Written Q1:

In terms of path, the biggest difference between BFS and DFS can be described as first path and shortest path. BFS uses the shortest path and DFS uses the first path. BFS tries to find the shortest path. By doing this, it is negatively affected in terms of time as it makes various attempts. However, when DFS finds the path, it considers the path it finds for the first time as the path. Other parameters are not important here. If there is only one solution, we can prefer DFS to BFS. Because in a labyrinth, he finds the only solution in a shorter time. However, if there is more than one path, other parameters come into play here. In this case I prefer BFS.

Written Q2:

A\* has almost the same features and results as UCS with a slight difference. A\* calculates by adding the cost with the heuristic, but UCS only uses the cost. Assuming the heuristic value is 0, both can be used. However, if the heuristic value is greater than zero, then A\* will be more effective than the other.

Written Q3:

I tried to use the startPosition and VisitCorners tuples as the state definition in the problem of finding all vertices. Since I couldn't do this part, I looked at the examples, but I couldn't fully understand. That's why I couldn't do those parts. I think we can get vertices with state[0] and state[1] using tuple to find vertices. In this way, control can take place.

Written Q4:

For the problem of finding all vertices, I should intuitively choose to use the maze distance of the furthest corner. While heuristic is acceptable and consistent for all test cases, it can sufficiently reduce the number of expanded nodes. But I couldn't implicit this situation in the code or it didn't work properly. That's why I asked my friends and according to their solution can solve using this situation. In my method, operation is performed according to the nearest corner. If such a case is near a corner, the heuristic will be the point to vertex distance and the distances of other unvisited vertices.

Written Q5:

It seems logical to solve using Manhattan distance in the previous question. This was because there were too many points. In this question we want to get more precise heuristic values.

Written Q6:

Heuristics affect the algorithm. A consistent heuristic can give an optimal solution. This allows the algorithm to run faster. If we need speed more than optimal, we can use unacceptable methods. If we need a high probability of optimality, then we must choose wisely intuitively.