Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Oleh Muhammad Rifky Muthahhari – 13519123

# ALGORITMA

Program hanya dibuat pada satu file yaitu “main.py”. Di dalamnya terdapat beberapa fungsi dan prosedur:

1. load\_data

Berfungsi untuk memuat data dari file test case

1. permutasi

Berfungsi untuk membuat seluruh permutasi yang mungkin

1. *cryptarithmetic*

Berfungsi untuk melakukan penyelesaian *cryptaritmetic* dengan algoritma *brute firce*

1. print\_greetings

Menampilkan sambutan awal

1. print\_menu

Menampilkan menu

Alur kerja program:

1. Program dimulai dengan menampilkan *greetings*. Kemudian program memasuki *loop* menu yang akan terus berulang sampai user memasukkan angka 9 (keluar)
2. Program akan membaca file test sesuai dengan angka yang diberikan user.
3. Dibuat varibel list alphabet\_sets, operand\_words, result\_word, first\_letters, dan list\_permutasi yang masing-masing berguna sesuai dengan namanya.
4. alphabet\_sets akan diisi dengan set alfabet yang muncul pada test case. Variabel ini kemudian akan diberikan pada fungsi permutasi yang akan membuat semua kemungkinan permutasi
5. list list\_permutasi akan diberikan pada fungsi cryptarithmetic.
6. Untuk setiap kemungkinan permutasi, fungsi cryptarithmetic akan membuat *dictionary* dengan *key* yang diisi set alfabet dan *value* yang diisi urutan angka pada permutasi
7. Huruf pada operand dan hasil akan diubah dengan angka sesuai dengan *dictionary* yang telah dibuat sebelumnya yang kemudian dibandingkan. Jika sama, program akan berhenti dan mengeluarkan solusi pada layar. Jika tidak, program akan kembali melanjutkan ke kemungkinan permutasi selanjutnya

# Source Program

import time

import re

alphabet\_sets = []

operand\_words = []

result\_word = []

first\_letters = []

list\_permutasi = []

# load data file

def load\_data(file\_name):

f = open(file\_name, "r")

contents = f.read()

lines = contents.splitlines()

for i in range(len(lines)):

lines[i] = lines[i].strip(" +")

for i in range(len(lines)):

if (lines[i][0] == "-"):

for j in range(i):

operand\_words.append(list(lines[j]))

for k in range(len(lines[-1])):

result\_word.append(lines[-1][k])

break

# searching first letters

for word in operand\_words:

if not word[0] in first\_letters:

first\_letters.append(word[0])

if not result\_word[0] in first\_letters:

first\_letters.append(result\_word[0])

# creating alphabet set

for word in operand\_words:

for i in range(len(word)):

if not word[i] in alphabet\_sets:

alphabet\_sets.append(word[i])

for letter in result\_word:

if not letter in alphabet\_sets:

alphabet\_sets.append(letter)

# permutation of 0-9 numbers

def permutasi(numset):

if len(numset) == 1:

return [numset]

numset\_next = permutasi(numset[1:])

x = numset[0]

perm = []

for p in numset\_next:

for i in range(len(p)+1):

perm.append(p[:i] + x + p[i:])

return perm

# checking permutation solutions

def cryptarithmetic(alphabetsets, operandwords, resultword):

time\_start = time.time()

n\_case = 0

for iterable in list\_permutasi:

current\_alphabet\_sets = alphabetsets.copy()

current\_operand\_words = operandwords.copy()

current\_result\_word = resultword.copy()

permutasi\_set = list(iterable)

# creating dictionary to swap letter to number

swap\_dict = {}

skip = False

for i in range(len(alphabet\_sets)):

# eliminate first letters that are 0

if (permutasi\_set[i] == '0' and (current\_alphabet\_sets[i] in first\_letters)):

skip = True

# first letters are not 0

else:

swap\_dict[current\_alphabet\_sets[i]] = permutasi\_set[i]

if not skip:

# swapping operand letters and result letters to number

for i in range(len(current\_operand\_words)):

current\_operand\_words[i] = [swap\_dict.get(x, x)

for x in current\_operand\_words[i]]

current\_result\_word = [swap\_dict.get(

x, x) for x in current\_result\_word]

# checking

n\_case += 1

operand\_sum = 0

result\_int = 0

for i in range(len(current\_operand\_words)):

operand\_sum += int("".join(current\_operand\_words[i]))

result\_int = int("".join(current\_result\_word))

