CS26110 Assignment Report

Introduction

The task of pathfinding can be solved in a variety of different manners using Artificial Intelligence. In this situation I had to develop and code a range of different heuristics for solving a maze which contains treasures. In exploring the problem two different types of heuristics must be considered: admissible heuristics and non admissible heuristics.

A heuristic is defined as admissible if it never over-estimates the cost of achieving the end goal therefore it is guaranteed that an admissible heuristic will find the shortest path to a solution wherever possible[1][2]. There are a variety of different admissible heuristics that I could have used on the maze. In terms of the admissible heuristics I will be using the following heuristics: Manhattan Distance and Straight Line Distance. The Manhattan Distance measures the distance between the two points using right angles[3][4]. Straight Line Distance calculates the distance between two points by using a diagonal line and Pythagoras theorem i.e. the diagonal lines distance (where in this case is c) would be calculated using $a^2+b^2=c^2$[5].

A non admissible heuristic is a heuristic which may potentially over-estimate the distance to the goal. In devising my non admissible heuristic for the maze explorer I decided that it would be logical for the heuristic to assume that when the maze explorer finds a piece of treasure that it is closer to the goal, this on the surface of the problem appears to be a rather rational approach to creating a heuristic however it is non-admirable due to the fact that collecting a treasure doesn't always guarantee that the maze explorer is closer to the final goal. In terms of the actual code I wrote for my non-admissible heuristic I wrote it such that if a treasure is found in a maze cell then the distance is decreased by using a scale of the maze size depending on how many treasures are left.

Admissible Heuristics

Manhattan Distance is an admissible heuristic due to the fact that it never over-estimates the distance to the goal. In the Maze project Manhattan Distance is implemented using the following lines of code:

```
'public int getManhattanDist(MazeCell other) {
      return Math.abs(other.x - x) + Math.abs(other.y - y);
   }'.[6]
```

This is admissible due to the fact that the method specifically compares the different co-ordinates of the current maze cell to the 'other' maze cell this therefore means that the heuristic never over-estimates the distance to the goal. The fact that the method uses the Math.abs method ensures that the calculation is specific as the Java API states that .abs 'Returns the absolute value of an int value.' [7] Consequently we can conclude that the ManhattanDist method in the code is admissible due to the fact that it never over-estimates the distance to the goal.

The other admissible heuristic I implemented is Straight Line Distance. I implemented the heuristic using the following lines of code:

```
'public class StraightLineDistance implements BestFirstHeuristic<MazeExplorer> {
   public int getHeuristic(MazeExplorer node, MazeExplorer goal) {
         int result;
```

```
        int finalX= (int) (goal.getLocation().X() - node.getLocation().X());
        int finalY= (int) (goal.getLocation().Y() - node.getLocation().Y());
        result=(int) Math.round(Math.sqrt((finalX*finalX) + (finalY*finalY)));
        return result;

}
}'
```

this is an admissible heuristic due to the fact that it never over-estimates the distance to the goal; the code works out the distance between different maze cells by using straight line distance which is a practical implementation of Pythagoras theorem. The code works out the specific distance between the goal node location and the node location by determining the diagonal line by using Pythagoras, as a result of using Pythagoras theorem it thus means that it does not over-estimate the distance to the goal and therefore the heuristic is admissible.

In contrasting and comparing the Manhattan Distance heuristic to the straight line distance it must be considered that both are different techniques for calculating distances outside of the area of Artificial Intelligence and can be used in various other situations such as geometry. Straight line distance only calculates the distance to the goal by using straight lines along the X and Y axis (i.e. only completely straight horizontal and vertical lines) where as Manhattan distance actually calculates the distance using a diagonal line which is determined by using Pythagoras theorem.

For the specific problem of navigating through a maze one would expect that the Manhattan Distance would be more efficient in the sense that it will have the exact same depth yet it will generally require less nodes as straight line heuristic is dependant on only sensing routes via using straight horizontal and vertical lines, where as Manhattan Distance can find more efficient ways to navigate through the maze. Below I have produced a few tables which shows the results of running experiments (in the maze) using the Manhattan Distance and Straight line heuristics , which prove what I just said. Above all of the tables I have specified what the maze I created was.

Maze was 10 by 10 with 2 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Straight Line Distance	266	76	77
Manhattan Distance	264	76	77

Maze was 10 by 10 with 5 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Straight Line Distance	1897	72	73
Manhattan Distance	1871	72	73

Maze was 10 by 10 with 5 treasures and approximatively 50% perfection.

