Introduction to Artificial Intelligence

TDT4136 - Exercise 3

Fredrik B. Tørnvall & Aleksander Skraastad

Preface

All our answers and solutions to the exercise parts have been integrated into a single application, which is written in python using the Tkinter GUI framework. To run an algorithm, simply load a board from the board menu, choose the algorithm in mind from the algorithm menu, and run it. If you want to view all the states (e.g openset, closedset as well as the path taken), please select “Show all states” from the options menu and run the algorithm again.

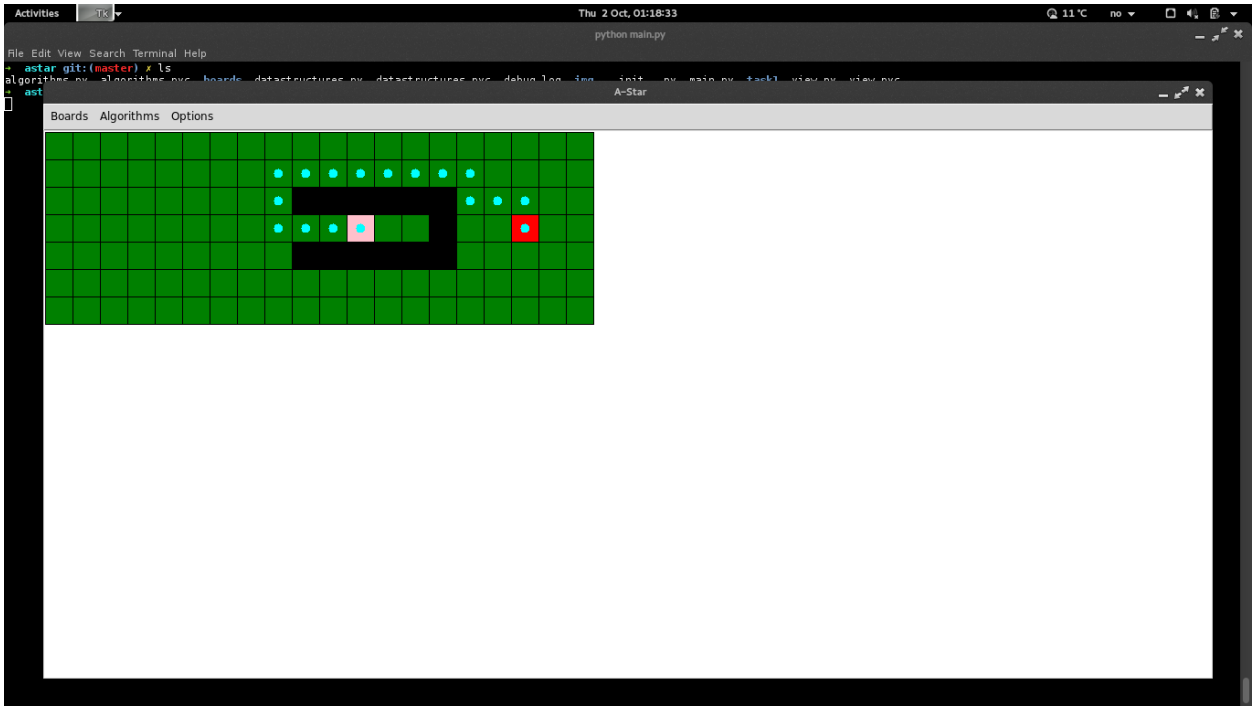
Problem A

Subproblem A1.1

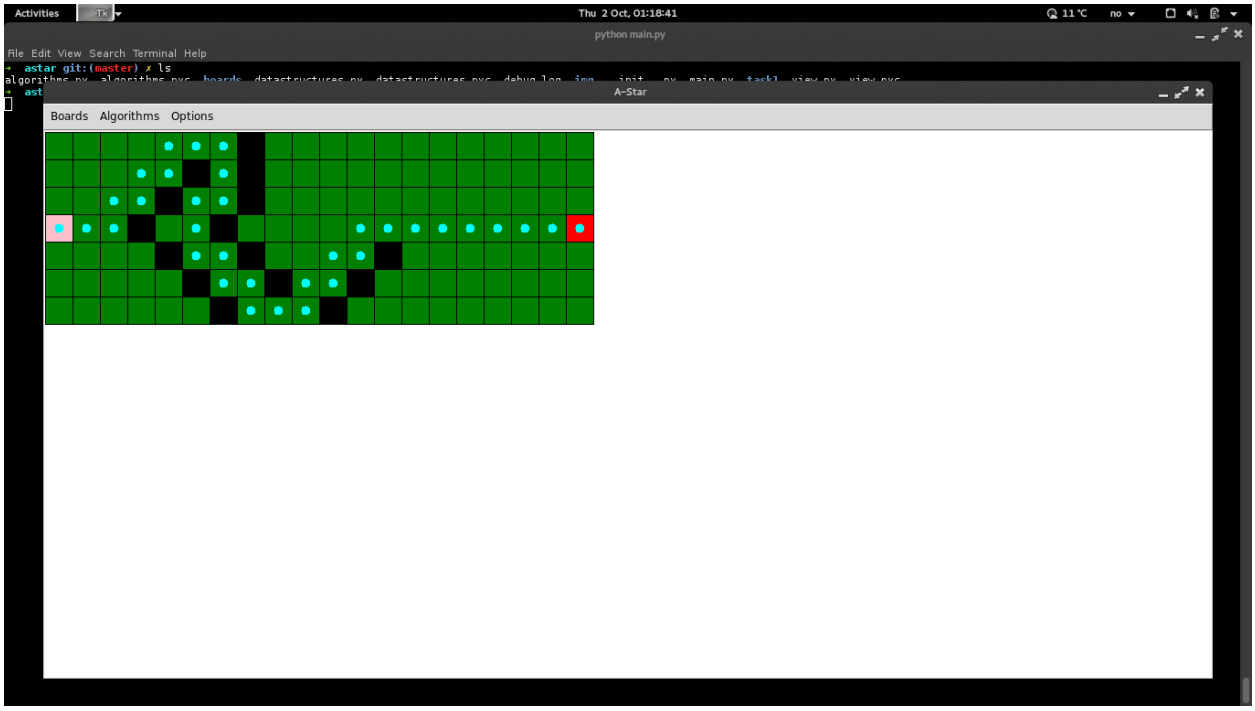
See the attatched zip-file with the complete source code. Our A* algorithm was written mostly ourselves, with inspiration from the excellent Wikipedia article on the algorithm.

