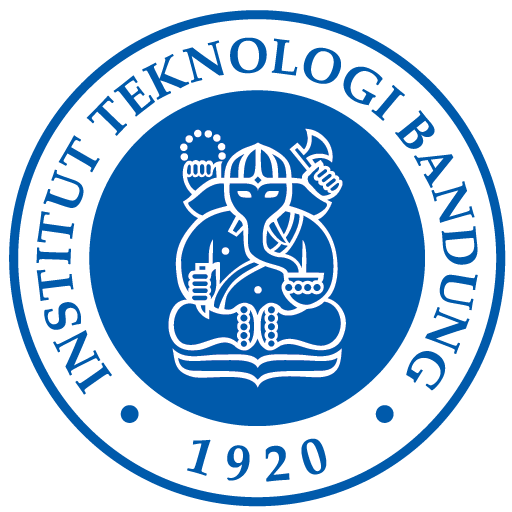
**TUGAS KECIL 1** **IF2211** **STRATEGI ALGORITMA**

**PENYELESAIAN CRYPTARITHMETIC DENGAN**

**ALGORITMA BRUTE FORCE**

****

**Disusun oleh:**

**Nama: Muhammad Furqon**

**NIM: 13519184**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2021**

Laporan Tugas Kecil 1 Strategi Algoritma

Cryptarithmetic

1. Algoritma Brute Force
2. Cari semua huruf yang unik atau berbeda yang terdapat pada semua operand
3. Cari permutasi dari angka 0 sampai 9 dengan panjang sebanyak jumlah huruf yang unik
4. Setiap huruf masing-masing diberi nilai, yaitu angka yang unik sesuai dengan salah satu permutasi
5. Nilai tersebut diperiksa dari kolom atau digit yang paling kanan, konstrainnya adalah jumlah nilai operand yang di atas harus sesuai dengan yang di bawah, simpan *remainder* jika menghasilkan jumlah yang lebih besar dari sepuluh. Syarat lainnya digit paling kiri (huruf pertama) nilainya tidak boleh nol
6. Ulangi langkah empat sampai semua persamaan diperiksa. Jika ada yang melanggar syarat ulangi langkah ketiga dengan permutasi yang berbeda

Contoh:

SEND

MORE+

-----

MONEY

Diperiksa persamaan D+E = Y+remainder1

Diperiksa persamaan N+R+remainder1 = E+remainder2

Dan seterusnya sampai semua persamaan diperiksa, syarat lainnya S dan M (huruf terkiri) adalah nol. Jika terdapat syarat yang dilanggar ambil angka unik dari permutasi yang lain.

1. SourceCode(Python):

Tucil1.py

#Muhammad Furqon (13519184)

#Cryptarithmatic

import time

import os

last\_hasil=0,0,0

length\_max=0

remainder=0

angka=0

jmlh=0

valid=True

jmlh\_loop=0

list\_angka = [0,1,2,3,4,5,6,7,8,9]

list\_huruf = []

correct =0

ada\_jawaban=False

#Permutasi elemen

def permutasi(elemen, r):

    n = len(elemen)

    indeks = list(range(n))

    perm = list(range(n, n-r, -1))

    yield tuple(elemen[i] for i in indeks[:r])

    stop=False

    while(not stop):

        stop=True

        for i in reversed(range(r)):

            perm[i] -= 1

            if perm[i] == 0:

                indeks[i:] = indeks[i+1:] + indeks[i:i+1]

                perm[i] = n - i

            else:

                j = perm[i]

                indeks[i], indeks[-j] = indeks[-j], indeks[i]

                yield tuple(elemen[i] for i in indeks[:r])

                stop=False

                break

#Referensi: https://stackoverflow.com/questions/104420/how-to-generate-all-permutations-of-a-list

#Dictionary huruf unik

isMain=True

substring="+"

substring\_skip="-"

list\_main=[]

list\_main\_output=[]

list\_hasil=[]

dict\_huruf = {}

#List untuk setiap kata, inisialisasi dictionary

os.chdir("..")

cur\_dir=os.getcwd()

os.chdir(cur\_dir+"\\test")

nama\_file = input("Masukkan nama file masukan: ")

cur\_dir=os.getcwd()

path=cur\_dir+ "\\" + nama\_file

file = open(path,"r")

