Laporan
Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II Tahun 2020/2021

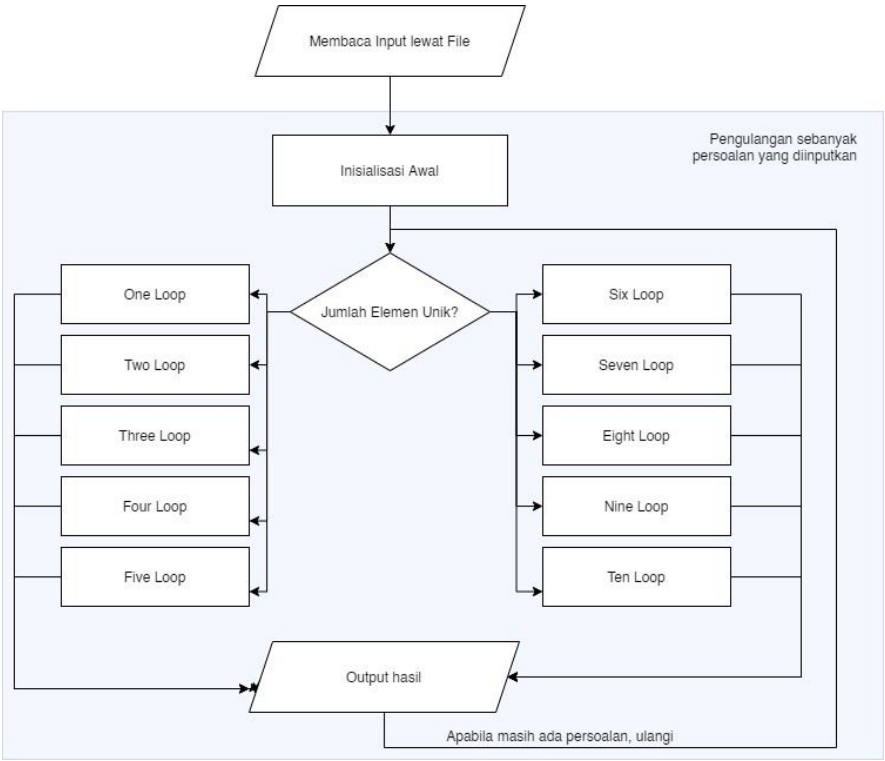
Penyelesaian *Cryptarithmetic* dengan Algoritma *Brute Force*

Shaffira Alya Mevia
13519083

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

1. Algortima *Brute Force* dalam Persoalan

Pada persoalan tugas kecil ini, untuk menyelesaikan masalah *cryptarithmetic* memerlukan beberapa langkah yang apabila di visualisasikan dengan *flowchart* adalah sebagai berikut.



Gambar 1.1 *Flowchart* sederhana untuk menggambarkan cara kerja program

Berikut adalah penjelasan langkah-langkah algoritma *brute force* yang digunakan secara lebih detail.

1) Membuat fungsi untuk inisialisasi data. Inisialisasi data yang digunakan untuk memudahkan proses menggunakan tipe data *dictionary* yang berisikan sebagai berikut.
    - Array huruf unik pada persoalan
    - Array kosong dengan ukuran yang sama dengan array huruf unik untuk melakukkan *assigning*
    - Array berisi huruf yang muncul diawal kata dalam bentuk index yang bereferensi kepada array unik
    - Array solusi-solusi yang berhasil
    - Array persoalan
    - Variabel waktu
    - Variabel jumlah iterasi

2) Membuat fungsi untuk mengecek kemungkinan solusi. Pada fungsi ini dilakukan pengecekan dengan mengganti huruf pada persoalan dengan angka yang

sesuai. Apabila hasilnya sesuai, maka solusi tersebut akan dimasukkan ke dalam array solusi-solusi yang berhasil.

3) Membuat fungsi untuk permutasi. Pada fungsi yang ini konsepnya sangatlah sederhana karena hanya memanfaatkan *konsep nested loop.* Dibuat sepuluh fungsi permutasi yang berbeda sesuai dengan jumlah huruf yang unik pada suatu persoalan.

4) Menjalankan *driver* program pada file utama. Program akan mengiterasikan permasalahan yang dimasukkan oleh pengguna kemudian akan melakukan inisialisasi awal dan menentukan jenis permutasi yang mana yang digunakan berdasarkan jumlah huruf yang unik.

