Laporan Tugas Kecil
IF2211 Strategi Algoritma
Cryptarithmetic

Rexy Gamaliel Rumahorbo (13519010)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2019

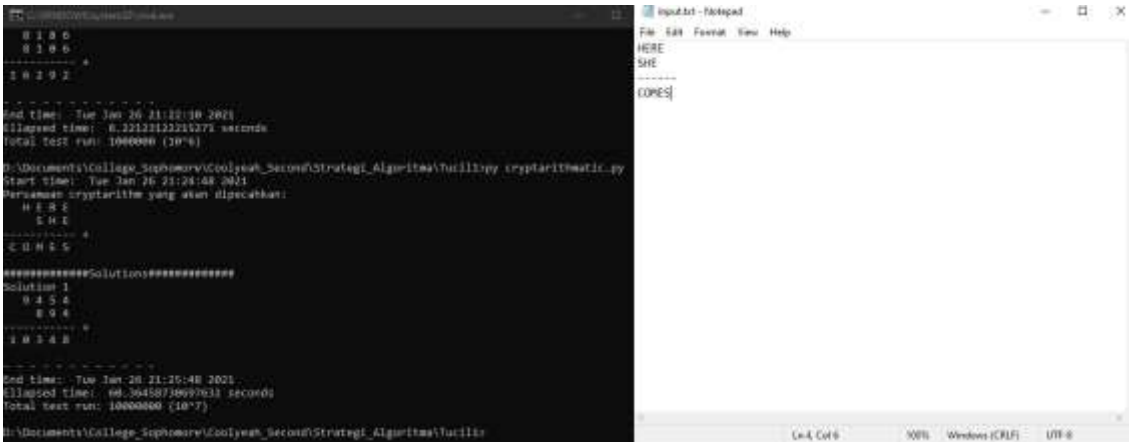
1. Algoritma Brute Force
    a. Preparasi:
        - Program membaca file
        - Menghitung waktu awal
        - Membuat list kata (word) “wordList” yang terdapat pada persamaan, dan dictionary “varDict” yang berisi pasangan variabel–nilai, serta menginisiasi list “solutions”.
        - Menampilkan persamaan pada soal.
    b. Iterasi (brute force):
        - Mengecek apakah konfigurasi nilai variabel-variabel pada varDict memenuhi, yakni tidak ada huruf inisial yang bernilai 0 dan masing-masing variabel bernilai unik. Jika ya, lanjut ke poin selanjutnya. Jika tidak, lanjut ke poin terakhir.
        - Mengecek apakah untuk konfigurasi nilai variabel-variabel yang valid tersebut memenuhi persamaan cryptarithm-nya. Jika ya, menampilkan bahwa program menemukan sebuah solusi sementara, dan menambahkannya ke list solutions. Jika tidak, lanjut ke poin terakhir.
        - Melakukan *increment* pada konfigurasi nilai variabel-variabel pada varDict, lanjut ke iterasi berikutnya. Iterasi dilakukan sebanyak 10^n kali, di mana n adalah jumlah variabel pada persamaan pada soal.
    c. Solusi:
        - Mengiterasi seluruh elemen list solutions dan menampilkan hasilnya
        - Menampilkan waktu berjalannya program dan banyak tes yang dilakukan

2. Link: https://github.com/Rexy-Gamaliel/Cryptarithmetic
3. Penilaian

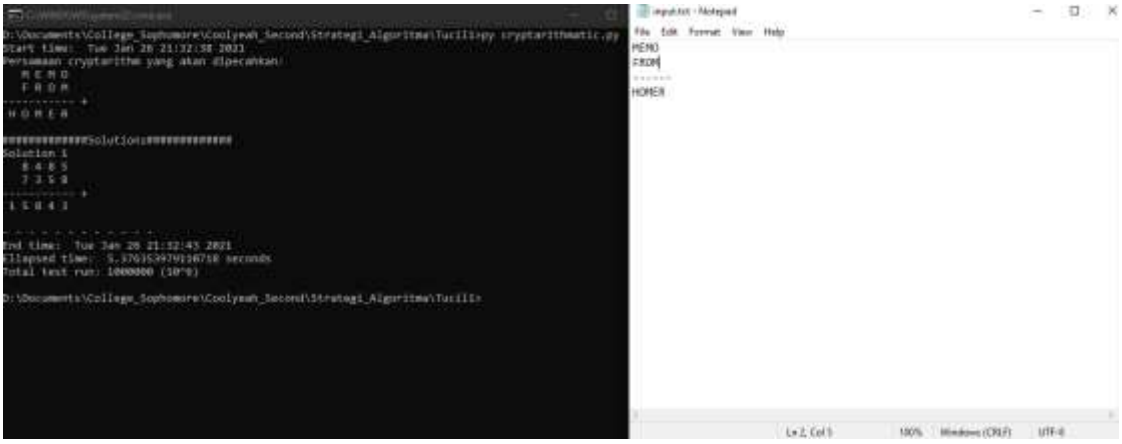
Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	v	
2. Program berhasil running	v	
3. Program dapat membaca file masukan dan menuliskan keluaran	v	
4. Solusi *Cryptarithmetic* hanya benar untuk persoalan *cryptarithmatic* dengan dua buah operan		v
5. Solusi *Cryptarithmetic* benar untuk persoalan *cryptarithmatic* untuk lebih dari dua buah operan	v	

