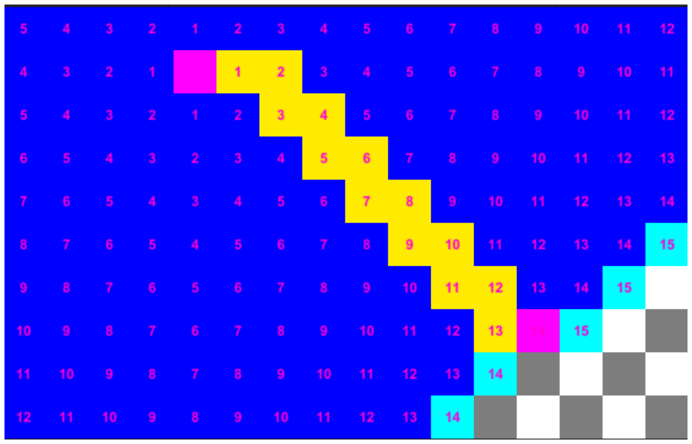
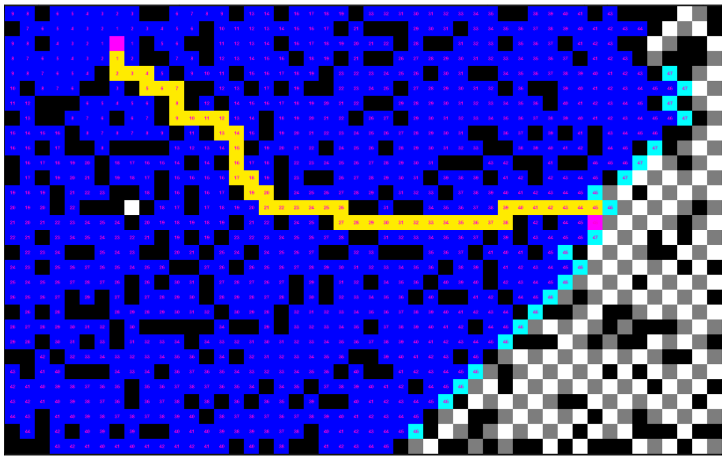
Lily Ready
February 12th 2023
Homework 1

Dijkstra's Algorithm:

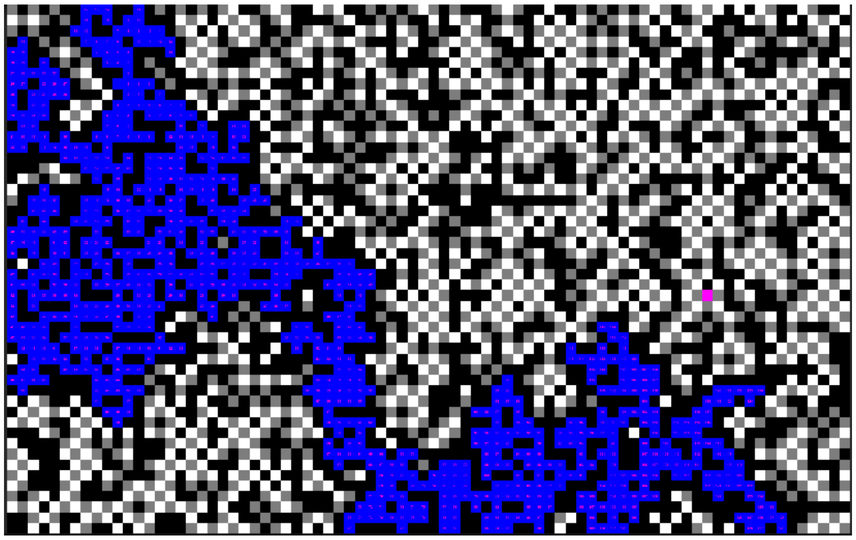
Scale 1, No Obstacles



Scale 3, 25% Chance of Obstacles



Scale 5, 40% Chance of Obstacles (No Route)

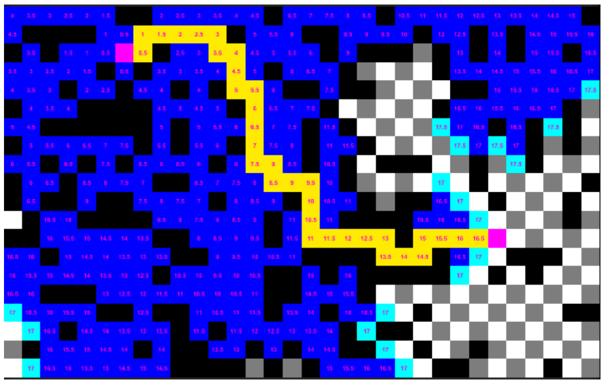


As Dijkstra's Algorithm is just a Breadth First Search algorithm modified to work over weighted graphs, and this graph doesn't have any weights, this is just a BFS implementation. This is going to create a much greater contrast between this implementation and the A* implementation than there otherwise would be. As there aren't any weights, it's completely undirected while A* uses a Best First algorithm, and so will move toward the goal instead of evenly spreading as Dijkstra's does.

