Assignment 3

Heuristic Search

Hunter Morabito

Heuristic Functions Used:

Euclidean Distance

The Euclidean distance takes the straight line distance between the vectors using the following formula:

$$H = \text{sqrt}((x2 - x1)^2 + (y2 - y1)^2)$$

Manhattan distance

The Manhattan distance is the sum of the distances of the two cells x and y components using the following formula:

$$H = |x2 - x1| + |y2 - y1|$$

Tie Breaker

The idea of the tie breaker is to make the heuristic function scale up as the path gets closer to the goal. It uses the following formula:

$$H \mathrel{*}= (1.0 + (\text{min cost of one step/expected max distance}))$$

Uniform Distance

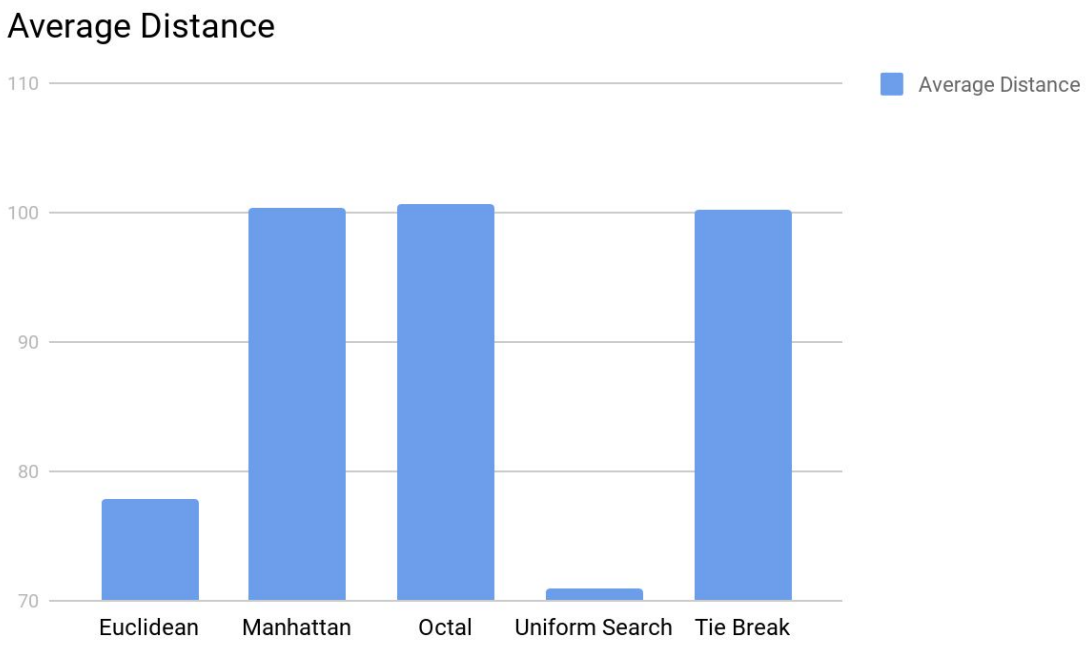
The uniform distance is the shortest possible path, given that here is no heuristic being given, the function solely advances based on shortest path.