	Nodes	Depth	Solution Length
Straight Line Distance	7138	31	31
Manhattan Distance	5424	31	31

Maze was 100 by 100 with 2 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Straight Line Distance	57780	4484	4485
Manhattan Distance	57621	4484	4485

Maze was 100 by 100 with 5 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Straight Line Distance	331600	4768	4769
Manhattan Distance	331316	4768	4769

Maze was 100 by 100 with 5 treasures and approximatively 50% perfection.

	Nodes	Depth	Solution Length
Straight Line Distance	642266	358	359
Manhattan Distance	513721	358	359

The fact that both straight line distance and Manhattan distance have the same solution length thus proves that they are both admissible as they are both finding the shortest path where possible and consequently are not over-estimating the path to the goal.

Non Admissible Heuristic

I created a non admissible heuristic which used determined whether a treasure had been found. If a treasure had been found then the heuristic comes to the assumption that if a user has more treasure they are closer to the goal, this however isn't necessarily the case and may lead to an overestimation of the goal and thus the heuristic is non admissible. I created the heuristic using the following code:

```
'public class NonAdmissible implements BestFirstHeuristic<MazeExplorer> {
   public int getHeuristic(MazeExplorer node, MazeExplorer goal) {
        int mazeSize;
        mazeSize=node.getMaze().getXSize()*node.getMaze().getYSize(); //Assigns mazeSize to =
the size of the maze.
        boolean hasTreasure= node.getMaze().isTreasure(node.getLocation());
        if(hasTreasure){
                int treasureRemaining=node.getMaze().getTreasures().size()-1;
                return node.getLocation().getManhattanDist(goal.getLocation())-
(mazeSize/treasureRemaining);
        }
        else {
                return node.getLocation().getManhattanDist(goal.getLocation());
        }
```

```
}
}'
```

The heuristic uses the Manhattan Distance and actually incorporates the size of the maze as a scalar to assume that when there is a treasure found that the maze is a certain fraction closer to completion depending on the size of the maze; due to the fact that this isn't necessarily always the case this can lead to over-estimating to the distance of the goal.

The heuristic does actually provide some useful results and I carried out experiments as shown below where I compared the non admissible heuristic to Manhattan heuristic to prove that using the non admissible heuristic can return useful results.

10 by 10 maze with 2 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Manhattan Distance	538	84	85
Non Admissible Heuristic	546	84	85

10 by 10 maze with 5 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Manhattan Distance	1211	80	81
Non Admissible Heuristic	1223	80	81

10 by 10 maze with 5 treasures and approximatively 50% perfection.

	Nodes	Depth	Solution Length
Manhattan Distance	463	22	23
Non Admissible Heuristic	480	22	23

100 by 100 maze with 2 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Manhattan Distance	56387	372	373
Non Admissible Heuristic	55764	372	373

100 by 100 maze with 5 treasures and 100% perfection.

	Nodes	Depth	Solution Length
Manhattan Distance	605015	722	723
Non Admissible Heuristic	607525	722	723

100 by 100 maze with 5 treasures and 50% perfection.

	Nodes	Depth	Solution Length
Manhattan Distance	208798	234	235
Non Admissible Heuristic	211395	235	235

Rather interestingly in only one situation (where it was the 100 by 100 maze with 2 treasures and 100% perfection) was the non admissible heuristic actually more efficient than Manhattan distance due to the amount of nodes being less this therefore shows that non admissible heuristics can on the odd occasion produce more efficient results than admissible heuristics.