# permutation satisfies

if (operand\_sum == result\_int):

print(" Permasalahan test case:")

print()

for word in operand\_words:

print(" {:>9s}".format("".join(word)))

print(" ---------- +")

print(" {:>10s}".format("".join(result\_word)))

print()

print(" Solusi:")

print()

for word in current\_operand\_words:

print(" {:>9s}".format("".join(word)))

print(" ---------- +")

print(" {:>10s}".format("".join(current\_result\_word)))

print()

print(" Tuple solusi -> ", swap\_dict)

print(" Case number:", n\_case)

time\_end = time.time()

print(" Durasi: {:.3f} seconds".format(

time\_end - time\_start))

break

def print\_greetings():

print(" ------------------------------------------------------------")

print(" ------------- WELCOME TO CRYPTARITMETIC SOLVER -------------")

print(" ------------------------------------------------------------")

def print\_menu():

print()

print(" Silakan pilih menu:")

print(" 1 - 8 : test case 1 - 8")

print(" 9 : keluar")

print()

# main program

print\_greetings()

active = True

while active:

allowed\_menu\_inputs = "[1-9]"

print\_menu()

while True:

# try:

user\_input = str(input(" input : ")).strip()

if not re.match(allowed\_menu\_inputs, user\_input):

print(" Pilih angka antara 1-9!")

else:

break

# except:

print(" Input tidak sesuai!")

if (user\_input != "9"):

print()

print(" Memuat data test . . .")

print()

file\_path = "../test/test" + user\_input + ".txt"

load\_data(file\_path)

for item in permutasi("0123456789"):

list\_permutasi.append(''.join(list(item)[:len(alphabet\_sets)]))

list\_permutasi = list(set(list\_permutasi))

cryptarithmetic(alphabet\_sets, operand\_words, result\_word)

else:

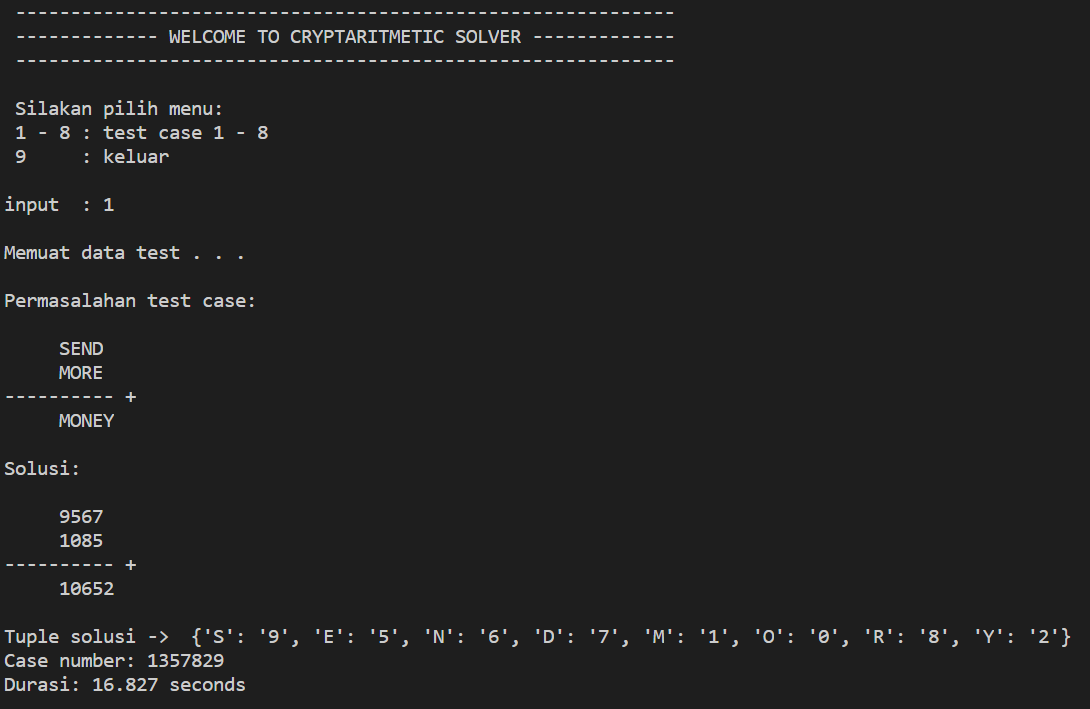
print(" Terima kasih telah menggunakan solver ini!")

