LAPORAN TUGAS KECIL

PENYELESAIAN *CRYPTARITHMETHIC* DENGAN ALGORITMA *BRUTE FORCE*

MENGGUNAKAN BAHASA PEMROGRAMAN *PYTHON 3*

DISUSUN OLEH

FARHAN NUR HIDAYAT DENIRA

NIM. 13519071

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

Program Penyelesaian *Cryptarithmethic* dengan Algoritma *Brute Force* yang telah saya buat ini menggunakan bahasa pemrograman Python 3.

1. Deskripsi Algoritma *Brute Force*

    Hal yang pertama dilakukan oleh program adalah melakukan *load* dari *file* dengan ekstensi *txt* menjadi dua atau tiga buah operan serta sebuah jawaban dimana jumlah operan bergantung pada soal di dalam *file text*, setelah itu operan-operan dan jawaban akan dimasukkan ke variabel *string* '*op1'*, '*op2'*, '*op3'*, dan '*jaw'*.

    Setelah melakukan *load*, program akan membuat sebuah *array of character* bernama '*huruf*" yang berisi huruf-huruf dari *string op1*, *op2*, *op3*, dan *jaw*. *Array* ini bersifat unik, yaitu tidak ada huruf yang sama di dalamnya. Setelahnya, program akan membuat sebuah objek bernama '*perm'* yang berisi list semua permutasi yang mungkin dari angka 0-9 dengan panjang *list* sesuai panjang *array* '*huruf*". Misalkan *array* '*huruf'* memiliki panjang 7, maka objek '*perm'* merupakan semua permutasi angka 0 sampai 9 sebanyak 7 angka, salah satu contohnya adalah *list* [0,1,4,3,6,8,7].

    Setelah itu, program akan melakukan inisialisasi variabel '*jumlahtes'* dengan nilai 0, yang akan selalu bertambah 1 setiap kali program melakukan *looping* dalam mencari permutasi yang tepat. Program juga menginisialisasi variabel '*awal'* yang merupakan waktu awal, dimana variabel ini digunakan untuk mencari waktu eksekusi program. Program juga menginisialisasi *boolean* '*valid'*, yang akan bernilai *True* jika tidak ada array dengan huruf awalan bernilai 0, seperti yang tertulis pada spesifikasi tugas.

    Kemudian, program akan masuk *loop*, yang mana akan selalu terulang sampai semua permutasi sudah dipakai. Saya memilih untuk tidak menghentikan program setelah menemukan sebuah permutasi yang tepat dikarenakan sebuah *cryptarithmethic* bisa saja memiliki lebih dari satu permutasi yang memenuhi. Pada *loop*, permutasi saat itu akan dimasukkan ke variabel '*lst'*.

    Untuk mencari permutasi yang tepat, pertama program akan merubah *string op1*, *op2*, *op3*, dan *jaw* menjadi *array of integer*, dimana *integer* di setiap indeks disesuaikan dengan permutasi pada *list* '*lst'*. Fungsi yang digunakan dalam melakukan hal ini adalah fungsi '*huruf_to_arrnum'*. Sebagai contoh, apabila permutasi saat itu adalah (1,2,3,4,5), *list* '*huruf*" adalah (a,b,c,d,e), *string* '*op1'* adalah "*acd*", maka akan dibuat array of integer '*arr1'* berisi (1,3,4). Hal yang sama juga dilakukan kepada operan lainnya.

    Array of integer yang tadi telah didapat akan diubah menjadi *integer* murni oleh program menggunakan fungsi '*arrnum_to_num'*. Misalkan *arr1* bernilai [1,3,6] akan diubah menjadi *integer num1* dengan nilai 136. Setelahnya, program akan mengecek apakah semua *array* tadi tidak ada yang elemen pertamanya bernilai 0, yang apabila terpenuhi maka *boolean* '*valid'* akan bernilai *True* dan program akan berlanjut, jika tidak maka akan melanjutkan *looping* ke permutasi setelahnya.

