# Part 1

1. Asd
   1. Asdasd
   2. Asdasd
   3. Asdasd
2. A Star
   1. [Insert Image of the pseudocode]

The A star algorithm would be implemented similar to the Dijkstra algorithm. Q would be implemented as a Priority Queue, where Q.GetFirst() retrieves the state with highest priority.

In this case, the priority is determined by an estimate of the minimal total path length from current state to the goal states. The estimate is calculated by adding the actual Cost to Come from the start state to that state with a estimated Cost to Go to the goal state calculated using a heuristic, and is inserted along with the state using Q.Insert().

Resolving duplicate is done in the same fashion as in Dijkstra, where the path with the lower cost to get to that state is chosen, the parent of the cell is updated to the parent which gives that lower path cost, and the corresponding cell within the Priority Queue is updated with that lower path cost.

* 1. When a heuristic is admissible, it means that it never gives an over-estimate to the actual cost-to-go value from current state to the goal and is non-negative.

An example of an admissible heuristic is the Euclidean distance because firstly it is never negative due to the addition of the inner squares, and also it never over estimates the cost to go from current state to the goal state as the path between two points can only be as little as the straight line distance between them.

An example of inadmissible heuristic is a heuristic which yields the Euclidean distance, but perceives one or more of the states on the only actual optimal path as having a very high cost (e.g. in extreme cases higher than the maximal total path length of the given problem) because although it is always positive, this heuristic yields an extensive over-estimate to the Cost to go of the current state to the goal.

This reason why admissible heuristics are important is because taking the two function described above, when using the inadmissible heuristic the A star algorithm could reach the goal state before fully exploring the actual optimal path due to the heuristic’s perceived high cost on the optimal states hence would have the lowest priority within the priority queue. This means that the algorithm would not provide an optimal path with minimal cost to the goal, which is undesired. On the other hand, for admissible heuristics such as Euclidean distance, would allow the A star algorithm to find an optimal path because …

An example of an inadmissible heuristic which still guarantees to provide an optimal solution is an admissible function with an additive constant, such as the Euclidean distance f(x) with a constant c = maximal path length of the given problem. Hence the total estimated path cost becomes cost\_to\_come + f(x) + c. The constant wouldn’t affect the way in which the algorithm explores as it wouldn’t change the priority of the states, however it makes the heuristic inadmissible as the heuristic now yields an overestimate to the path.

* 1. The optimal heuristic that A\* can use is the actual cost to go from current state to the goal. Could it be used in practice? No because ….
  2. See code
  3. Evaluate performance of implementation …