2. *Source Code* Program

Untuk repository dari program ini dapat diakses melalui GitHub. Sementara itu berikut adalah *source code* untuk main.py.

```
import time
import loops # File fungsi iterasi

print("===== CRYPTARITHMETIC SOLVER =====")
print("This program is using a brute force algorithm, so it may take some time
depending on your computer power.")
print("You can add more than 1 (one) problem each file. Use
`./folder_name/file_name` if it's on another directory.")
print("")

#-------------- INPUT PROBLEMS --------------#
# Read File
filename = input("Enter filename (.txt only): ")
f = open(filename+".txt", "r")

# Assign to Array
i = 0; j = 0; strAll = []; strTemp = []
for x in f:
    if (x == '\n'):
        x = f.readline()
        strAll.append(strTemp); strTemp = []

    if (x[0] == '-'):
        x = f.readline()
```

```
        x = x.strip()
        strTemp.append(x)
    else:
        x = x.strip().strip('+')
        strTemp.append(x)
strAll.append(strTemp)

#-------------- ASSIGN AND TIMER --------------#
# Timer On
start = time.time()

# Asign to Unique Array
def initCryp(arr):
    # Find Unique
    temp = []; res = []
    for i in arr:
        if (not(i[0] in temp)): temp.append(i[0])
        for j in i:
           #if (not([j, 0] in res) and not(j.isspace())):
           if (not(j in res) and not(j.isspace())):
                res.append(j)

    arrFirst = []
    for i in temp:
        arrFirst.append(res.index(i))

    # Init Dict
    d = {'unique' : res, 'assign' : [0 for i in range(len(res))],
            'firstPos' : arrFirst, 'combi' : [],
            'arr' : arr, 'time' : 0, 'iter' : 0}

    return d

#-------------- SOLVER --------------#
# Permutate
for problem in strAll:
    print('--- Solving Problem ---')
    s = ' + '.join(problem[:len(problem)-1])
    s = s + ' = ' + problem[len(problem)-1]
```

```
print("Problem: ", s)

d = initCryp(problem)
length = len(d['unique'])
if (length == 1):
    loops.one_loop(d)
elif (length == 2):
    loops.two_loop(d)
elif (length == 3):
    loops.three_loop(d)
elif (length == 4):
    loops.four_loop(d)
elif (length == 5):
    loops.five_loop(d)
elif (length == 6):
    loops.six_loop(d)
elif (length == 7):
    loops.seven_loop(d)
elif (length == 8):
    loops.eight_loop(d)
elif (length == 9):
    loops.nine_loop(d)
elif (length == 10):
    loops.ten_loop(d)
else:
    print("Error.")

# Results
print("")
print('--- Results ---')
print(len(d['combi']),' solution(s) found.' )
for i in range(len(d['unique'])):
    for j in range(len(d['combi'])):
        print(d['unique'][i],"=",d['combi'][j][i]," ",end="")
    print('')

print("Iterations: ", d['iter'])

d['time'] = time.time() - start
```

```
    print("Time taken: ", d['time'])
    #print(d)


#-------------- DELAY --------------#
input('Press ENTER to exit. To try again, you need to close and reopen the
program.')
```

3. Hasil Program

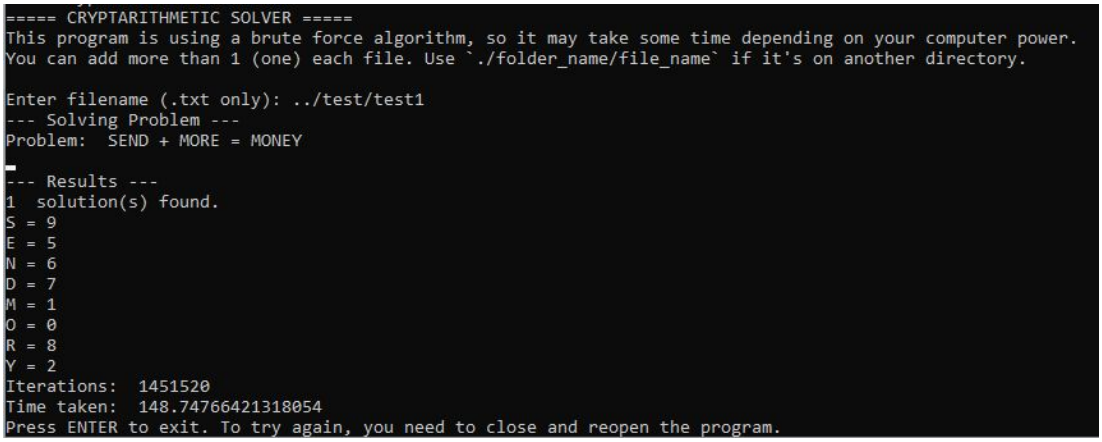
Berikut adalah beberapa *screenshot* dan persoalan yang digunakan dalam *testing* program.

