Cryptocrossword

Jonas Karlsson

Robin Sund

Emil Österberg

Contents

[Introduction 4](#_Toc381816345)

[User manual 4](#_Toc381816346)

[Solve a crossword 4](#_Toc381816347)

[Create a crossword 4](#_Toc381816348)

[Documentation 4](#_Toc381816349)

[Data Structures 4](#_Toc381816350)

[Algorithms 4](#_Toc381816351)

[Function Description 5](#_Toc381816352)

[The solve function 5](#_Toc381816353)

[The generate function 5](#_Toc381816354)

[Difficulties 5](#_Toc381816355)

# 

# 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 the most efficient? 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 assembles it, as intricate as possible, into a cryptographic crossword of whole numbers.

# 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.

It takes in consideration the words in the columns as well, from top to bottom. And would thus return a list with tuples containing each number in the crossword paired with the specific character it represents.

## Create a crossword

To create a puzzle you must call the generate function with a list of words, it will then step through all the different possible ways of constructing the crossword.

# Documentation

## Data Structures

## Function Description

### The solve function

The solve function is rater straight forward in how it attempts to solve any given crossword. Primarily, the first thing we needed was a sort of dictionary, in order for the function to be able to find the matching words. We acquired a small dictionary from Uppsala Universitet’s students’ portal[[1]](#footnote-1), from the 07/08 years edition of this course.

#### Algorithm

The basis of solve is using a depth-first search (DFS) for a valid binding between characters and integers.

Firstly solve

### 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 most optimal 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.

#### Algorithm

As we want to find and compare all the possible solutions and find the best, unlike solve where we want to find any indiscriminate solution, the algorithm can be compared to that of a breadth first search (BFS).

The algorithm runs through a series of steps in order to convert the list of text strings into a complete crossword.

Firstly it calls on the on the stringToInteger function with the list of string lists. To be able to reach the letters the function explodes the words into characters. It then steps through the lists and compares the characters in each string to a memory list created with the function call. If the function encounters a new letter it assigns this to a new integer value and remembers it for the entire run. If, by chance, it finds a character that already is in the memory list, it swaps the element in question with the saved value. By this means the function will return a list of lists, containing the words converted into integers and the value of a character is determined by the order of appearance of that specific character.

With a list of integer lists, generate calls a series of auxiliary functions. These calculate all the viable placements of list in every possible order. findPlacement’ returns the all the combinations of building the crossword from a given list, and the main’ function calls findPlacement’ with every permutation of the input list (a list of n strings would cause n! calls from main’).

During the previous step, generate save the puzzle containing the least amount of characters, only counting those that are not zero, and constantly comparing it with any new results. The reason for counting the non-zero characters is that the more intersections a given crossword possesses, the better it is, by our definition. This means that when the function ends, it will be left with the most concise crossword.

It is worth noting that simply counting the breath and height of the crossword not necessarily returns the most optimal crossword, as there is a possibility that we may be left with a series of list completely separated from each other. Even if that solution is more conserving of space, it barely counts as a crossword in our standards.

Lastly it prints the output to a text file for the user to save for later use or discard. If a save already exists, the crossword will be appended to the previous saves enabling the user to create a library of cryptographic crosswords.

# 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.

# References and Resources

1. Uppsala Universitet, [www.studentportalen.se](http://www.studentportalen.se), 2014-02-24.

1. Studentportalen, Uppsala Universitet [↑](#footnote-ref-1)