Subproblem A1.2

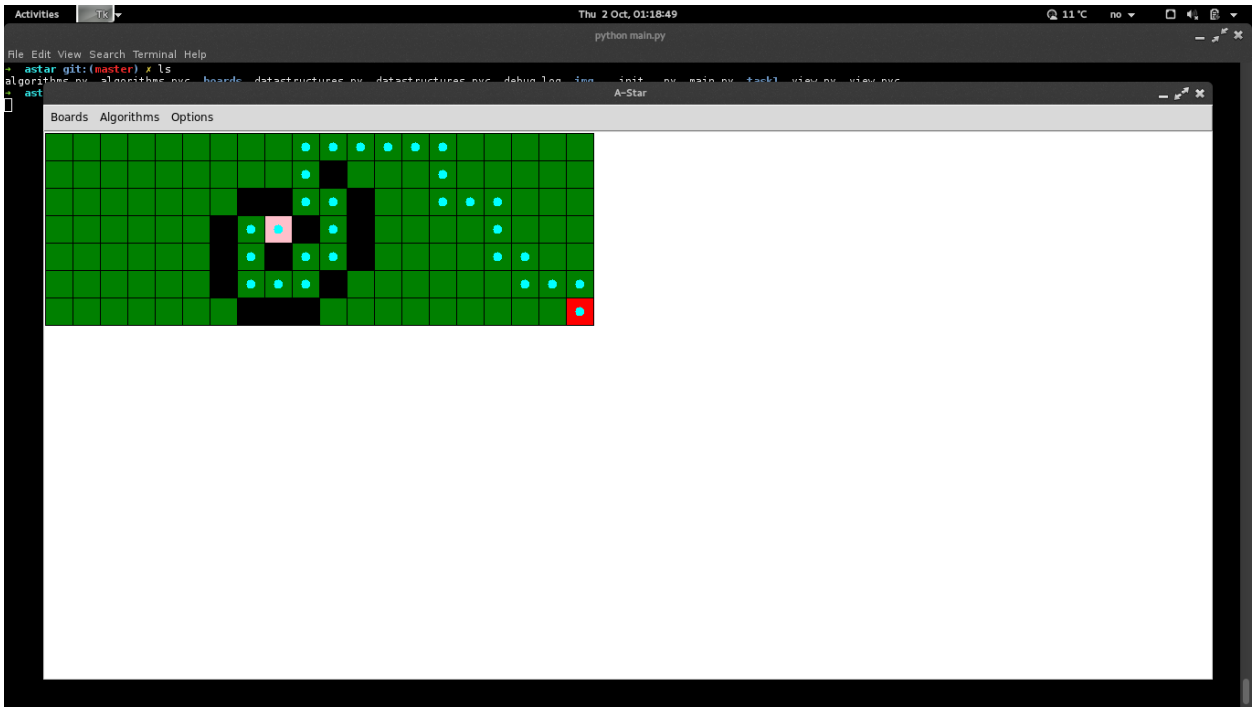
Board 1-1.txt



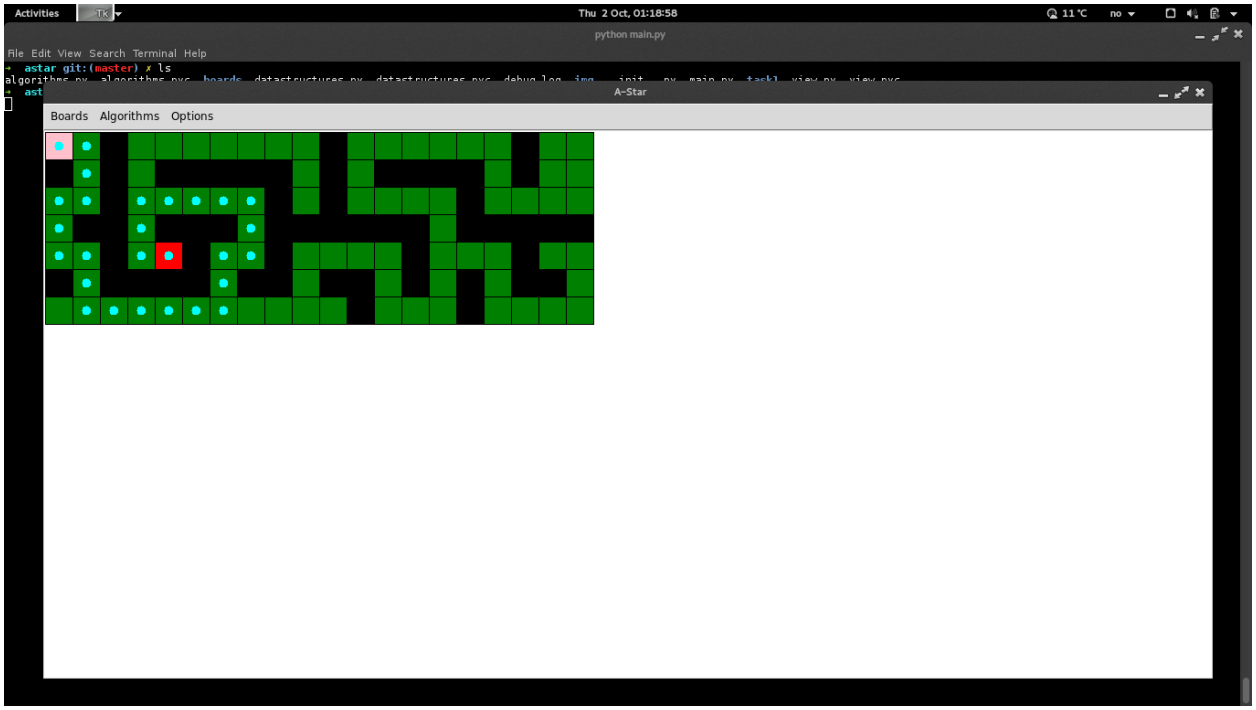
Board 1-2.txt



Board 1-3.txt



Board 1-4.txt

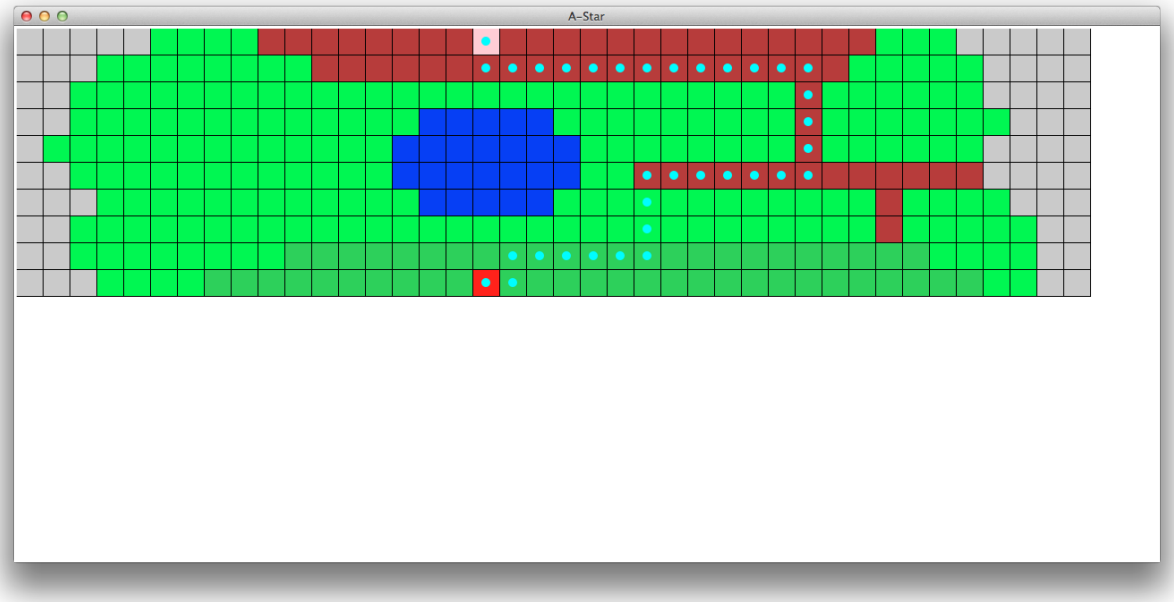


Subproblem A2.1

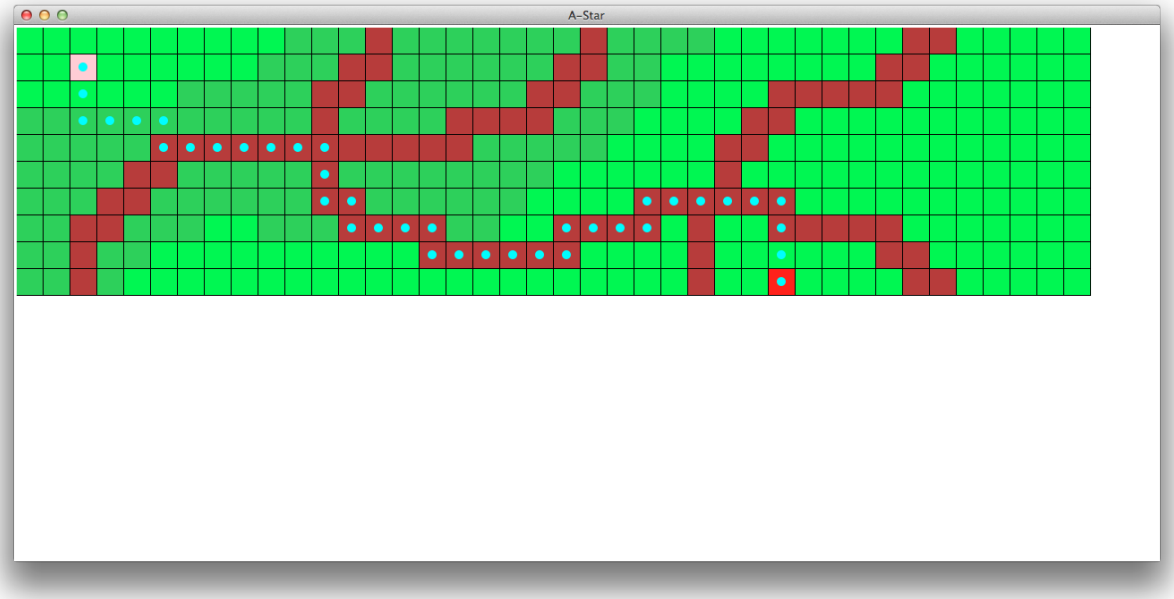
As stated previously, the the source code for this task is also wrapped into the same application provided in the attatched zip file.