1) SEND + MORE = MONEY

Masukan dari program berupa file yang berisi seperti demikian.

```
 SEND
 MORE+
------
MONEY
```

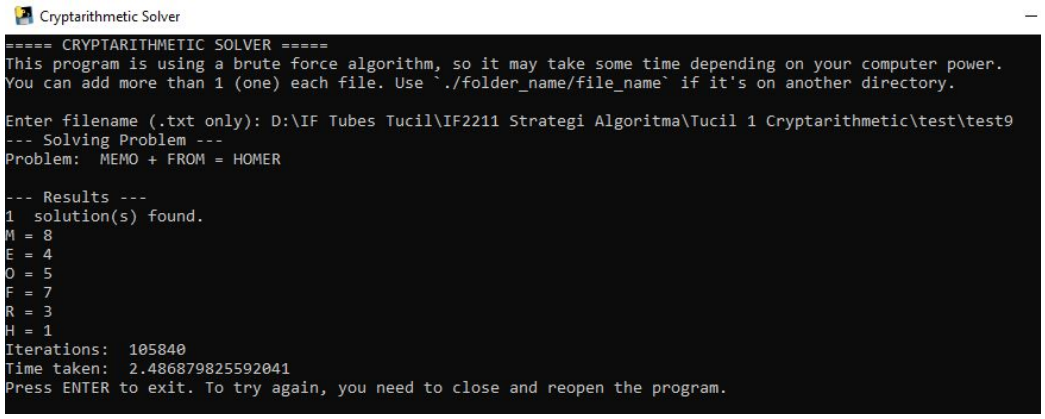
Kemudian hasil yang dikeluarkan oleh program adalah sebagai berikut.



Gambar 3.1 Hasil keluaran dari program.

2) MEMO + FROM = HOMER

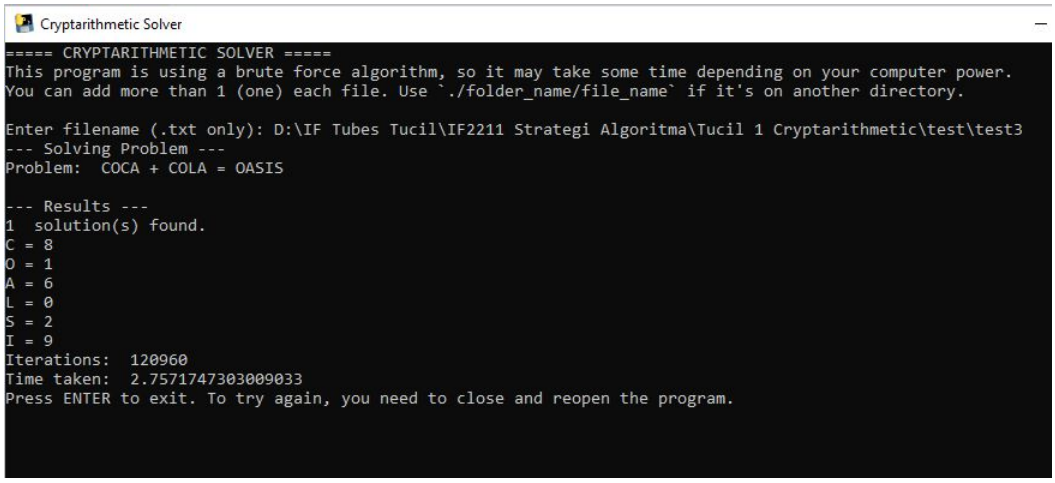
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.2 Hasil keluaran dari program.

