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
2. permutasi
   Berfungsi untuk membuat seluruh permutasi yang mungkin
3. *cryptarithmetic*
   Berfungsi untuk melakukan penyelesaian *cryptaritmetic* dengan algoritma *brute firce*
4. print_greetings
   Menampilkan sambutan awal
5. 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])):
```

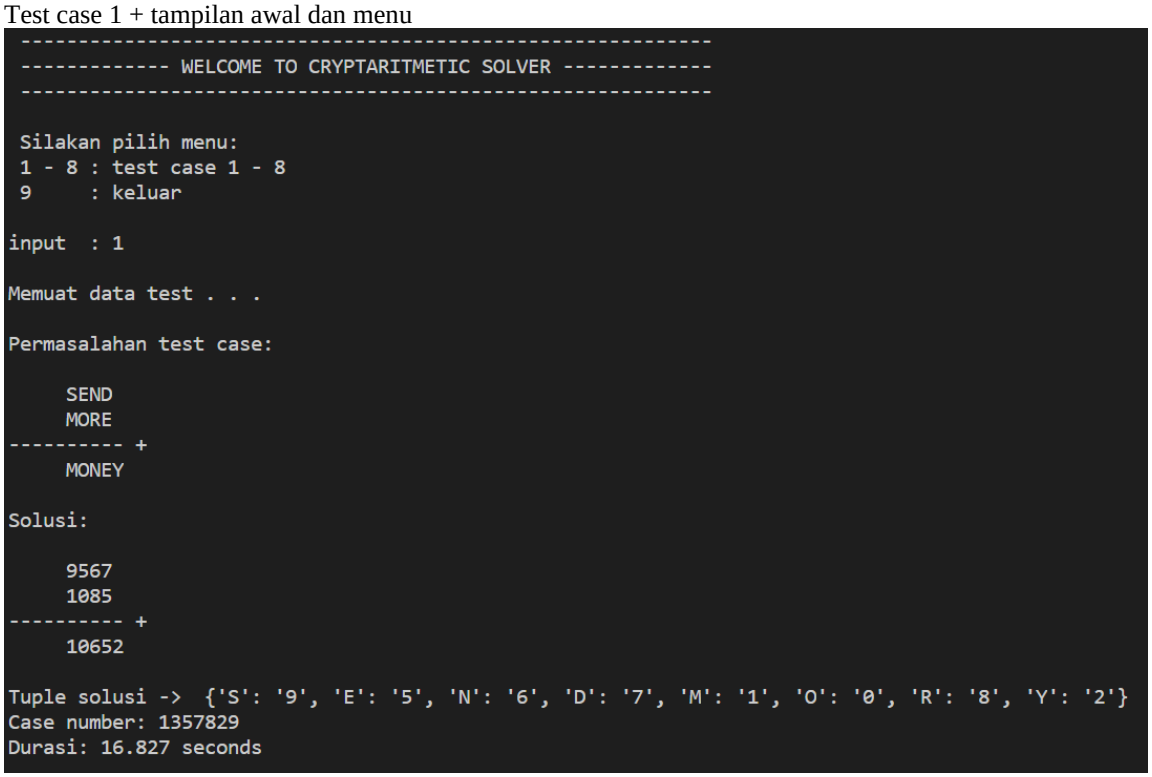
```
                result_word.append(lines[-1][k])
            break

