© There are many, many examples of where the path found by Repeated A* is a shortest path, but not the optimal path from start to goal. In general, the argument for finding the optimal path is very controversial because in real life circumstances there is a variable that we cannot test in our code, which is travel time. (Delays, closed ways, etc.) An example for our Repeated A* is a graph or gridworld where there are weights to each path taken. A* algorithm will look for the next "closest" node to the target node (example A -4-> B -10-> D -11-> G), only 3 paths to G, however there could be a path (A-1->C-3->E-4->D-1->F-1->G), 5 paths to G. Depending on how the A* search sends out the agent to go search. Also, we have to consider sometimes weights on a path or edge are not equal to distance. They may be costs to go a certain way, so that means that the shortest path is not necessarily the optimal path. Another thing that we can consider are negative weights, however in A* search we would not get stuck in a cycle. And one last thing that I can think of where the shortest path may not be the optimal path is when self-loops and parallel edges are present inside a gridworld or graph.