Screenshoot Result

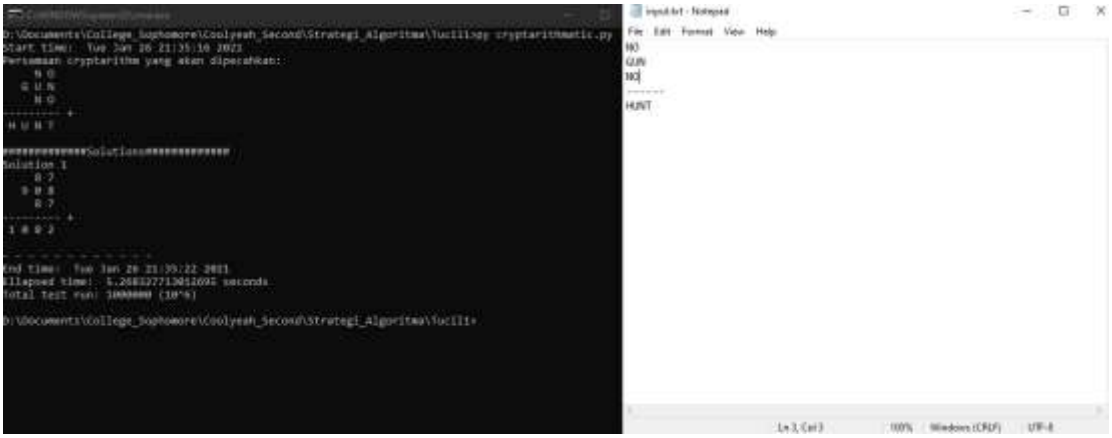
1) HERE + SHE = COMES



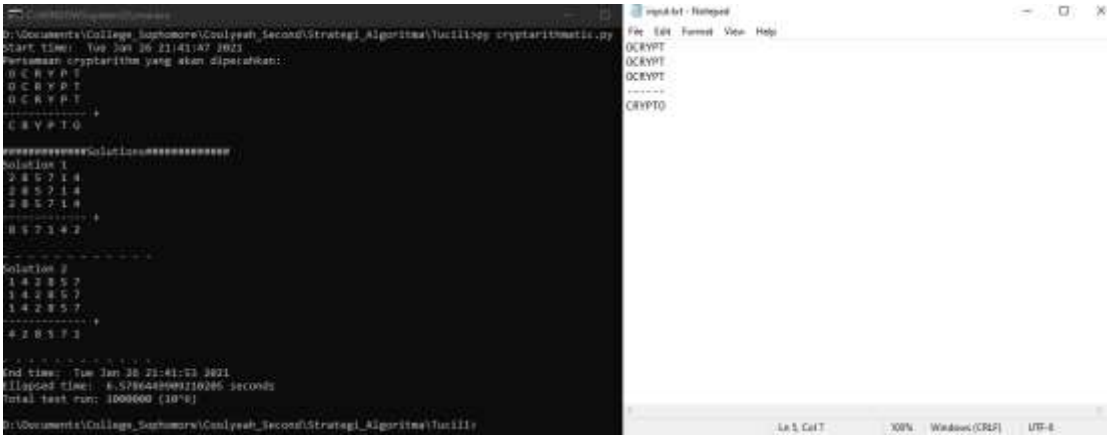
2) MEMO + FROM = HOMER



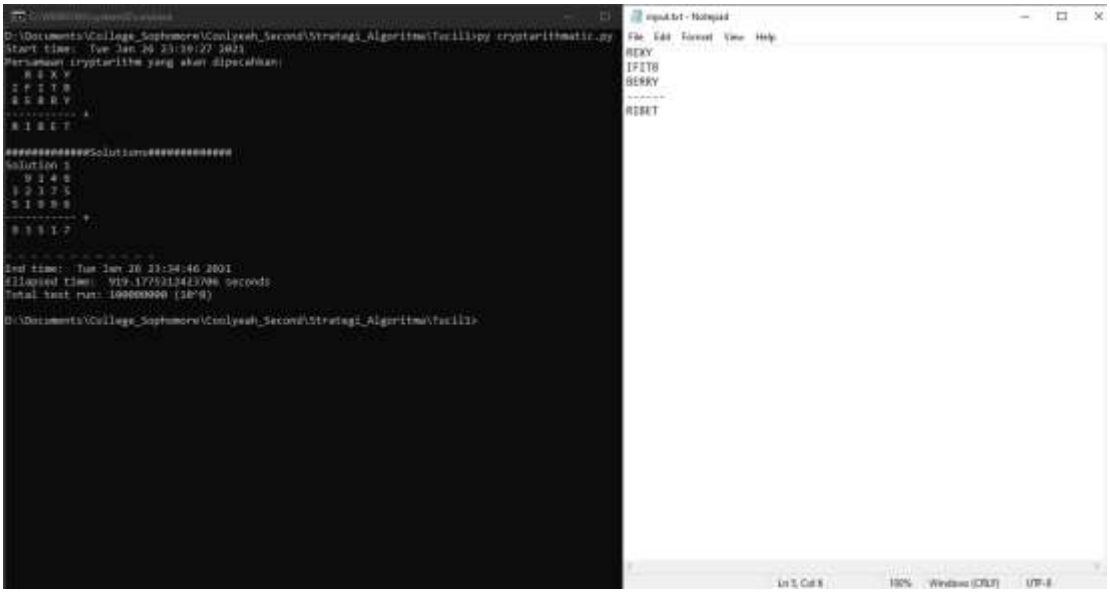