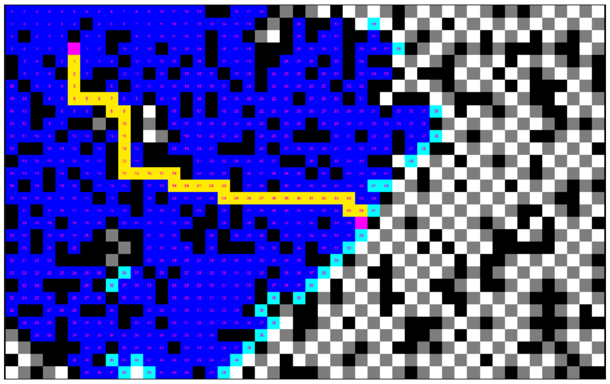
A* Uniform:

As A* differs from Dijkstra in that it uses a heuristic to determine which paths to value over others, without this heuristic (as a uniform cost search is), we've just created Dijkstra's algorithm again. There are little differences in the implementation, but as each node is expanded uniformly, it behaves the same as Djikstra's algorithm.

Scale 2, 20% Chance of Obstacles

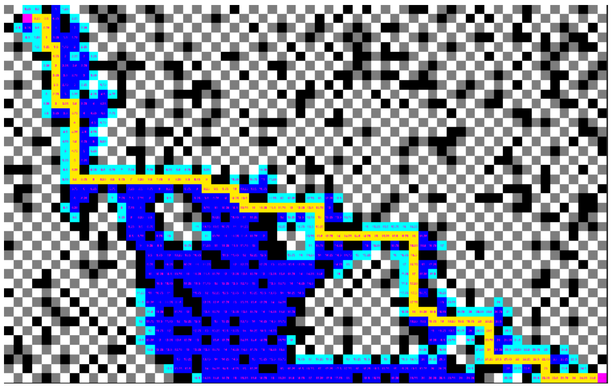


Scale 3, 25% Chance of Obstacles

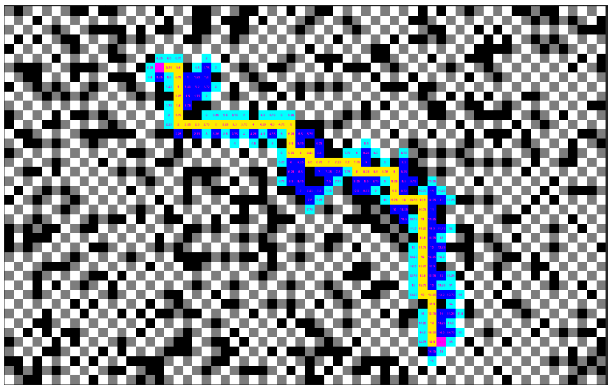


A* Manhattan Distance:

Scale 4, 20% Chance of Obstacles



Scale 4, 25% Chance of Obstacles

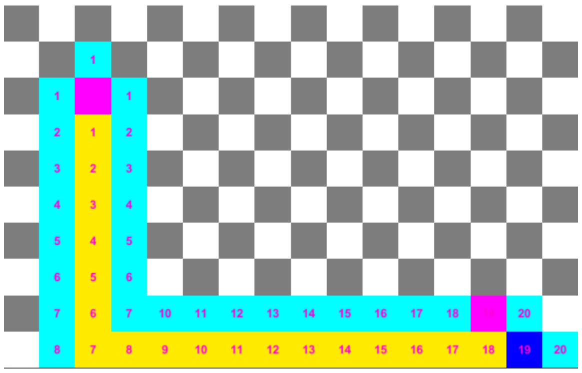


The Manhattan distance describes the way distance is described for cities on a grid system, most famously, Manhattan. This has the downside that many of the blocks between start and end will have the same value, which you can see in the large swaths of blue throughout the above images. It isn't nearly as much as the BF searches, but there's a decent bit of backtracking and trying other routes. As it goes straight for the goal, it expands a lot fewer nodes than either of the previous searches, and tends to find the goal faster as well.

A* Cross Product

Cross product tries to eliminate a lot of the ties that we have with Manhattan distance, and you can see that when it searches, it tends to stay pretty close to the final path. This indicates that it's more strongly directed, with fewer ties between nodes. It still has some issues, it definitely finds the fastest path in terms of the search, like the Manhattan, but also like the Manhattan it doesn't always find the most direct path. In the screenshots, you can clearly see there are pretty big sections of the final path that we could cut out and have the final path be shorter. In my testing, cross product performed faster than Manhattan when the paths were the same distance and when going to the same node. I know this is what the rubric says, but I honestly thought it wouldn't with the given implementation, since cross product also calls the Manhattan distance method. I was able to find specific mazes where it wasn't consistently faster, but it was explained by an increased number of expanded nodes per route.

Scale 1, No Chance of Obstacles



Scale 3, 20% Chance of Obstacles