Parte 1: Análisis - Laberinto

Kyle Echeverri
Andrés Gomez
Sebastian Osorio

1. **¿Cómo se está representando cada nodo del terreno a recorrer en la aplicación interactiva?**
   Cada punto en la cuadrícula que conforma el terreno es un nodo. Desde el código, cada nodo es un objeto que parte de una clase Nodo.

2. **¿Cuál es la estructura de datos que se está usando para almacenar cada nodo del terreno?**
   Se utiliza un diccionario para almacenar cada nodo del terreno, en donde la llave es la posición en la cuadrícula, y el valor es el objeto nodo en la posición indicada.

3. **¿Cómo se definen el punto de inicio y el punto final del recorrido?**
   Son variables serializadas, por lo que se definen en el inspector de Unity.

4. **¿Qué ventaja tiene usar la estructura de datos anterior para este problema?**
   Un diccionario permite guardar datos vinculados, como en este caso es la posición con un objeto. Ya que el problema implica recorrer un terreno, es importante la posición de cada nodo de la cuadrícula.

5. **Escribe los pasos que componen el algoritmo BFS en la solución del problema.**
   Se agrega a una cola el punto inicial, y luego se inicia un ciclo *while*, el cual persiste mientras sigan habiendo elementos dentro de la cola **y** todavía no se haya llegado al punto final del camino. Dentro del ciclo:
   - Se saca de la cola el nodo que esté disponible.
   - Se revisa si el nodo actual es el punto final. Si lo es, el ciclo acaba una vez termine la iteración actual.
   - Se exploran los nodos vecinos. Esto se hace por un método que revisa cada dirección (arriba, abajo, izquierda y derecha) a partir del nodo actual. En cada dirección:
     - Verifica si hay un nodo (o sea, si la posición revisada es la llave para un nodo en el diccionario).
     - Si hay un nodo, revisa si este ya ha sido explorado. Si no, se agrega a la cola, se marca el nodo desde donde se encontró, y se marca como explorado.
   - Luego, se reinicia el ciclo.

6. **¿Para qué se usa una cola en el algoritmo BFS de la aplicación interactiva?**
   El funcionamiento de una cola se caracteriza porque el primer objeto que entra es el primero en salir (FIFO), lo que significa que después de visitar un nodo, inmediatamente se exploran todos los vecinos, en lugar de irse muy profundo dentro del terreno. Esto permite una exploración más gradual y menos arbitraria.

7. **¿Cómo se determina si un nodo ya fue explorado?**
   Dentro de la clase Nodo hay un booleano para verificar si un nodo ha sido explorado o no. Esto permite que cada objeto pueda seguir rastro de si ha sido explorado de manera individual. Ese booleano originalmente viene declarado como falso, pero una vez sea encontrado dentro del algoritmo BFS, se marca como verdadero.

8. **¿Cómo se determina qué nodos serán explorados?**
   Dentro del método de BFS, se usa otro método llamado ExploreNeighbourNodes(). Este método usa un arreglo de Vector2, donde en cada índice apunta a una dirección a explorar (arriba, abajo, izquierda y derecha). Para cada una de las direcciones, se hace lo siguiente: a la posición del nodo actual, se le suma una de las direcciones del arreglo, y se revisa si hay un nodo para esa posición dentro del diccionario. Luego, se agrega a la cola, los cuales se irán sacando en orden de llegada.

9. **¿Qué estructura de datos se está utilizando para almacenar los nodos que componen la ruta calculada desde el punto de inicio al punto final?**
   Se utiliza una lista para almacenar la ruta desde el inicio hasta el final.

10. **¿Qué ventaja tiene usar la estructura de datos de la pregunta anterior?**
    Las listas crecen de manera dinámica, lo cual es importante para que el algoritmo sea flexible, ya que permite guardar rutas de distintos tamaños.

11. **Al crear el path ¿Por qué es necesario usar el método Reverse()?**
    Porque al crear la lista, esta empieza desde el punto final y va trazando la ruta hasta el principio. Para que quede en el orden correcto hay que invertirlo, lo cual se hace con el método Reverse().

12. **Explica cómo funciona la aplicación interactiva luego de dar click en play.**
    Al darle click a play, la cuadrícula colorea el punto inicial, el punto final, y la ruta entre los dos puntos. Luego, la esfera recorre cada cuadro de la ruta hasta llegar al final, donde pasa al siguiente nivel.

13. **¿Qué es una corrutina y cómo y para qué se está usando en la aplicación interactiva?**
    Una corrutina es un método que pausa su ejecución en cierto momento para devolverle control al resto del programa, y en el siguiente frame reanuda su ejecución desde el punto donde lo había dejado. Para la pausa, se debe declarar una línea de *yield*, el cual sería el punto donde pausa el método. Una corrutina se diferencia de un método convencional, los cuales se ejecutan de inicio a fin dentro del mismo frame.

    La aplicación utiliza una corrutina para el movimiento del jugador. En este, tiene un ciclo *for*, que recorre cada nodo en la lista de la ruta. Cada paso entre nodos lo hace gradual al ser una corrutina, y el tiempo entre cada paso es una variable serializada.