active = False

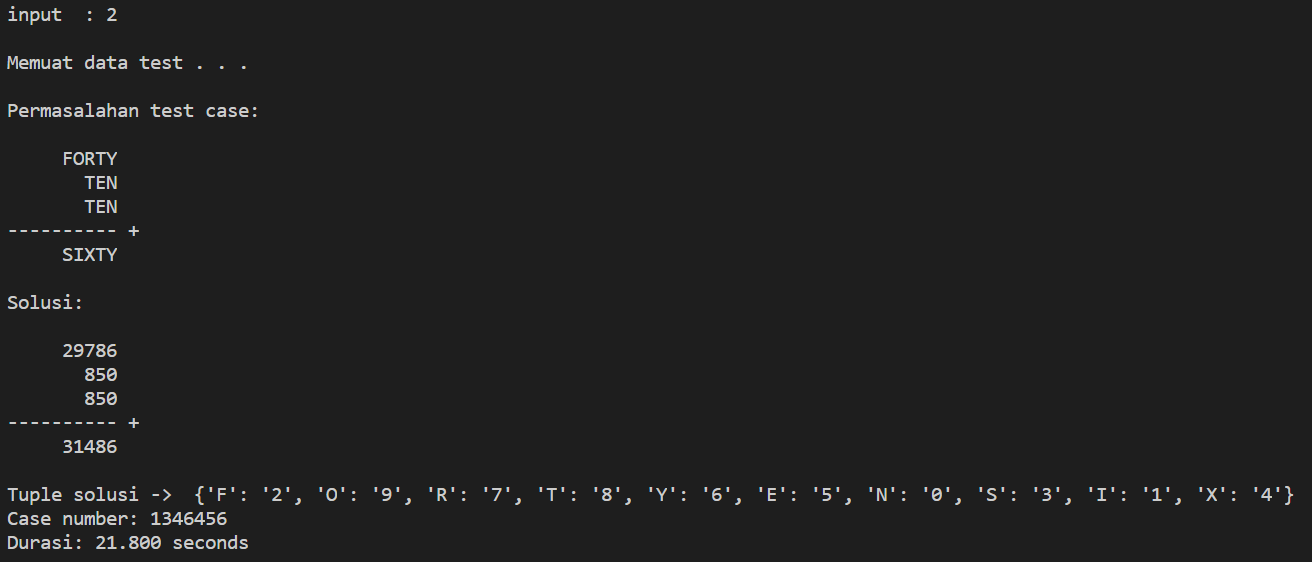
Program dibuat dalam Bahasa pemrogramam Python.