3) COCA + COLA = OASIS

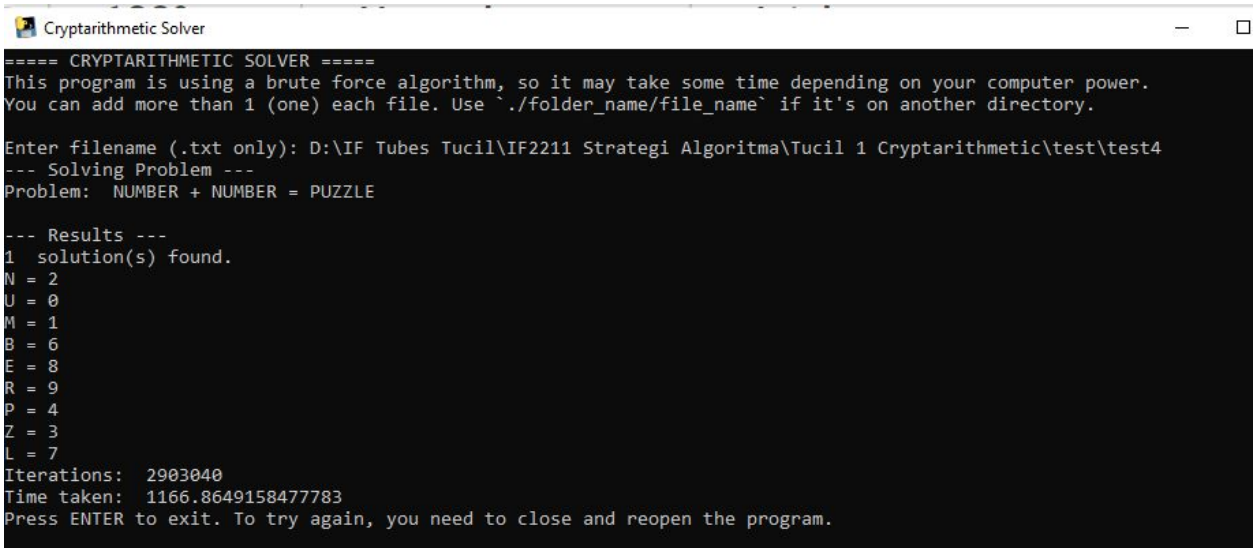
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.3 Hasil keluaran dari program.

4) NUMBER + NUMBER = PUZZLE

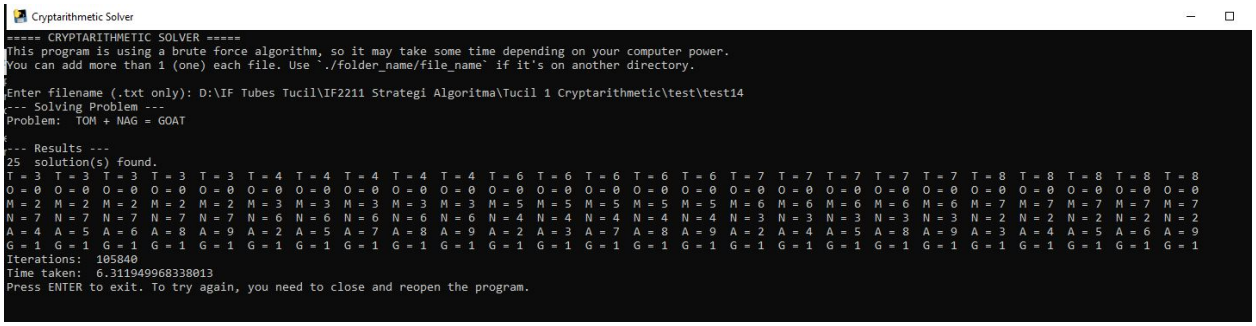
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.4 Hasil keluaran dari program.

