Cryptocrossword

Jonas Karlsson

Robin Sund

Emil Österberg

Contents

[Introduction 3](#_Toc381732446)

[User manual 3](#_Toc381732447)

[Create a puzzle 3](#_Toc381732448)

[Documentation 3](#_Toc381732449)

[Data Structures 3](#_Toc381732450)

[Algorithms 3](#_Toc381732451)

[Function Description 3](#_Toc381732452)

[The solve function 3](#_Toc381732453)

[The generate function 4](#_Toc381732454)

[Difficulties 4](#_Toc381732455)

# Introduction

Have you ever encountered a cryptographic puzzle? In that case you may have pondered how you would go about solving it, and what solution would be superior? In this project we have constructed a program that allows you not only solve, but create crypto puzzles of your own! By using this document and our software, you can quickly grasp the fundamentals of crypto crosswords in addition to making ones for your friends to solve as well!

This program is able to create and solve cryptographic puzzles. For solving it uses a dictionary and recursively tests all possible combinations of bindings between words and number until a solution is reached.

The program creates puzzles by receiving a list of words from the user and then tries to make a puzzle as concise as possible.

# User manual

## Solve a crossword

To solve a crossword, the solve function needs to be called with a list of integer lists, where each list represents a row of the crossword.

The function call could look something like this:

solve [ [1, 2, 3, 4], [5, 2, 2, 3], [3, 2, 6, 5] ]

The function would match the possible bindings between characters and integers to recognize that the sought words would be POLY, FOOL, and LOAF. And would thus return a list with tuples containing each number in the crossword paired with the specific character it represents.

Val it =

## Create a crossword

To create a puzzle you must call the generate function with \_\_\_\_\_\_

# Documentation

How we did it. Including:

## Data Structures

## Algorithms

## Function Description

### The solve function

### The generate function

We discussed what kind of crypto crossword we were wanted the function to return. After some deliberating, we unanimously agreed that the “best” crossword was that which had the most intersecting words, as this is the most basic aspect of any crossword. Hence we came to the conclusion that the function should recursively check all the possible order of words, in addition to all the possible ways of combining these into a puzzle. If the function did this successfully, the final crossword would have as many intersections as possible given the words with which it was built.

# Difficulties

A notable aspect of the crossword is that all the lists making up the different rows have to be of the same length. Although, this does not limit the use of different words as all the lists will be filled out with 0’s until they are of the same length as the longest list.

The most difficult part was perhaps the creation of the generate function. To make the best crossword possible with the given words showed itself to be quite the challenge, as the program were forced to recursively check all possible solutions before deciding on the most compact one.