

**TUGAS KECIL 1**  
**IF2211 STRATEGI ALGORITMA**  
**Penyelesaian *Cryptarithmic* dengan Algoritma *Brute Force***  
**SEMESTER 2 TAHUN 2020/2021**

oleh

Irvin Andryan Pratomo / 13519162



**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**BANDUNG**  
**2021**

## I. Algoritma *Brute Force*

Algoritma *brute force* yang digunakan pada program ini pada dasarnya adalah dengan melakukan *iterate* angka-angka pada *range* tertentu. *Range* angka tersebut ditentukan berdasarkan jumlah huruf unik yang diperoleh dari kata pada operan dan hasil.

Langkah-langkah kerja :

1. Program akan membaca input dari text file yang berisi operan dan hasil, kemudian menyimpan isi dari text file tersebut ke dalam sebuah list.
2. Program “membersihkan” karakter yang tidak perlu seperti ‘+’ dan ‘-----’, kemudian menyimpan hasilnya ke dalam sebuah listOfKata, yaitu list berisi kata-kata yang terdapat di dalam text file input.
3. Dari listOfKata, program kemudian mengambil huruf-huruf yang terdapat di dalam kata-kata tersebut dan menyimpannya ke dalam list baru bernama hurufUnik. Selain itu program juga mencatat huruf pertama yang digunakan pada kata dan menyimpannya di list baru bernama hurufAwal.
4. Program melakukan looping yang dimulai dari sebuah angka yang ditentukan berdasarkan banyak huruf yang digunakan. Sebuah variabel stringOfDigit menyimpan string ‘1234567890’, Ketika ada 6 huruf yang digunakan, loop akan dimulai dari angka 123456 hal ini untuk mengurangi waktu yang tidak perlu jika looping dilakukan dari angka nol. Batas akhir loop adalah  $10^{(\text{banyak huruf unik yang digunakan})}$ .
5. Ketika program looping dan sedang berada pada indeks dengan digit yang tidak berulang (misal 273098), maka digit-digit tersebut akan dimasukkan ke dalam list bernama angkaUnik [2, 7, 3, 0, 9, 8]. Kemudian huruf yang ada di list hurufUnik akan di-assign sesuai dengan angka di dalam list angkaUnik yang indeksinya sama. Misal hurufUnik = [H, R, E, U, P, S], maka huruf H = 2, R = 7, E = 3, U = 0, P = 9, S = 8. Namun hal ini akan batal jika ada angka nol dan huruf yang di-assign dengan angka nol adalah anggota dari list hurufAwal (karena leading 0 tidak boleh). Jika batal, maka program akan melanjutkan looping dan mengulangi langkah nomor 5 ini. Angka nol tidak akan mungkin untuk di assign pada huruf di indeks pertama hurufUnik, karena huruf itu sudah pasti adalah huruf awal dari kata di operan pertama (hal ini juga menjadi alasan loop bisa dilakukan tidak dari nol).
6. Ketika berhasil dengan tidak ada huruf pertama yang di assign dengan angka nol, program kemudian menggunakan listOfKata dan angka-angka tersebut akan disusun

berdasarkan kemunculan huruf yang bersesuaian, misal kata PURE akan membentuk 9073.

7. Program kemudian menjumlahkan angka-angka tersebut, jika hasil operan + operan + ... + operan (n) = hasil, maka akan menuliskannya pada layar, jika gagal, maka kembali ke langkah nomor 5.

