**Algorithms Design and Analysis**

**ALGO PROJECT**horizontal line

**Group Members:**  
Mohammad Usama k180154  
Abdul Musawwir Siddiqui k180185

horizontal line

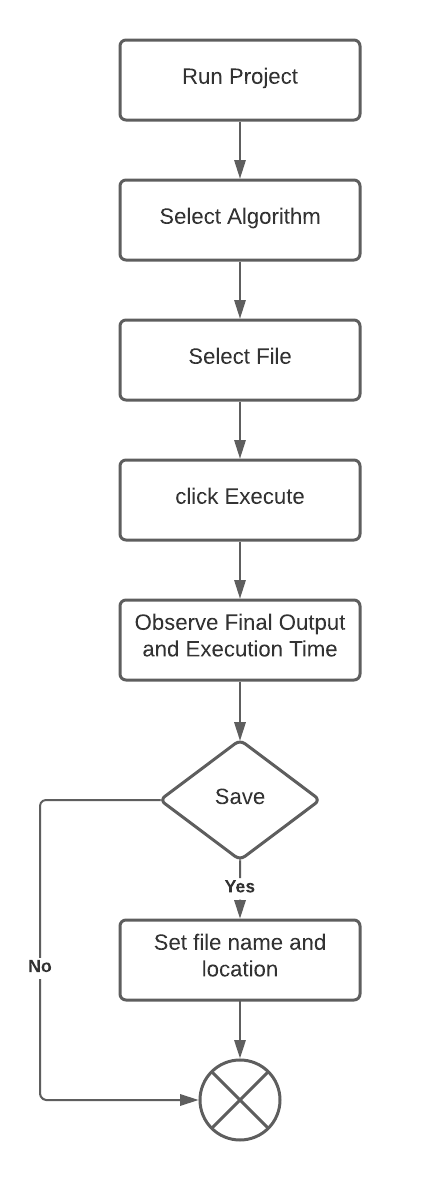
# Abstract

Dynamic Programming (DP) is an algorithmic technique for solving an optimization problem by breaking it down into simpler subproblems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its subproblems. We have selected 10 such algorithms and a handful of random inputs to observe efficiency of computation achieved by using dynamic programming approach while implementing these algorithms. In this report we will further discuss the working of the system in detail and observe the estimated time of the algorithms.

# Introduction

This project is implemented using python and PyQT as a library for GUI. It provides a nice, user friendly Interface, in which the user selects their chosen algorithm, and file for input (assuming it is a valid file for chosen algorithm). Here the Source code of the chosen algorithm is also shown. On execution the project gives the output along with the execution time of the algorithm. This report will further discuss the working of the system in detail, the variety of inputs generated and used, and compare and contrast the results obtained along with their efficiency.

# Proposed System



# 

# Experimental Setup

Algorithms require inputs relevant to their scope of computation. We therefore generated the following types of inputs for respective algorithms:

### Input Type 1:

Two sequences by using alphabets of the names of the group members in random order and repetition with the random length from 30 to 100 characters.  
Names: MOHAMMADUSAMA  
 ABDULMUSAWWIR  
Algorithms: Longest Common Subsequence  
 Shortest Common Supersequence  
 Levenshtein Distance

### Input Type 2:

A sequence of n random numbers from 0 to 100 (n varies from 30 to 100).

Algorithms: Longest Increasing Subsequence  
 Matrix Chain Multiplication  
 Partition Problem  
 Coin-change-making-problem

### Input Type 3:

Generate a set of n points (n is a random number varying from 10 to 100) with weights and values ranging from 1 to 100. Capital W is the last three digits of our roll number i.e. 168.