Lines = file.readlines()

start = time.time()

for kata in Lines:

    skip=False

    kata=kata.strip()

    if(substring\_skip in kata):

        skip=True

    if(not skip):

        if(isMain):

            if(substring in kata):

                isMain=False

                kata=kata.replace('+','')

            list\_main.append(list(kata))

            list\_main\_output.append(list(kata))

        else:

            list\_hasil=list(kata)

        for huruf in kata:

            dict\_huruf[huruf] = dict\_huruf.get(huruf,0) + 1

#Buat dict kosong, list huruf

for key in dict\_huruf:

    list\_huruf.append(key)

    dict\_huruf[key] = -1

list\_hasil\_output=list(list\_hasil)

#Cek dari kanan/last

for x in list\_main:

    x.reverse()

list\_hasil.reverse()

#Ujung kiri tidak boleh 0, angka mulai dari 1

#Length di sini adalah indeks ujung, len-1

length\_main\_max = len(max(list\_main,key=len))-1

length\_max = len(list\_hasil)-1

# Permutasi

perm = permutasi(list\_angka,len(dict\_huruf))

for x in list(perm):

    #Setup

    correct=0

    angka=0

    valid=True

    i=0

    #ubah nilai dictionary

    for y in x:

        dict\_huruf[list\_huruf[angka]]=y

        angka += 1

    while(i<=length\_max and valid):

        if(i <= length\_main\_max):

            for sublist in list\_main:

                if(i<=len(sublist)-1):

                    temp = dict\_huruf[sublist[i]]

                    if(i==(len(sublist)-1)):

                        if(temp==0):

                            valid=False

                            remainder=0

                else:

                    temp=0

                jmlh += temp

            jmlh += remainder

            if jmlh>=10:

                remainder=jmlh//10

            else:

                remainder=0

            if(i!=length\_max):

                jmlh=jmlh%10

            #penjumlahan elemen

        else:

            jmlh=remainder

        #cek dengan elemen dari hasil

        last\_hasil = dict\_huruf[list\_hasil[i]]

        if(i==(len(list\_hasil)-1)):

            if(last\_hasil==0):

                valid=False

                remainder=0

        if(valid):

            if(jmlh==last\_hasil):

                correct+=1

                if(correct==length\_max+1):

                    valid=False

                    remainder=0

            else:

                valid=False

                remainder=0

        jmlh=0

        i+=1

    if(correct==length\_max+1):

        #Print soal

        count=0

        print("Soal:")

        for output in list\_main\_output:

            selisih=length\_max-len(output)+1

            print(" "\*selisih,end="")

            for x in output:

                print(x, end="")

            if(count==len(list\_main\_output)-1):

                print("+",end="")

            print()

            count+=1

        print("-" \* (length\_max+1))

        for x in list\_hasil\_output:

            print(x, end="")

        print()

        #Print jawaban

        count=0

        print("Jawaban:")

        for output in list\_main\_output:

            selisih=length\_max-len(output)+1

            print(" "\*selisih,end="")

            for x in output:

                print(dict\_huruf[x], end="")

            if(count==len(list\_main\_output)-1):

                print("+",end="")

            print()

            count+=1

        print("-" \* (length\_max+1))

        for x in list\_hasil\_output:

            print(dict\_huruf[x], end="")

        print()

        #Print tes dan waktu

        print('Jumlah total tes: {}'.format(jmlh\_loop))

        end = time.time()

        print('Waktu pengerjaan adalah {:.2f} s'.format(end-start))

        correct=0

        ada\_jawaban=True

    jmlh\_loop += 1

if(not ada\_jawaban):

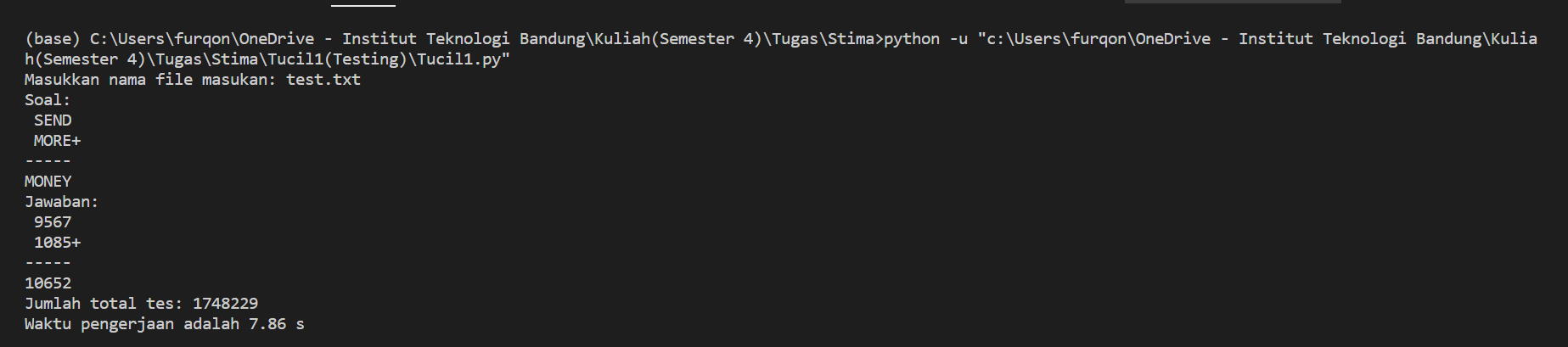
    print("Tidak ditemukan solusi")

input("Tekan enter untuk mengakhiri program")

