Pseudocode: Safe pathfinding for Sniffer pup and Ally

Variables:

private static readonly int[] dRow = { -1, -1, -1, 0, 1, 1, 1, 0 };

private static readonly int[] dCol = { -1, 0, 1, 1, 1, 0, -1, -1 };

private static readonly string[] directionNames =

{ "Up-Left", "Up", "Up-Right", "Right", "Down-Right", "Down", "Down-Left", "Left" };

**dRow**: Array of integers that represents the row movements (up, down, diagonals)

**dCol**: Array of integers that represents the column movement (left, right, diagonals)

**directionNames**: Array of String that stores the direction name for logging movements

**eg: (-1,-1) = Up-Left, (-1,0) = Up, (-1,1) = Up-Right…**

**Function FindPath(minefield, totalRows, totalCols, snifferRow, snifferCol, allyRow, allyCol, visited, path):**

1. isInBounds(x,y)

bool isInBounds(int x, int y)

{

return x >= 0 && x < row && y >= 0 && y < col;

}

* it is a helper function that returns true if the position (x,y) is within the minefield bounds

1. Default/Base case

if sniffer pup's position (snifferRow, snifferCol) is location at the exit location (bottom right corner):

**snifferRow == row - 1 && snifferCol == col - 1**

* Print "Sniffer Pup and Ally reached the end"
* Return true

1. Recursive Exploration

* Based on the movement we have define previously under dRow & dCol, the method should loop through all 8 possible direction (Up, Down, Left, Right, Diagonals)
* Followed by the position that Sniffer pup is currently at, calculate the new position under newRow & newCol for Sniffer pup's potential move

int newRow = snifferRow + dRow[i];

int newCol = snifferCol + dCol[i];

* Check if the move is valid by ensuring the new position is within (isInBounds) and ensure the position is not a bomb and has not visited before

**if (isInBounds(newRow, newCol) && !visited[newRow, newCol] && minefield[newRow, newCol] != 1)**

* when the new position is clear, mark the position as visited by setting **visited[newRow, newCol]** to true
* Then we can move ally to Sniffer pup's previous position **allyRow, allyCol = snifferRow, snifferCol**
* Then we should check whether Ally’s new position is overlapping with Sniffer pup’s position, if true, reset the marked position and skip the move.

**if (allyRow == newRow && allyCol == newCol)**

**{**

**visited[newRow, newCol] = false; // Reset visited status**

**continue;**

**}**

* Print out Sniffer pup’s new position and Ally’s new position for tracking/logging purposes
* Add Sniffer pup’s current position to a list **path.Add(newRow, newCol);** this is to track down the safe position that Sniffer pup has moved to, and we will use it during backtracking purpose if there is needed
* Perform recursive calling with the updated position for Sniffer pup to continue find the exit pathway

**if (findPath(minefield, row, col, newRow, newCol, allyRow, allyCol, visited, path))**

**{**

**return true;**

**}**

* If Sniffer pup has reached a dead end (safe path but it is not the exit path), we will need to perform backtracking.

**path.RemoveAt(path.Count - 1);**

will remove Sniffer pup’s current position from the path

**visited[newRow, newCol] = false;**

this will mark the visited position as false and it helps Sniffer pup to go back and find other pathways

**Console.WriteLine($"Backtracking from ({newRow}, {newCol}).");**

Prints out the message to notify that Sniffer pup has backtracked to find other pathways

* If there is no more available pathway to reach exit, it will return false

1. **Main Program**

* Define the mine field

Create a 2D array minefield with bombs (1) and safe cells (0)

int n = 6, m = 6;

int[,] minefield =

{

{0, 0, 0, 0, 0, 0 },

{1, 0, 1, 1, 1, 1 },

{0, 1, 0, 1, 1, 1 },

{0, 1, 1, 0, 0, 1 },

{0, 1, 0, 0, 0, 1 },

{1, 1, 1, 1, 0, 0 }

};

* Initialize the variables

visited: 2D boolean array to track visited positions.

path: list to store Sniffer Pup's path.

snifferRow and snifferCol: starting position for Sniffer Pup at (0, 0).

allyRow and allyCol: starting position for Ally at (0, 0).

Set visited[0, 0] to true (starting position).

* Display the minefield

for (int i = 0; i < minefield.GetLength(0); ++i)

{

for (int j = 0; j < minefield.GetLength(1); ++j)

Console.Write(minefield[i, j] + " ");

Console.Write("\n");

}

* Call the findPath method and pass in the minefield, starting positions for both Sniffer pup and Ally and other parameters to find the exit

**bool result = findPath(minefield, n, m, snifferRow, snifferCol, allyRow, allyCol, visited, path);**

* Check result, if findPath returns true, print out the success message and the list of movement coordinates

Console.WriteLine("Safe path found:");

foreach (var pos in path)

{

Console.WriteLine($"({pos.Item1}, {pos.Item2})");

}

* Otherwise print out “No safe path found.”