Subproblem A2.2

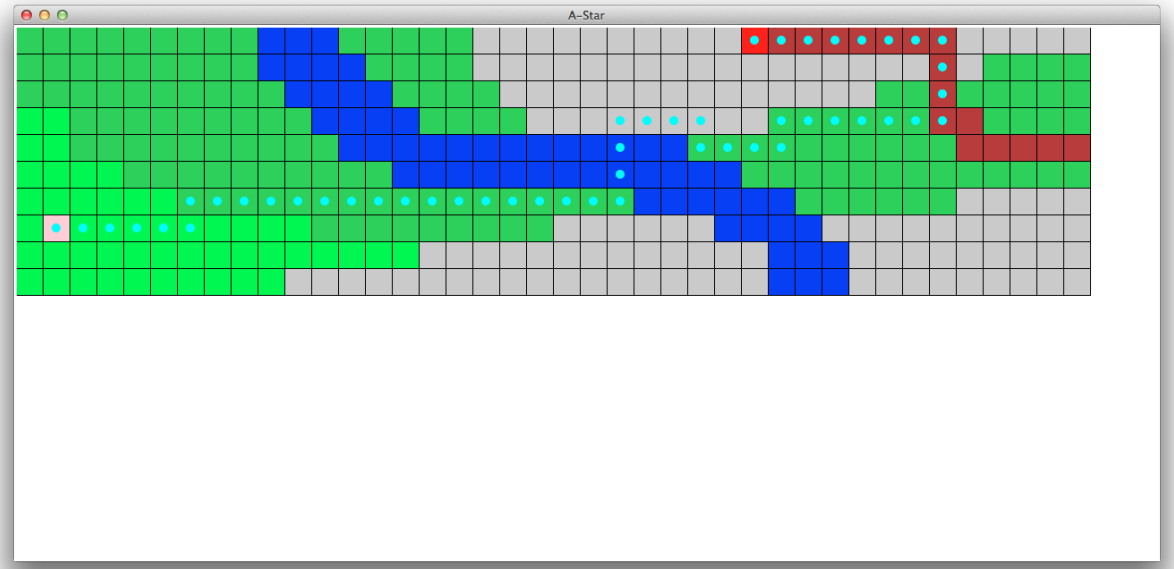
Board 2-1.txt



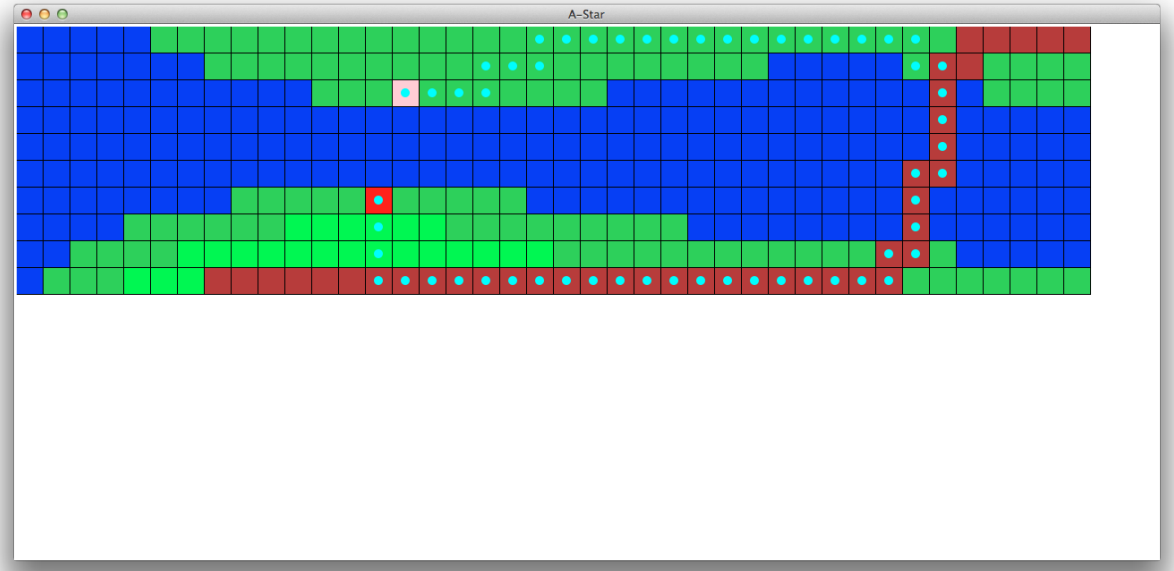
Board 2-2.txt



Board 2-3.txt



Board 2-4.txt



Subproblem A3.1

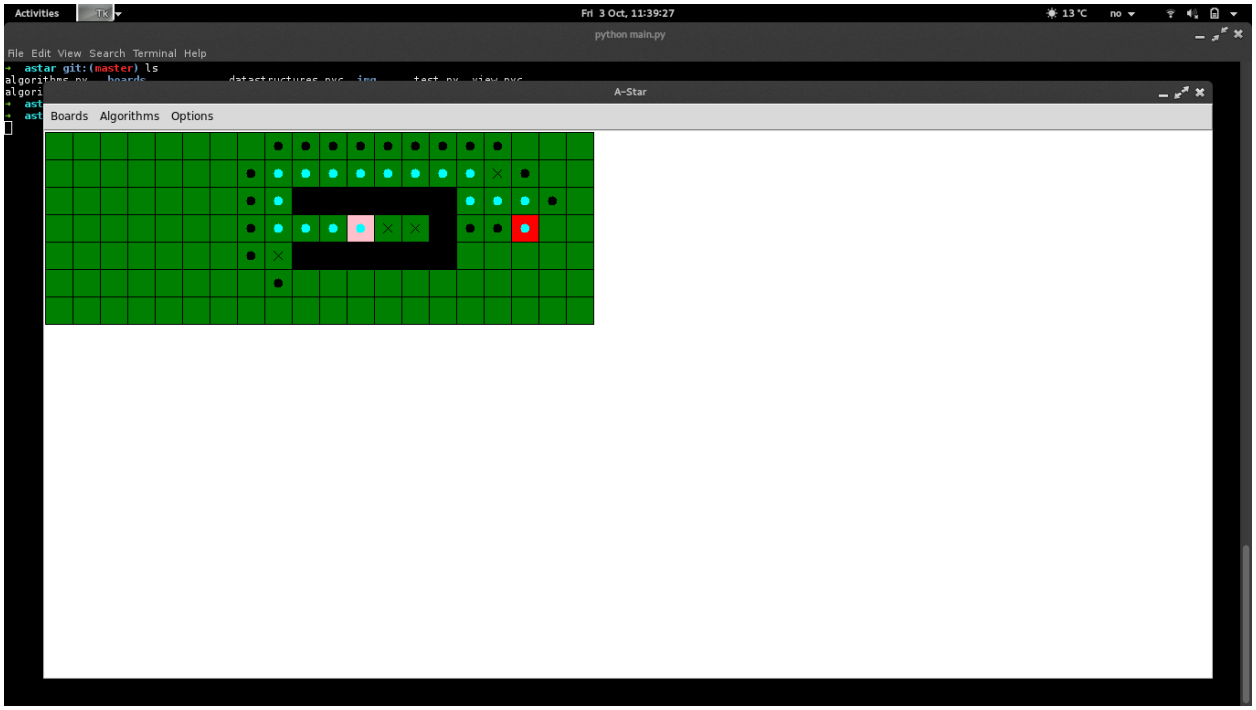
As stated previously, the the source code for this task is also wrapped into the same application provided in the attatched zip file.

Subproblem A3.2

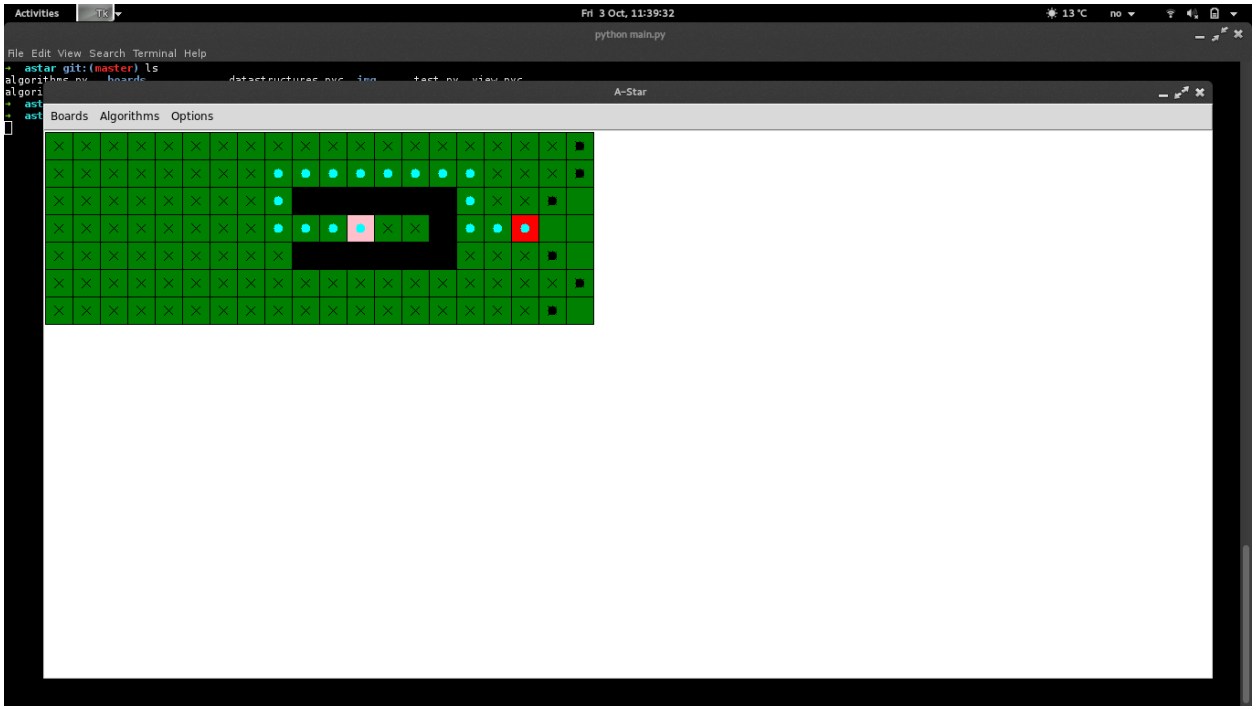
LEGEND: Black dots are in openset, crosses are in closedset, and cyan dots are the path taken.

