Crossword-Backtracking

Solution for Assignment 3 (Word Puzzle) in CS 480 at ODU. Implements recursive backtracking to find arrangements of words adhering to the constraints inherent in the shape of crossword puzzles.

Usage

The main driver is the main function in src/crossword.design.py.

Just execute that file with python, e.g. (from the root directory): `python src/crossword.design.py`

Args/Options

There are a total of seven (7) optional arguments:
`--puzzle [PUZZLE]`

> where [PUZZLE] is the path to a text file containing a list of variables that together specify the puzzle's structure

`--words [WORDS]`

> where [WORDS] is the path to a text file containing a list of word candidates for assignment to variables

`--heuristic [HEURISTIC]`

> where [HEURISTIC] is either mrv, degree, mrv+degree, or first_unassigned. Determines the heuristic used to choose the next unassigned variable during backtracking

`--disable_forward_check`

> disables forward checking, so that the backtracking algorithm will no longer try to avoid dead ends

`--sort_domains`

> enables domain sorting based on letter frequency analysis from: www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html in an attempt to favor candidates that minimally diminish the domains of mutually constrained variables

`--hide_stats`

> hides additional stats normally displayed underneath the solution

`--randomize`

> randomizes domain order for each variable (warning: causes wildly varying runtimes)

The default execution (when no additional args are supplied) is equivalent to:
`python src/crossword_design.py --puzzle resources/puzzles/heart.txt --words resources/words/words.txt --heuristic mrv`

File Structure

Three (3) puzzles are supplied in resources/puzzles, namely mini.txt, larger.txt, and heart.txt, with the first two being relatively trivial/simple compared to the last.
Two (2) word lists are supplied in resources/words, namely mini_words.txt and words.txt, with the latter being fairly comprehensive and the former being extremely small (and insufficient for all of the supplied puzzles except mini.txt).

Supplying Your Own Puzzles/Word Lists

Puzzles and word lists can be anywhere, so long as you supply the requisite paths via the args discussed above.

Both are expected to be text files.
Word lists should be solely populated with one word per line and can contain an arbitrary number of words. Note that the program does not expect, and cannot account for, words longer than 45 characters long. Words are expected to be lowercase only with no capitalization.

The general line for a puzzle file should look like the following:
`a,b,c,d,e`

> where a is the variable's number (specified by the puzzle), b is its orientation (0 for across, 1 for down), c is its starting x coordinate, d is its starting y coordinate, and e is its length.

Predetermined letters can be added as extra constraints by first adding the following line: `[Letters]` followed by an arbitrary number of lines of the form: `a,b,c`

> where a and b are the are the respective x and y coordinates, and c is the letter itself (lowercase is expected).

See the files in resources/puzzles for examples.