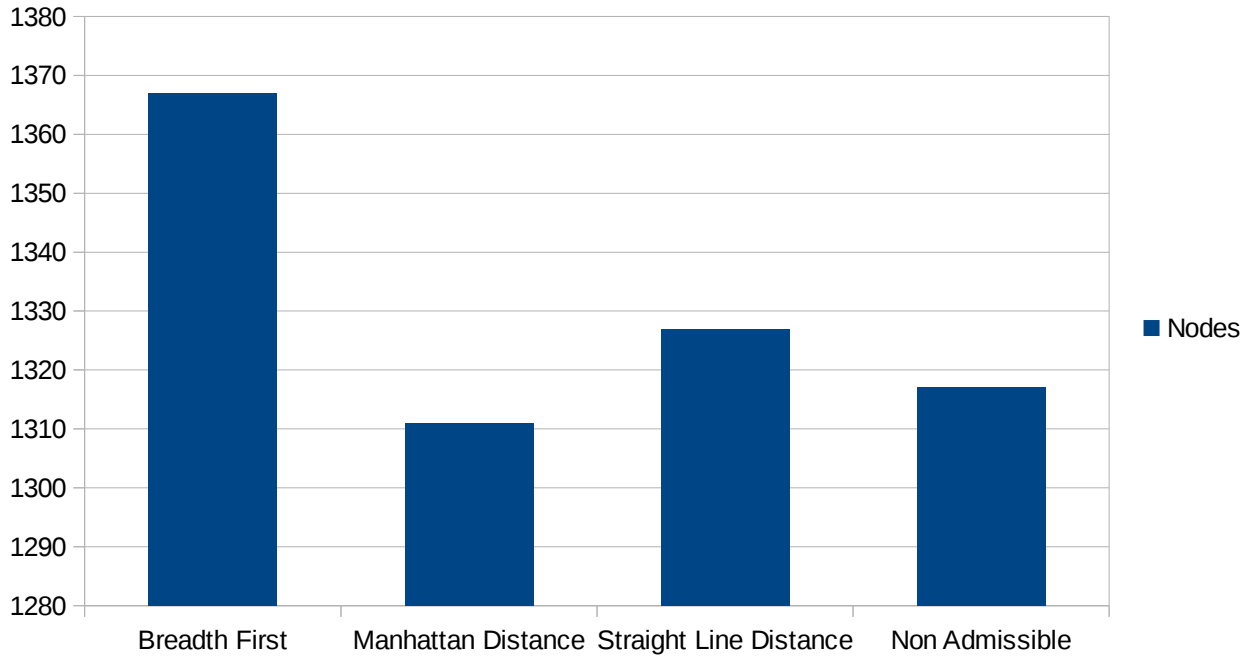
Experiments

I will now discuss and investigate all of the different heuristics I have implemented by running a series of experiments using the randomly generated mazes. I will also be comparing these results to the breadth first search technique which actually uses no heuristics at all as this will clearly highlight how useful heuristics can be. I will be determining the difficulty of the maze in three main ways: the size of the maze, the amount of treasure in the maze to be collected and the perfection of the maze. I will initially alter the difficulty variables so that the maze becomes more and more difficult.

I initially started with a rather simple maze which was of size 10 by 10, 100% perfection and contained 3 treasures, below I have shown the results in a table and on a graph. I didn't carry out any experiments with 0 treasures due to the fact that the heuristics actually use the treasures as information and consequently we would generally get the same result for the different heuristics and search methods if the experiment contained 0 treasures.

Search/Heuristic	Nodes	Depth	Solution Length
Breadth First	1367	106	107
Manhattan Distance	1311	106	107
Straight Line Distance	1327	106	107
Non Admissible	1317	106	107

As you can see from the table above the only thing which varied between the search techniques and heuristics was the nodes consequently on the graph below I have only shown the nodes.

