Patryk Stroński 242564

Lab 2 part 2 Fill in puzzle

Introduction

Fill in puzzle is similar to sudoku, even though we do not insert numbers, but letters which needs to make up words. Each word list (later called 'lexicone') along with the crossword shape is given. The task consumed much computational power and the crosswords do spend much time on being evaluated.

Definition as CSP

Each word slot is a variable. A word is a string consisting of at least two signs, delimited by board ending or a hash. The words overlap each other and the joint parts are the constraints to be met.
Lexicone – set of all words that needs to be used
n – number of all words (horizontally and vertically)
V(0..n-1) ⊂ Lexicone
D(0..n-1) ⊂ Lexicone

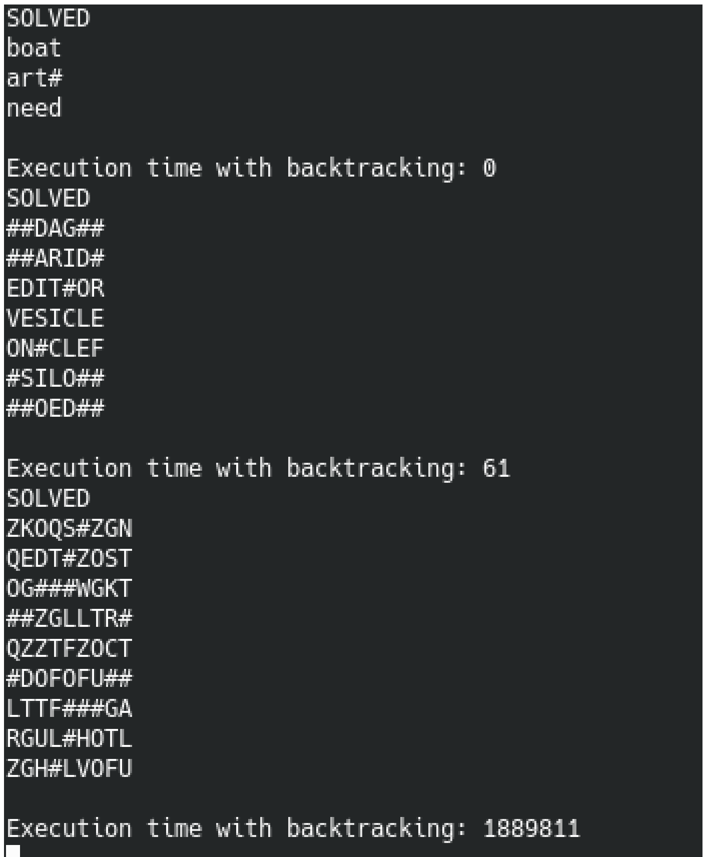
Constraint 1: each word can appear only once
V(k) ⊂ Lexicone && V(k') ⊂ Lexicone && k' != k && V(k) != V(k')

Constraint 2: The words are predefined by their junctions.
Here we assume: x,y – cartesian coordinates, ind – one-dimensional coordinate.
Ind = x + y * width && x < width && y < height
Any ind,where
C(k) – coordinate set for a word of index k
For k, where: ind ⊂ C(k) && ind ⊂ C(k') <=> V(k) != V(k') && C(k, ind) == C (k',ind) == CharAt(ind)

Constraint 3:
Words cannot be put in the places of hashes:
N – number of words
CharAt(ind) == '#' => ind !⊂ C(0..N)

My solution: backtracking with forward checking

Here I implemented only one solution – backtracking with forward checking. It finalized till the third puzzle. In order to get the results I used several optimization methods:
1. Each domain is held on the vector of Variables. The Variable structure holds information about word slot length, the slot's domain and index at which the word starts. The domain is calculated firstly in the beginning and it is being edited each time some word it used.
2. As the domains for words are taken from Variable struct, the general domain is also passed and processed. It is needed in order to perform final validation



```
SOLVED
boat
art#
need

Execution time with backtracking: 0
SOLVED
##DAG##
##ARID#
EDIT#OR
VESICLE
ON#CLEF
#SILO##
##OED##

Execution time with backtracking: 61
SOLVED
ZKOQS#ZGN
QEDT#ZOST
OG###WGKT
##ZGLLTR#
QZZTFZOCT
#DOFOFU##
LTTF###GA
RGUL#HOTL
ZGH#LVOFU

Execution time with backtracking: 1889811
```

3. I insert only horizontal words and assume the vertical ones will be filled itself. Also, the given crosswords make it possible as there is no pattern like #_# which would make this approach invalid.
4. The vertical words I validate – I extract words from them and check whether they match with the already narrowed down domain (the words already used are not in this final domain).

The time in the screenshot is reflected in milliseconds – as we see the third puzzle took around 0.5h which is very much comparing to previous puzzles.
On the other hand, I also tried using simple backtracking without changes on domain (only I was removing the used words), the second one was even extremally slow.