Board 1-1.txt

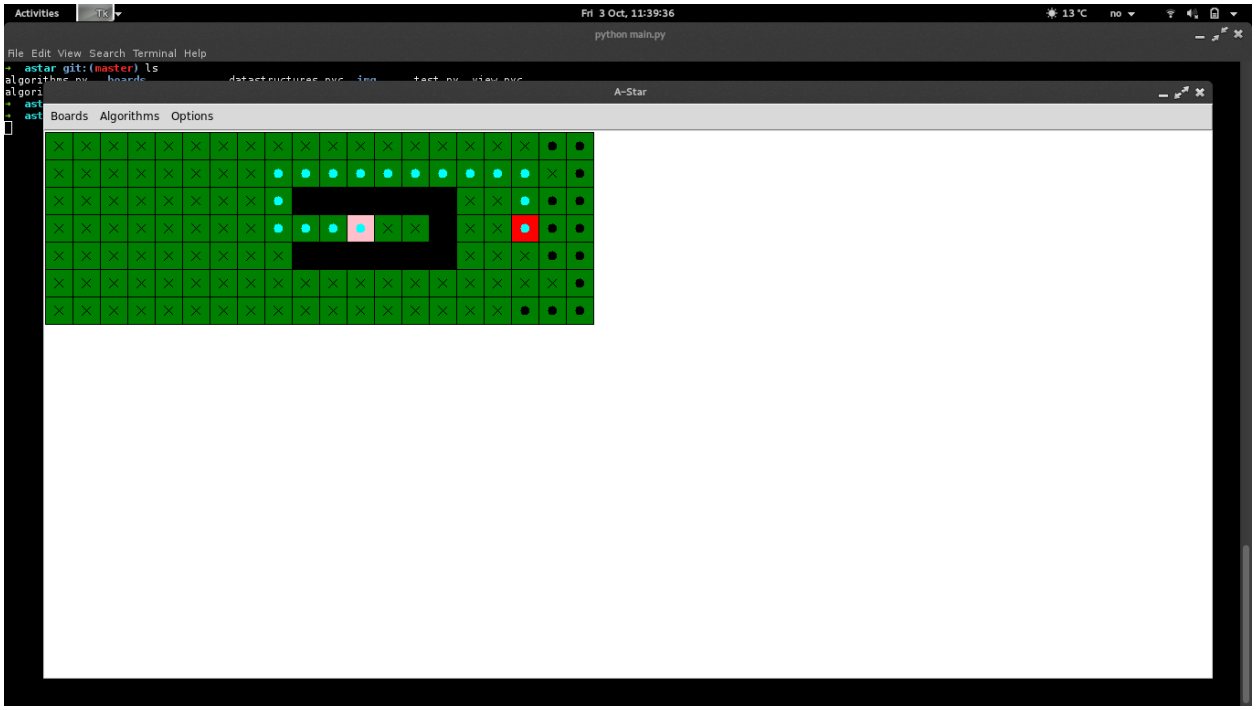
A*



BFS



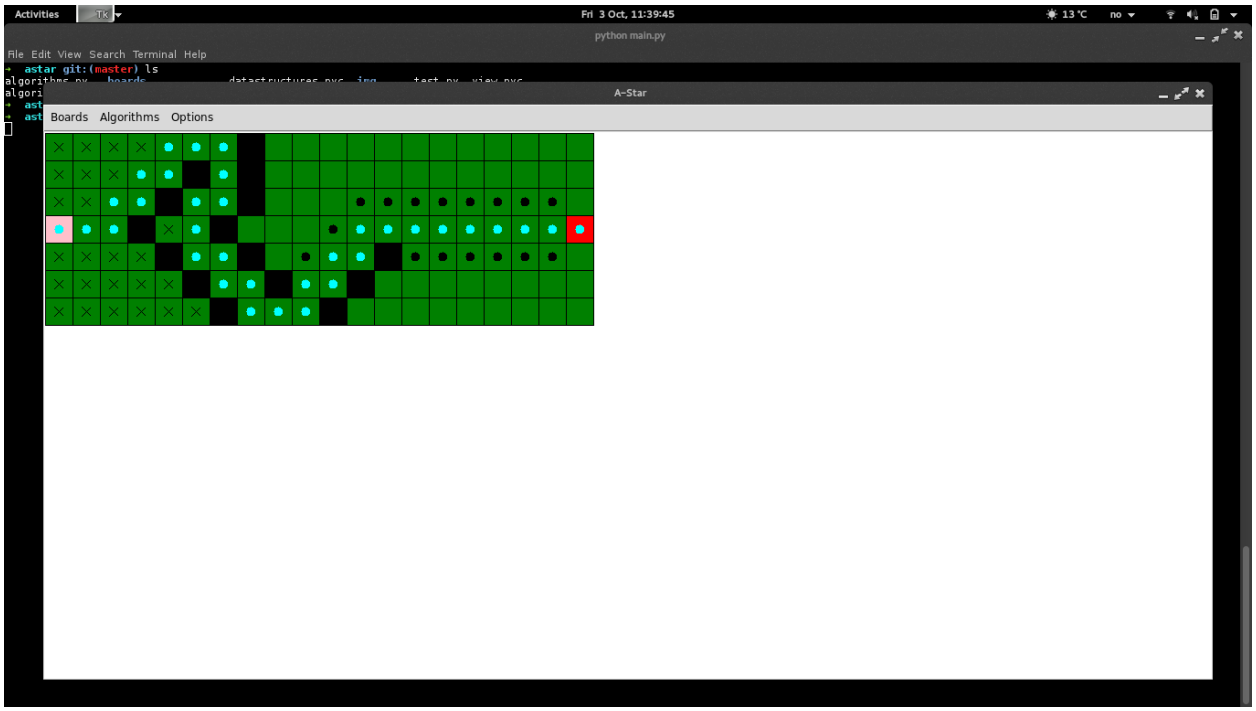
Dijkstra



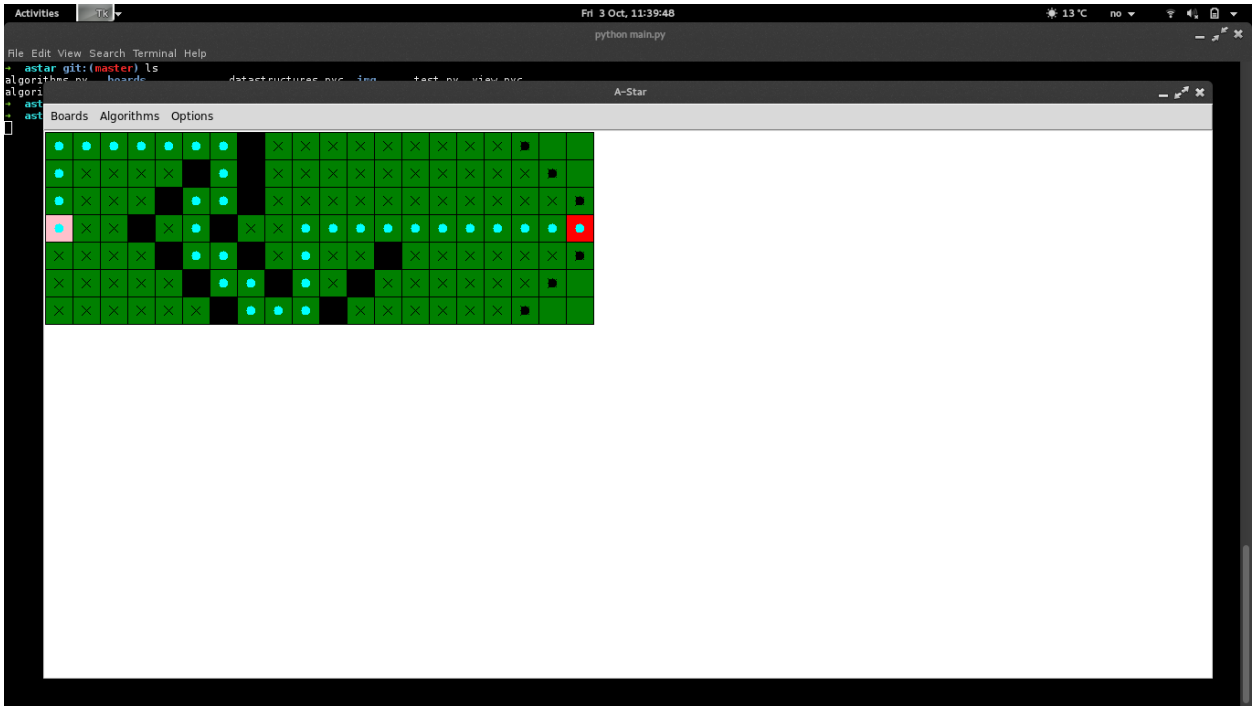
Comment:
Here we clearly see that A* has the best-first option, and the heuristics really come in hand, as all the nodes walkable have the same cost. Dijkstra and BFS look very similar here.