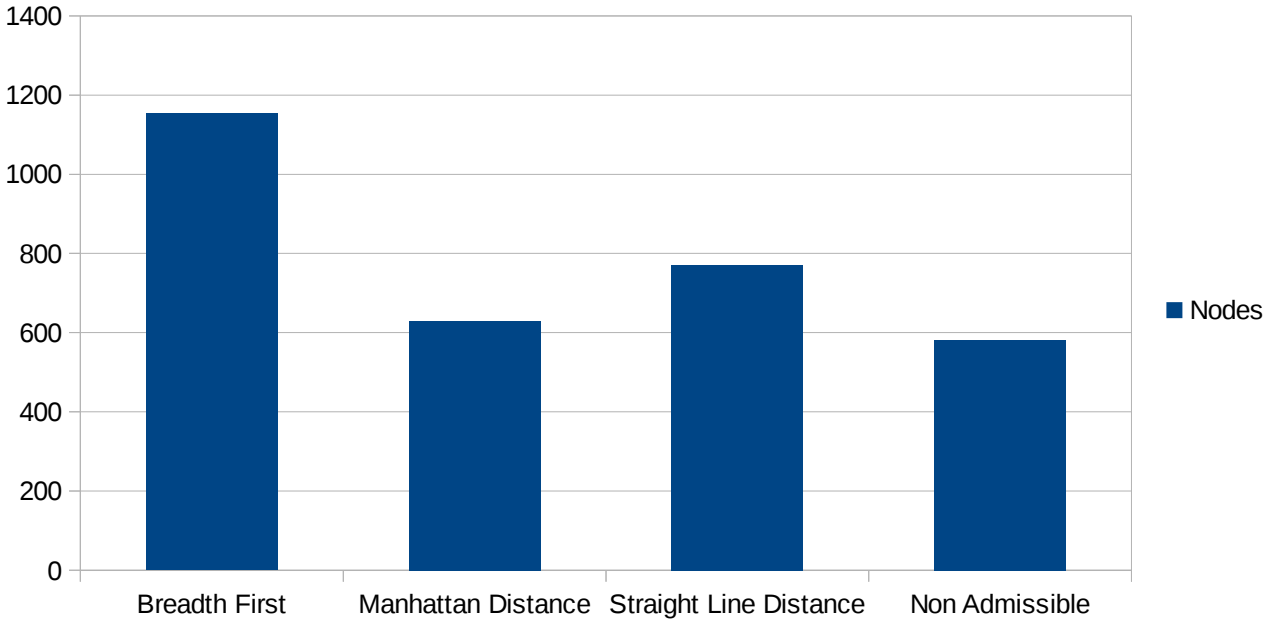


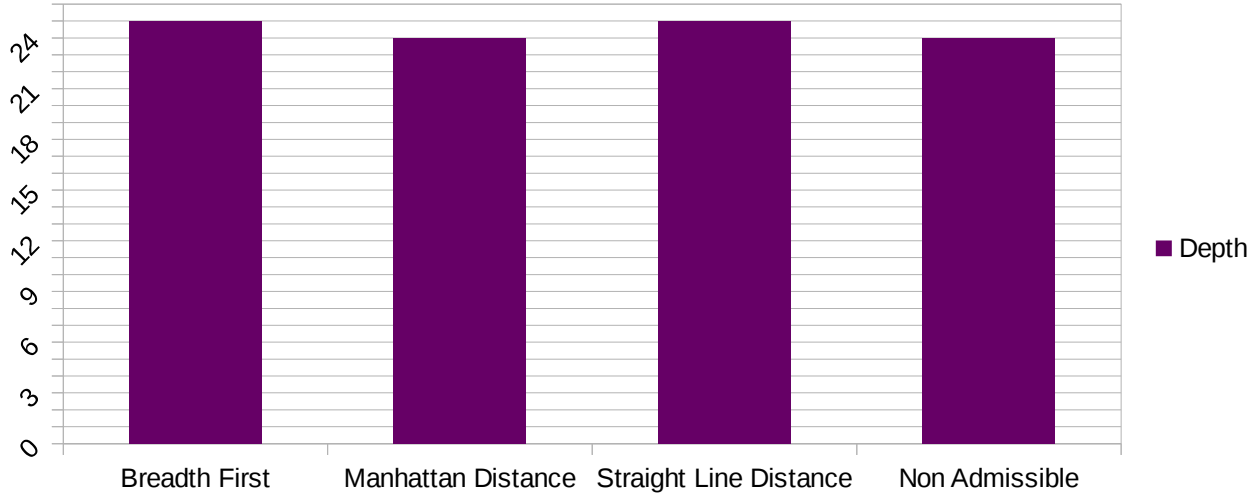
Clearly the less nodes the more efficient the search has been consequently from this experiment it can be concluded that Breadth First was the worst and the Manhattan Distance was the best heuristic.

The was a relatively easy maze due to it being rather small and only containing 3 treasures. The next experiment is more complex as I have lowered the perfection of the maze to approximately 80%, once again I have showed the results below.

Search/Heuristic	Nodes	Depth	Solution Length
Breadth First	1153	25	25
Manhattan Distance	630	24	25
Straight Line Distance	771	25	25
Non Admissible	582	24	25

In this case there was a slight difference between depth and some of the search techniques and heuristics as well as nodes, consequently I have created two graphs one to show the nodes and the other to show the depth.



