Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun 2020/2021

Penyelesaian *Cryptarithmetic* dengan Algoritma *Brute Force*

Nama : Daru Bagus Dananjaya

NIM : 13519080

Kelas : K02

Pendahuluan

Cryptarithmetic (atau cryptarithm) adalah sebuah puzzle penjumlahan di dalam matematika dimana angka diganti dengan huruf. Setiap angka dipresentasikan dengan huruf yang berbeda. Deskripsi permainan ini adalah: diberikan sebuah penjumlahan huruf, carilah angka yang merepresentasikan huruf-huruf tersebut.

Contoh dan solusinya :

```
  S E N D          9 5 6 7
+ M O R E        + 1 0 8 5
---------        ---------
M O N E Y        1 0 6 5 2
```

Jadi, S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2

Deskripsi Langkah-Langkah

Dalam menyelesaikan persoalan cryptArithmetic menggunakan metode brute force ini, saya memanfaatkan *data structure* berupa array. Array digunakan untuk menyimpan operand, menyimpan karakter unik beserta *corresponding value*-nya untuk kemudian dicek kesesuaian antara problem dan hasilnya, serta menyimpan hasil dari persoalan yang diberikan.

Program ini dibuat menggunakan konsep permutasi, karena maksimum terdapat 10 huruf unik, maka terdapat 10! cara atau 3.628.800 buah cara untuk menyusun 10 angka yang merepresentasikan 10 buah huruf. Untuk kasus huruf unik yang berjumlah kurang dari 10, akan tetap dihasilkan 10! cara penyusunan angka tetapi hanya akan diambil sesuai dengan kebutuhan, hal tersebut menunjukkan bahwa berapapun jumlah hurufnya, akan tetap dihasilkan sebuah solusi meskipun tidak efisien karena ada proses *redundant* yang tetap dilakukan.

Langkah-langkah Program :

1. Membaca file persoalan *Cryptarithmetic* yang akan diselesaikan, lalu dilakukan *parsing* pada setiap *operand* untuk dimasukkan ke dalam list yang akan diproses kemudian.
2. Dilakukan pengecekan kemunculan sebuah huruf yang kemudian akan menghasilkan sebuah list karakter unik. Karakter unik tersebut dimasukkan ke dalam *listOfNode* yang berisi sebuah huruf dan *corresponding value*-nya yang diinisialisasi dengan nilai 0.
3. *Generate* semua kemungkinan permutasi 10 angka disusun 10 sambil dilakukan assignment ke *corresponding value* dari *listOfNode*. Setiap mencapai karakter unik terakhir, dilakukan pencocokan kemungkinan, jika ditemukan kecocokan, maka program akan keluar.
4. Pencocokan kemungkinan dilakukan dengan cara mengubah setiap huruf menjadi angka yang merepresentasikannya kemudian dihitung apakah jumlah dari seluruh *operand* sama dengan hasil. Jika ditemukan kecocokan, maka program akan keluar, namun jika tidak ditemukan kecocokan, permutasi akan dilanjutkan.

Source Code

```
import time

# listOfNode = [{letter, corresponding value}]
use = [0 for i in range(10)]

listOfWord = []
listOfNode = []
opCounter = 1


def noLeadingZero(listOfWord):
   noZero = []
   for i in range(len(listOfWord)):
       noZero.append(listOfWord[i][0])

   return noZero

def isValid(listOfNode, countChar, listOfWord) :
   global val
   global opCounter
   val = [0 for i in range(len(listOfWord))]
   temp = 0
   totalVal=0

   for z in range(len(listOfWord)):
       m = 1
       i = len(listOfWord[z])-1
       while (i >= 0) :
           char = listOfWord[z][i]
           for j in range(countChar) :
               temp = j
               if (listOfNode[j][0] == char) :
                   break
           val[z] += m * listOfNode[temp][1]
           m *= 10
           i-=1
```