# Screenshots

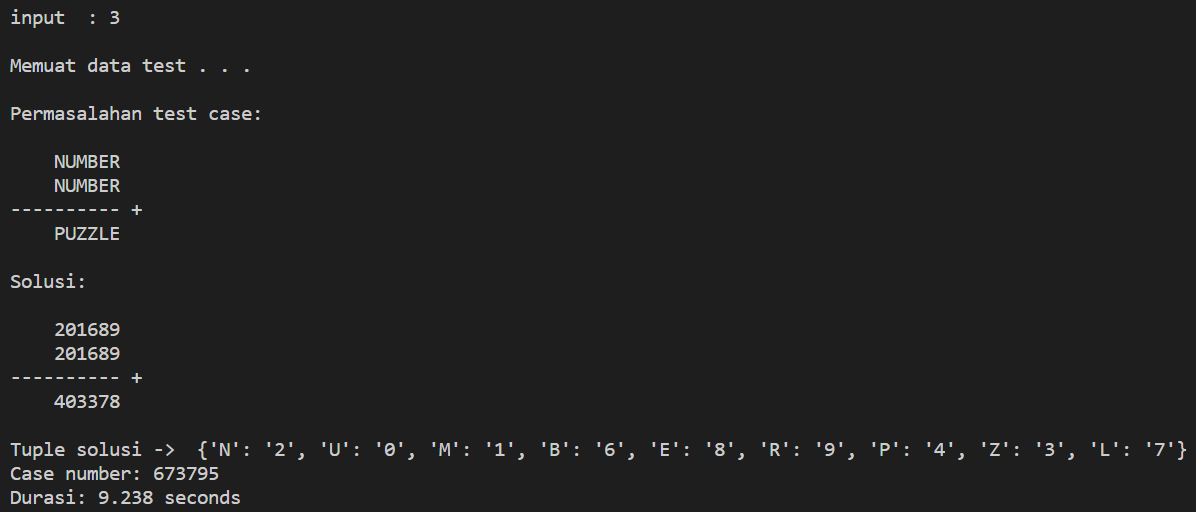
* Test case 1 + tampilan awal dan menu



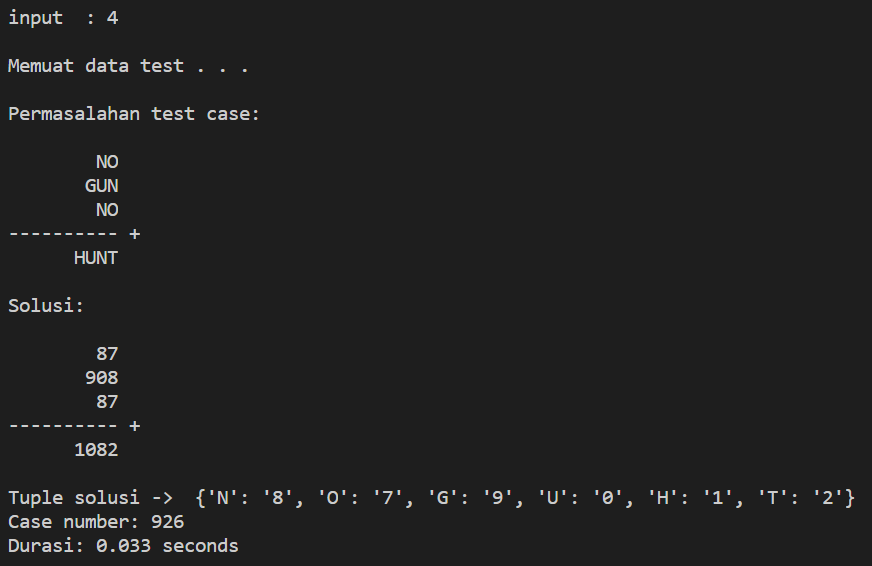
* Test case 2



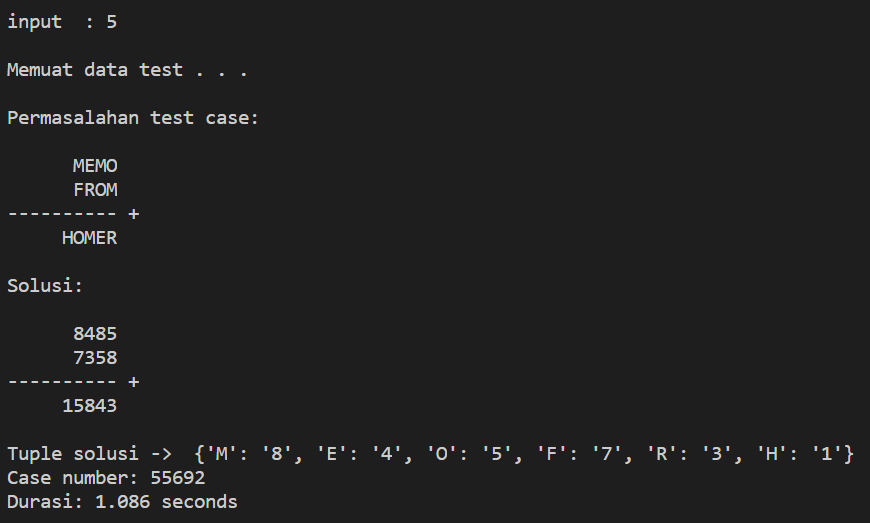
* Test case 3



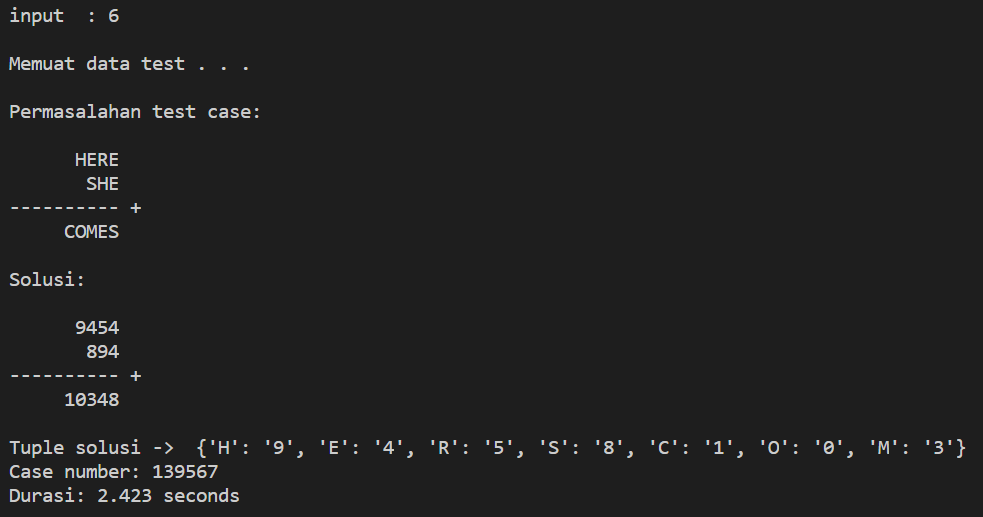
