A* Search

Dataset Summary:

Name	n	m	format	wallVal
3x3.txt	3	3	.txt	0
5x5.txt	5	5	.txt	0
100x100.txt	100	100	.txt	0
NewYork_1_256.csv	256	256	.csv	1
1000x1000.txt	1000	1000	.txt	0
Sydney_0_1024.csv	1024	1024	.csv	1

Test Suites:

“3x3 Correct Shortest Path from top left to bottom right”

Manually calculated the shortest path from top left (0,0) to bottom right (2,2):
solution = {{0,0}, {0,1}, {1,1}, {1,2}, {2,2}};
Checked to see if: solution.size() == threeShortestPath.size();
5 == 5
TRUE

“3x3 Correct Shortest Path using BFS”

Compared A* search shortest path to the length of the path using BFS.
Check to see if: BFS_result.size() == A*_result.size()
5 == 5
TRUE

“3x3 Heuristic Calculations”

Making sure the Heuristic calculations load correctly into the Point data type, and see if the Point::getHeuristic() function works properly.

```
Point startPoint(0, 0, 1, 4); //expected h value of startPoint is 4
Point midPoint(1, 1, 1, 2); //expected h value of midPoint is 2
Point endPoint(2, 2, 1, 0); // expected h value of endPoint is 0
```

Checked start-, mid-, and end-point, all worked correctly
TRUE

“5x5 Correct Shortest Path from top left to bottom right”

Comparing the length of the output of BFS and A* search.

Output lengths are equivalent if they are both the shortest paths.

TRUE

“100x100 Correct Shortest Path from top left to bottom right”

Comparing the length of the output of BFS and A* search.

Output lengths are equivalent if they are both the shortest paths.

TRUE

“1000x1000 Correct Shortest Path from top left to bottom right”

Comparing the length of the BFS and A* search output.

Output lengths are equivalent if they are both the shortest paths.

“Sydney CSV to PNG”

Testing readFromCSV as well as writing an image to a file.

No REQUIRE statements, but we verified the image visually.

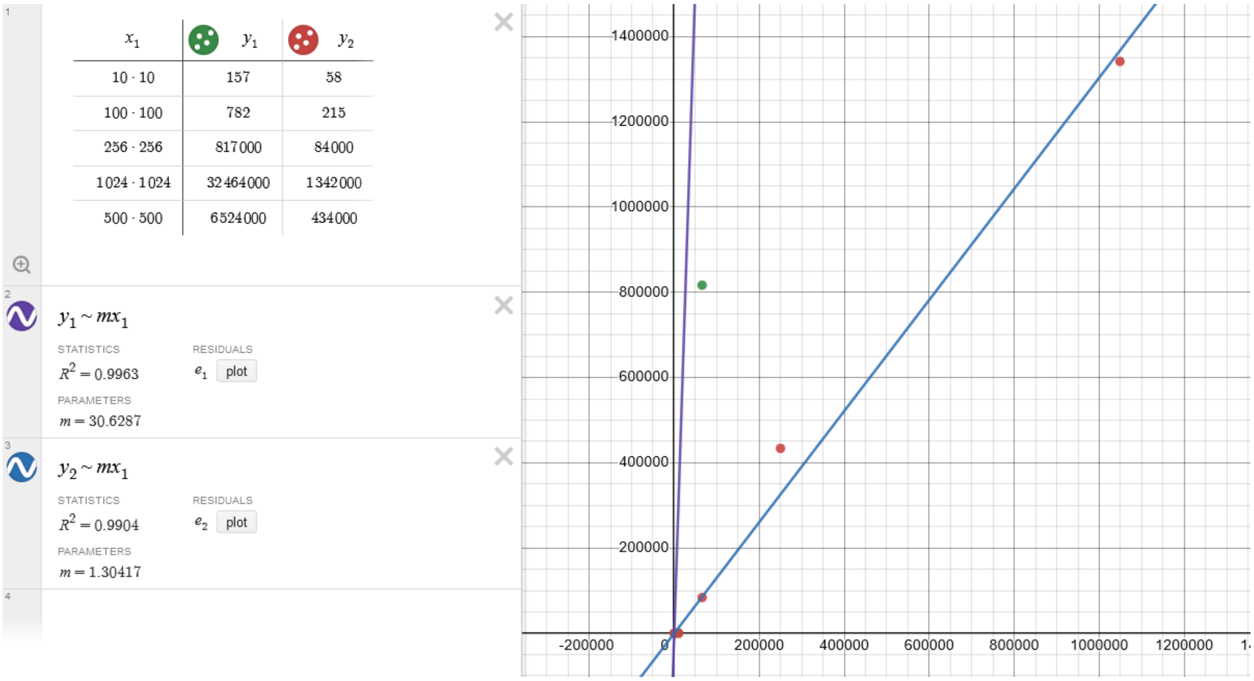
“Sydney Correct Shortest Path from top left to bottom right”

Comparing the length of the output of BFS and A* search.

Output lengths are equivalent if they are both the shortest paths.

Purple: A* Search

Blue: BFS Search



x_1	🟩 y_1	🟥 y_2
$10 \cdot 10$	157	58
$100 \cdot 100$	782	215
$256 \cdot 256$	817000	84000
$1024 \cdot 1024$	32464000	1342000
$500 \cdot 500$	6524000	434000

$y_1 \sim mx_1$

STATISTICS

$R^2 = 0.9963$

RESIDUALS

e_1 plot

PARAMETERS

$m = 30.6287$

$y_2 \sim mx_1$

STATISTICS

$R^2 = 0.9904$

RESIDUALS

e_2 plot

PARAMETERS

$m = 1.30417$

b = (the average number of successors per state)
d = is the depth of the optimal solution. However, the efficiency of A* heavily depends on the accuracy of the heuristic function.

In the worst case, the A* algorithm can degrade to the time complexity of a basic breadth-first search, which is O(b^d). This occurs when the heuristic function is not informative and does not guide the search toward the optimal solution. Our solution does not match this: in fact, it goes slower than the basic BFS search algorithm. We believe this is due to a combination of a mis-implementation of the priority queue and a weak heuristic function. However, the algorithm still finds the shortest path.