**MENYELESAIKAN *CRYPTARITHMETHIC* DENGAN MENGGUNAKAN ALGORITMA *BRUTE FORCE***

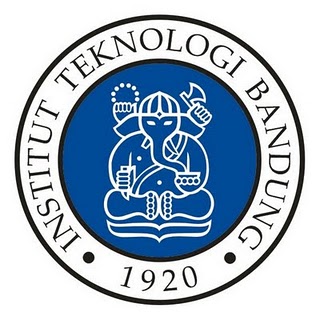
TUGAS KECIL

Diajukan sebagai tugas dari mata kuliah Strategi Algoritma di jurusan Informatika Institut Teknologi Bandung

Oleh :

**Fakhri Nail Wibowo**

**135190**

****

**PROGRAM STUDI INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG  
2021**

1. **Algoritma Brute-Force**

Pemecahan cryptarithmethic dengan algoritma *brute force* berarti mencoba semua kemungkinan yang ada satu per satu, dengan cara:

1. Membaca file lalu masukkan soal dan huruf pertama ke dalam string masing-masing
2. Buat array berisi angka 0 sampai 9 untuk dipermutasikan
3. Ubah string soal menjadi dictionary dengan key huruf unik dan value antara 0 sampai 9
4. Ubah string soal menjadi angka dengan melakukan substitusi menggunakan dictionary
5. Cek hasil penjumlahan dan persyaratan memenuhi atau tidak
6. Jika tidak, lakukan permutasi pada array berisi angka dan masukkan angka tersebut sebagai value pada dictionary
7. Lakukan kembali substitusi soal menjadi angka dengan menggunakan dictionary
8. Cek kembali hasil penjumlahan dan persyaratan memenuhi atau tidak
9. Ulangi langkah ke-6 jika masih belum memenuhi
10. Jika sudah memenuhi, print hasil ke layar

Untuk permutasi array menggunakan algoritma permutasi *next permutation*, dengan cara:

1. Cari indeks *i* terbesar yang memenuhi a[i] < a[i+1], jika tidak ada maka permutasi selesai
2. Cari indeks *j* terbesar yang memenuhi a[i] < a[j]
3. Tukar nilai dari a[i] dan a[j]
4. Balik sekuens array mulai dari a[j+1] hingga elemen terakhir
5. **Source Code**

*import* time

*# mencari index terbesar yang memenuhi a[i] < a[i+1]*

def largestIndexPlusOne(array):

    index = -1

*for* i in range(len(array)-1):

*if* array[i] < array[i+1] and i > index:

            index = i

*return* index

*# mencari index terbesar yang memenuhi a[i] < a[j]*

def largestIndexAll(array,index):

    maxIndex = -1

*for* i in range(index, len(array)):

*if* array[index] < array[i] and i > maxIndex:

            maxIndex = i

*return* maxIndex

*# ubah string ke int*

def strToInt(operand, lettersValue):

    value = 0

    factor = 1

*for* letter in reversed(operand):

        value += lettersValue.get(letter) \* factor

        factor \*= 10

*return* value

*# cek hasil penjumlahan memenuhi semua syarat*

def evalEquation(equation, lettersValue):

    operands = 0

    listOfValue = []

    value = strToInt(equation[len(equation)-1], lettersValue)

*for* i in range(len(equation)-1):

        listOfValue.append(strToInt(equation[i], lettersValue))

        operands += strToInt(equation[i], lettersValue)

*if* operands == value:

*for* i in range(len(listOfValue)):

*if* i == len(listOfValue)-1:

                print(str(listOfValue[i])+'+')

                print('-----')

*else*:

                print(listOfValue[i])

        print(value)

*return* True

*else*:

*return* False

*# cek digit pertama bukan nol*

def firstDigitNotZero(array, dict):

*for* letter in array:

*if* dict.get(letter) == 0:

*return* False

*return* True

file1 = open('../test/test8.txt', 'r')

equation = []

firstLetters = set()

*# baca file dan setup*

*for* line in file1:

*if* line != '-----\n':

        equation.append(line.strip('+\n'))

        firstLetters.add(line[0])

start = time.time()

file1.close()

firstLetters = list(firstLetters)

possibleNums = [0,1,2,3,4,5,6,7,8,9]

uniqueEquation = []

listEquation = (list(''.join(equation)))

*for* letter in listEquation:

*if* letter not in uniqueEquation:

        uniqueEquation.append(letter)

lettersValue = dict(zip(uniqueEquation,possibleNums))

isFound = False

tries = 0

*# print soal*

*for* i in range(len(equation)):

*if* i == len(equation)-2:

        print(equation[i]+'+')

        print('-----')

*else*:

        print(equation[i])

print('')

*# cek permutasi pertama*

*if* evalEquation(equation,lettersValue):

    tries += 1

*else*:

*# cek sisa permutasi lainnya dengan next permutation*

*while* largestIndexPlusOne(possibleNums) != -1 and not isFound:

        tries += 1

        i = largestIndexPlusOne(possibleNums)

        j = largestIndexAll(possibleNums, i)

*# swap possibleNums[i] dan possibleNums[j]*

        possibleNums[i], possibleNums[j] = possibleNums[j], possibleNums[i]

*# reverse list mulai dari i+1*

        possibleNums[i+1:] = possibleNums[i+1:][::-1]

        lettersValue = dict(zip(uniqueEquation,possibleNums))

*# cek hasil penjumlahan*

*if* firstDigitNotZero(firstLetters, lettersValue) and evalEquation(equation, lettersValue):

            isFound = True

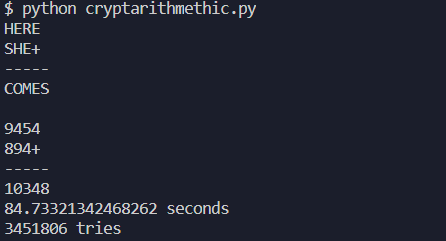
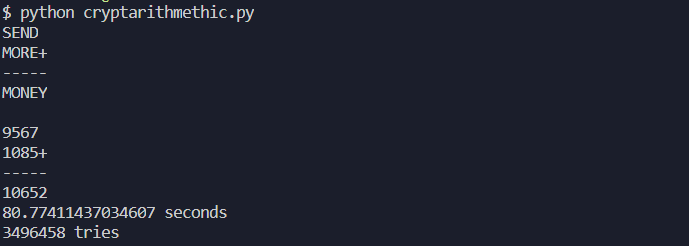
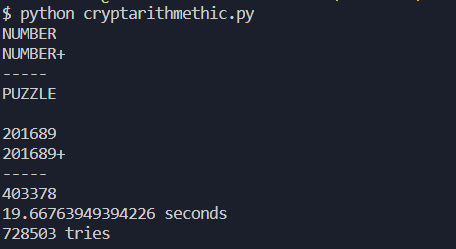
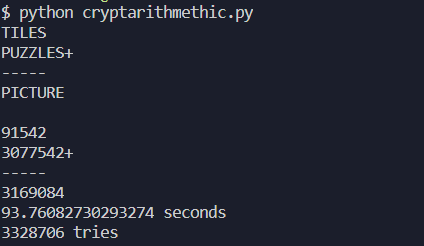
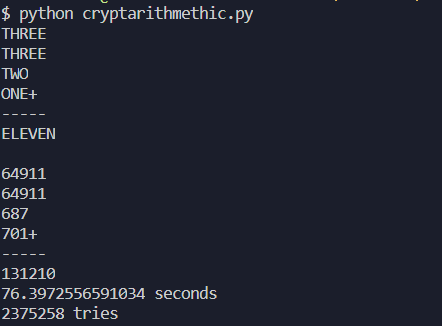
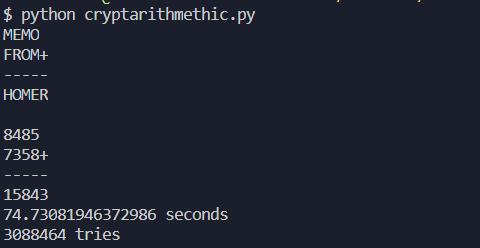
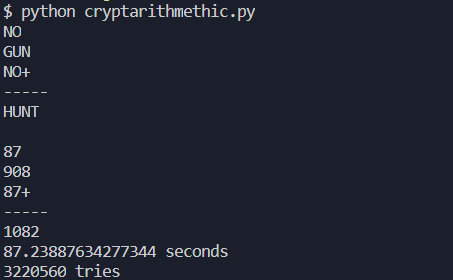
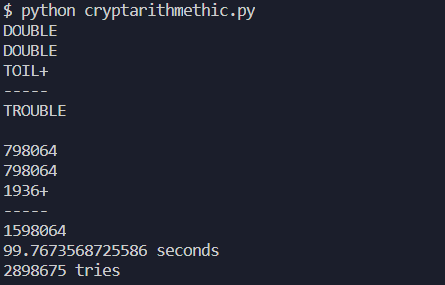
*else*:

            i = largestIndexPlusOne(possibleNums)

print(time.time() - start, "seconds")

print(tries, 'tries')

1. **Screenshot**



1. **Alamat Kode Program**
2. **Cek List**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil *running* | V |  |
| 1. Program dapat membaca file masukan dan menuliskan luaran. | V |  |
| 1. Solusi *cryptarithmetic* hanya benar untuk persoalan *cryptarihtmetic* dengan dua buah *operand*. |  | V |
| 1. Solusi *cryptarithmetic* benar untuk persoalan *cryptarihtmetic* untuk lebih dari dua buah operand. | V |  |
|  |  |  |