

Decision Management Community's Challenge Jan-2024

"Christmas Word Search"

The solution proposed by Copilot

I suspected it would be quite difficult (or awkward) to write the proper decision model with a DMN-like tool. But before writing it in Java or Python, I asked Copilot to do it for me. Here is the resulting dialog.



You

MMMSXXMASM
MSAMXMSMSA
AMXSXMAAMM
MSAMASMSMX
XMASAMXAMM
XXAMMXXAMA
SMSMSASXSS
SAXAMASAAA
MAMMMXMMMM
MXMXAXMASX



Copilot

Thanks for sharing the grid! What would you like to do with it? Are you looking for a specific pattern, solution, or analysis? Let me know how I can assist you further.



You

You need to find how many times the word "XMAS" appears in the grid, horizontally, vertically, or diagonally, written backwards, or even overlapping other words in any orientation.



Copilot

✓ Executing Task: **find how many times the word 'XMAS' appears in the grid, horizontally, vertically, or diagonally, written backwards, or even overlapping other words in any orientation.**

I ran a search through the grid you provided, and the word "XMAS" appears **18 times** in various orientations and directions.

If you need any further analysis or have another question, feel free to ask!



You

Can you show me a Java programm that does such a search?



Copilot

Sure! Here's a Java program that searches for the word "XMAS" in a grid, considering all possible directions and orientations:

Java



```
public class WordSearch {  
    private static final int[] dx = {0, 1, 1, 1, 0, -1, -1, -1};  
    private static final int[] dy = {1, 1, 0, -1, -1, -1, 0, 1};  
  
    public static void main(String[] args) {  
        char[][] grid = {  
            {'M', 'M', 'M', 'S', 'X', 'X', 'M', 'A', 'S', 'M'},  
            {'M', 'S', 'A', 'M', 'X', 'M', 'S', 'M', 'S', 'A'},  
            {'A', 'M', 'X', 'S', 'X', 'M', 'A', 'A', 'M', 'M'},  
            {'M', 'S', 'A', 'M', 'A', 'S', 'M', 'S', 'M', 'X'},  
            {'X', 'M', 'A', 'S', 'A', 'M', 'X', 'A', 'M', 'M'},  
            {'X', 'X', 'A', 'M', 'M', 'X', 'X', 'A', 'M', 'A'},  
            {'S', 'M', 'S', 'M', 'S', 'A', 'S', 'X', 'S', 'S'},  
            {'S', 'A', 'X', 'A', 'M', 'A', 'S', 'A', 'A', 'A'},  
            {'M', 'A', 'M', 'M', 'M', 'X', 'M', 'M', 'M', 'M'},  
            {'M', 'X', 'M', 'X', 'A', 'X', 'M', 'A', 'S', 'X'}  
        };  
        String word = "XMAS";  
        System.out.println("The word '" + word + "' appears " + findWordInGrid(grid, word) + " times in the grid.");  
    }  
}
```

```

private static int findWordInGrid(char[][] grid, String word) {
    int count = 0;
    int rows = grid.length;
    int cols = grid[0].length;
    int len = word.length();

    for (int x = 0; x < rows; x++) {
        for (int y = 0; y < cols; y++) {
            for (int dir = 0; dir < 8; dir++) {
                int k, rd = x + dx[dir] * (len - 1), cd = y + dy[dir] * (len - 1);
                if (rd >= 0 && rd < rows && cd >= 0 && cd < cols) {
                    for (k = 0; k < len; k++) {
                        if (grid[x + k * dx[dir]][y + k * dy[dir]] != word.charAt(k)) {
                            break;
                        }
                    }
                    if (k == len) {
                        count++;
                    }
                }
            }
        }
    }
    return count;
}
}

```

When I executed this Java code, I received the expected result:

“The word 'XMAS' appears 18 times in the grid.”