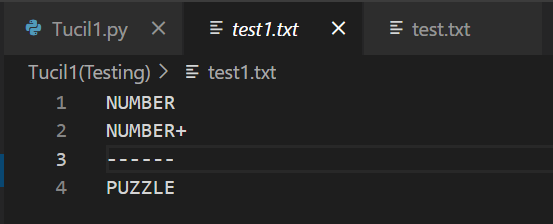
1. Skrinshut Program:

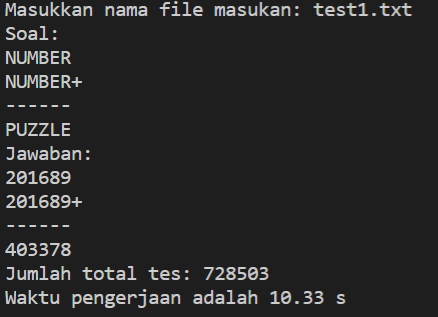
1.



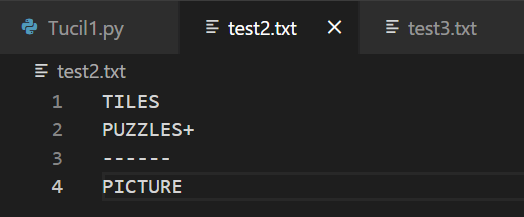


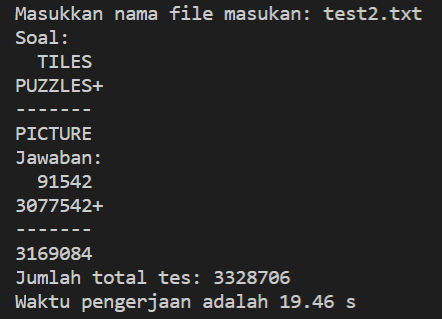
2.



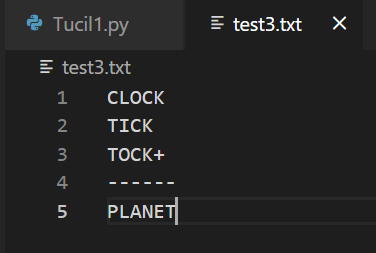


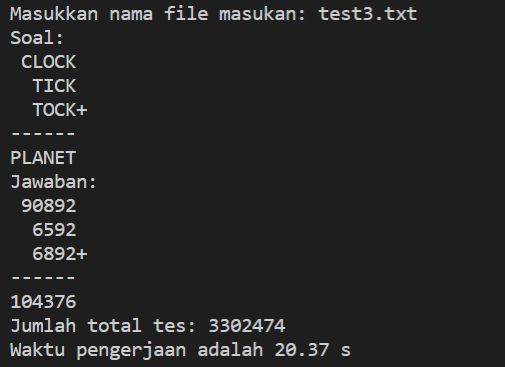
3.



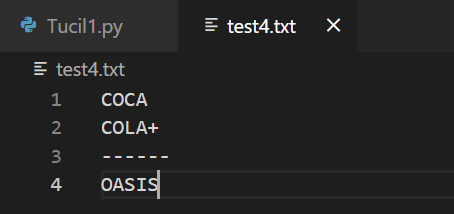


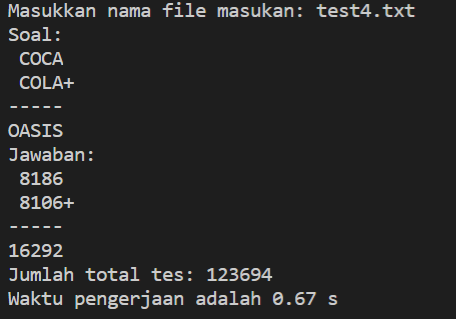
4.