    # 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.

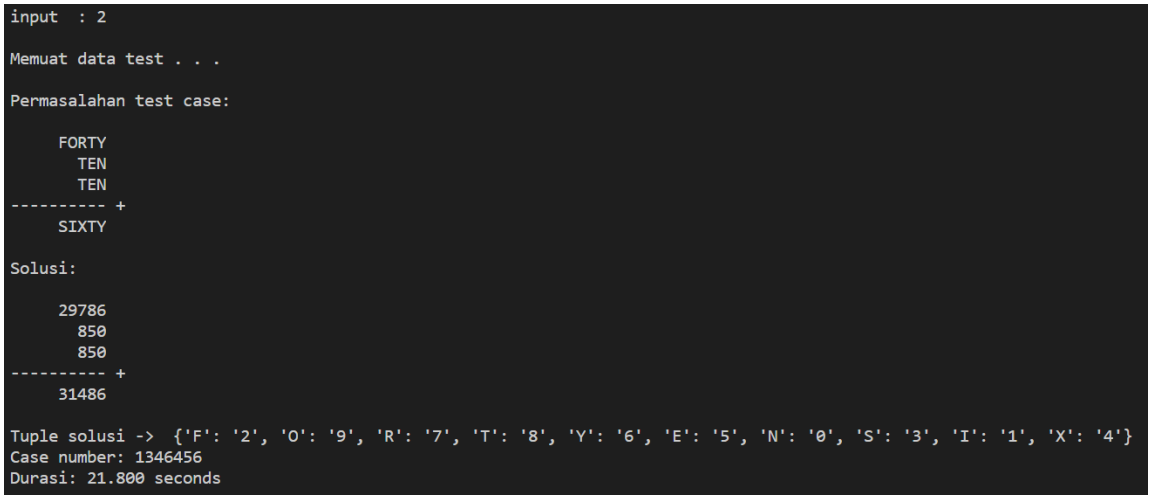
Screenshots

- Test case 1 + tampilan awal dan menu



```
 ------------------------------------------------------------
 ------------ WELCOME TO CRYPTARITMETIC SOLVER ------------
 ------------------------------------------------------------

 Silakan pilih menu:
 1 - 8 : test case 1 - 8
 9     : keluar

input  : 1

Memuat data test . . .

Permasalahan test case:

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

Solusi:

     9567
     1085
---------- +
     10652

Tuple solusi ->  {'S': '9', 'E': '5', 'N': '6', 'D': '7', 'M': '1', 'O': '0', 'R': '8', 'Y': '2'}
Case number: 1357829
Durasi: 16.827 seconds
```

- Test case 2



```
input  : 2

Memuat data test . . .

Permasalahan test case:

     FORTY
       TEN
       TEN
---------- +
     SIXTY

Solusi:

     29786
       850
       850
---------- +
     31486

Tuple solusi ->  {'F': '2', 'O': '9', 'R': '7', 'T': '8', 'Y': '6', 'E': '5', 'N': '0', 'S': '3', 'I': '1', 'X': '4'}
Case number: 1346456
Durasi: 21.800 seconds
```

- Test case 3

```
input  : 3

Memuat data test . . .

Permasalahan test case:

    NUMBER
    NUMBER
---------- +
    PUZZLE

Solusi:

    201689
    201689
---------- +
    403378

Tuple solusi ->  {'N': '2', 'U': '0', 'M': '1', 'B': '6', 'E': '8', 'R': '9', 'P': '4', 'Z': '3', 'L': '7'}
Case number: 673795
Durasi: 9.238 seconds
```

- Test case 4

```
input  : 4

Memuat data test . . .

Permasalahan test case:

        NO
       GUN
        NO
---------- +
      HUNT

Solusi:

        87
       908
        87
---------- +
      1082

Tuple solusi ->  {'N': '8', 'O': '7', 'G': '9', 'U': '0', 'H': '1', 'T': '2'}
Case number: 926
Durasi: 0.033 seconds
```

- Test case 5

```
input  : 5

Memuat data test . . .

Permasalahan test case:

      MEMO
      FROM
---------- +
     HOMER

Solusi:

      8485
      7358
---------- +
     15843

Tuple solusi ->  {'M': '8', 'E': '4', 'O': '5', 'F': '7', 'R': '3', 'H': '1'}
Case number: 55692
Durasi: 1.086 seconds
```

- Test case 6

```
input  : 6

Memuat data test . . .

Permasalahan test case:

      HERE
       SHE
---------- +
     COMES

Solusi:

      9454
       894
---------- +
     10348

Tuple solusi ->  {'H': '9', 'E': '4', 'R': '5', 'S': '8', 'C': '1', 'O': '0', 'M': '3'}
Case number: 139567
Durasi: 2.423 seconds
```

- Test case 7

```
input  : 7

Memuat data test . . .

Permasalahan test case:

     CLOCK
      TICK
      TOCK
---------- +
    PLANET

Solusi:

     90892
      6592
      6892
---------- +
    104376

Tuple solusi ->  {'C': '9', 'L': '0', 'O': '8', 'K': '2', 'T': '6', 'I': '5', 'P': '1', 'A': '4', 'N': '3', 'E': '7'}
Case number: 444013
Durasi: 7.452 seconds
```

- Test case 8

```
input  : 8

Memuat data test . . .

Permasalahan test case:

      COCA
      COLA
---------- +
     OASIS

Solusi:

      8186
      8106
---------- +
     16292

Tuple solusi ->  {'C': '8', 'O': '1', 'A': '6', 'L': '0', 'S': '2', 'I': '9'}
Case number: 69368
Durasi: 1.010 seconds
```

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

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan (no syntax error)	V	
Program berhasil running	V	
Program dapat membaca file masukan dan menuliskan luaran	V	
Solusi cryptarithmetic hanya benar untuk persoalan cryptarihtmetic dengan dua buah operand	V	
Solusi cryptarithmetic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand.	V	