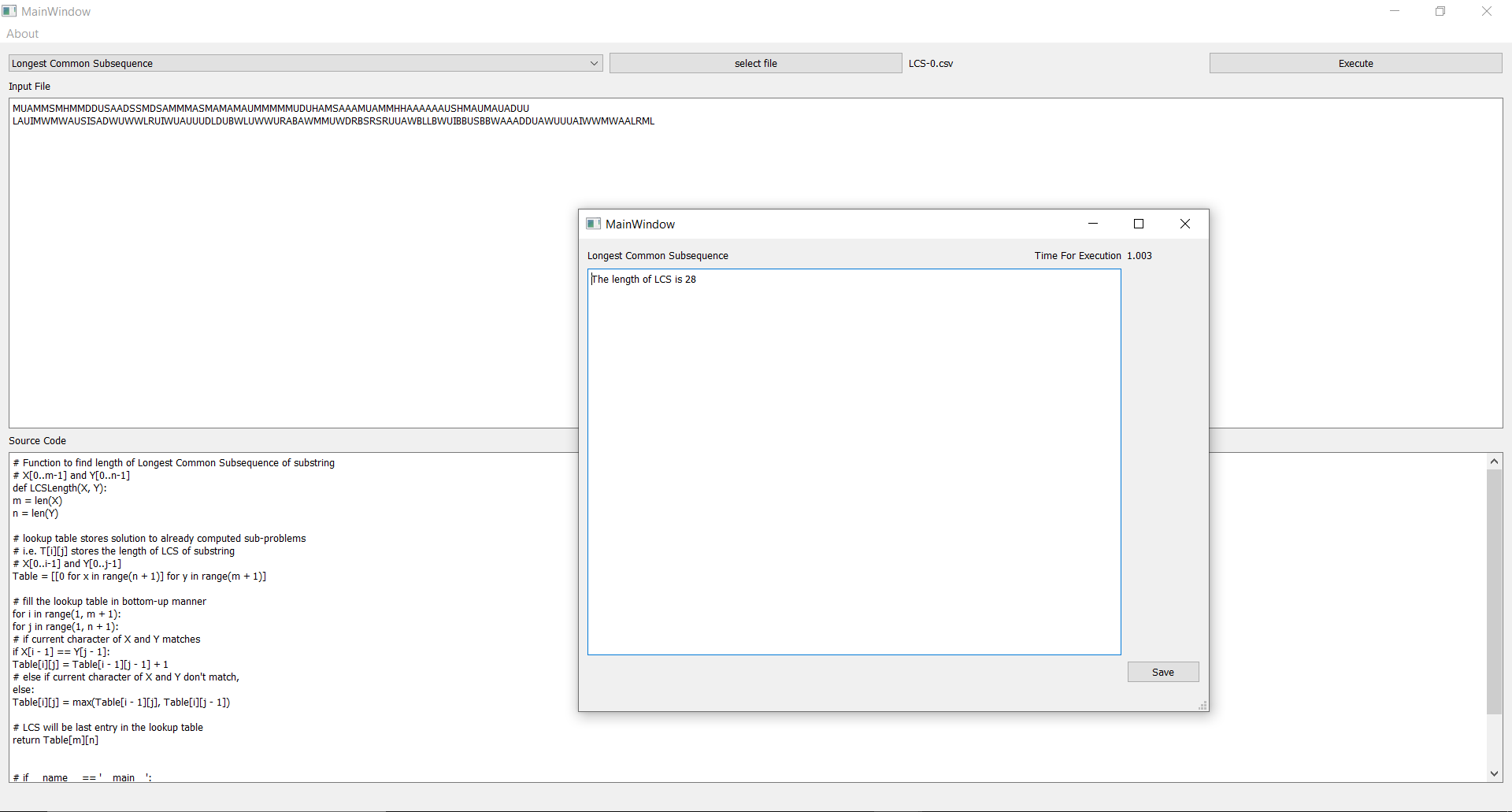
Algorithms: 0-1 Knapsack Problem  
 Rod Cutting

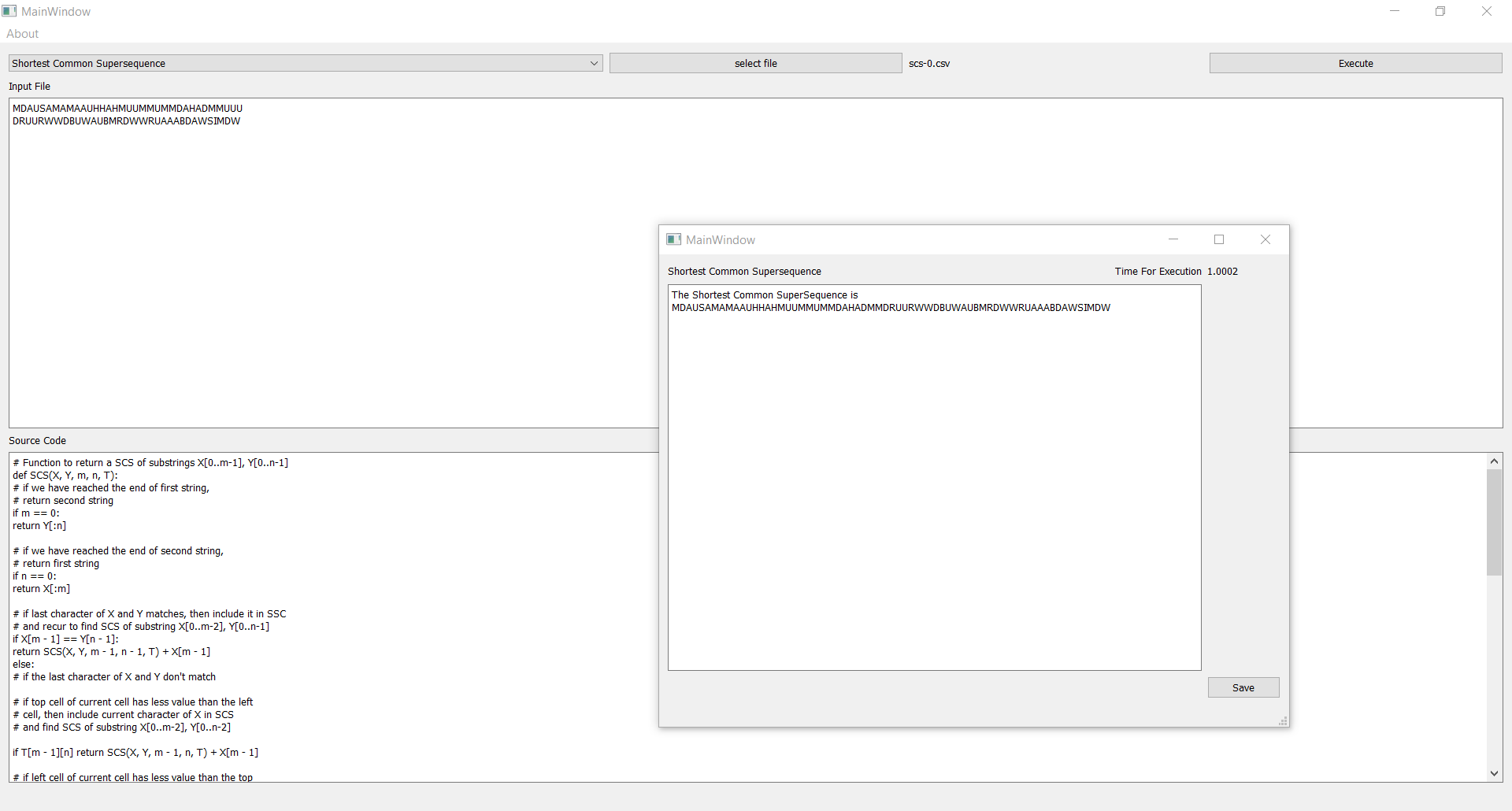
### Input Type 4:

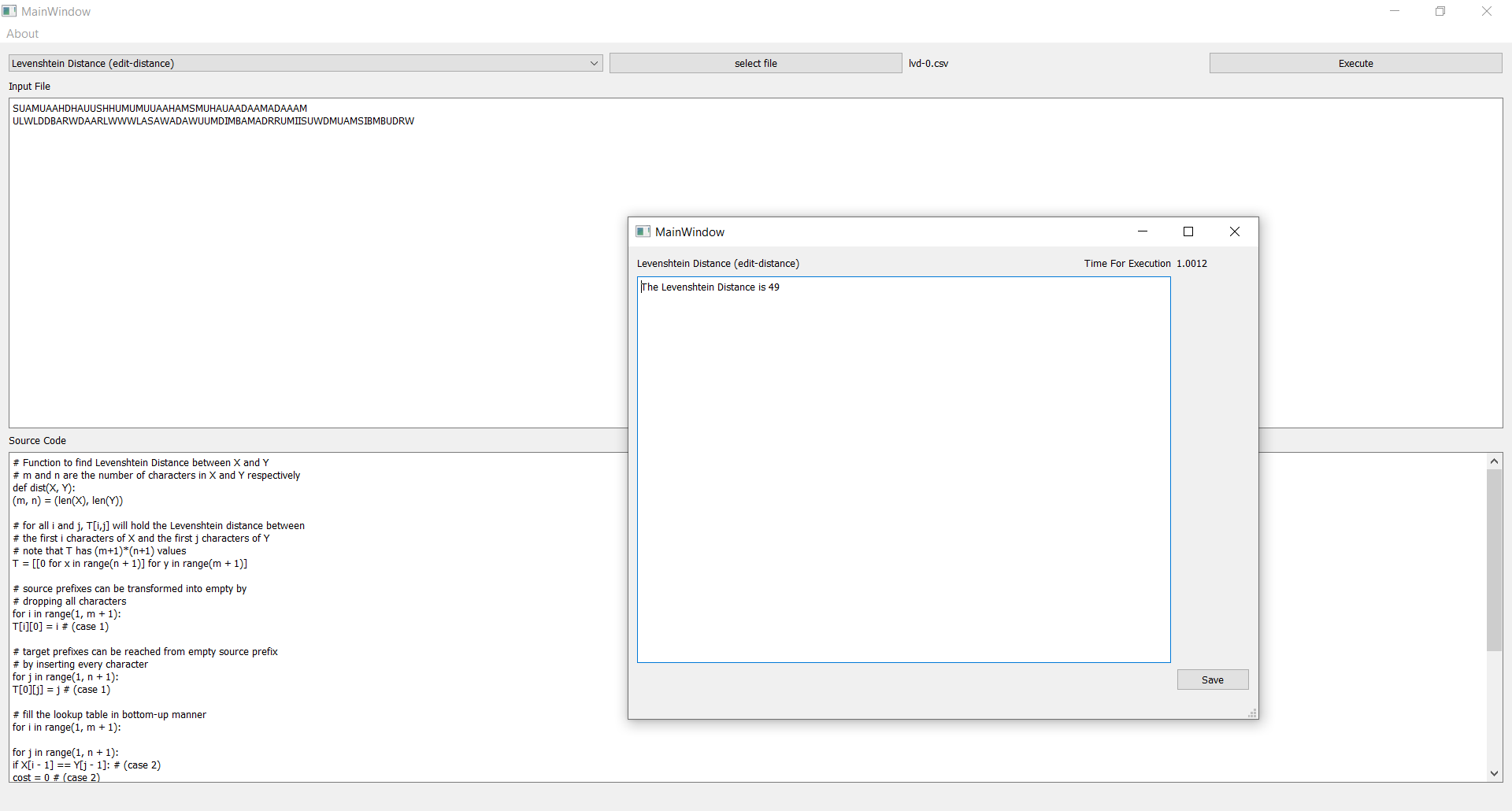
A set of randomly generated strings from alphabets a to z as S and our names as Input for word break.

