2801ICT Computing Algorithms – Assignment 1

Timothy Tadj -s5178358

Algorithmic Design

Overview

The task is to create a program that takes in an input file that contains a number and limits and finds combinations of the primes that sum to that number within the defined limits.

Algorithm description

The algorithm should first be able to define the number that is being summed to and the limits of number of primes added together, the algorithm should be able to sum multiple of the same prime number. A good way to do this might be to make a recursive function to create combinations by looking through a list of primes. The reason why this approach is good is because it can backtrack to find more combinations. We can check each combination as we go through and append it to a list and save that combination if it fulfills all the requirements of the search.

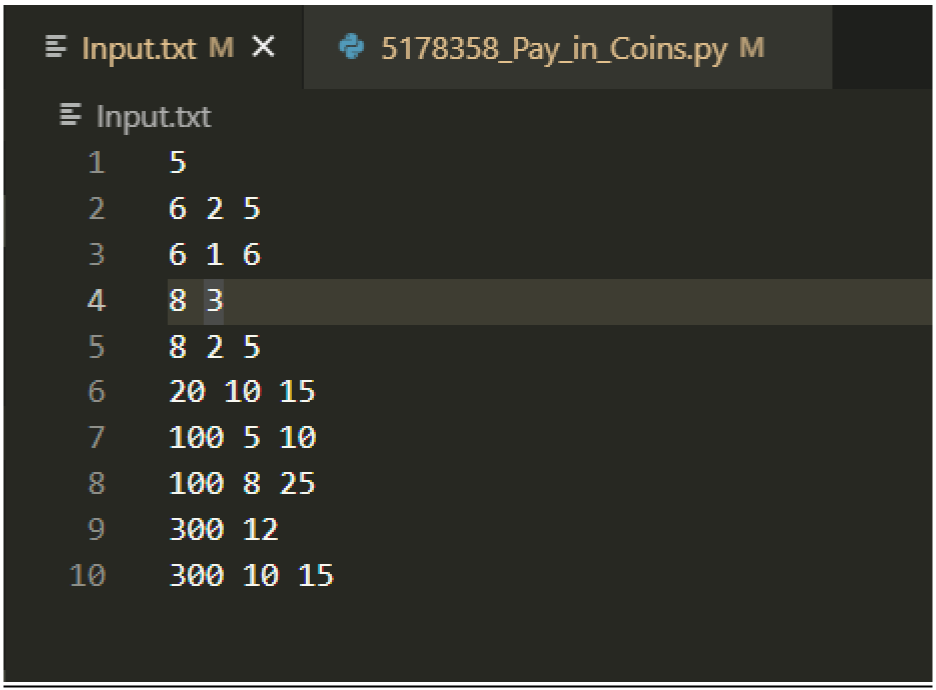
Pseudo-Code

```
//This code could be implemented as a class to access global variables safely.

Global variables; sumto, primes, lowerbound, upperbound, combination_count


Sum_rec(current_sum, start, output, result)
    If current_sum == sumto
        If sizeof(result) <= upperbound and sizeof(result) >= lowerbound
            Combination_count ++
            Output.append(result)
    For i from start to size of (primes):
        Temp_sum <- current_sum + primes[i] //add next prime
    If temp_sum <= sumto and sizeof(result) < upperboud //if we go over bounds, stop
        Result.append(primes[i]) //apply change
        Sum_rec(tem_sum, I, output, result) //do again
        Result.pop() //discard change
Primesum() //initialise everything
    Output <- array
    Result =<-array
    Sum_rec(0, 0, Output, Result)
    If (sumto not in primes) and (lowerbound ==1)    //add gold coin
        Combination_count ++
                Output.append(sumto)

        Return output //array of valid combinations(represented as arrays)
```

Result and Algorithm Analysis

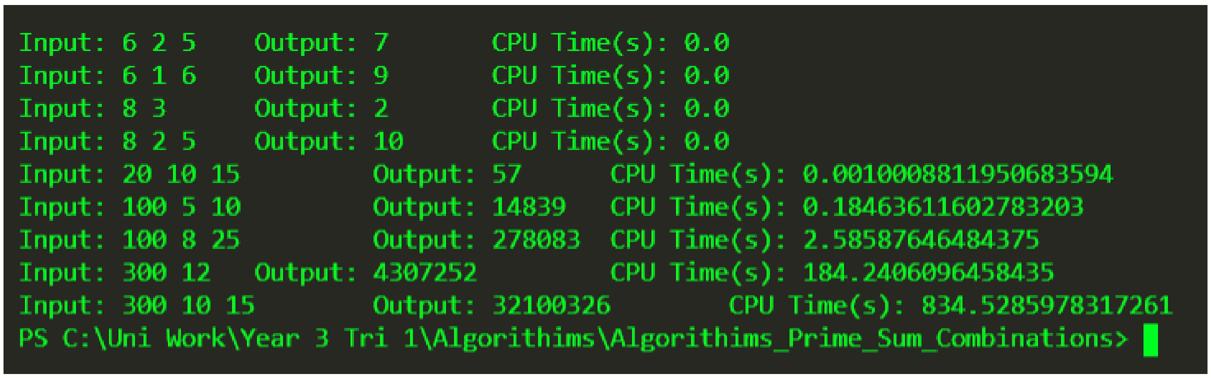
Result

<u>Input File:</u>



```
5
6 2 5
6 1 6
8 3
8 2 5
20 10 15
100 5 10
100 8 25
300 12
300 10 15
```

<u>Output:</u>



```
Input: 6 2 5    Output: 7       CPU Time(s): 0.0
Input: 6 1 6    Output: 9       CPU Time(s): 0.0
Input: 8 3      Output: 2       CPU Time(s): 0.0
Input: 8 2 5    Output: 10      CPU Time(s): 0.0
Input: 20 10 15         Output: 57      CPU Time(s): 0.0010008811950683594
Input: 100 5 10         Output: 14839   CPU Time(s): 0.18463611602783203
Input: 100 8 25         Output: 278083  CPU Time(s): 2.58587646484375
Input: 300 12   Output: 4307252         CPU Time(s): 184.2406096458435
Input: 300 10 15        Output: 32100326        CPU Time(s): 834.5285978317261
PS C:\Uni Work\Year 3 Tri 1\Algorithims\Algorithims_Prime_Sum_Combinations>
```

As can be seen the results of my algorithm match the example results provided in the task sheet. So, this Algorithm is correct as it produces the correct result.

Performance Analysis

The performance of the algorithm is better than the performance of the example output provided in the task sheet, so it is assumed that my algorithm implementation is valid in terms of speed. However, when solving for an input of “300 10 15“my algorithm used up 5 GB of memory to solve the problem. This is due to my use of recursion as every time a function is recursively called, it is added to the stack, and this happens thousands of times. Due to my system having enough RAM this is not an issue, however if solving for a problem with looser bounds this will create a significant problem for processing speed as the system will have to save memory to permanent storage which will slow down processing significantly. Overall, the performance of this algorithm seems to fulfill the requirements for the assignment.