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, serta program tidak memiliki batasan jumlah operan.

1. Deskripsi Algoritma *Brute Force*

    Hal yang pertama dilakukan oleh program adalah melakukan *load* dari *file* bernama '*soal.txt'* ke dalam *array 'Op',* sehingga *array* ini berisi string operan dan jawaban. Setelahnya, program membuat *array 'huruf"* yang berisi karakter dari string-string di *array 'Op'*. *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 melakukan *brute force*. 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 semua *string* pada '*Op'* menjadi *array of integer*, dimana *integer* di setiap indeks disesuaikan dengan permutasi pada *list* '*lst'*, lalu dimasukkan ke *array 'numOp'*. Fungsi yang digunakan dalam melakukan hal ini adalah fungsi '*string_to_arrnum'*. Sebagai contoh, apabila permutasi saat itu adalah (1,2,3,4,5), *list* '*huruf"* adalah (a,b,c,d,e), *string* pertama dari *array 'Op'* adalah "*acd*", maka akan dibuat array of integer '*arrnum'* berisi (1,3,4).

    Array of integer yang tadi telah didapat akan diubah menjadi *integer* murni oleh program menggunakan fungsi '*arrnum_to_num'*. Misalkan *arrnum* bernilai [1,3,6] akan diubah menjadi *integer num* 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 *looping* akan berlanjut ke permutasi selanjutnya. Program juga menambahkan nilai *integer 'sum'* dengan *'num'.*

    Langkah terakhir, apabila nilai dari *integer 'sum'* sama dengan dua kali nilai *integer* '*numj'* yaitu nilai 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. Waktu yang tercatat pada keluaran tidak termasuk waktu mencari semua permutasi dan hanya merupakan waktu pencarian permutasi yang tepat menggunakan metode *brute force.* Apabila semua permutasi sudah dicoba, maka program akan

berhenti. *'sum'* akan bernilai dua kali dari *'numj'* karena pada *loop* sebelumnya, *array 'Op'* mengandung *string* jawaban sehingga nilai dari *string* jawaban juga ditambahkan ke *'sum'*.

Jadi secara garis besar, pertama program membaca file, lalu membuat array berisi huruf unik dari semua string yang dibaca, mencari semua 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, karena ada kemungkinan jawaban lebih dari satu.

2. Source Code Program
   - Permutasi

```python
# PERMUTASI
def permutation(lst):
    if len(lst) == 0:
        perm = []
    elif len(lst) == 1:
        perm = [lst]
    else:
        perm = []
        for i in range(len(lst)):
            m = lst[i]
            tail = lst[:i] + lst[i + 1:]
            for p in permutation(tail):
                perm.append([m] + p)
    return perm


# PERMUTASI DENGAN RANGE
def permrange(lst, r):
    take = len(lst) - r
    # cari faktorial
    fak = 1
    for i in range(1, take + 1):
        fak = fak * i
    # permutasi
    perm = permutation(lst)
    l = []
    for i in range(0, len(perm), fak):
        perm[i] = perm[i][:-take]
        l.append(perm[i])
    return l
```

- Read dari text file

```
# READ FROM FILE
def read():
    global Op
    file1 = open('soal.txt', 'r')
    Op = file1.readlines()

    # hapus \n
    for i in range(len(Op)):
        Op[i] = Op[i].strip()
    # hapus elemen ----- di array
    Op.pop(len(Op) - 2)
    Op[len(Op) - 2] = Op[len(Op) - 2].replace('+', '')
```

- Merubah string menjadi integer

```
# string jadi arr of int, misal 'ab' jadi [1,2]
def string_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

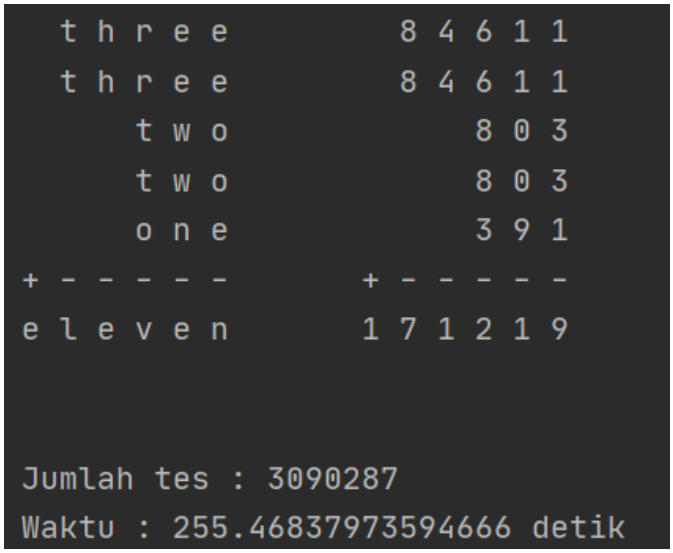
```python
# PRINT PERSAMAAN
def print_persamaan(ophuruf, opangka):
    jarak = len(max(ophuruf, key=len))