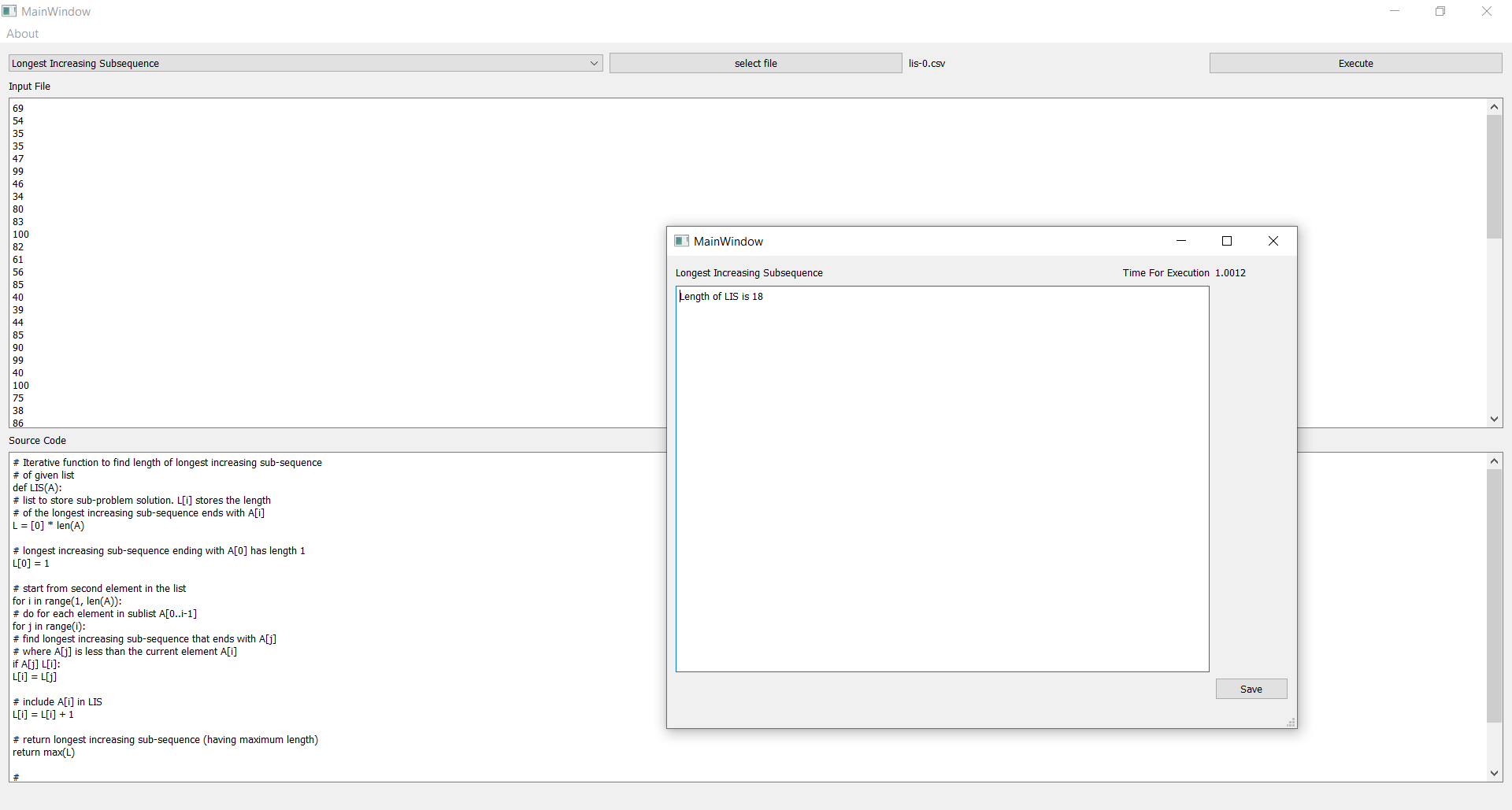
Algorithm: Word Break Problem

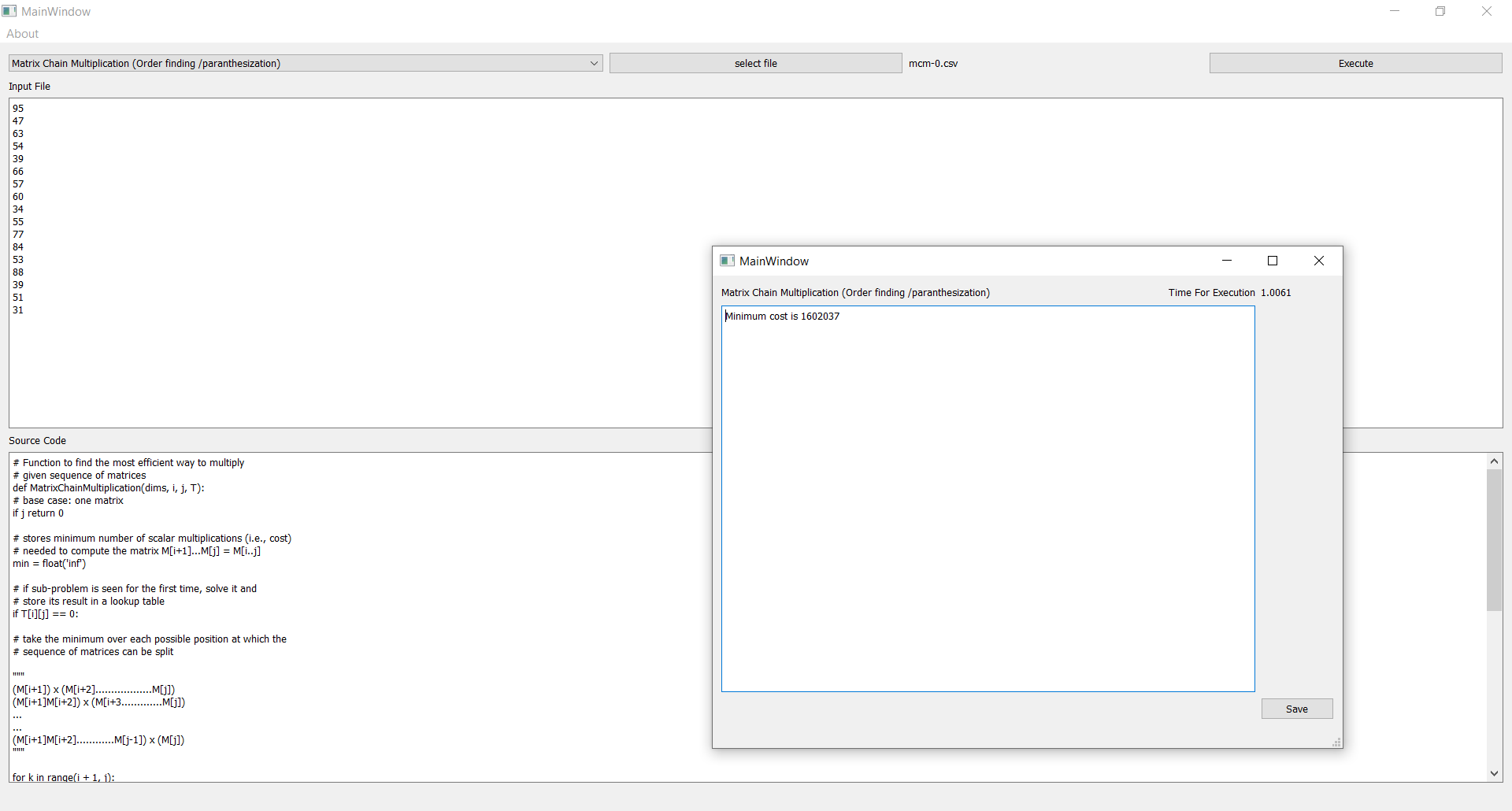
# Results

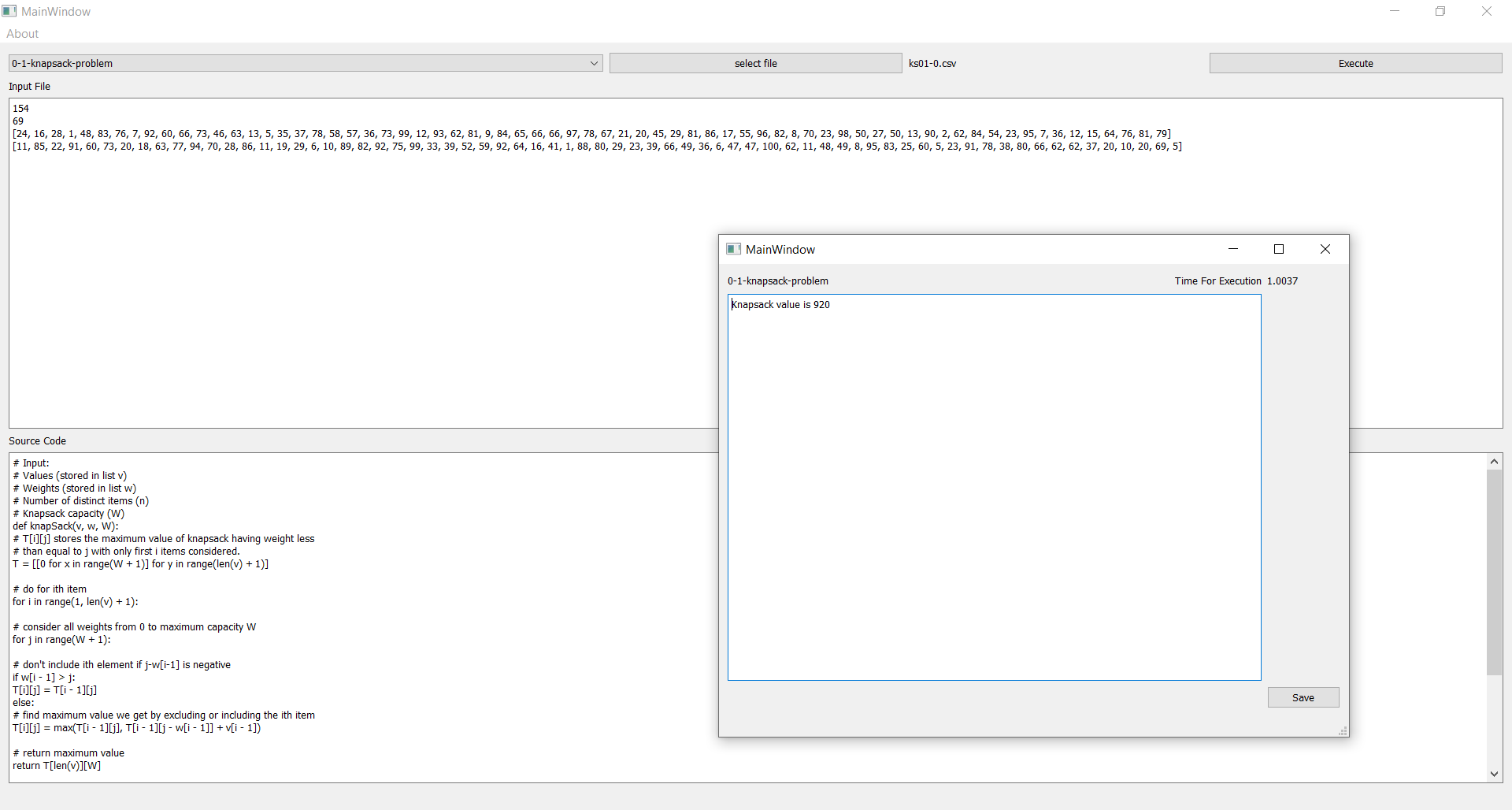


