CSC 270 Final Project: Word Search Solver

Michael Vessia

December 5, 2014

Introduction

A word search puzzle is a word game that consist of the letters of words placed on a grid, which usually has a square shape. The objective is to find all of the valid words inside the grid. Typically the words are placed vertically, horizontally, and diagonally. This solver, written in C, aims to find all of the specified words in a given word search.

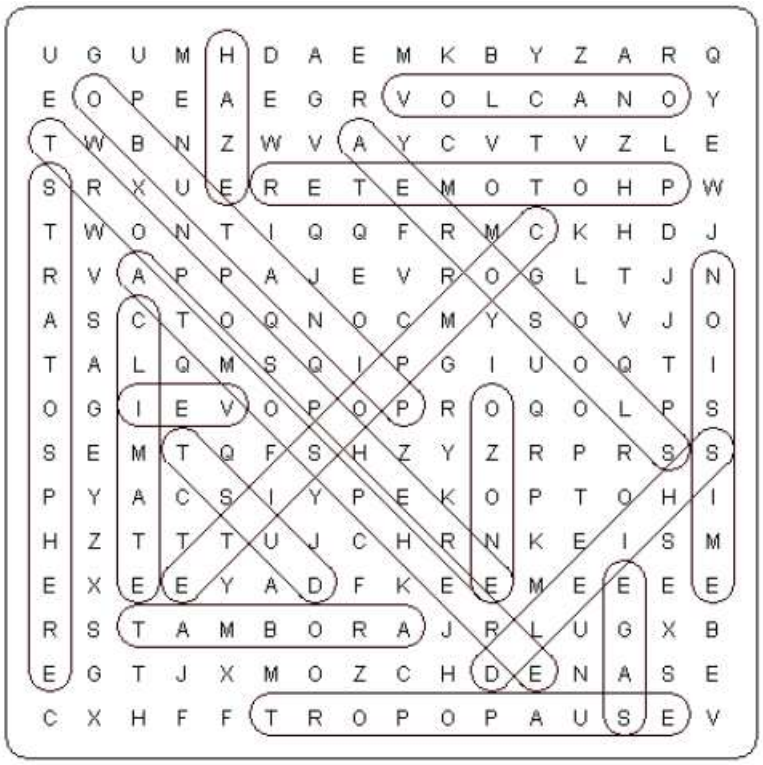

Figure 1: Example of a solved word search puzzle

Puzzle Solver

The Program will assume:

- For simplicity, the puzzle will always be a 10x10 square.
- Two files will be given, one containing the puzzle, and one containing the words to be searched for.
- The puzzle letters will be in upper case, with no delimiters.
- The words will be comprised of alphabetic characters.
- The words will be at least two characters long.
- The word list will not contain blank lines.
- All the words in the list must be in the puzzle.
- The word list will not contain duplicate words.

Algorithm

In order to find the words of the puzzle, the program will search for every occurrence of the first letter of the word. Once the first letter is found, the second letter is searched for in all directions surrounding the first. Once we have found the first and second letter next to each other, we are able to discern the orientation (up, down, left, right, diagonally) that the word may possibly be going in. This can be used to search for the rest of the word, cutting search time down significantly, as instead of looking in 8 directions for every letter of the word we now only need to look in 1 direction. If the rest of the word is found, the row, column, and orientation of the word will be outputted to the user, so they can see where the word was in the puzzle.

Functions

findWord

Searches for a word in the puzzle and determines the orientation.

Arguments:

char word[] - The word we are searching for
char puzzle[][] - The puzzle we are searching
orientation - struct containing the words orientation in the puzzle
int row, col - will hold the row and column the word was found in for printing.
Output: returns 0 if not found, 1 if found.

searchRestOfWord

Recursively searches the rest of the word once the orientation is found.

Arguments:

char word[] - The word we are searching for
char puzzle[][] - The puzzle we are searching
orientation - struct containing the words orientation in the puzzle
int row, col - will hold the row and column the word was found in for printing.
Output: returns 0 if not found, 1 if found.

readPuzzle

Reads in the puzzle from the file.

Arguments:

char fileName[] - The filename where the puzzle is located.
char puzzle[][] - The puzzle we are reading into.
Output: None.

printPuzzle

Prints out the puzzle to standard output.

Arguments:

char puzzle[][] - The puzzle we are printing.
Output: None.