* Test case 4



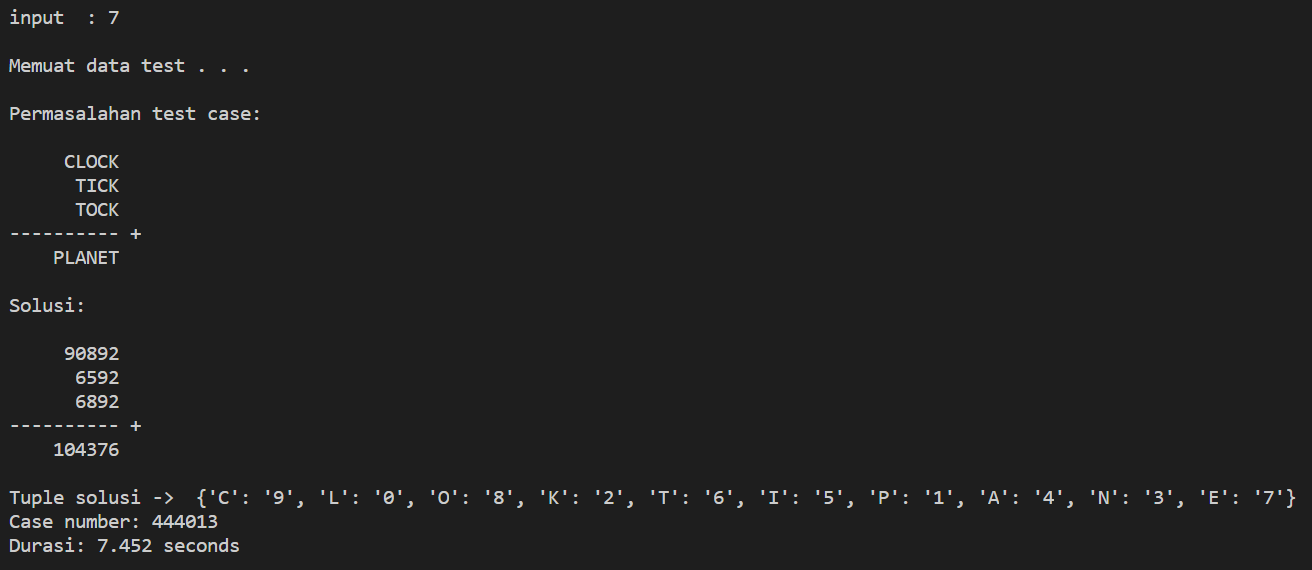
* Test case 5



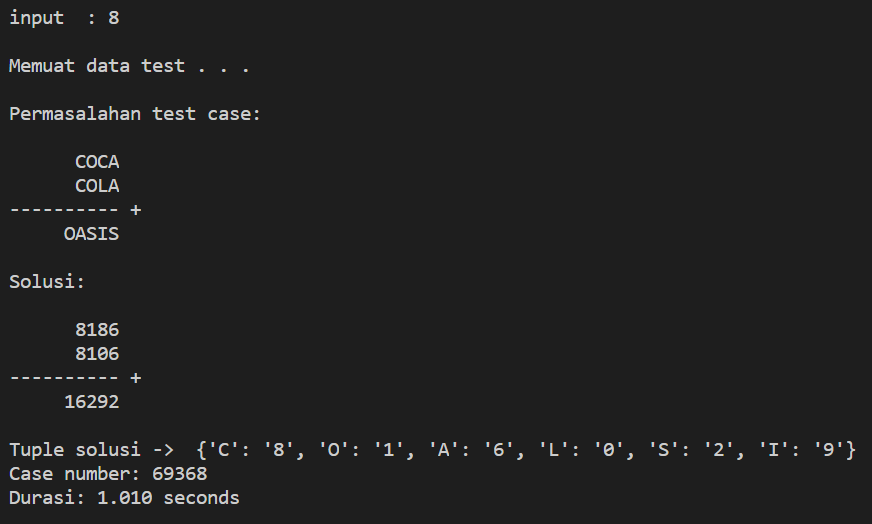
* Test case 6



* Test case 7



* Test case 8



Alamat repository: <https://github.com/rifkymuth/Tucil1>