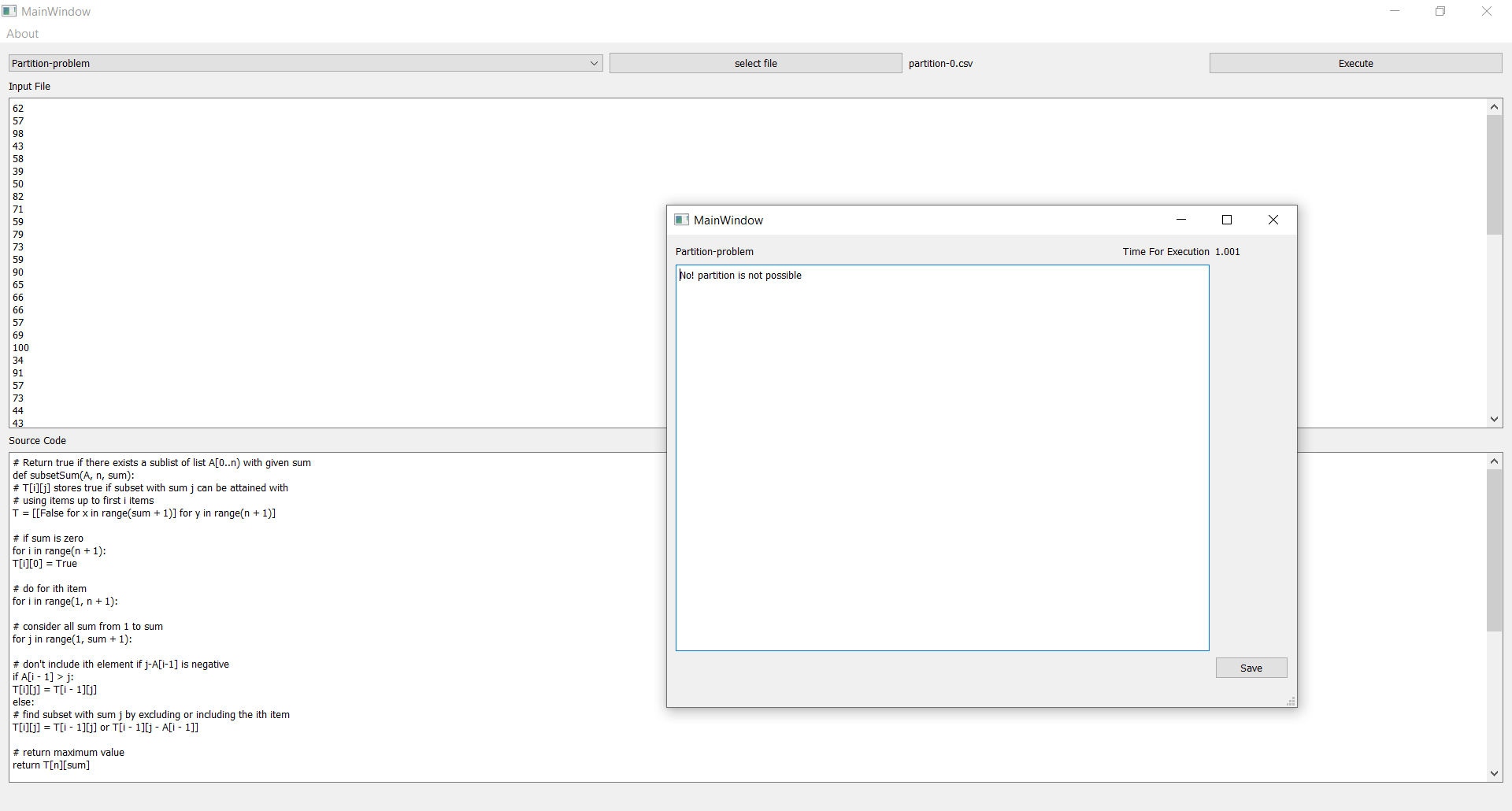


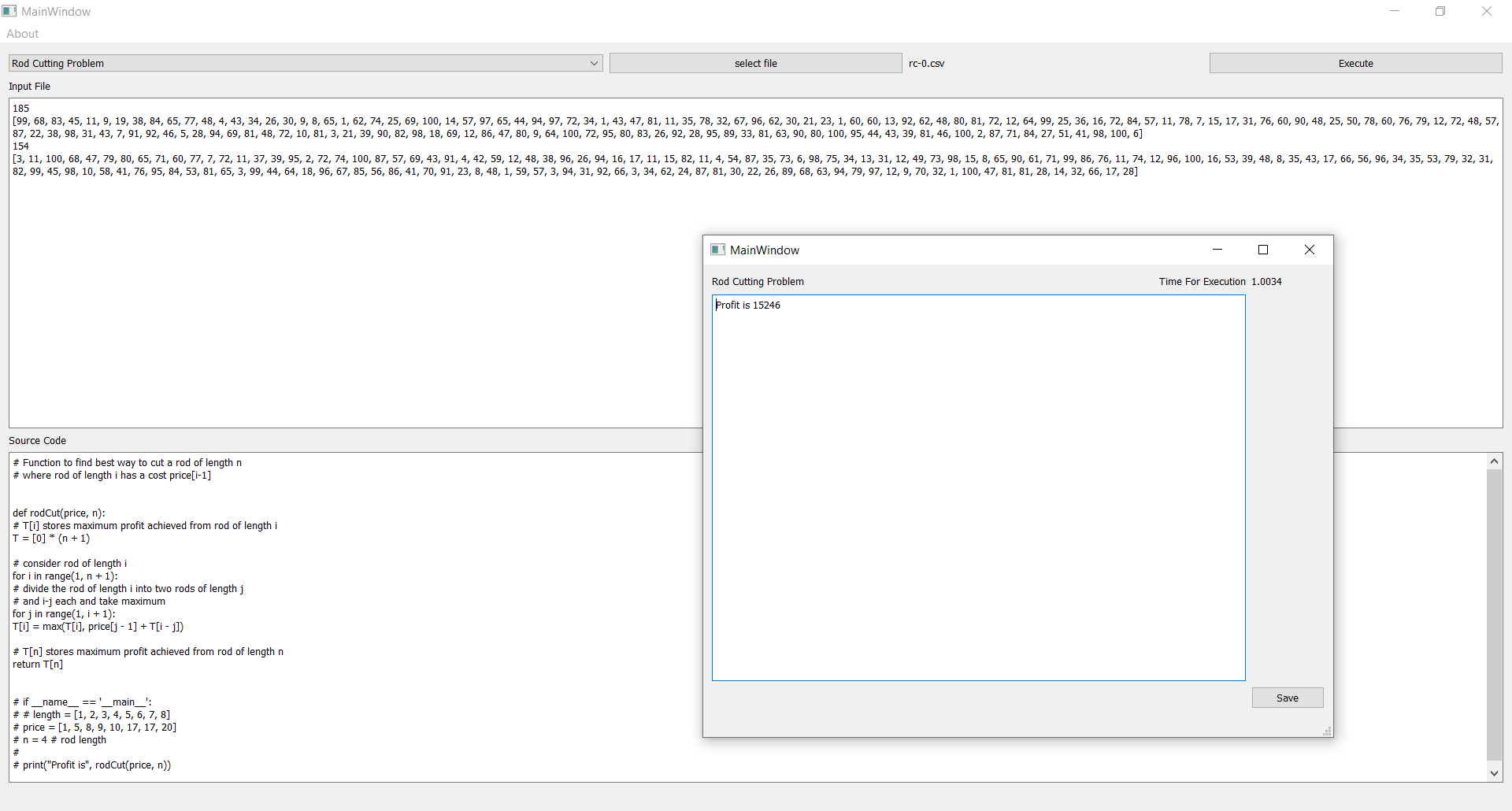


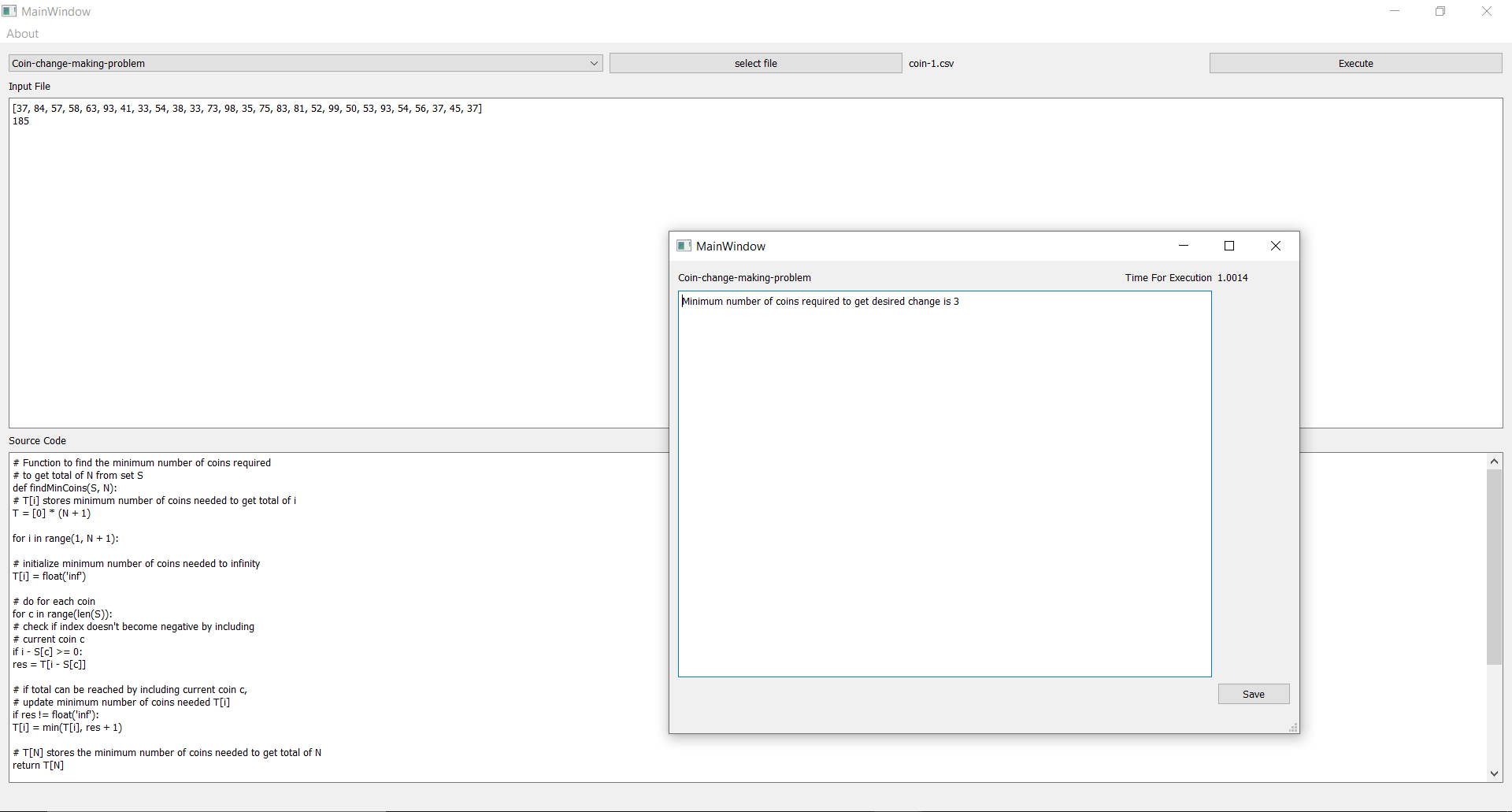


