En este documento vamos a explicar cómo se ha estructurado nuestro proyecto, además de hacer una introducción al uso de la interfaz y la realización de pruebas sobre nuestro código fuente.

Dividiremos esta explicación por fases:

· Fase 1: Comenzamos en esta parte de nuestro código fuente pidiendo por consola

ciertos parámetros que más tarde nos van a ser útiles, como son el tamaño del problema y el estado objetivo. Con estos datos, calculamos el tablero con el que vamos a trabajar. A raíz de este tablero, podemos calcular la matriz R, sabiendo que las transiciones prohibidas tienen recompensa de -1, las posibles tienen recompensa de 0, y las transiciones que llegan al objetivo tienen recompensa de 100.

Ahora, pasamos a inicializar nuestra matriz Q como una matriz de ceros de las mismas dimensiones que nuestra matriz R. Tras esto, estamos listos para aplicar nuestro algoritmo de entrenamiento. Para la ejecución del algoritmo de entrenamiento diseñado, nos apoyaremos en dos funciones auxiliares creadas por nosotros: maxQValue y acciones. Para entrenar la matriz Q, necesitamos saber cuántos episodios de entrenamiento queremos llevar a cabo, así como el valor del parámetro Gamma, por lo que pedimos por consola estos datos. Tras esto, se entrena Q y se calcula el rendimiento, que mostraremos más adelante con una gráfica.

Pasamos ahora a calcular el camino eficiente, para el que necesitamos recibir el valor del estado en el que iniciaremos el camino. El algoritmo diseñado devuelve el camino en forma de array, un contador de saltos y el valor total de recorrer ese camino.

· Fase 2: El funcionamiento de esta fase es muy similar al de la fase anterior. La única modificación que se ha llevado a cabo es la inclusión de dos parámetros adicionales a la hora de entrenar la matriz Q. En este caso, estos parámetros son llamados alpha y epsilon, ambos se encuentran en el rango [0,1] y serán configurables por el usuario a la hora de entrenar la matriz Q.

· Fase 3: Las modificaciones introducidas en esta fase se encuentran en su mayor parte en el algoritmo para obtener la matriz R (obtainRMatrix). Permitimos que el usuario pueda decidir las recompensas de las transiciones horizontales-verticales, y las de las oblicuas. Así conseguimos que el camino a veces pueda variar en función

de estas recompensas, del estado objetivo y del estado inicial que se escoja. Igual que en los algoritmos anteriores, estas recompensas serán configurables por el usuario.

Pasamos ahora a explicar el funcionamiento de las interfaces gráficas diseñadas.

Estas interfaces están disponibles para facilitar el funcionamiento automático de las fases 1 y 2. El usuario solo necesitará ejecutar la única celda que se encuentre en el cuaderno de jupyter, y le aparecerá una interfaz en la que podrá introducir todos los datos necesarios para la configuración del problema y el entrenamiento de la matriz Q. Al pulsar el botón de enviar, los resultados de la prueba aparecerán por consola. Mostramos información de los datos introducidos, los resultados obtenidos en las matrices R y Q con estos datos, así como el rendimiento final de la matriz Q y el camino más eficiente para un estado inicial que deberá proporcionar el usuario.