## II. Source Code

Program dibuat menggunakan bahasa Python.

```
import time
import os

#-----membuka file input dan memasukkan isinya ke dalam listOfContain
os.chdir("../")
inputFile = open("test/textinput1.txt", "r")
isiFile = inputFile.read()
listOfContain = isiFile.split("\n")
inputFile.close()

#-----membentuk list yang berisi operand
listOperand = [''] * (len(listOfContain) - 2)
for i in range(len(listOfContain) - 2):
    listOperand[i] = listOfContain[i]
listOperand = [s.replace('+', '') for s in listOperand] #menghapus tanda +
listOperand = [s.strip(' ') for s in listOperand] #menghapus space kosong

#-----menyimpan hasil dalam variabel "hasil"
hasil = listOfContain[len(listOfContain) - 1]

def solver (operand, hasil):

    startTime = time.perf_counter()
    #-----membentuk listOfKata berisi hanya operand dan hasil tanpa tanda '+'
    dan '-----'
    listOfKata = [0 for i in range(len(listOperand) + 1)]
    for i in range(len(listOperand)):
        listOfKata[i] = listOperand[i]
    listOfKata[len(listOfKata) - 1] = hasil

    #-----membentuk list hurufUnik berisi huruf-huruf unik yang ada
    hurufUnik = []
    for i in range(len(listOfKata)):
        for j in range(len(listOfKata[i])):
            if (listOfKata[i][j] not in hurufUnik):
                hurufUnik.append(listOfKata[i][j])

    #-----membentuk list berisi huruf-huruf awal kata
```

```

hurufAwal = []
for i in range(len(listOfKata)):
    if(listOfKata[i][0] not in hurufAwal):
        hurufAwal.append(listOfKata[i][0])

counter = 0 #menghitung banyak percobaan

stringOfDigit = '1234567890'
initialNumber = ''
for dig in range(len(hurufUnik)):
    initialNumber += stringOfDigit[dig]
i_unik = int(initialNumber)

while(i_unik < 10**(len(hurufUnik))):
    angkaUnik = [] #list untuk menyimpan angka yang digunakan
    num = str(i_unik)
    j_unik = 0
    masihbenar = True
    while(j_unik < len(num) and masihbenar == True):
        if (hurufUnik[j_unik] in hurufAwal and num[j_unik] == '0'):
            #print(num[j_unik])
            masihbenar = False #ketika angka di index n pada list angkaUnik
= 0 dan huruf di index n pada list hurufUnik merupakan huruf pertama suatu kata
        else:
            if(num[j_unik] not in angkaUnik):
                angkaUnik.append(num[j_unik])
                j_unik += 1
            else:
                masihbenar = False

    if (masihbenar == True):
        #menggabungkan angka-angka dari list angkaUnik ke dalam listOfString
angka sesuai urutan kemunculan huruf pada suatu kata
        listOfStringAngka = ['' for i in range(len(listOfKata))]
        for i in range(len(listOfKata)):
            for j in range(len(listOfKata[i])):
                idx = hurufUnik.index(listOfKata[i][j])
                listOfStringAngka[i] += str(angkaUnik[idx])

    result = int(listOfStringAngka[len(listOfStringAngka)-1])
    total = 0
    #menjumlahkan
    for i in range(len(listOfStringAngka) -1):
        total += int(listOfStringAngka[i])
    if(total == result):
        #output
        for i in range(len(listOfStringAngka) -1):
            if (i == len(listOfStringAngka) - 2):
                print(listOfOperand[i].rjust(5), '+', end = ' '.rjust(8))
                print(listOfStringAngka[i].rjust(10), '+')

```

```

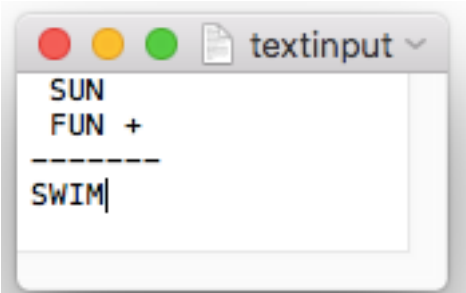
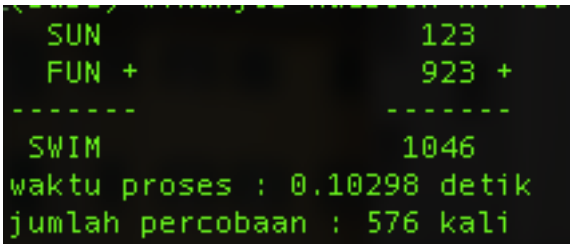
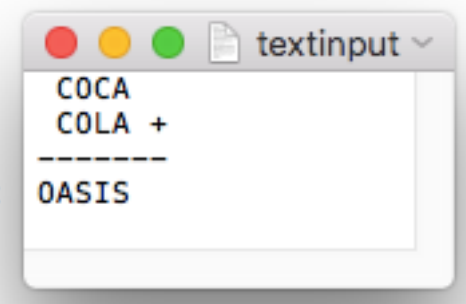
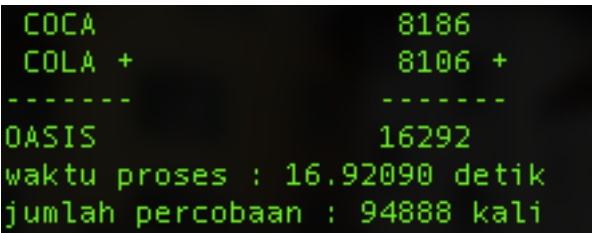
else:
    print(listOperand[i].rjust(5), end = ' '.rjust(10))
    print(listOfStringAngka[i].rjust(10))
    print("-----".rjust(5), end = ' '.rjust(10))
    print("-----".rjust(10))
    print(hasil.rjust(5), end = ' '.rjust(10))
    print((listOfStringAngka[len(listOfStringAngka)-1]).rjust(10))

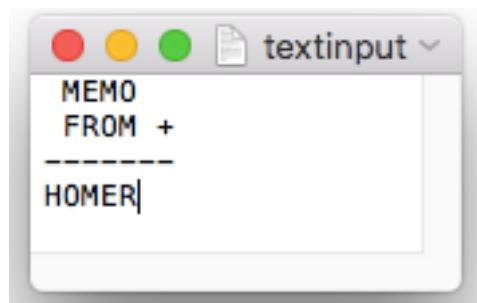
durasi = time.perf_counter() - startTime
print("waktu proses : %.5f detik" % durasi)
print("jumlah percobaan : %d kali" % counter)
break
else:
    counter += 1
    i_unik += 1
else:
    i_unik += 1