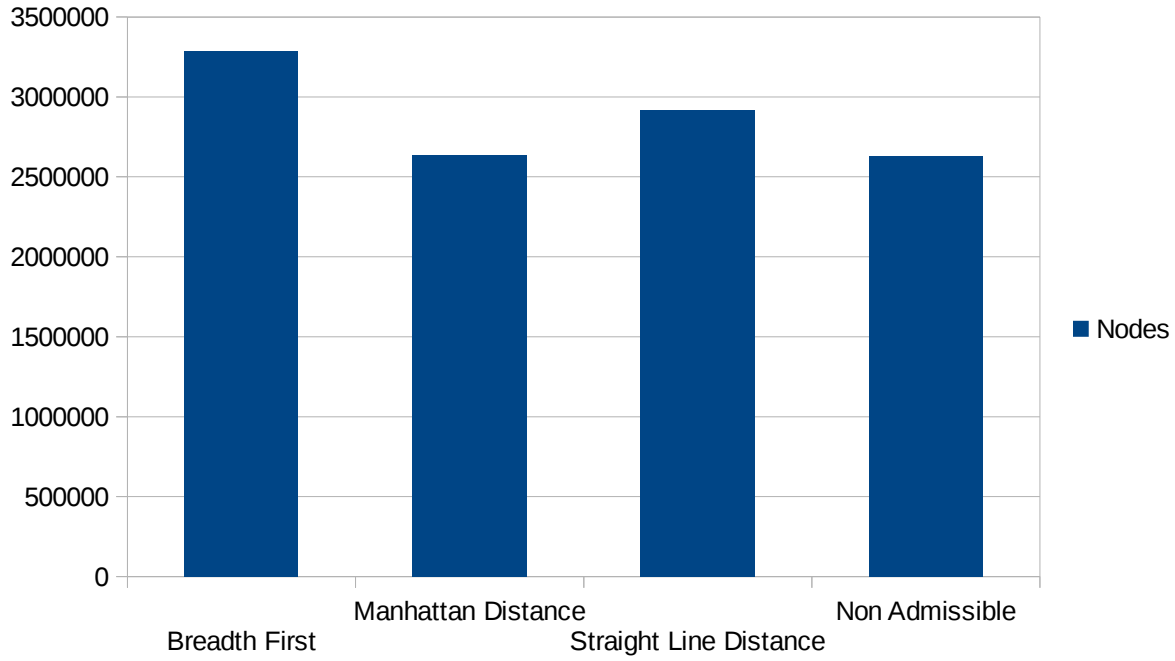


In this particular experiment due to the fact that the Non Admissible heuristic I created actually has a lower depth and a less amount of nodes consequently the Non Admissible heuristic is the most efficient search heuristic in this experiment, this therefore consequently proves that in some situations a Non Admissible heuristic could actually be more useful than an admissible one.

I then experimented by making the maze even more difficult, I done this by making the maze 100 by 100 having 7 treasures and by having approximatively 70% perfection.

Search/Heuristic	Nodes	Depth	Solution Length
Breadth First	3287598	479	479
Manhattan Distance	2637493	478	479
Straight Line Distance	2914110	478	479
Non Admissible	2630797	478	479

As with the previous experiment both the nodes and the depth are different however in this case I only had one graph to show the difference in the nodes as I felt that due to the difference in the depth only being one that there is no real need for a visualisation. (I did not have the graph below start at 0 as I felt that that would be a waste of space and having it set at a higher number would help in terms of expressing the difference between nodes.)