    # print operan
    for i in range(len(ophuruf) - 1):
        # print spasi
        print(" " * 2 * (jarak - len(ophuruf[i])), end='')
        # print per huruf
        for j in ophuruf[i]:
            print(j, end=' ')
        print('     ', end=' ')

        # print spasi
        print(" " * 2 * (jarak - len(ophuruf[i])), end='')
        # print per angka
        for j in opangka[i]:
            print(j, end=' ')
        print()

    # print garis pembatas
    print("+ " + "- " * (jarak - 1), end='      ')
    print("+ " + "- " * (jarak - 1))

    # print jawaban
    # print spasi
    print(" " * 2 * (jarak - len(ophuruf[-1])), end='')
    # print per huruf
    for j in ophuruf[-1]:
        print(j, end=' ')
    print('     ', end=' ')

    # print spasi
    print(" " * 2 * (jarak - len(ophuruf[-1])), end='')
    # print per angka
    for j in opangka[-1]:
        print(j, end=' ')
    print()
```

- Program Utama

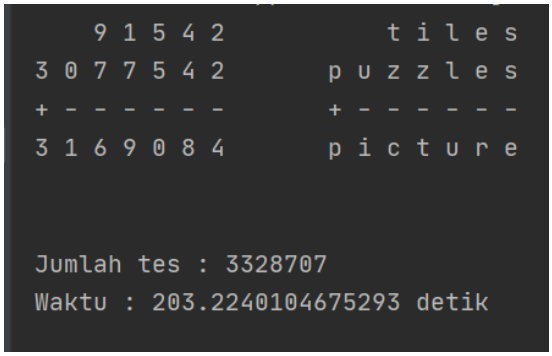
```
def main():
    read()  # load file ke array Op

    # membuat array unik berisi huruf-huruf dari semua operator dan jawaban
    huruf = []
    for string in Op:
        for letter in string:
            if not letter in huruf:
                huruf.append(letter)

    # membuat permutasi angka 0-9 dengan range panjang array huruf

    perm = permrange([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], len(huruf))

    # Menginisialisasi jumlahtes dan waktu awal
    jumlahtes = 0
    awal = time()

    # mengecek satu-persatu list permutasi
    for lst in list(perm):
        valid = True
        jumlahtes += 1
        sum = 0
        numOp = []
        # ubah jadi array int
        for string in Op:
            arrnum = string_to_arrnum(string, huruf, lst)
            numOp.append(arrnum)
            # huruf pertama operator tidak boleh bernilai 0
            if arrnum[0] != 0:
                num = arrnum_to_num(arrnum)
                sum += num
            else:
                valid = False
                break
```

```
        if valid:
            # mengecek num dari jawaban
            arrj = string_to_arrnum(Op[-1], huruf, lst)
            numj = arrnum_to_num(arrj)

            # mengecek apakah sesuai soal
            if sum == 2 * numj:
                print_persamaan(Op, numOp)
                print()
                akhir = time()
                print("\nJumlah tes :", jumlahtes)
                print("Waktu :", (akhir - awal), "detik")
                print()

    print("Finished")


main()
```

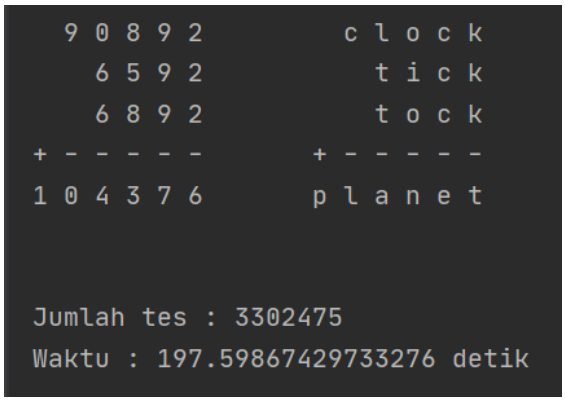
3. *Screenshot input dan output*
    - contoh 1



    - contoh 2



    - contoh 3



- 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
```

- contoh 11

```
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
```

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.