```

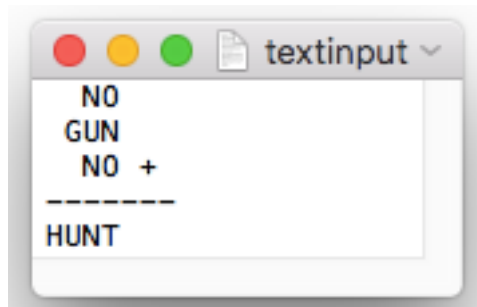
solver(listOperand, hasil)

### III. Input / Output

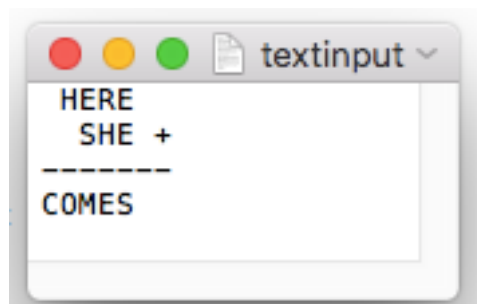
INPUT	OUTPUT
	
	



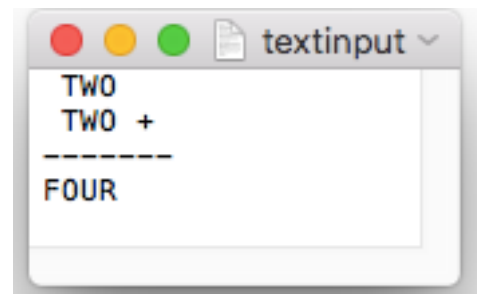
```
MEMO 8485
FROM + 7358 +
-----
HOMER 15843
waktu proses : 18.11283 detik
jumlah percobaan : 86658 kali
```



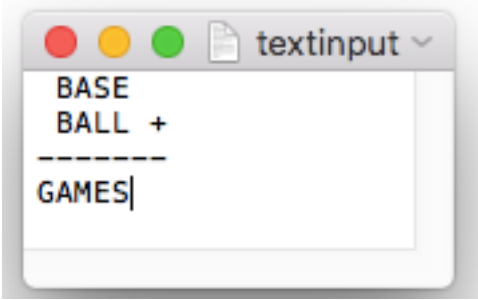
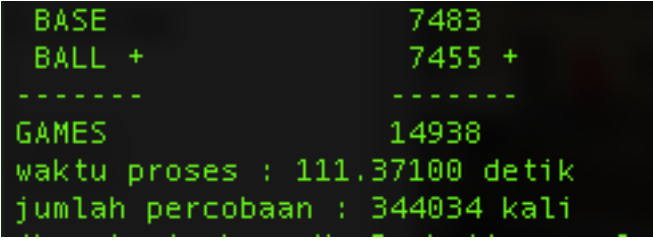
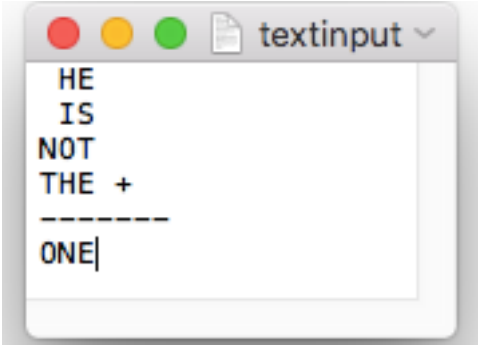
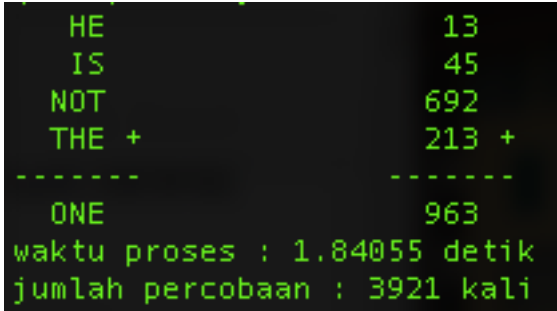
```
NO 87
GUN 908
NO + 87 +
-----
HUNT 1082
waktu proses : 19.24927 detik
jumlah percobaan : 90929 kali
```



```
HERE 9454
SHE + 894 +
-----
COMES 10348
waktu proses : 151.09539 detik
jumlah percobaan : 393736 kali
```



```
TWO 734
TWO + 734 +
-----
FOUR 1468
waktu proses : 16.32993 detik
jumlah percobaan : 83952 kali
```

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 cryptarithmic hanya benar untuk persoalan cryptarihtmetic dengan dua buah operand.		√
5. Solusi cryptarithmic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand.	√	

Repository berisi source code :

[https://github.com/irvinandryan/Tucil1\\_Stima](https://github.com/irvinandryan/Tucil1_Stima)