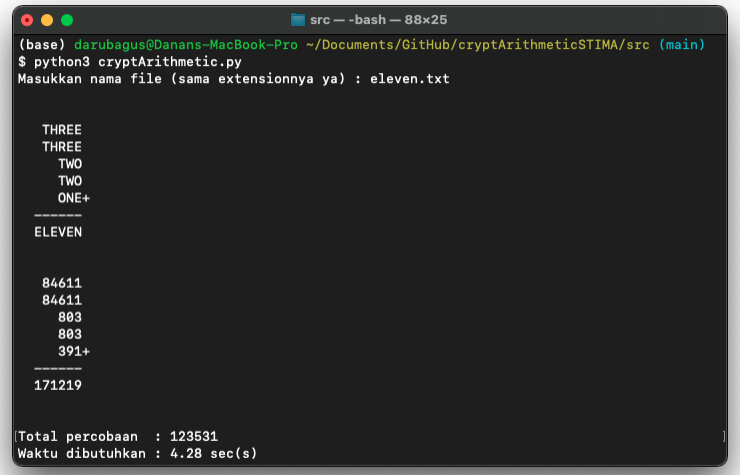
```
    # buat ngecek hasil
    for i in range(len(listOfWord)-1) :
        totalVal += val[i]

    if (totalVal == val[len(listOfWord)-1]) :

        return True
    else:
        opCounter += 1
        return False


def Permutate(uniqueChar, listOfNode, n, listOfWord) :
    global opCounter
    if (n == uniqueChar-1):#Basis
        if ((listOfNode[n][0] not in(noLeadingZero(listOfWord)))) :
            for i in range(10):
                if (use[i] == 0):
                    listOfNode[n][1] = i
                    if (isValid(listOfNode,uniqueChar,listOfWord)) :
                       return True
        else :
            for i in range(1,10):
                if (use[i] == 0):
                    listOfNode[n][1] = i
                    if (isValid(listOfNode,uniqueChar,listOfWord)) :
                       return True
    else:
        if (listOfNode[n][0] not in(noLeadingZero(listOfWord))):
            for i in range(10):
                if (use[i]==0):
                    listOfNode[n][1] = i
                    use[i] = 1
                    if (Permutate(uniqueChar,listOfNode, n+1,listOfWord)):
                        return True
                    use[i] = 0
```

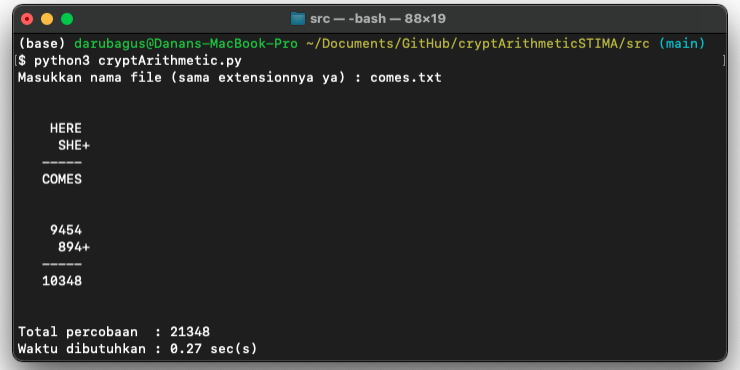
```
        else :
            for i in range(1,10):
                if (use[i]==0):
                    listOfNode[n][1] = i
                    use[i] = 1
                    if (Permutate(uniqueChar,listOfNode, n+1,listOfWord) ):
                        return True
                    use[i] = 0

def problemSolver(listofWord,listOfNode):
    uniqueChar = 0

    letterFrequency = [0 for i in range(26)]

    for i in range(len(listOfWord)):
        for j in range(len(listOfWord[i])):
            letterFrequency[ord(listOfWord[i][j]) - ord('A')] += 1

    for i in range(26):
        if (letterFrequency[i] > 0) :
            uniqueChar +=1

    if (uniqueChar > 10) :
        # Operand terlalu banyak
        print("Invalid input")

    # listOfNode = [{letter, corresponding value}]
    listOfNode = []

    for i in range(26):
        j = 0
        if (letterFrequency[i] > 0) :
            # listOfNode[j][0] = chr(i + ord('A'))
            # [['A',0]]
            listOfNode.append([chr(i + ord('A')), 0])
            j += 1
            # listOfNode.append([chr(i + ord('A')), 0])
```