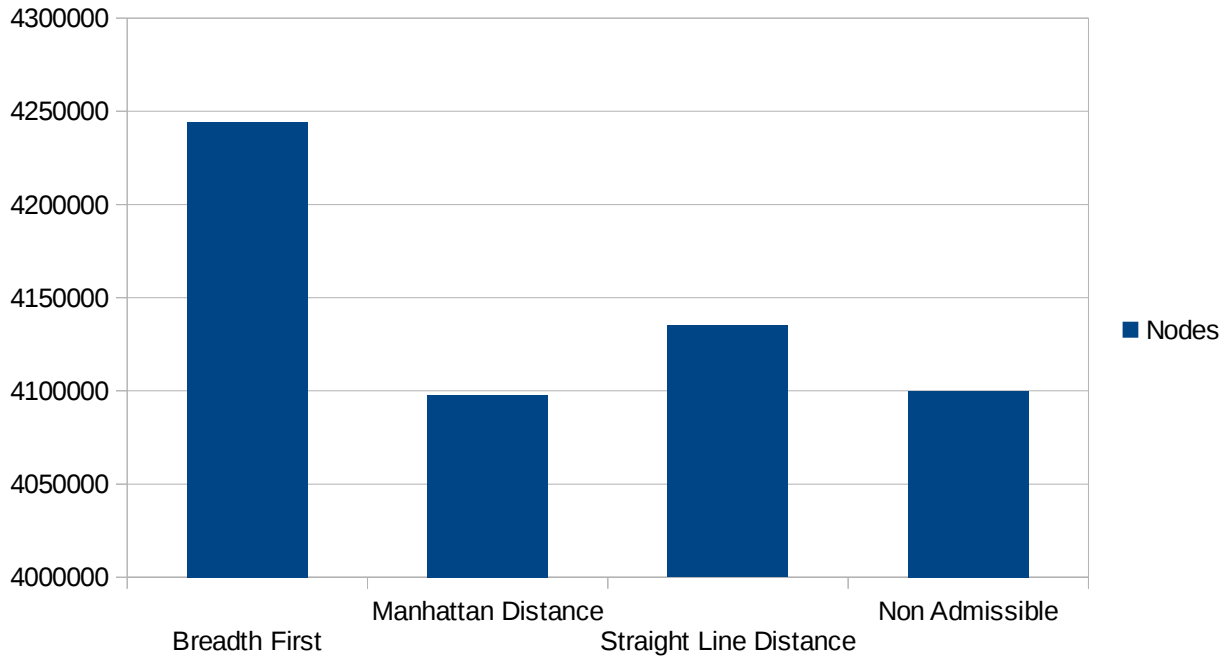


The graph above doesn't actually clearly highlight that there is a difference in the amount of nodes between the Manhattan Distance and the Non Admissible heuristic this is due to the fact that the number of nodes is so gigantic that the amount between them is practically irrelevant. Overall though the non admissible heuristic is better as it used 263079 nodes where as Manhattan Distance used 2637493 and the solution depth on both Non Admissible and Manhattan Distance was 478.

I decided that I would do one final experiment which would be an extremely difficult maze. I tried to experiment on a maze that was 500 by 500 approximately 50% perfection and contained 25% perfection, however any time I tried to solve the maze it led to my CPU working at 546.4% and it still did not solve the problem after 2 minutes (that is despite me running the experiment on a laptop which has an i7 processor and 8gb RAM). So unfortunately after experimenting to see what maze difficulty I could actually run I came to the conclusion that the hardest maze I could create was 100 by 100 with 10 treasures and 50% perfection, consequently the graph and the table below relate to that.

Search/Heuristic	Nodes	Depth	Solution Length
Breadth First	4244286	5749	5749
Manhattan Distance	4097826	5748	5749
Straight Line Distance	4134907	5748	5749
Non Admissible	4100044	5748	5749

As there was a variety in nodes I also created a graph to help with the visualisation of that.



In this experiment the Manhattan Distance has proved to be the most effective heuristic due to it having the least amount of nodes.

Overall, it must be said that both admissible and non admissible have their merits, but in all of the experiments I carried out Manhattan Distance is better that Straight Line Distance in terms of using admissible heuristics. Ultimately using heuristics is defiantly a more efficient way of path finding using Artificial Intelligence because in all of my experiments Breadth First Search was the least effective search technique.

Note: Word Count does not include title or references

References

[1] Stuart Russell and Peter Norvig,Artifical Intelligence A Modern Approach Third Edition, Pearson Education Inc 2010, Page 94.
[2] George F Luger and William A Stubfield, Artifical Intelligence Structures and Stratergies for Complex Problem Solving Second Edition, The Benjamin/Cummings Publishing Company, inc 1993, Page 132.
[3] Paul E. Black, "Manhattan distance", in *Dictionary of Algorithms and Data Structures* [online], Vreda Pieterse and Paul E. Black, eds. 31 May 2006. (accessed 5/11/14) Available from: http://www.nist.gov/dads/HTML/manhattanDistance.html
[4] Barile, Margherita. "Taxicab Metric." From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein. (acessed 5/11/14) Avalaible from: http://mathworld.wolfram.com/TaxicabMetric.html
[5] Oxford Dictionary http://www.oxforddictionaries.com/definition/english/Pythagoras%27-theorem (acessed 6/11/14)
[6] Maze Project provided via Blackboard. The getManhattanDist method is in the maze.core package and the MazeCell.java class.
[7] http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html (acessed on 6/11/14.)