Luna McBride

**AI Problem A Report**

4b:

1. Random: This approach uses random values as a heuristic to try to bring Pacman to his goal. Of course, since this is literally random, this is not a consistent heuristic. It is also slower than the others, as it is not specifically looking for the end goal. Random heuristic is also not admissible, as it can run different numbers of nodes depending on the number generated, and thus never necessarily gives the most optimal solution.
2. Euclidean: This approach gives a straight line to the end as the heuristic. It uses the same formula every time, so for the same start and end points, it will calculate the same points to go to next, thus being consistent. This is the middle of the road for the three in terms of speed, as trying to maintain a straight line in the grid-like world is not the easiest thing to do in the world. This is also not admissible, as it puts going in a straight line above overall optimality, thus doing some rash and jagged moves rather than thinking like a grid.
3. Manhattan: This approach takes advantage of the grid structure of Pacman to take a more calculated path. This uses a mathematical formula between the start and the goal, thus being consistent in its calculations and bringing forth the same path every time for the same maze. This is also the fastest of the three choices, as it takes advantage of the Pacman grid structure in a similar way to the blocks of Manhattan (Times for all 3 in the order listed in the report: 0.6, 0.5, 0.4 seconds). This also builds the best outcome in terms of node number, making it the most admissible of the three heuristics tested om the Pacman board.

5:

1. Depth First Search: In the open maze, this search falls apart. It goes right for one of the potholes before anything else. It then proceeds to squiggle back and forth until it reaches the goal. This occurs because the search goes down one path until it hits a dead end, which explains the pothole. The successors are then held in a specific order, which, by the nature of a stack, have specific directions occur before others (here being east and west), causing the squiggle back and forth.
2. Breadth First Search: The algorithm explores nodes outward rather in streams like the depth first. In the open maze, this creates an explosion hitting almost every node with earlier checks near the epicenter (the start). The path then follows this sort of expansion, following the one path that got to the goal first. The path is pretty straight forward with very few squiggles because of this, turning to get around the walls and move toward the goal only.
3. Uniform Cost Search: This search reacts exactly like the breadth first search in behavior. The open maze looks the exact same and the path goes the exact same. This is because this search is made to prioritize cost. All costs are the exact same on the open maze, so it searches all of the choices. If specific tiles had specific costs, this would be a different story, but since they are all the same, this becomes the same as a breadth first search.
4. A\*: A\* is interesting as the path depends on the heuristic. A\*, by implementation, is a uniform cost with an added heuristic to guide the search toward the goal. With no heuristic, like with the basic A\*, the result looks the exact same as the uniform cost search. This makes the heuristics the deciding factor on shape and path.
   1. Random: The random heuristic looks nearly identical to the uniform cost. Nearly every node is accounted for and the path is a direct ray. However, the background redness is very different. The redness implies which nodes were first to be explored. Where the others had a nice gradient, random creates a blotchy mess by the nature of randomly checking nodes.
   2. Euclidean: The Euclidean heuristic draws a direct line from the finish to Pacman’s current position. This destroys the need for checking every nook and cranny, as it has an idea where the goal is. Though it can still try to go into a pothole by not knowing where walls are when figuring the path out, it tries to go in as straight of a line as possible. Of course, the path ends up a bit jagged because Pacman cannot go diagonal like a line can, but it gets the job done.
   3. Manhattan: This heuristic works very similar to the Euclidean heuristic, however, does not try for direct straight lines. No straight lines lead to less jagged movement (no more left down left down left down), but also allows for optimal movement in a grid system. Both those heuristics have explored areas that look similar (a little bit into potholes, but a tail toward the goal once the walls are maneuvered around), however, the path for Manhattan lowers the need for turns and uses corners for a better path in a grid system.