Laporan Penyelesaian *Cryptarithmetic* dengan Algoritma Brute Force

TUGAS KECIL STRATEGI ALGORITMA

Oleh:

Michael Philip G

13519121

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Langkah – Langkah Penyelesaian Cryptarithmethic dengan Algoritma Brute Force

Secara sederhana, persoalan Cryparithmetic dapat diselesaikan dengan Algoritma Brute Force, yaitu dengan mencoba-coba semua kemungkinan angka yang mungkin untuk disubtitusikan kepada setiap huruf yang unik. Algoritma ini akan memiliki kompleksitas P(10,N) dimana N menyatakan banyaknya huruf yang unik. Berikut merupakan penjelasan tahapan dalam bahasa python untuk menyelesaikan persoalan Cryptarithmethic dengan algoritma brute force.

1. **readFromFile**
   Fungsi ini menerima 1 buah parameter string yang menyatakan nama file yang terdapat pada directory ../test/NAMA_FILE.txt. Fungsi ini akan mengembalikan tuple berupa setiap kata (operand) dan juga hasil dari penjumlahan kata-kata tersebut.
2. **generateDict**
   Fungsi ini menerima 1 buah parameter string yang menyatakan huruf-huruf unik yang hendak dipetakan menjadi dictionary, dengan key dari dictionary tersebut merupakan huruf-huruf yang ada pada parameter tersebut dan untuk semua key akan di assign default value berupa -1. Fungsi ini mengembalikan dictionary.
3. **isNoFirstZero**
   Fungsi ini menerima 3 buah parameter berupa dictionary, array of string, dan string. Dictionary yang dimaksud adalah dictionary yang menyimpan kemungkinan pemetaan angka terhadap huruf yang unik, array of string merupakan variable yang berisikan semua operand yang ada, dan string pada parameter ketiga menyatakan hasil penjumlahan semua operand tersebut. Fungsi ini akan mengecek apakah ada pemetaan huruf awal menjadi angka 0, jika iya, fungsi ini akan mengembalikan False, tapi jika tidak, fungsi ini akan mengembalikan True.
4. **checkPossibility**
   Fungsi ini menerima 3 buah parameter berupa dictionary, array of string, dan string. Dictionary yang dimaksud adalah dictionary yang menyimpan kemungkinan pemetaan angka terhadap huruf yang unik, array of string merupakan variable yang berisian semua operand yang ada dalam kondisi terbalik (reversed), dan string pada parameter ketiga menyatakan hasil penjumlahan semua operand dalam kondisi terbalik (reversed). Jika semua pemetaannya sukses dan sesuai dengan peraturan penjumlahan, maka fungsi ini akan

mengembalikan True, tetapi jika tidak, fungsi ini akan mengembalikan False.

5. **Permutations**

   Fungsi ini menerima 2 buah parameter berupa array dan integer. Array yang dimaksud merupakan kumpulan angka yang hendak diambil n buah untuk dicari semua permutasinya, integer pada parameter kedua menyatakan jumlah elemen yang hendak dihasilkan dari array pada parameter kedua. Fungsi ini akan mengembalikan list of list, atau array of array, dimana isinya adalah semua permutasi yang mungkin dengan panjang n pada array yang diberikan.

6. **Main**

   Fungsi ini merupakan fungsi utama yang didalamnya ada beberapa hal yang dilakukan seperti membalikkan kalimat, mencoba-coba semua kemungkinan yang ada dan mengembalikan pesan kepada layar

7. **parseFile**

   Fungsi ini merupakan fungsi khusus yang dipakai untuk memetakan file .txt yang awalnya dibaca menjadi angka yang relevan kemudian di cetak di layar. Fungsi ini menerima 2 parameter berupa string yang menyatakan nama file, dan dictionary yang berfungsi untuk memetakan huruf kepada angka yang bersesuaian.

B.Source Code

```
import math
import time
from itertools import permutations

def readFromFile(name):
  filename = "../test/{}.txt".format(name)
  print("\nOpening file {}".format(filename))
  with open(filename) as f:
    lines = f.readlines()
    result = ""
    i = 0
    print("\nINPUT :\n")
    for line in lines:
      lines[i] = line.strip()
      print(line.rstrip("\n"))
      i+=1
  return (lines[:-2],lines[-1:][0])

def parseFile(name, dictMap):
  filename = "../test/{}.txt".format(name)
  with open(filename) as f:
    lines = f.readlines()
    for line in lines:
      line = list(line.rstrip("\n"))
      i = 0
      for c in line:
        if(c!=" " and c!="+" and c!="-"):
          line[i] = dictMap[line[i]]
        i+=1
      target = ''.join(str(e) for e in line)
      print(target)
  print("\n")

def generateDict(text):
  res = {}
  for i in text:
    res[i] = -1
  return res

def isNoFirstZero(res, case, result):
  valid = True
  length = len(case)
  for i in range(length):
    valid = res[case[i][0]]!=0
    if(not(valid)):
```

```
      break
  return valid

def checkPossibility(res, caseReversed, resultReversed):
  found = True
  length = len(resultReversed)
  carry = 0
  for i in range(length):
    temp = carry
    for word in caseReversed:
      if(i<len(word)):
        temp+=res[word[i]] #convert char to it's integer value
    carry = int(temp/10)
    temp = temp%10
    if(temp!=res[resultReversed[i]]):
      found = False
      break
  return found

def main():
  print("Enter filename : ")
  name = input()
  (case, result) = readFromFile(name) #reading file

  totalTime = time.time()

  case[len(case)-1] = case[len(case)-1][:-1] #removing +
  caseReversed = [""] * len(case)
  resultReversed = result[::-1]
  uniqueChar = result
  i=0
  for a in case:
    caseReversed[i] = a[::-1]
    uniqueChar+=a
    i+=1

  uniqueChar = list(set(uniqueChar))
  totalChar = len(uniqueChar)
  res = generateDict(uniqueChar)
  totalCase = len(case)
  maxlength = len(result)

  found = False
  slots = [0,1,2,3,4,5,6,7,8,9]
  target = list(permutations(slots,totalChar))
  totalTest = 0

  for possibility in target:
```

```
    counter = 0
    #assgning possibility to dict
    for i in uniqueChar:
      res[i] = possibility[counter]
      counter+=1
    #checking if there's any leading 0
    valid = isNoFirstZero(res, case, result)
    if(not(valid)):
      continue
    #validating possibility
    found = checkPossibility(res, caseReversed, resultReversed)
    totalTest+=1
    if(found):
      break

  totalTime = time.time()-totalTime

  if(found):
    print("\nRESULT :\n")
    parseFile(name, res)
    print("Total Execution time : {:.6f}s".format(totalTime))
    print("Total Test Executed : {} possibilities".format(totalTest))
    print("Dictionary : {}\n".format(res))
  else:
    print("\nMaaf sepertinya algoritma saya tidak sempurna, atau test casenya
salah (melebihi 10 unique character), jawaban tidak ditemukan\n")

main()
```