    Langkah terakhir, apabila nilai dari jumlah semua variabel *num* operan sama dengan nilai variabel *numj* yaitu *integer* dari jawaban, maka program akan memberi keluaran persamaan penjumlahan menggunakan fungsi *print_array*, waktu yang dibutuhkan yaitu nilai dari variable *akhir* dikurang variabel *awal*, serta jumlah tes yang dilakukan. Apabila semua permutasi sudah dicoba, maka program akan berhenti.

    Jadi secara garis besar, pertama program membaca file, lalu membuat array berisi huruf unik dari semua string yang dibaca, mencari permutasi, menyamakan nilai array huruf dan permutasi sesuai indeks, mengecek kebenaran permutasi dengan cara merubah operan-operan

menjadi integer dan menambahkan integer operan-operan tersebut. Program akan berhenti setelah semua permutasi dicoba.

2. Source Code Program
   - Read dari text file

```
from itertools import permutations
from time import time
```

```
# Read from text file
def read():
    global op1, op2, op3, jaw
    file1 = open('soal.txt', 'r')
    Lines = file1.readlines()
    count = len(Lines)

    op1 = Lines[0].strip()
    op2 = Lines[1].strip()
    if count == 4:
        op3 = ''
        op2 = op2.replace('+', '')
    else:  # 5 baris
        op3 = Lines[2].strip().replace('+', '')
    jaw = Lines[count - 1].strip()
```

   - Merubah string menjadi integer

```
# string jadi arr of int, misal 'ab' jadi [1,2]
def huruf_to_arrnum(op,huruf_unik,solusi):
    arr = [-1 for i in range(len(op))]
    for i in range(len(huruf_unik)):
        for j in range(len(op)):
            if op[j] == huruf_unik[i]:
                arr[j] = solusi[i]
    return arr

# ubah array nomor jadi angka, misal [1,2] jadi 12
def arrnum_to_num(arr):
    if arr == []:
        return 0
    num = 0
    for i in range(len(arr)):
        pangkat = (len(arr) - (i+1))
        num += arr[i] * (10**pangkat)
    return num
```

   - Menampilkan keluaran dalam bentuk persamaan

```
def printbaris(r1,jarak):
    print(" " * 2 * (jarak - len(r1)), end='')
    for i in r1:
        print(i,end=' ')
```

```
def print_array(r1,r2,r3,jaw1,s1,s2,s3,jaw2):
    jarak = max(len(r1),len(r2),len(r3),len(jaw1))

    printbaris(r1,jarak)
    print('     ',end=' ')
    printbaris(s1,jarak)
    print()

    printbaris(r2, jarak)
    print('     ', end=' ')
    printbaris(s2, jarak)
    print()

    if r3 != []:
        printbaris(r3, jarak)
        print('     ', end=' ')
        printbaris(s3, jarak)
        print()

    print("+ "+"- "*(jarak-1),end='      ')
    print("+ "+"- "*(jarak-1))
    printbaris(jaw1, jarak)
    print('      ', end='')
    printbaris(jaw2, jarak)
    print()
```

- Program Utama

```
def main():                                                    ⚠ 58  ✓ 58

    read()
    # membuat array unik berisi huruf-huruf dari semua operator dan jawaban
    huruf = []
    for letter in (op1+op2+op3+jaw):
        if not letter in huruf:
            huruf += [letter]

    # membuat permutasi angka 0-9 dengan range panjang array huruf
    perm = permutations([0,1,2,3,4,5,6,7,8,9],len(huruf))

    # Menginisialisasi jumlahtes dan waktu awal
    jumlahtes = 0
    valid = False
    awal = time()
    # mengecek satu-persatu list permutasi
    for lst in list(perm):
        jumlahtes += 1

        # ubah jadi array int
        arr1 = huruf_to_arrnum(op1,huruf,lst)
        arr2 = huruf_to_arrnum(op2,huruf,lst)
        arr3 = huruf_to_arrnum(op3, huruf, lst)
        arrj = huruf_to_arrnum(jaw,huruf,lst)
```

```
    # huruf pertama operator tidak boleh bernilai 0
    if arr1[0] != 0 and arr2[0] != 0 and arrj[0] != 0:
        # ubah arr jadi int
        num1 = arrnum_to_num(arr1)
        num2 = arrnum_to_num(arr2)
        numj = arrnum_to_num(arrj)
        if op3 == '':
            num3 = 0
            valid = True
        else:
            num3 = arrnum_to_num(arr3)
            if arr3[0] != 0:
                valid = True

        if valid and num1 + num2 + num3 == numj:
            print_array(arr1,arr2,arr3,arrj,op1,op2,op3,jaw)
            print()
            akhir = time()
            print("\nJumlah tes :", jumlahtes)
            print("Waktu :", (akhir - awal), "detik")
            print()

print("Finished")
```