5) TOM + NAG = GOAT

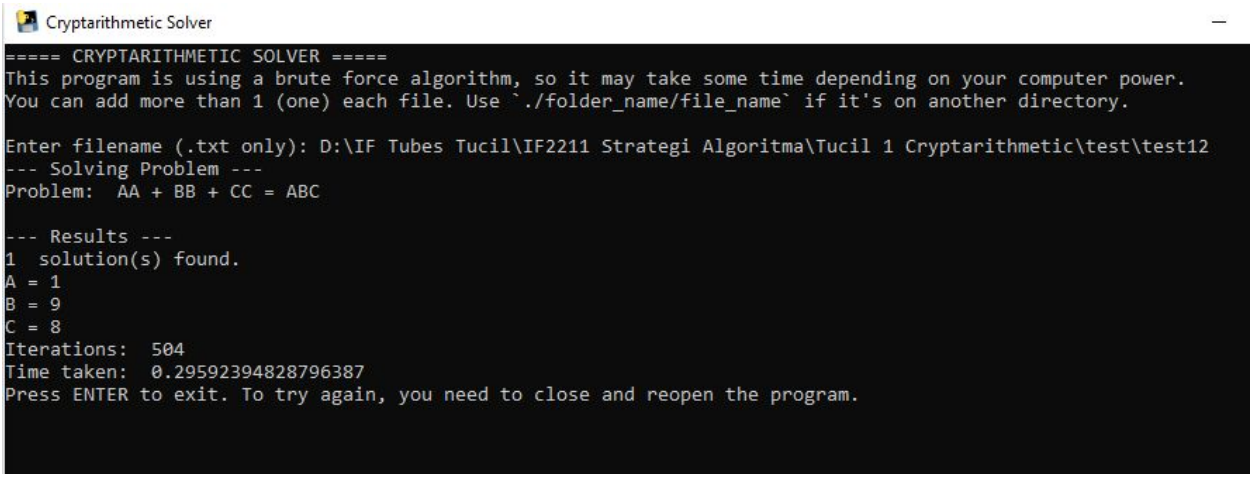
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.5 Hasil keluaran dari program.

6) AA + BB + CC = ABC

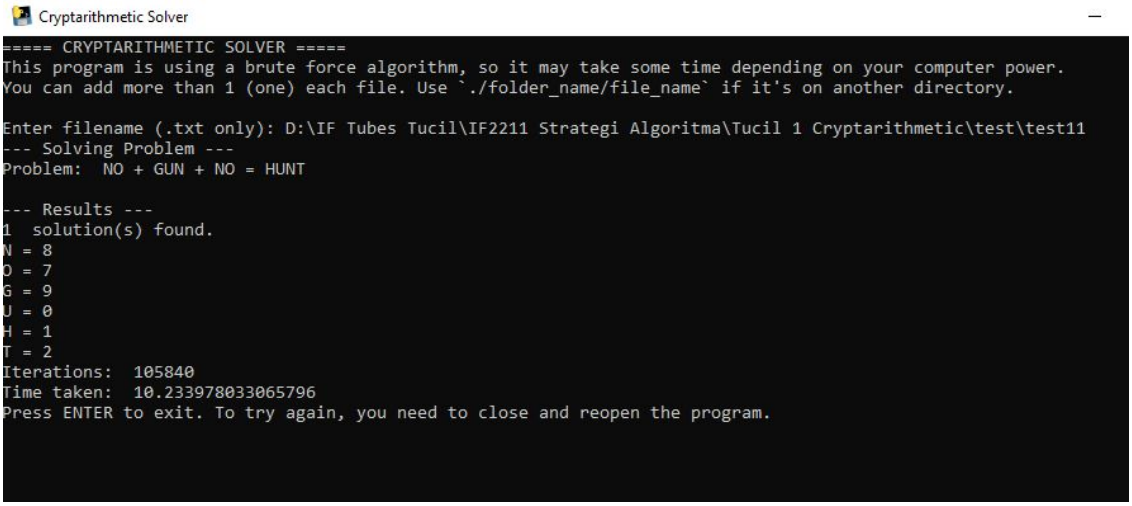
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.6 Hasil keluaran dari program.