Board 1-2.txt

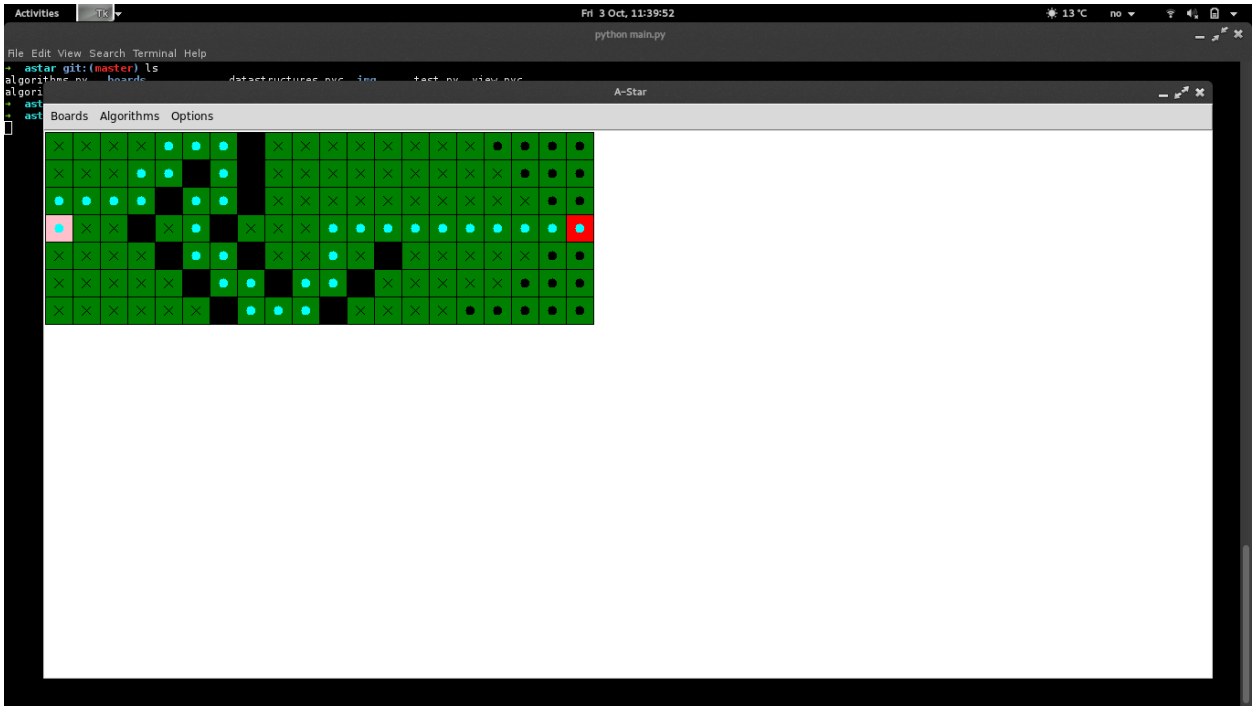
A*



BFS



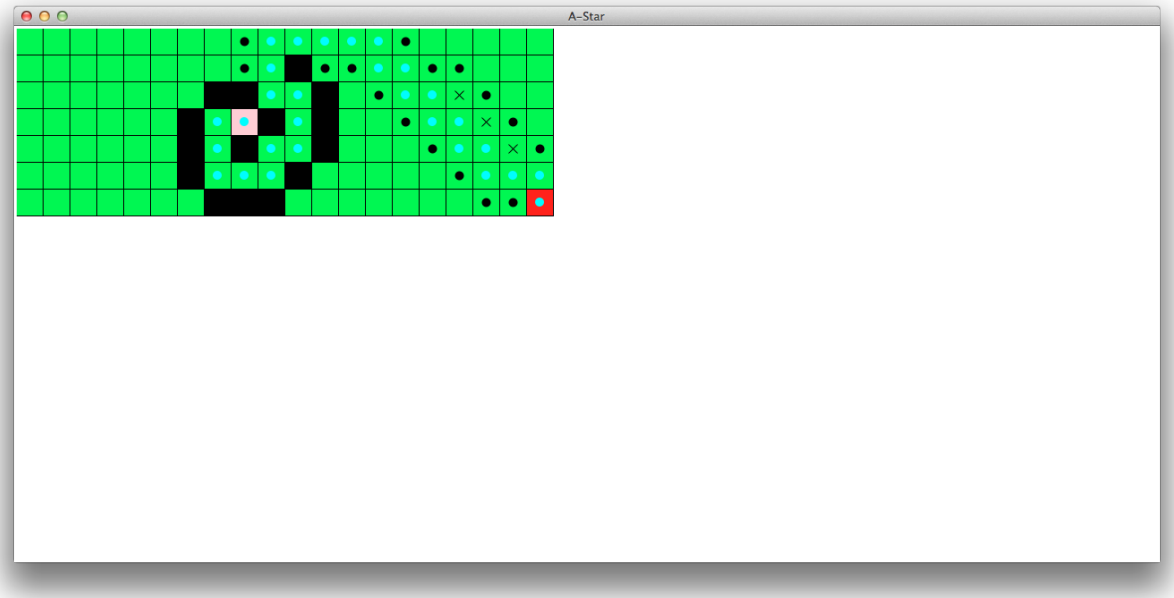
Dijkstra



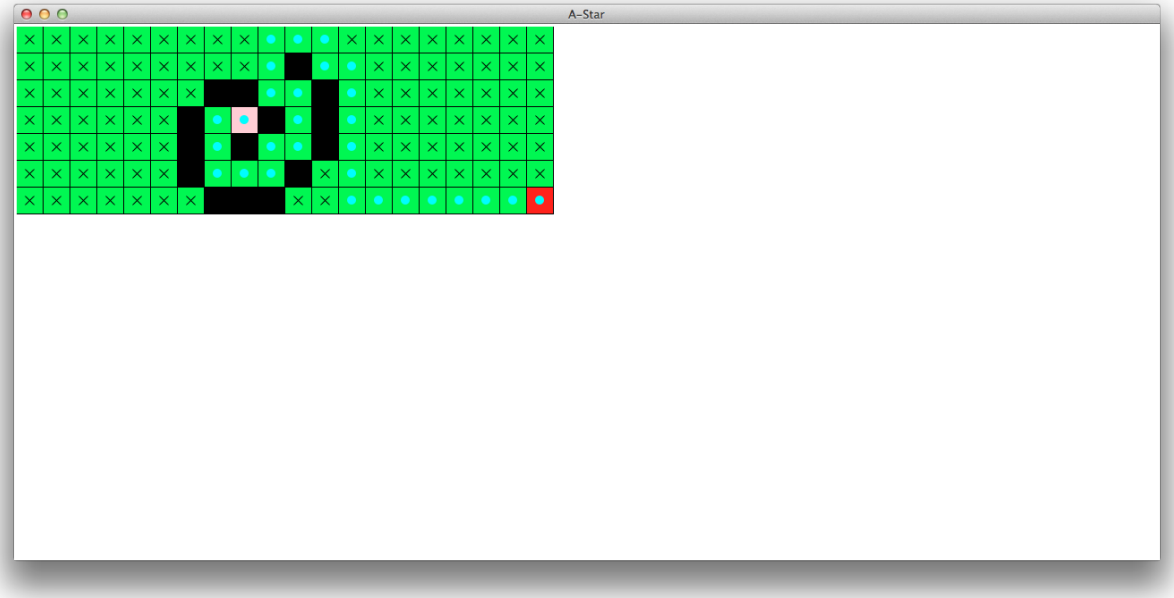
Comment:
On this board, we see that the A* algorithm clearly takes advantage of the H-function (in our case, Euclidean distance). But the differences in BFS and Dijkstra are more apparent. Dijkstra adds all the nodes into the queue at start, and BFS will on a grid like this with no diagonal movement create a diamond (like on cards) shape.

Board 1-3.txt

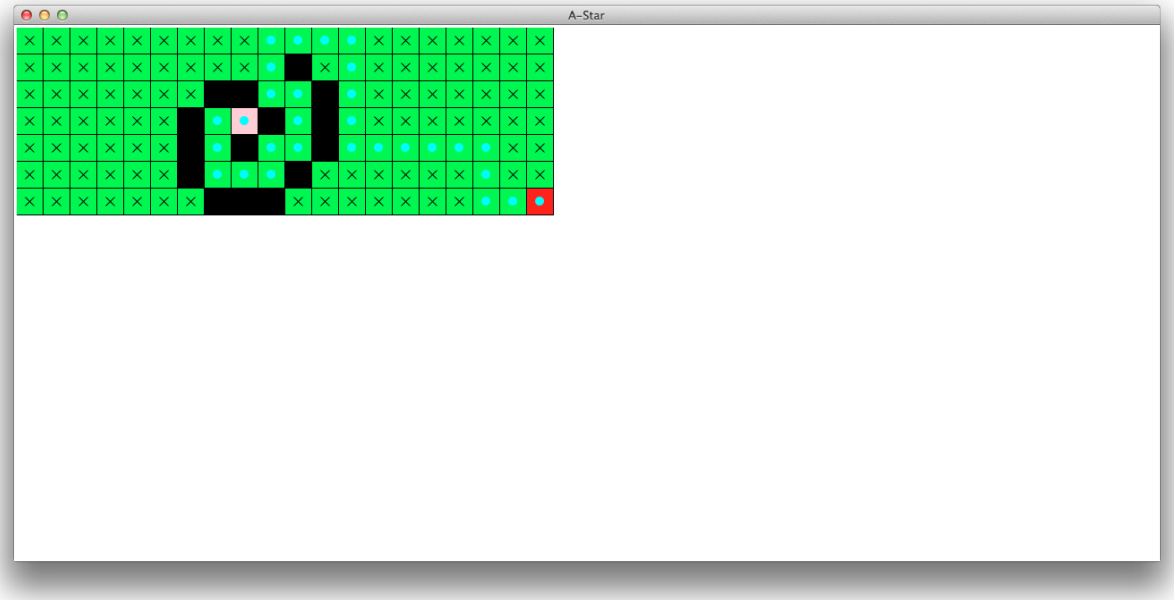
A*



BFS



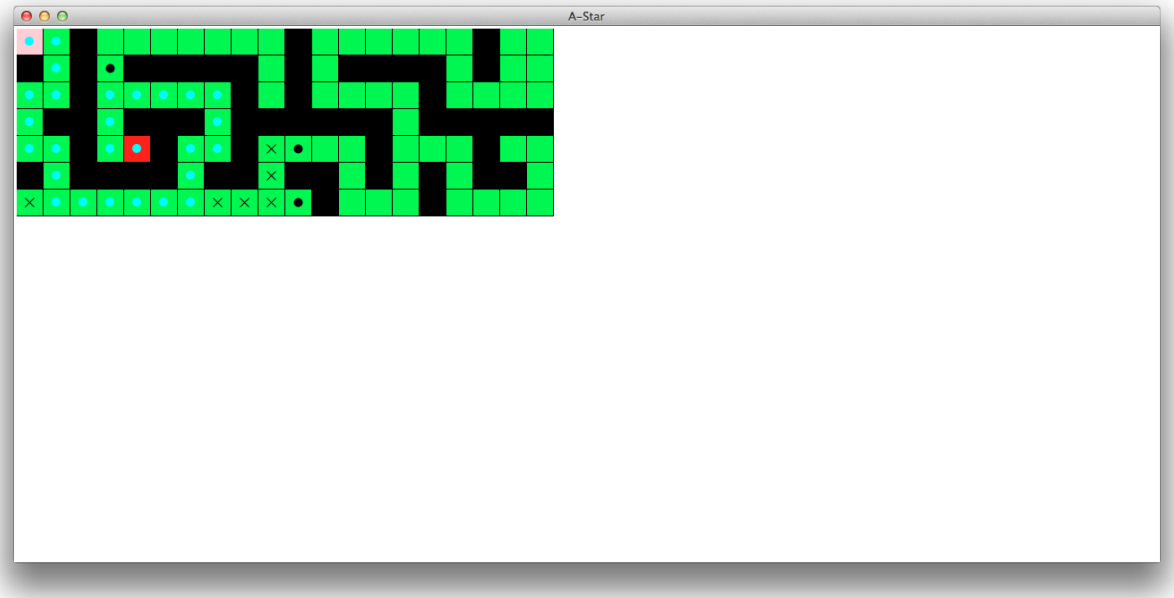
Dijkstra



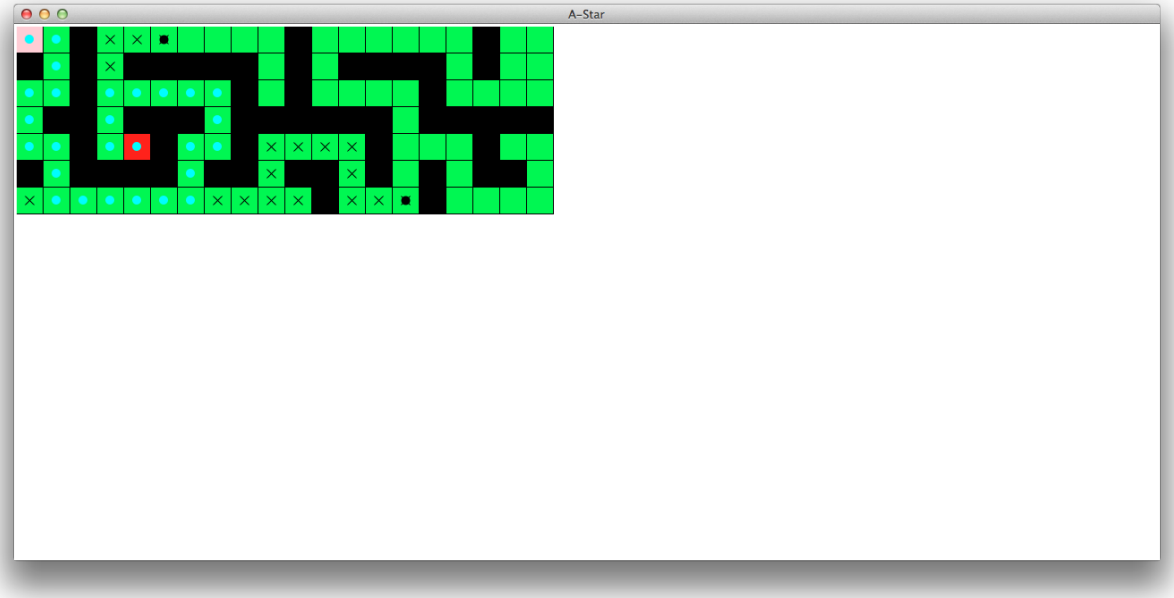
Comment:
This board is quite interesting when comparing BFS and Dijkstra. Since the goal node is at the end of our matrix, both Dijkstra and BFS will check this node last. BFS because of its diamond shape exploration of the matrix, and Dijkstra because that node was the last one to get added into the queue.

