Q1: DFS algorithm generally finds a path quicker than BFS but does not necessarily find the best path. DFS cannot find a solution in an open maze, as it will continuously go in a single direction to infinity but it expands less nodes on bigMaze when compared to BFS because it checks the deep nodes first before checking alternative paths. If the maze is open or if we are required to find the optimal path, then BFS is better as it will always find the goal in b^s time complexity where b is the max number of successors and s is the depth of the goal in the tree. If any path would suffice and the maze has many splits and dead ends, then DFS is preferable.

Q2: If A* works with a consistent heuristic, both A* and UCS finds the lowest cost optimal path. UCS expands depending on cost differences and generally expands more nodes when compared to A*. A* uses a heuristic to guess the remaining cost, which makes its speed heavily reliant on the heuristic. If the runtime of the heuristic is significant enough to offset the benefit of expanding less nodes than UCS is preferable. With a good heuristic, A* works faster than UCS.

Q3: My state consists of the position of Pacman and a tuple of four booleans. Each of these booleans checks if a specific corner has been visited. This implementation works because it refreshes the visited locations when a corner is visited because the boolean part of the state changes, allowing previously visited locations to be visited during the new travel (from a corner to another). When the four booleans are all True, then the goal state is reached.

Q4: My heuristic takes the Manhattan distance to all not visited corners and then divides it by 2. I settled on this heuristic because it is directly related to our goal, is simple and works relatively well. It is admissible because dividing the distances by 2 makes the worst case scenario admissible, which is being in a maze with one row or column (not both) and being positioned at one of its corners while the other corner is unvisited. It is consistent because it uses Manhattan distances and any move can only reduce the Manhattan distance to two corners at the same time, therefore decreasing the heuristic by at most 2, which the division makes at most 1.

Q5: My heuristic takes Manhattan distances to all foods present in the maze and adds them, then divides them by the number of foods present in the maze. I settled on this heuristic because it directly takes the goal into consideration, works relatively fast, and is consistent and admissible. The worst case is a long single row maze where Pacman is on one corner and all the foods are concentrated on the other corner, which makes the distance to all foods very close to the furthest away one, which is the solution. Dividing the heuristic by the number of foods makes the heuristic admissible as it is bound to be less than the solution and consistent because each move decreases the heuristic by the number of foods.

Q6: Consistent heuristics guarantee the shortest path result, but an inadmissible heuristic can find a solution by expanding less nodes. While not necessarily being correct, an inadmissible heuristic can be preferred for its speed and it can give a close estimate to the correct solution even if it is not the shortest path. In a real-time environment where speed is crucial, and an estimate is good enough an inadmissible heuristic can be preferred while if we need correct results without being time bound a consistent heuristic is better.