C. Input/Output

```
Enter filename :
tes1

Opening file ../test/tes1.txt

INPUT :

 SEND
 MORE+
------
MONEY

RESULT :

 9567
 1085+
------
10652


Total Execution time : 2.405542s
Total Test Executed : 889245 possibilities
Dictionary : {'N': 6, 'O': 0, 'M': 1, 'S': 9, 'Y': 2, 'D': 7, 'E': 5, 'R': 8}
```

```
Enter filename :
tes2

Opening file ../test/tes2.txt

INPUT :

 ME
 ME+
----
BEE

RESULT :

 50
 50+
----
100


Total Execution time : 0.000000s
Total Test Executed : 97 possibilities
Dictionary : {'B': 1, 'M': 5, 'E': 0}
```

```
Enter filename :
tes3

Opening file ../test/tes3.txt

INPUT :

FORTY
  TEN
  TEN+
------
SIXTY

RESULT :

29786
  850
  850+
------
31486


Total Execution time : 0.538120s
Total Test Executed : 12703 possibilities
Dictionary : {'N': 0, 'I': 1, 'X': 4, 'Y': 6, 'R': 7, 'O': 9, 'S': 3, 'F': 2, 'E': 5, 'T': 8}
```

```
 Enter filename :
 tes4

 Opening file ../test/tes4.txt

 INPUT :

  CLOCK
   TICK
   TOCK+
 ------
 PLANET

 RESULT :

  24326
   8926
   8326+
 ------
 041578


 Total Execution time : 6.355394s
 Total Test Executed : 2099655 possibilities
 Dictionary : {'E': 7, 'A': 1, 'C': 2, 'T': 8, 'O': 3, 'P': 0, 'K': 6, 'N': 5, 'L': 4, 'I': 9}
```

```
Enter filename :
tes5

Opening file ../test/tes5.txt

INPUT :

  NO
 GUN
  NO+
-----
HUNT

RESULT :

  16
 781
  16+
-----
0813


Total Execution time : 0.030007s
Total Test Executed : 349 possibilities
Dictionary : {'N': 1, 'H': 0, 'T': 3, 'G': 7, 'O': 6, 'U': 8}
```

```
Enter filename :
tes6

Opening file ../test/tes6.txt

INPUT :

 CROSS
 ROADS+
------
DANGER

RESULT :

 96233
 62513+
------
158746


Total Execution time : 2.411631s
Total Test Executed : 805846 possibilities
Dictionary : {'O': 2, 'R': 6, 'A': 5, 'S': 3, 'E': 4, 'N': 8, 'D': 1, 'C': 9, 'G': 7}
```

```
Enter filename :
tes7

Opening file ../test/tes7.txt

INPUT :

 MEMO
 FROM+
------
HOMER

RESULT :

 4247
 3174+
------
07421


Total Execution time : 0.116029s
Total Test Executed : 41376 possibilities
Dictionary : {'M': 4, 'H': 0, 'O': 7, 'R': 1, 'F': 3, 'E': 2}
```

```
Enter filename :
tes8

Opening file ../test/tes8.txt

INPUT :

NUMBER
NUMBER+
--------
PUZZLE

RESULT :

201689
201689+
--------
403378


Total Execution time : 0.943149s
Total Test Executed : 240491 possibilities
Dictionary : {'U': 0, 'B': 6, 'R': 9, 'L': 7, 'M': 1, 'N': 2, 'P': 4, 'E': 8, 'Z': 3}
```

D.Link Alamat Drive

https://github.com/michaelpege/STIMA_TUCIL1_13519121

E. Tabel Ceklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	V	
2. Program berhasil running	V	
3. Program dapat membaca file masukan dan menuliskan luaran.	V	
4. Solusi cryptarithmetic hanya benar untuk persoalan cryptarihtmetic dengan dua buah operand		V
5. Solusi cryptarithmetic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand	V	