Board 1-4.txt

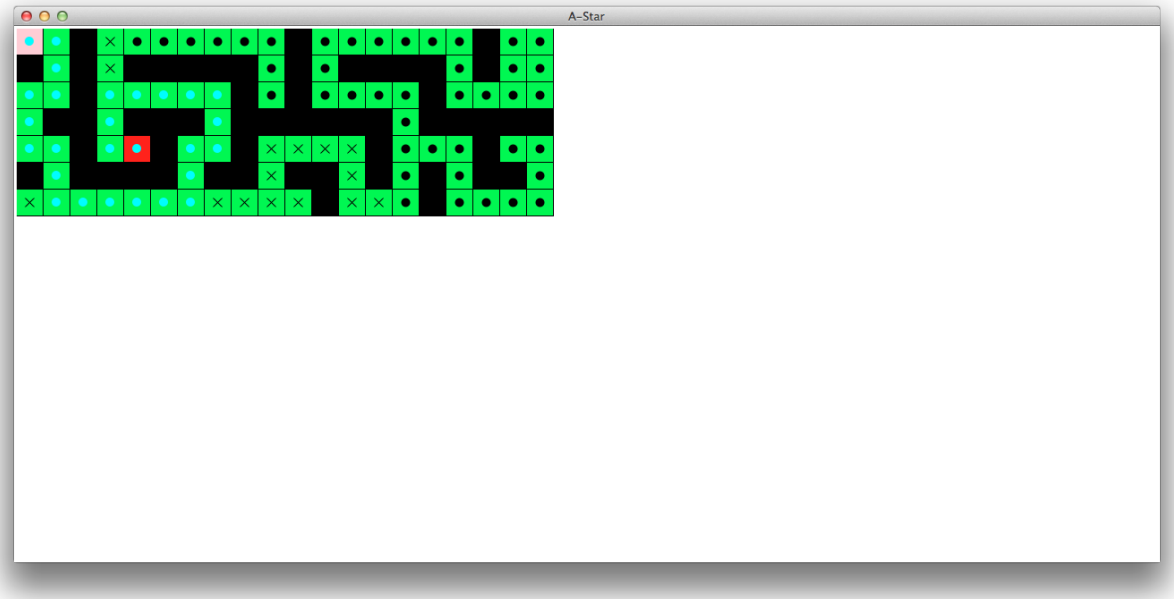
A*



BFS



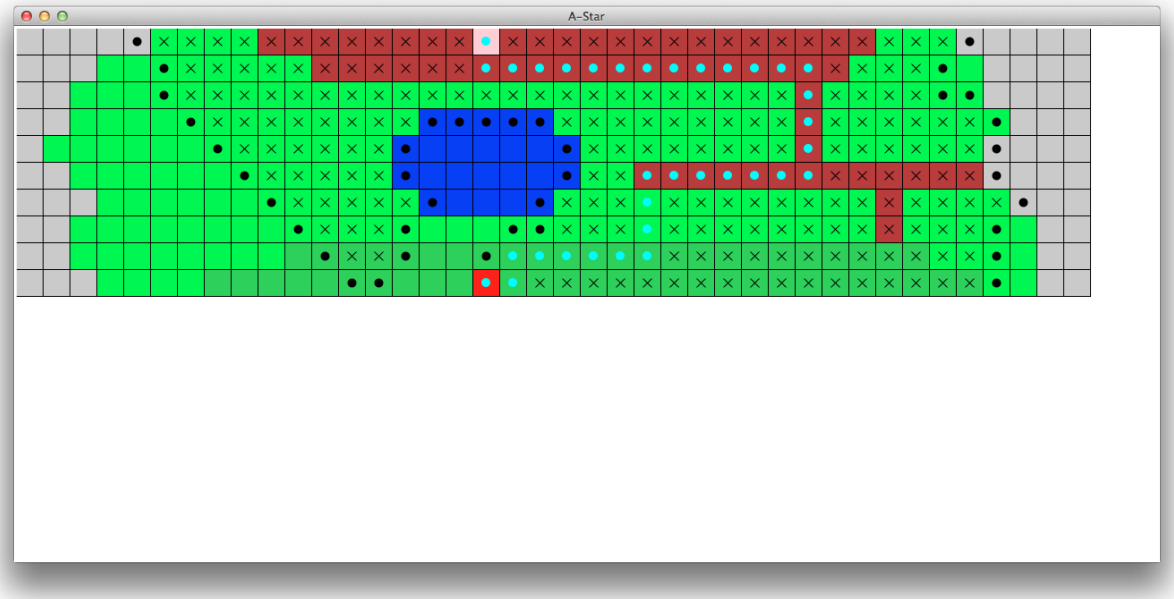
Dijkstra



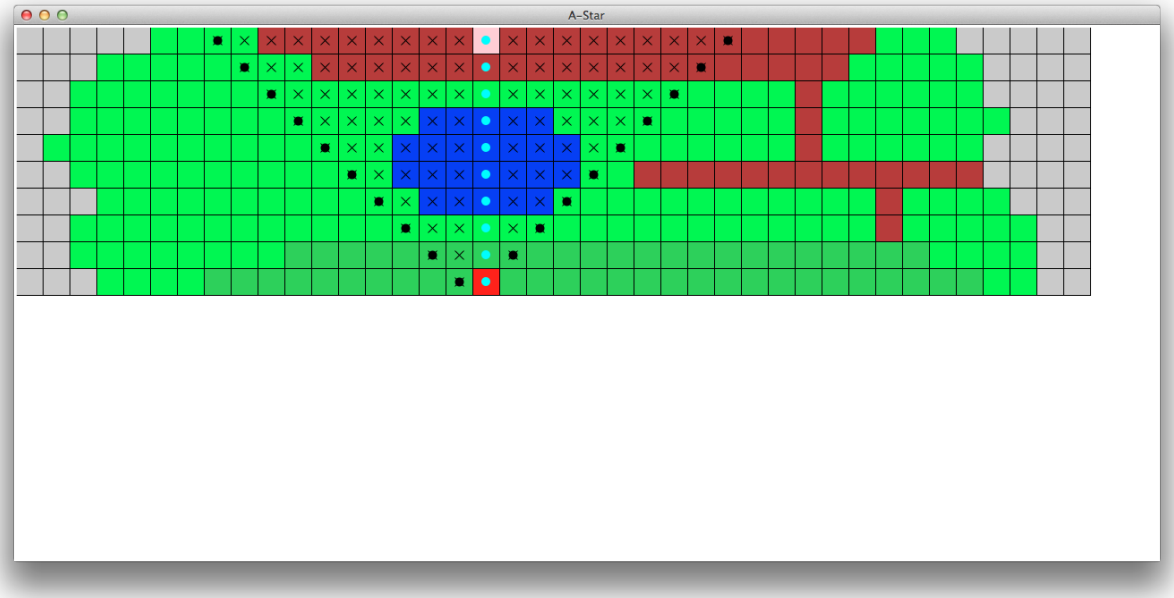
Comment:
On this board, the similarities between BFS and A* are more apparent than with Dijkstra. Since Dijkstra adds all the nodes at start, we see many nodes in the openset for Dijkstra.