7) NO + GUN + NO = HUNT

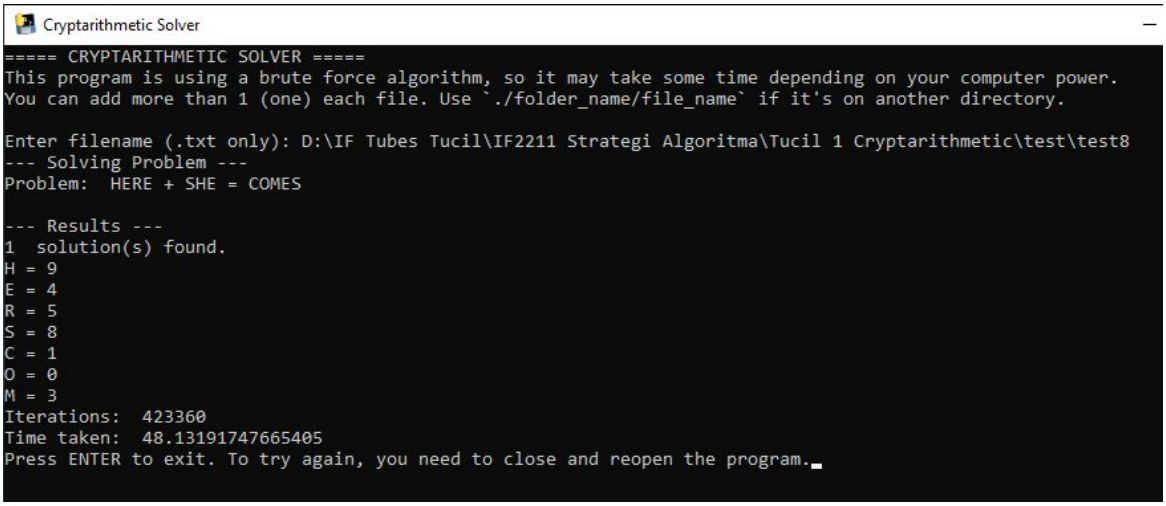
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.'



Gambar 3.7 Hasil keluaran dari program.

8) HERE + SHE = COMES

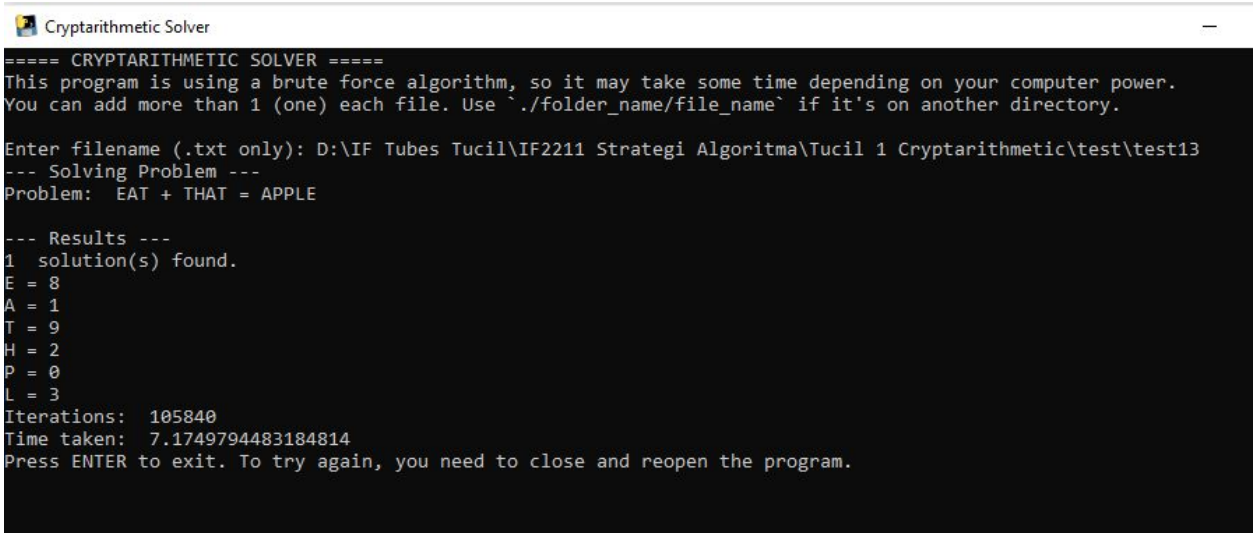
Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.8 Hasil keluaran dari program.

9) EAT + THAT = APPLE

Untuk masukkannya, memiliki format yang sama seperti uji kasus 1. Hasilnya adalah sebagai berikut.



Gambar 3.9 Hasil keluaran dari program.