Octal

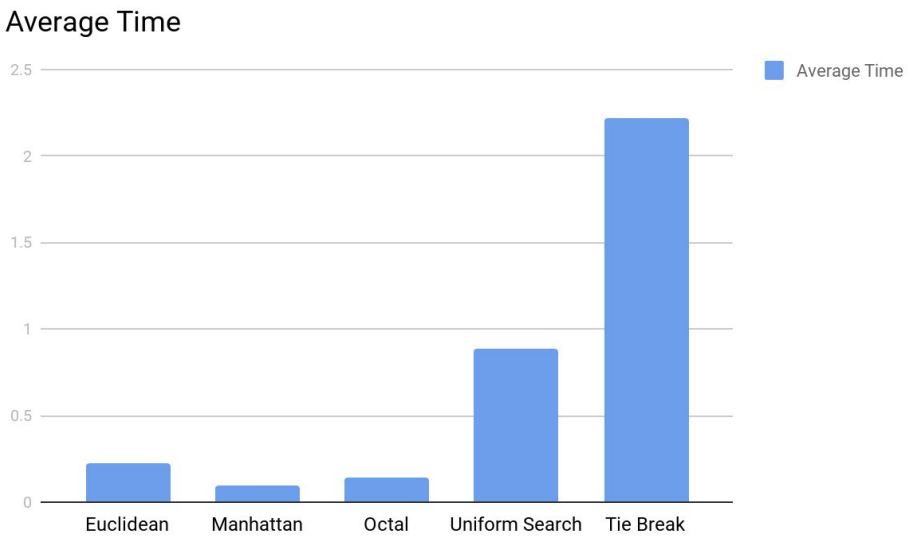
Since we are allowed to make diagonal moves, the octal (or diagonal) heuristic makes sense to use

$$H = (|x1 - x2| + |y1 - y2|) + (\text{sqrt}(2) * \min(x,y))$$

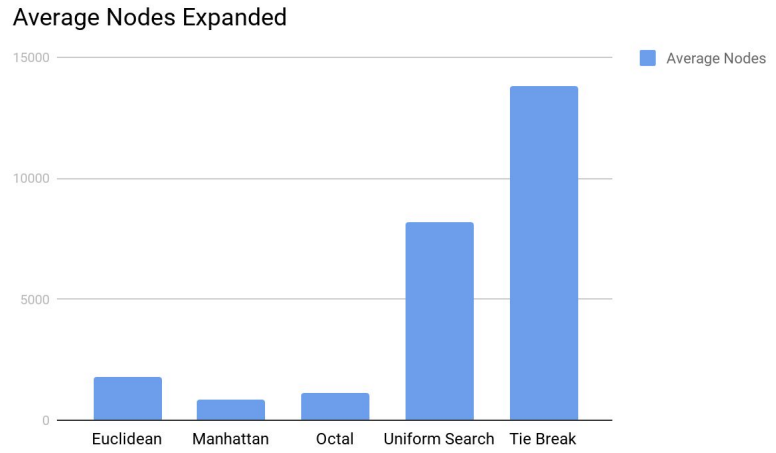
Benchmark results over 5 maps with 10 different goals each:



	Average Distance
Euclidean	77.81
Manhattan	100.389
Octal	100.581
Uniform Search	71
Tie Break	100.1572

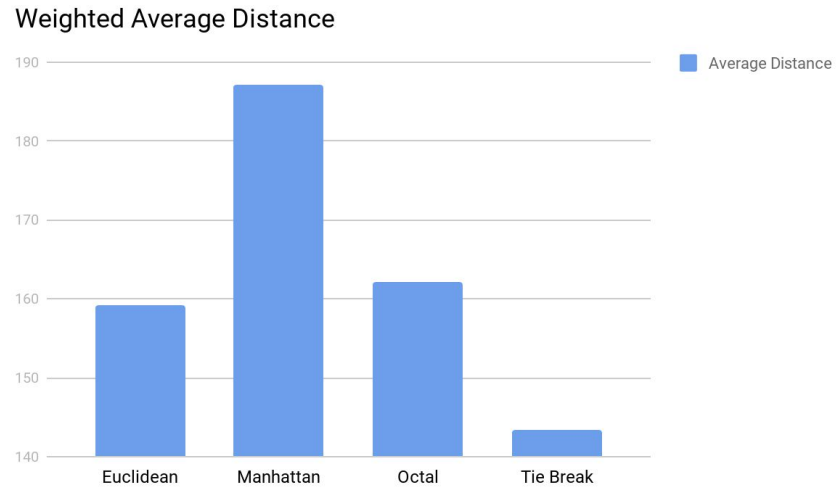


	Average Time
Euclidean	0.2207
Manhattan	0.09978
Octal	0.14303
Uniform Search	0.885
Tie Break	2.215