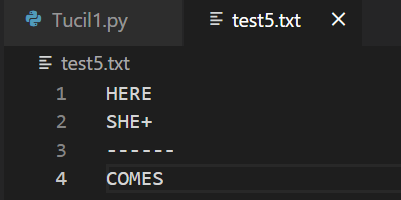


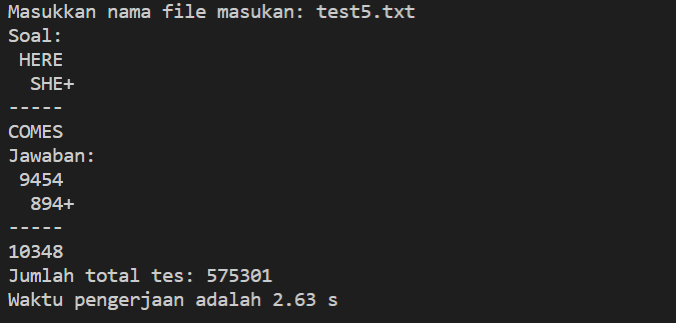
5.



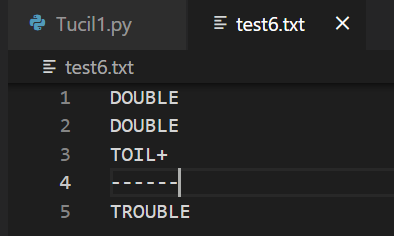


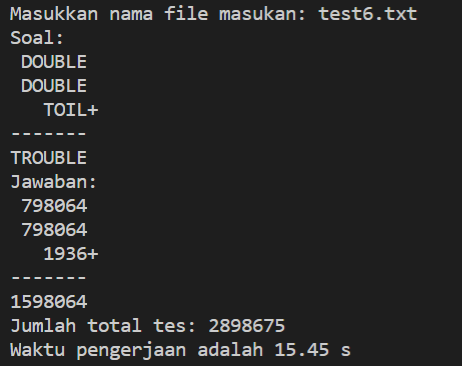
6.



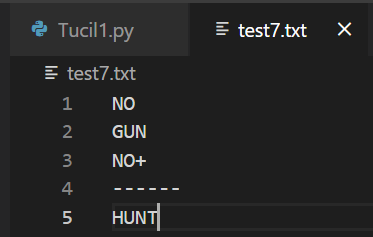


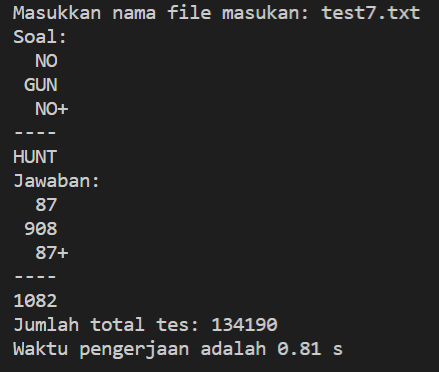
7.



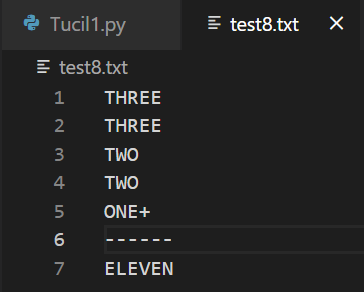


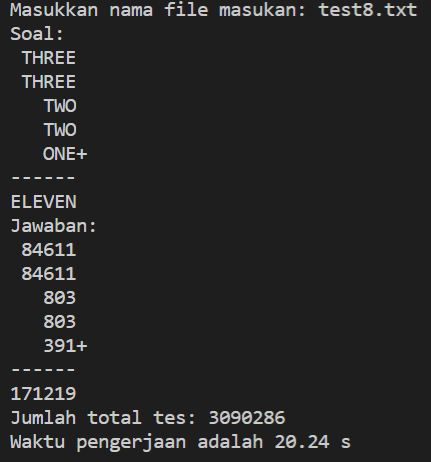
8.





9.





1. Link Drive:

<https://drive.google.com/drive/folders/1AjK0GVVXSn9aJEkOEGZhuTJI9i0NTEWF?usp=sharing>

Drive Pengumpulan

<https://drive.google.com/drive/u/0/folders/1Ugq787WRmed5-ZR_mZizJmjxCMRnfhIo>

|  |  |  |
| --- | --- | --- |
| 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. | √ |  |

Referensi:

https://puzzling.stackexchange.com/questions/51/alphametic-verbal-arithmetic-general-strategy

http://www.cryptarithms.com/default.asp?pg=1

https://people.revoledu.com/kardi/tutorial/CryptArithmetic/AttractionsIntentionsRegeneration.html

<https://stackoverflow.com/questions/104420/how-to-generate-all-permutations-of-a-list>