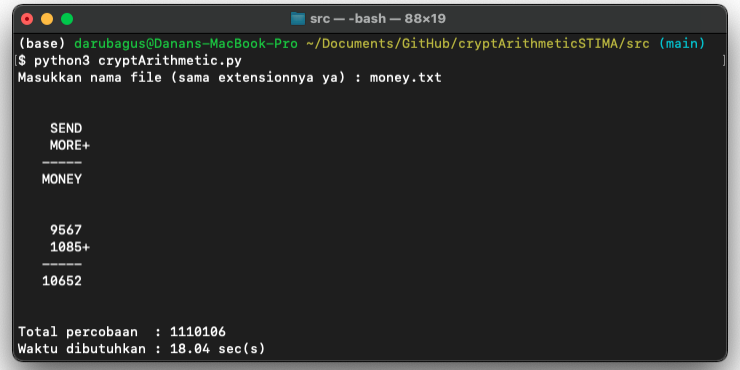
```
    return Permutate(uniqueChar, listOfNode, 0, listOfWord)

def displayAngka() :
    for i in range(len(listOfWord)-1):
        if (i != len(listOfWord)-2):
            txt = str(val[i]).rjust(8)
            print(txt)
        else :
            txt = str(val[i]).rjust(8)
            print(txt+"+")

    length = (len(max(listOfWord, key=len)))
    string = ''
    for i in range(length) :
        string += '-'
    string = string.rjust(8)
    print(string)

    txt = str(val[len(listOfWord)-1]).rjust(8)
    print(txt)
    print("\n")

def displayWord(listOfWord):
    for i in range(len(listOfWord)-1):
        if (i != len(listOfWord)-2):
            txt = (listOfWord[i]).rjust(8)
            print(txt)
        else :
            txt = (listOfWord[i]).rjust(8)
            print(txt+"+")

    length = (len(max(listOfWord, key=len)))
    string = ''
    for i in range(length) :
        string += '-'
    string = string.rjust(8)
    print(string)
```

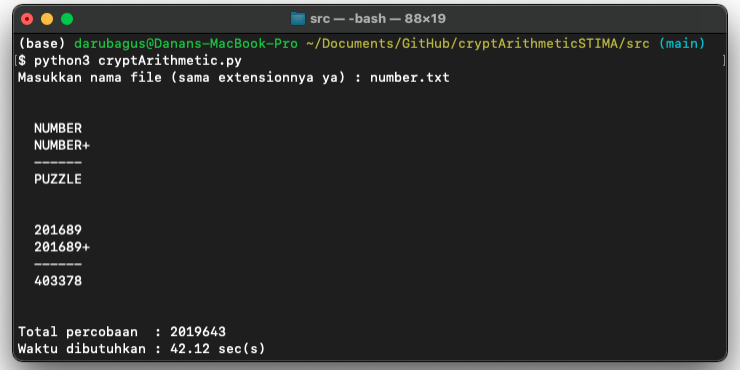
```
   txt = (listOfWord[len(listOfWord)-1]).rjust(8)
   print(txt)
   print("\n")

#MAIN PROGRAM

# open file
fileName = input("Masukkan nama file (sama extensionnya ya) : ")
problem = "../test/"+fileName
inString = open(problem, "r")

line = inString.readlines()

i = 0
while (i < len(line)):
   if (line[i][0] != '-'):
       # do nothing
       cleanString = line[i].replace(" ","")
       cleanString = cleanString.replace("+","")
       cleanString = cleanString.replace("\n","")

       listOfWord.append(cleanString)
   i+=1
inString.close()

# START TIME
startTime = time.time()
problemSolver(listOfWord,listOfNode)
print("\n")
displayWord(listOfWord)
displayAngka()
# HITUNG RUNNING TIME
runningTime = (time.time() - startTime)
print("Total percobaan  : "+str(opCounter))
print("Waktu dibutuhkan : %.2f sec(s)" %(runningTime))
```