3. *Screenshot input dan output*
    - contoh 1

```
2 0 1 6 8 9       n u m b e r
2 0 1 6 8 9       n u m b e r
+ - - - - -       + - - - - -
4 0 3 3 7 8       p u z z l e


Jumlah tes : 728504
Waktu : 35.58997654914856 detik


Process finished with exit code -1
```

- contoh 2

```
    9 1 5 4 2          t i l e s
3 0 7 7 5 4 2      p u z z l e s
+ - - - - - -      + - - - - - -
3 1 6 9 0 8 4      p i c t u r e


Jumlah tes : 3328707
Waktu : 203.2240104675293 detik
```

- contoh 3

```
  9 0 8 9 2          c l o c k
    6 5 9 2            t i c k
    6 8 9 2            t o c k
+ - - - - -        + - - - - -
1 0 4 3 7 6        p l a n e t


Jumlah tes : 3302475
Waktu : 197.59867429733276 detik
```

- contoh 4

```
2 9 7 8 6        f o r t y
    8 5 0            t e n
    8 5 0            t e n
+ - - - -        + - - - -
3 1 4 8 6        s i x t y


Jumlah tes : 1083579
Waktu : 56.67174553871155 detik
```

- contoh 5

```
   8 7          n o
 9 0 8        g u n
   8 7          n o
+ - - -      + - - -
1 0 8 2      h u n t


Jumlah tes : 134191
Waktu : 4.388931512832642 detik
```

- contoh 6

```
  8 4 8 5        m e m o
  7 3 5 8        f r o m
+ - - - -      + - - - -
1 5 8 4 3      h o m e r


Jumlah tes : 128687
Waktu : 4.693318843841553 detik
```

- contoh 7

```
  9 4 5 4        h e r e
    8 9 4          s h e
+ - - - -      + - - - -
1 0 3 4 8      c o m e s


Jumlah tes : 575302
Waktu : 22.498474597930908 detik
```

- contoh 8

```
  8 1 8 6        c o c a
  8 1 0 6        c o l a
+ - - - -      + - - - -
1 6 2 9 2      o a s i s


Jumlah tes : 123695
Waktu : 4.479819059371948 detik
```

- contoh 9

```
  9 6 2 3 3          c r o s s
  6 2 5 1 3          r o a d s
+ - - - - -        + - - - - -
1 5 8 7 4 6        d a n g e r


Jumlah tes : 3519768
Waktu : 168.58429169654846 detik
```

- contoh 10

```
  7 9 8 0 6 4          d o u b l e
  7 9 8 0 6 4          d o u b l e
      1 9 3 6              t o i l
+ - - - - - -        + - - - - - -
1 5 9 8 0 6 4        t r o u b l e


Jumlah tes : 2898676
Waktu : 190.32725477218628 detik
```

4. Link Source code
   Laporan serta source code dapat dilihat pada link berikut :
   Google Drive :
   https://drive.google.com/drive/folders/1FIq9VQNcOpZA8oiLxRVZmeplqmOLqQLX?usp=sharing
   Github :
   https://github.com/ahandnr/CryptarithmeticSolver

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

Sekian dan Terima Kasih.