Board 2-1.txt

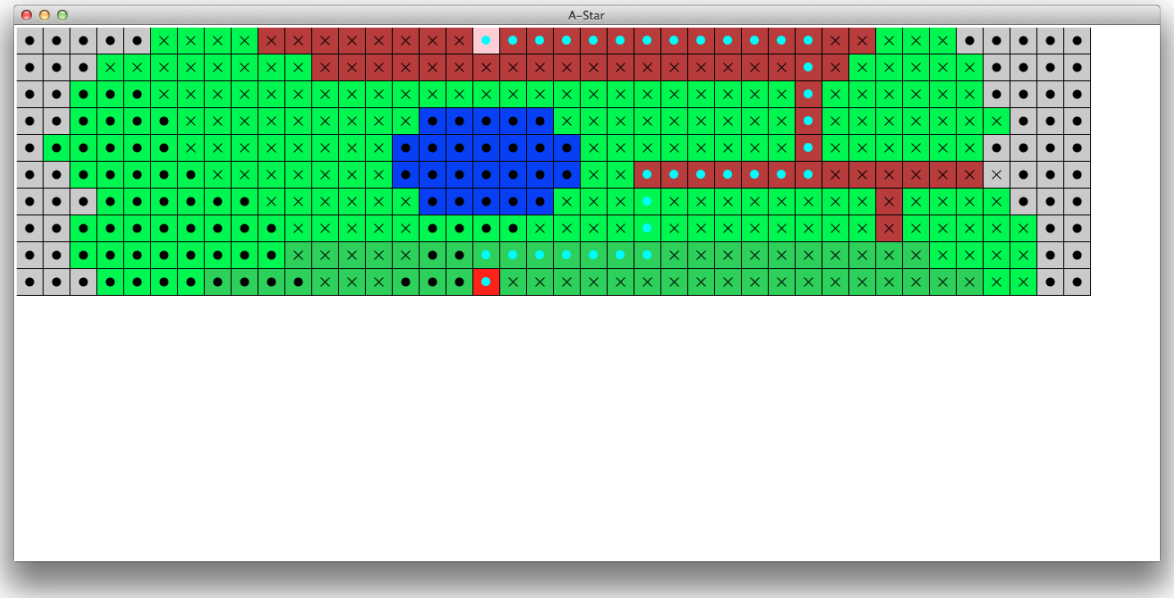
A*



BFS



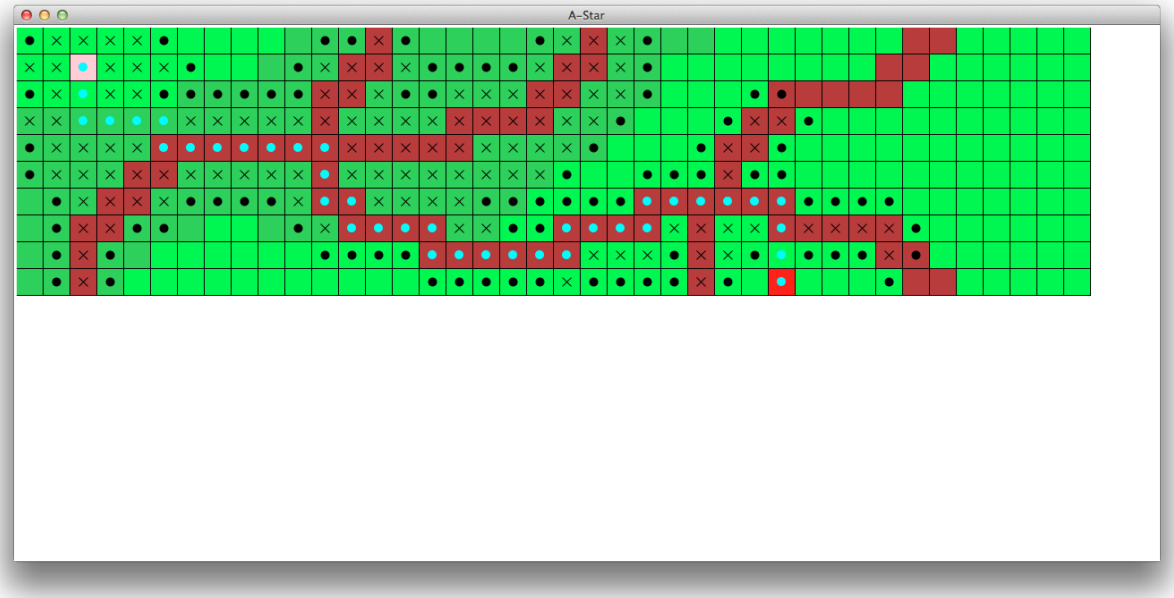
Dijkstra



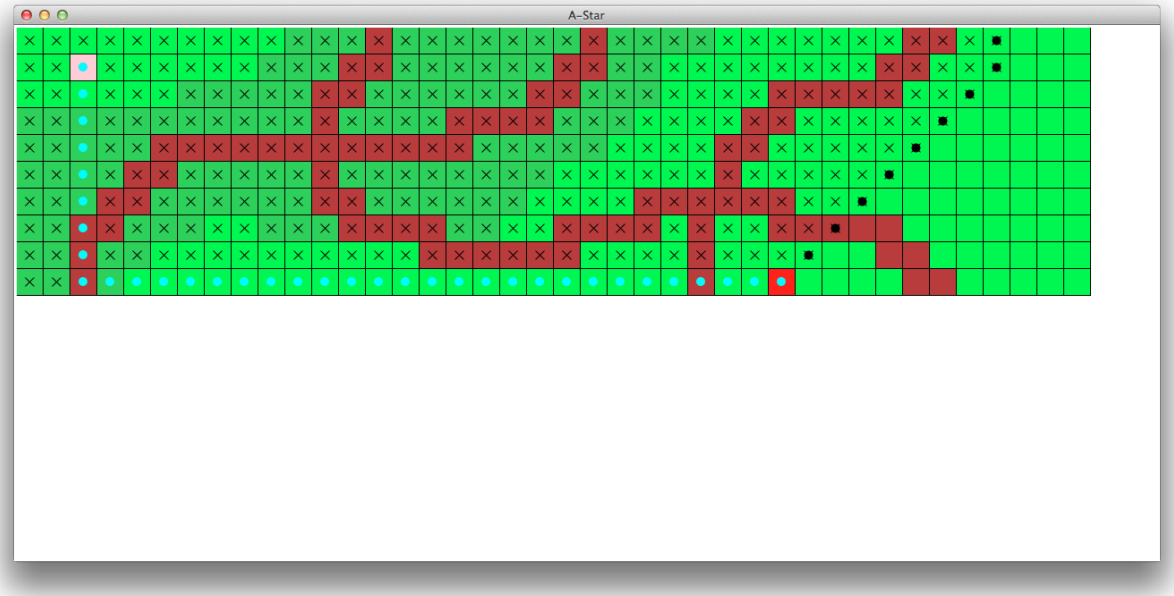
Comment:
This board shows more detail about how the different algorithms work. The diamond shaped exploration of BFS is clearly shown here. The similarities between Dijkstra and A* are apparent, albeit A* has a more narrow search than Dijkstra, due to how the H-function is taken into account.

Board 2-2.txt

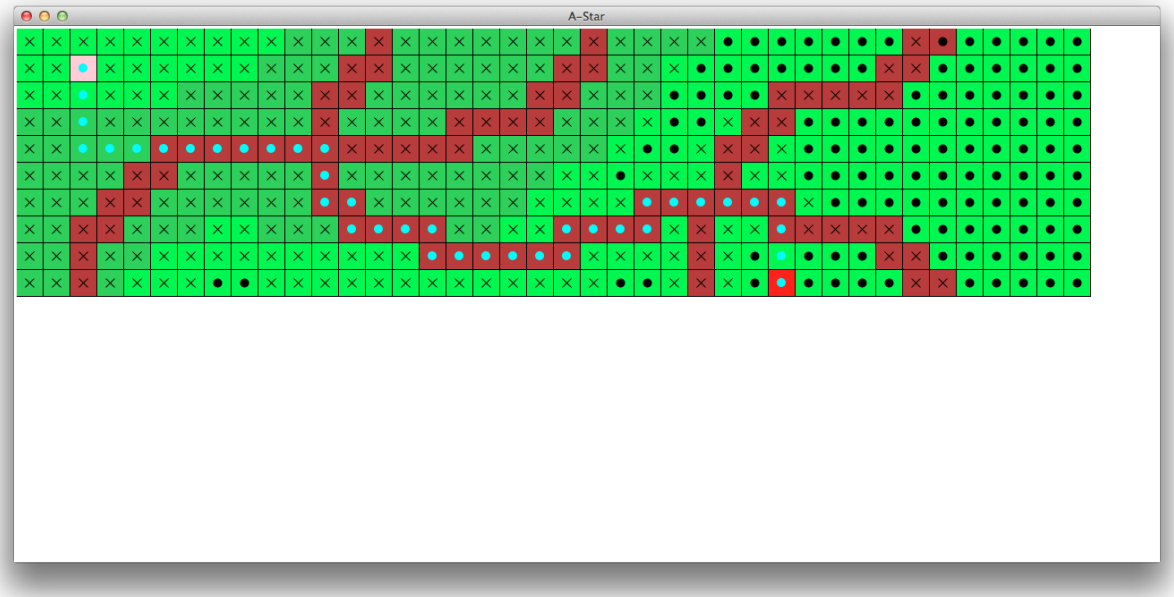
A*



BFS



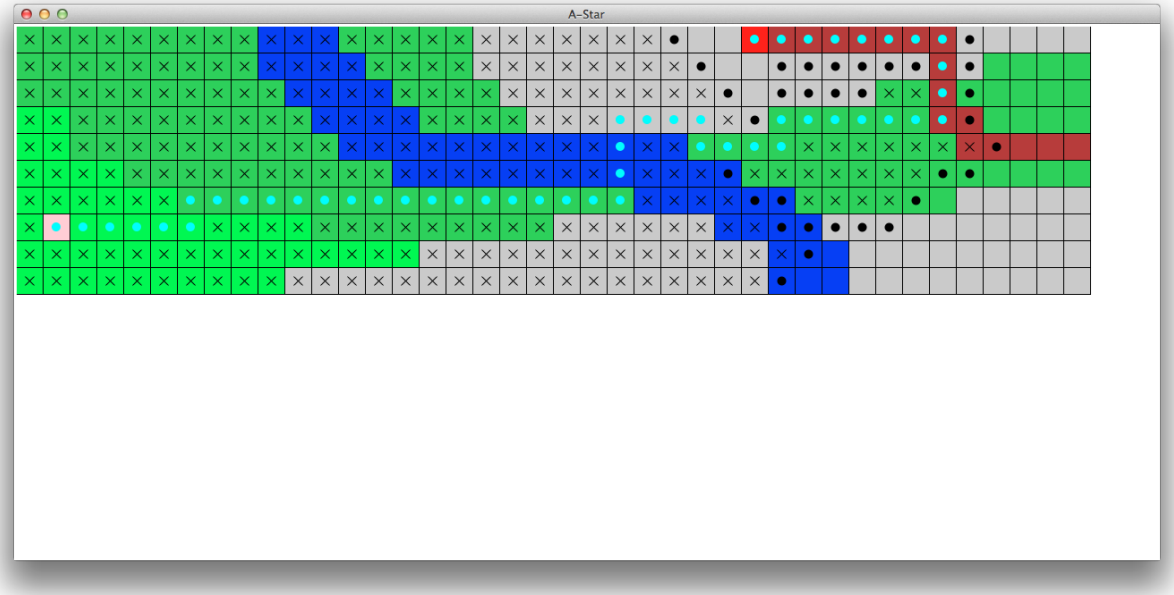
Dijkstra



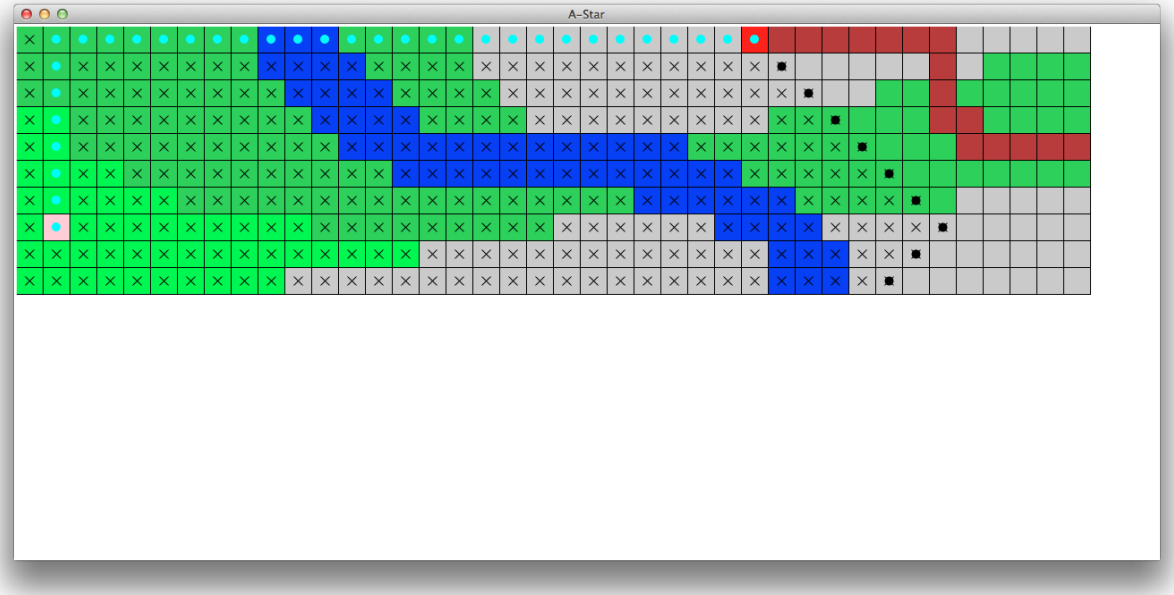
Comment:
This is one of the boards which show a clear difference between all 3 algorithms. A* best first approach, with the included euclidean distance function. Dijkstra, which chooses almost exactly the same path, but explores way more, and has many nodes in openset. And BFS' complete exploration of a depth level before proceeding to the next.

