```
/*
    Time complexity: O(N * M)
    Space complexity: O(N * M)

    where N and M are the rows and columns respectively of the board
*/

int validPoint(int x, int y, int n, int m) {
    return (x >= 0 && x < n && y >= 0 && y < m);
}

bool dfs(vector<vector<char>> &board, vector<vector<bool>> &used, string &word, int x,
int y, int wordIndex, int n, int m) {
    if (wordIndex == 11) {
        return true;
    }

    used[x][y] = true;

    bool found = false;

    int dXdY[8][2] = {{-1,-1},{-1,0},{-1,1},{0,-1},{0,1},{1,-1},{1,0},{1,1}};

    for (int i = 0; i < 8; ++i) {
        int newX = x + dXdY[i][0];
        int newY = y + dXdY[i][1];

        if (validPoint(newX, newY, n, m) && board[newX][newY] == word[wordIndex] &&
!used[newX][newY]) {
            found = found | dfs(board, used, word, newX, newY, wordIndex + 1, n, m);
        }
    }

    used[x][y] = false;

    return found;
}

bool hasPath(vector<vector<char>> &board, int n, int m) {
    bool foundPath = false;
    string word = "CODINGNINJA";
    vector<vector<bool>> used(n, vector<bool>(m, false));

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (board[i][j] == word[0]) {
                foundPath = dfs(board, used, word, i, j, 1, n, m);
                if (foundPath) break;
            }
        }

        if (foundPath) break;
    }

    return foundPath;
}
```