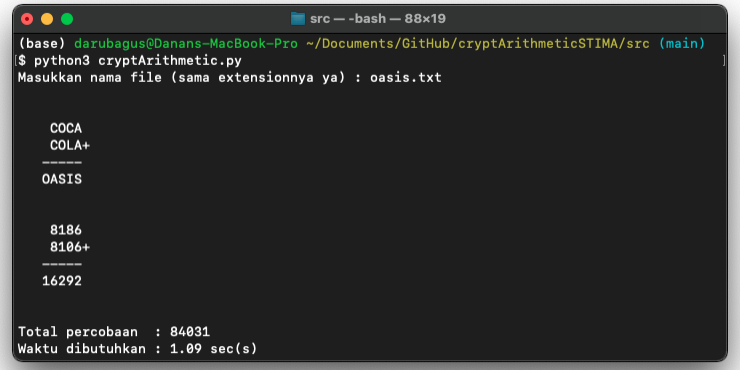
Hasil Running Program

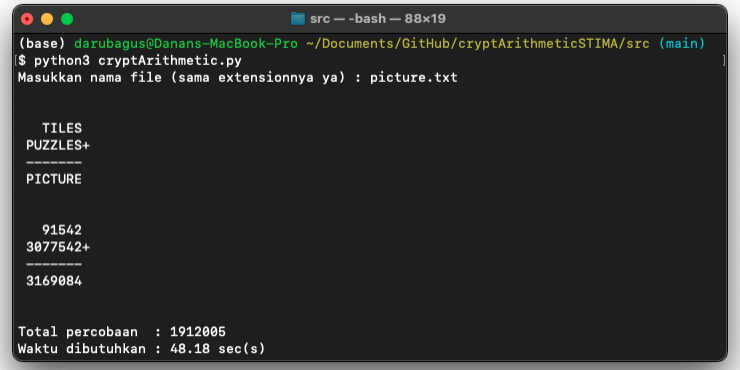


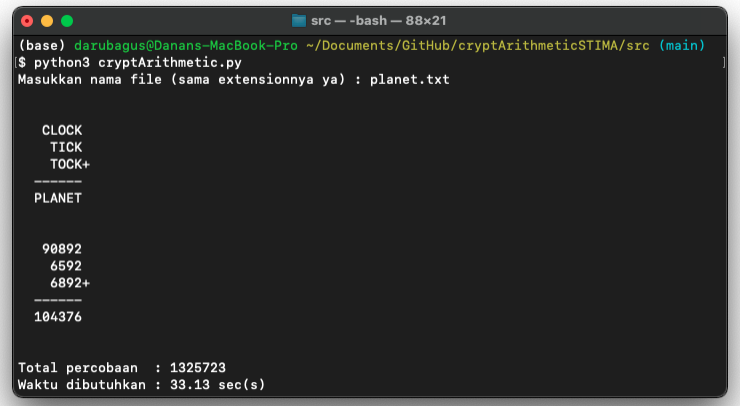


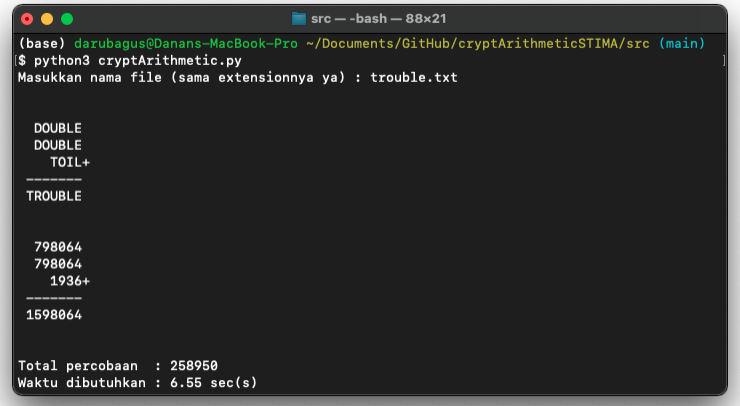



src — -bash — 88×19
(base) darubagus@Danans-MacBook-Pro ~/Documents/GitHub/cryptArithmeticSTIMA/src (main)
$ python3 cryptArithmetic.py
Masukkan nama file (sama extensionnya ya) : number.txt
NUMBER
NUMBER+
PUZZLE
201689
201689+
403378
Total percobaan : 2019643
Waktu dibutuhkan : 42.12 sec(s)


src — -bash — 88×19
(base) darubagus@Danans-MacBook-Pro ~/Documents/GitHub/cryptArithmeticSTIMA/src (main)
$ python3 cryptArithmetic.py
Masukkan nama file (sama extensionnya ya) : oasis.txt
COCA
COLA+
OASIS
8186
8106+
16292
Total percobaan : 84031
Waktu dibutuhkan : 1.09 sec(s)


src — -bash — 88×19
(base) darubagus@Danans-MacBook-Pro ~/Documents/GitHub/cryptArithmeticSTIMA/src (main)
$ python3 cryptArithmetic.py
Masukkan nama file (sama extensionnya ya) : picture.txt
TILES
PUZZLES+
PICTURE
91542
3077542+
3169084
Total percobaan : 1912005
Waktu dibutuhkan : 48.18 sec(s)





Link File

1. Github
   https://github.com/darubagus/cryptArithmeticSTIMA
2. Google Drive
   https://drive.google.com/drive/folders/1FRTPH6tBPT5nzUo56UwmlWLGZym5aey-?usp=sharing

Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (syntax error)	✔	
2. Program berhasil *running*	✔	
3. Program dapat membaca file masukan dan menuliskan luaran	✔	
4. Solusi *Cryptarithmetic* hanya benar untuk persoalan *cryptarithmetic* dengan dua buah operand		✔
5. Solusi *cryptarithmetic* benar untuk persoalan *cryptarithmetic* untuk lebih dari dua buah operand	✔	