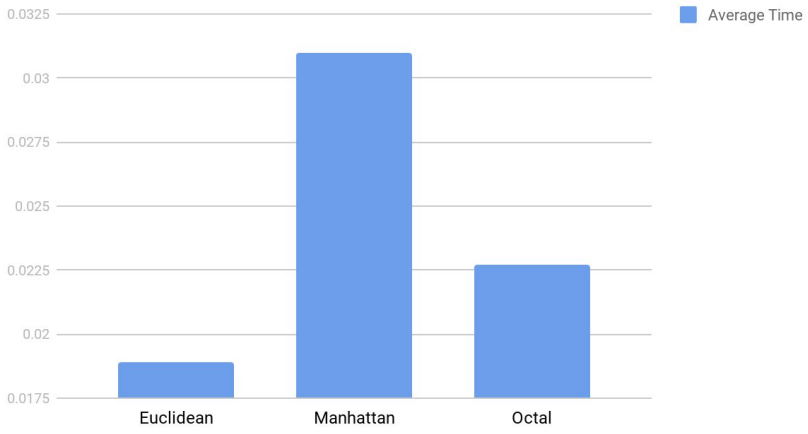
	Average Nodes
Euclidean	1764.58
Manhattan	852.59
Octal	1154
Uniform Search	8197
Tie Break	13819.75

Weighted Heuristics at w = 1.5



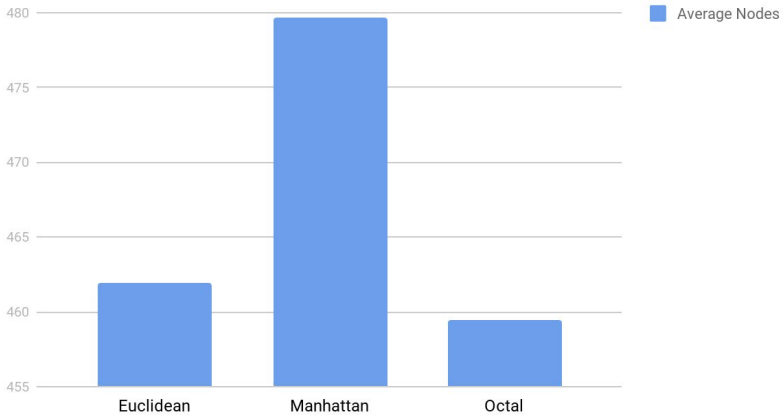
	Average Distance
Euclidean	159.11
Manhattan	187.06
Octal	162.05
Tie Break	143.42

Weighted Average Time



	Average Time
Euclidean	0.0189
Manhattan	0.031
Octal	0.0227

Weighted Average Nodes Expanded



	Average Nodes
Euclidean	461.93
Manhattan	479.707
Octal	459.41

Results (Lower is better for all the above charts):

Euclidean:

Euclidean as it turns out is a very bad Heuristic to use. Since the distance from the goal doesn't vary enough from the amount of speed needed to get to the next cell, the search doesn't give the f value enough of a push in the right direction. The euclidean value shines when weighted. It is has an extremely good time/distance ratio, as it is the first to find the goal node on average.

Manhattan:

Since we are in a rigid grid space, this heuristic performs very well. The amount that the distance between neighbors works very well with this heuristic. When weighted, this heuristic does not perform as well as Euclidean in terms of time/distance.

Octal:

This heuristic performed somewhere in between the Manhattan and Euclidean Heuristics, which makes good sense as it combines the two. Weighted this heuristic tells the same story.

Uniform Search:

Uniform Search is the best way to get the shortest distance between start and goal nodes. The obvious problem with this is that it stores a large amount of nodes in memory, and if the start and goal cells are the maximum distance from each other in a grid, then the algorithm will visit every node possible

Tie Break:

The concept of Tie Break still makes sense to me. As the path gets closer to the goal, hone in on it to get there faster. Unfortunately, I do not believe I had the best implementation.