**Assignment 6**

Camilo Aguilar León

201820551

**Laberinto**

Para la implementación del laberinto se trabajo directamente con el algoritmo de Q-learning y con su respectiva Q tabla. Para la determinación de la convergencia de este problema y el siguiente del taxi simplemente se ejecuto varias veces para determinar cuando aproximadamente convergía. Aparte, debido a la simetría de los problemas, en muchos casos la convergencia se podía evidenciar cuando celdas a la misma distancia de una celda terminal tenían el mismo q valor máximo.

Para el problema del laberinto se uso un factor de descuento de 0.7, una tasa de aprendizaje de 0.1, una épsilon de 0.1, una recompensa por salir de 100 y una recompensa por cada paso de -1. Se determinaron estos valores para poder lograr un aprendizaje rápido que favorezca el encontrar el camino más corto para la salida, pero siempre permitiendo la exploración. El resultado de esto se puede ver en un tablero que muestra para cada casilla el q valor máximo entre todas las acciones posibles para ese estado. Para determinar la política de una casilla simplemente hay que encontrar la casilla adyacente con mayor q valor y por lo tanto esta será la casilla que seguir.

Table

Description automatically generated

Como se puede ver, el comportamiento es el esperado ya que sigue las casillas que minimizan el camino a la salida respetando las paredes del laberinto. La table de q-valores se puede ver en el notebook ya que es muy extensa.

**Taxi**

Para la implementación de este problema se siguió algo muy similar a lo que se hizo en el laberinto en tanto a las paredes que no se podían pasar. Se siguieron los lineamientos definidos en el enunciado sin embargo se hicieron unos ajustes para facilitar la ejecución del modelo y por problemas computacionales que se tuvo a la hora de ejecutar el algoritmo. Lo primero que se hizo fue realizar una penalización de -1 por cada paso así buscar que se dejara al pasajero lo más rápido posible. También se hizo un cambio en las recompensas donde recoger al pasajero da un total de 5 y dejarlo en una parada da 10. Esto con el objetivo de mitigar un poco lo de las recompensas negativas y de que se propagara mejor el objetivo. Por último, debido a problemas computacionales se decidió que el agente siempre se recogiera en la estación B y que se tuviera que dejar en cualquiera de las otras tres estaciones. Esto reduce la cantidad de ejecuciones para que el modelo converja y no difiere mucho del modelo original.

En caso de que se decidiera implementar que el lugar de recogida fuera aleatorio se debería agregar al estado la ubicación del pasajero y que las recompensas de las acciones de recoger o dejar al pasajero en las estaciones cambien dependiendo de este estado. Esto claramente aumenta el tamaño de la tabla de q-valores y dificulta la ejecución del algoritmo.

Para la ejecución de este algoritmo laberinto se usó un factor de descuento de 0.8, una tasa de aprendizaje de 0.2 y una épsilon de 0.1. Por lo que el resultado de ejecutar este algoritmo se puede ver en dos tableros, uno para cuando el taxi se encuentra vacío y otro para cuando ya recogió al pasajero.

*Vacío*

*Table

Description automatically generated*

*Lleno*

*A picture containing table

Description automatically generated*

El comportamiento en ambos casos es bastante intuitivo. Cuando este vacío simplemente busca acercarse a la estación B que es donde va a recoger al pasajero siempre y por lo tanto la política siempre buscará ser la que tenga el camino más corto a esta celda. Por otro lado, cuando el taxi está lleno se puede ver como la política difiere más en tanto a cuál es la estación más cercana que no sea la B. Por lo que al final decidirá ir a la estación que se encuentre más cercana o la que tenga menos probabilidades de encontrarse con un obstáculo.

La tabla de q-valores se puede encontrar en el notebook.