Q-Learning Deep Dive

Q-Learning was created in 1989 by Chris Watkins, which he then expanded into a thesis in 1992. It is a reinforcement learning algorithm that uses Bellman's Equation (created by Richard E. Bellman) to calculate rewards and punishments not only for the next action in sequence, but also taking into account a number of future actions in the decision making. It is a model free approach to machine learning, so there is no internal model for the agent to start with, it simply starts going, and slowly learns what moves are the best. It takes this information and generally trends towards whatever state gives it the best reward.

Q-Learning stores all of this information in what is called a Q-Table, which stores these modified rewards (calculated by Bellman's equation) for each action that can be taken in each state. These states in my approach are simply what square the agent is on, and the actions are the moves it can make (most of the time that is all 4 cardinal directions).

In most applications of Q-Learning, there is a sort of two-phase approach. First there is the exploration phase, where the agent about it's environment, and then there is the execution phase, where the agent takes what it knows about the environment and applies it, getting itself to the goal as efficiently as possible. In my application, these two phases are sort of blurred. At the start, the AI has a very high chance of simply moving randomly to any square it hasn't explored yet, and this chance of a random move gradually decreases over time. Eventually, this can build out a fairly fleshed out view of the most important parts of the environment (those with positive rewards) and it can find one of the most optimal paths from the start to the end fairly easily.

There were a lot of bugs to work out, mostly with how I was updating the State-Action-value table, and how I was updating the Q-Table with new values. Those were the nastiest bugs to find especially. And then the bug of the agent sometimes just moving back and forth between two squares, even though there are better moves to be making in both scenarios was a particularly deceptive one, but it was actually linked to how I was updating the tables, so I found the fix to that one relatively quickly once I fixed the previous errors.