Board 2-3.txt

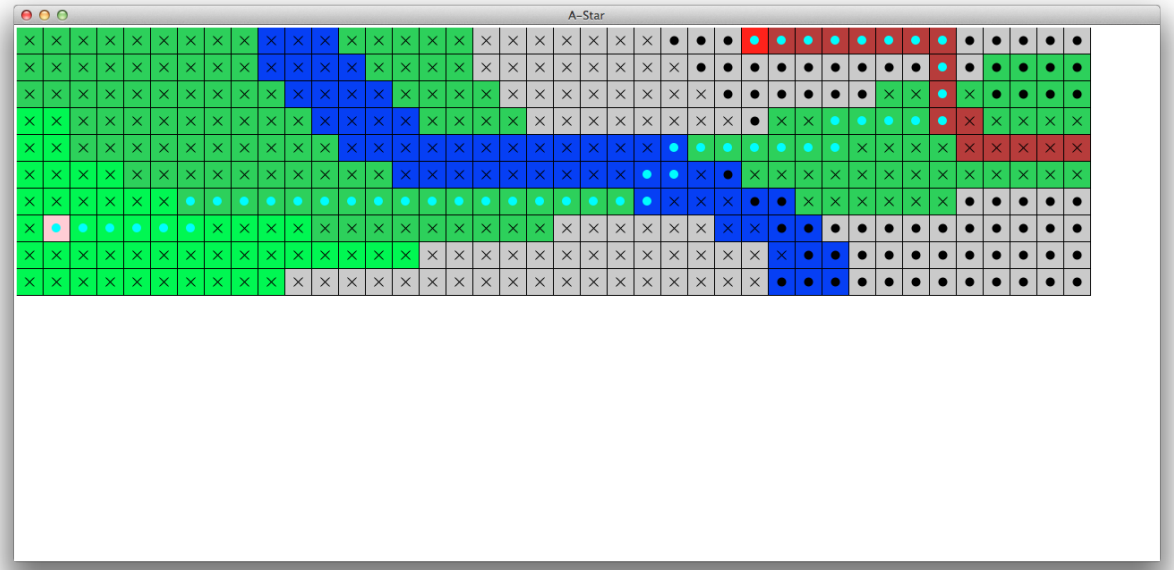
A*



BFS



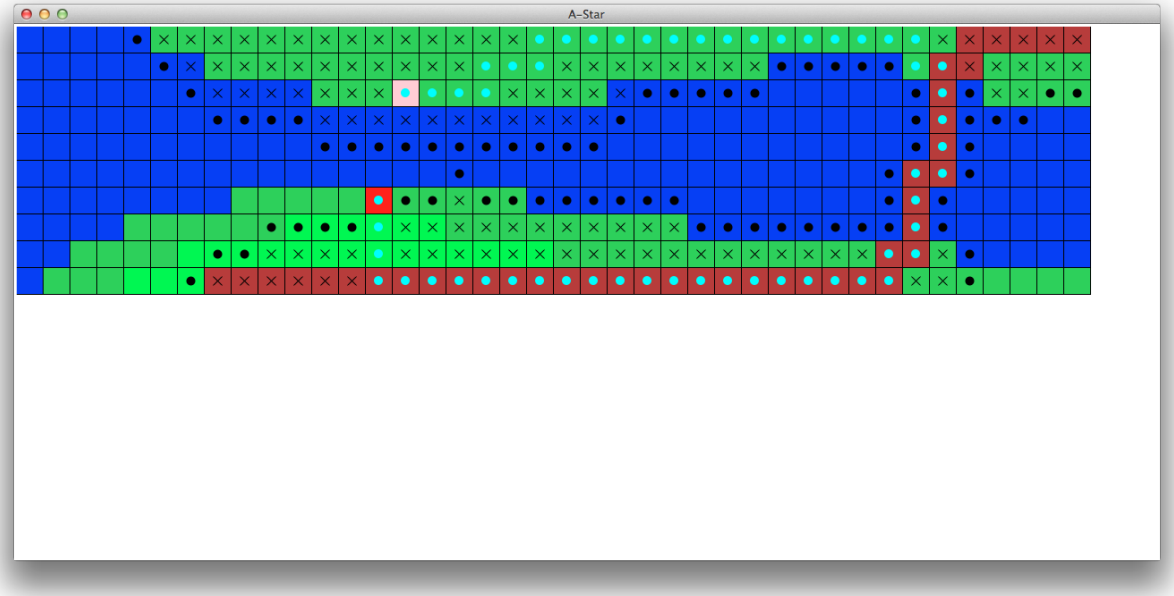
Dijkstra



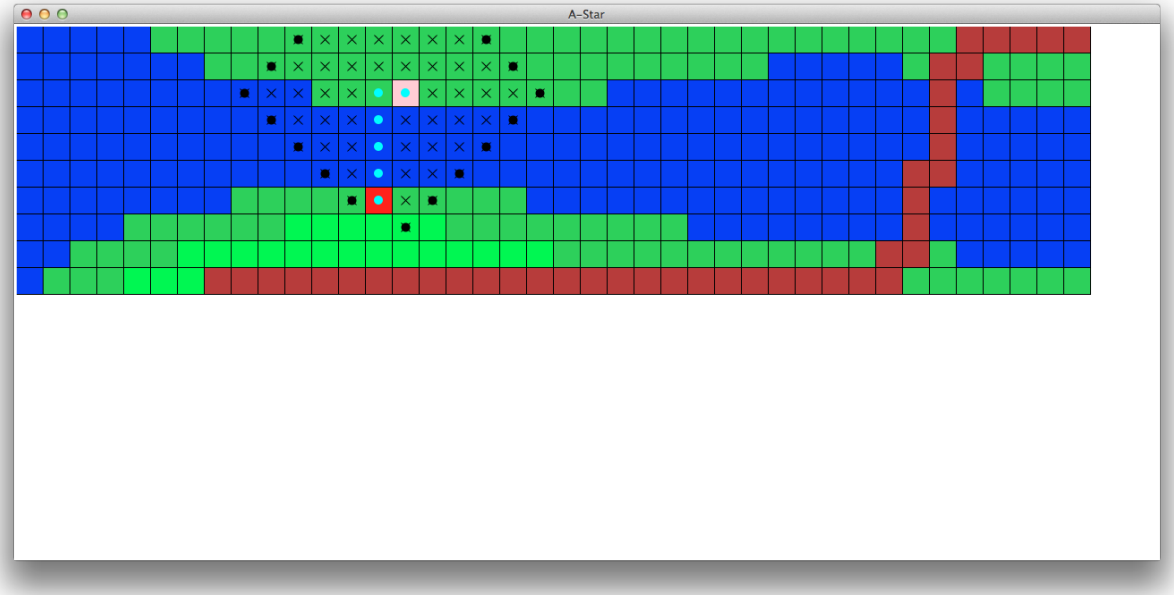
Comment:
Again we see that Dijkstra has more nodes in the openlist than A* and BFS. BFS ignoring the cost of moving over mountain has ten mountain tiles in the path. A* and Dijkstra has a similar path again with a small difference when crossing the water tiles.

Board 2-4.txt

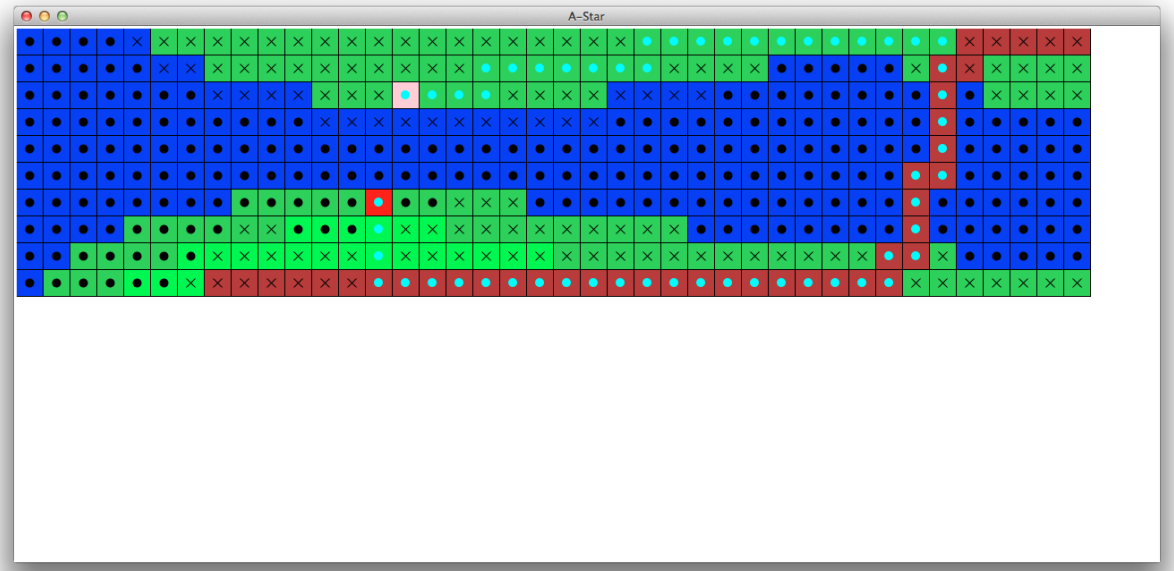
A*



BFS



Dijkstra



Comment:
On this board we see that BFS finds the goal fast, but ignores the cost of moving over the water. A* and Dijkstra have a similar path in this case, but Dijkstra checks more nodes than A*.