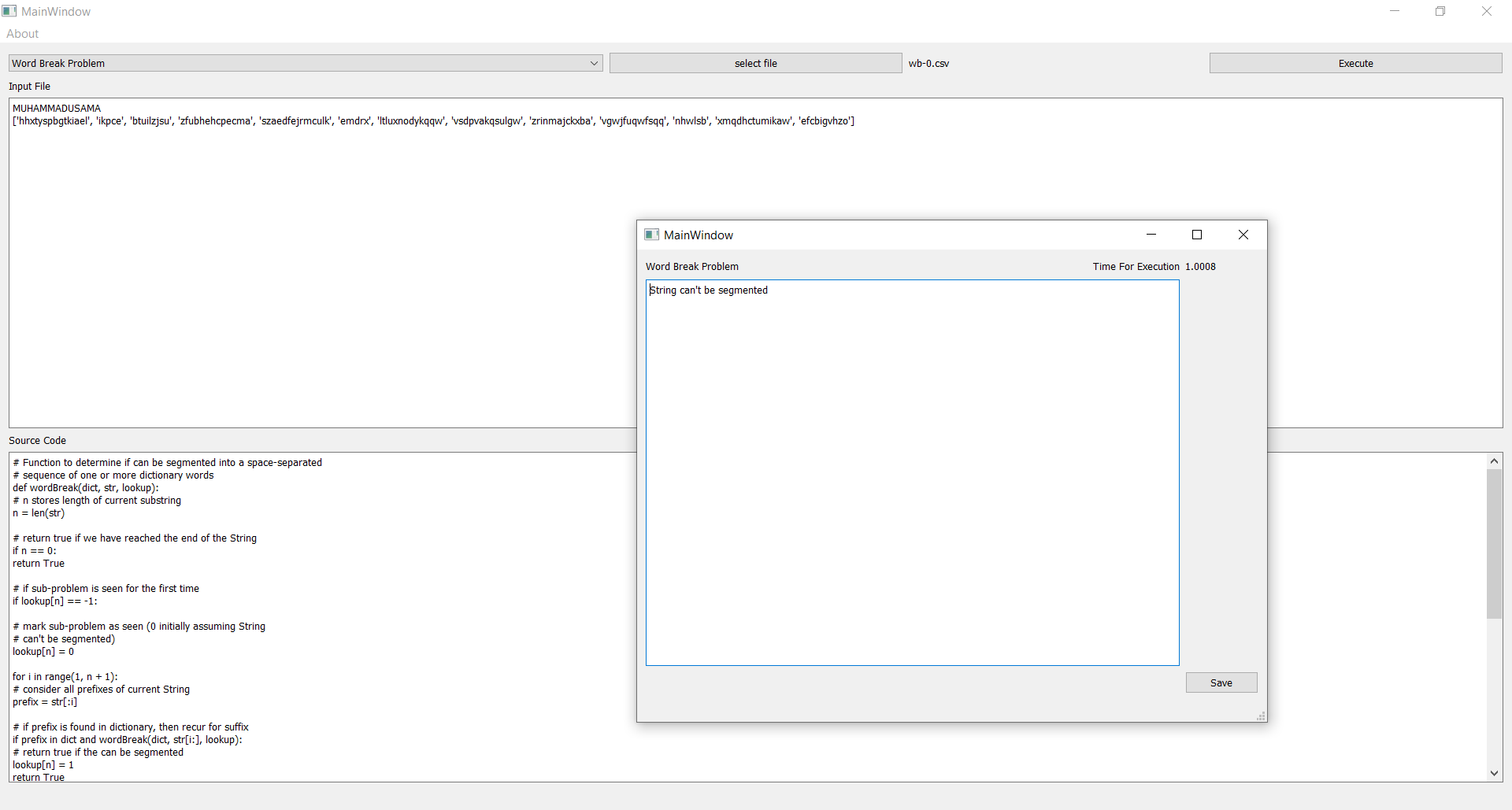












# Conclusion

All in all, a significant reduction in the computation was observed after Dynamic Programming was used i.e. result of the similar subproblems were stored rather than being computed again and again. Moreover, time consumption also decreased in contrast to recursive approaches.

# References

<https://medium.com/techie-delight/top-10-dynamic-programming-problems-5da486eeb360>