.8 State SearchTree.TreeNode.getState ( )

Method to return the state of the node object

Returns

the State object of the node

Here is the caller graph for this function:

3.11.3.9 TreeNode SearchTree.TreeNode.peekChild ( )

Method to peek(return but not remove) the head of the "children" linkedList

Returns

The head of the "children" LinkedList

Here is the caller graph for this function:

3.11.3.10 void SearchTree.TreeNode.removeFirstChild ( )

Method to remove the head of the children linkedList

Here is the caller graph for this function:

3.11.3.11 void SearchTree.TreeNode.setExplored (boolean *b*)

Method to set the "explored" variable of the object

Parameters

b	The boolean value to set

3.11.3.12 boolean SearchTree.TreeNode.siblingIsEmpty ( )

Method to check if the **TreeNode** (p. 17) object has siblings. Returns true if the linkedList is empty

Returns

Boolean "True" if the list is empty

The documentation for this class was generated from the following file:

- C:/Users/Stefan/Documents/GitHub/CSM6120_Assignment2/src/SearchTree/TreeNode.java