3) NO + GUN + NO = HUNT



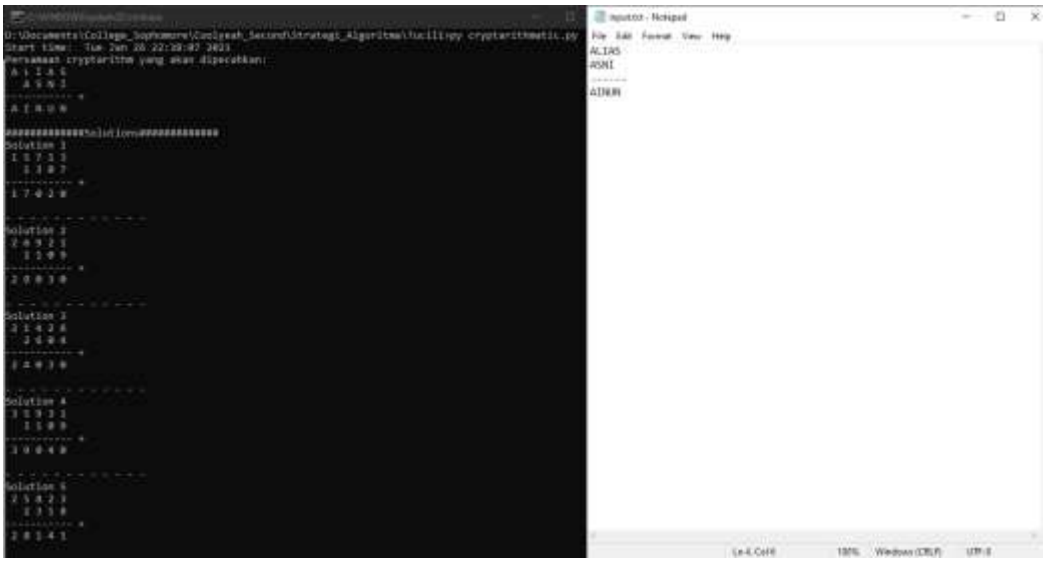
4) OCRYPT + OCRYPT + OCRYPT = CRYPTO

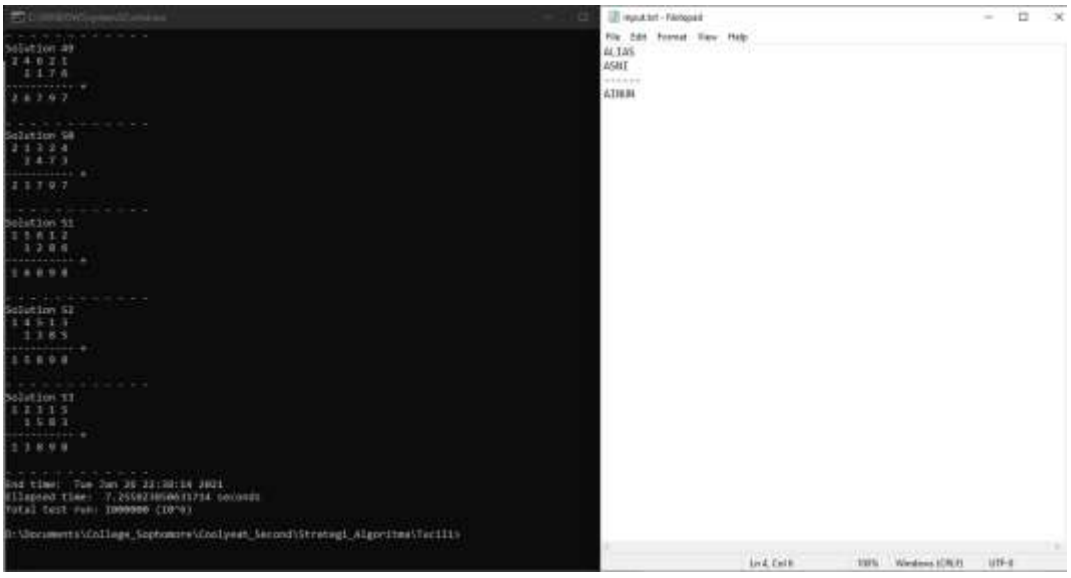


5) REXY + IFITB + BERRY = RIBET

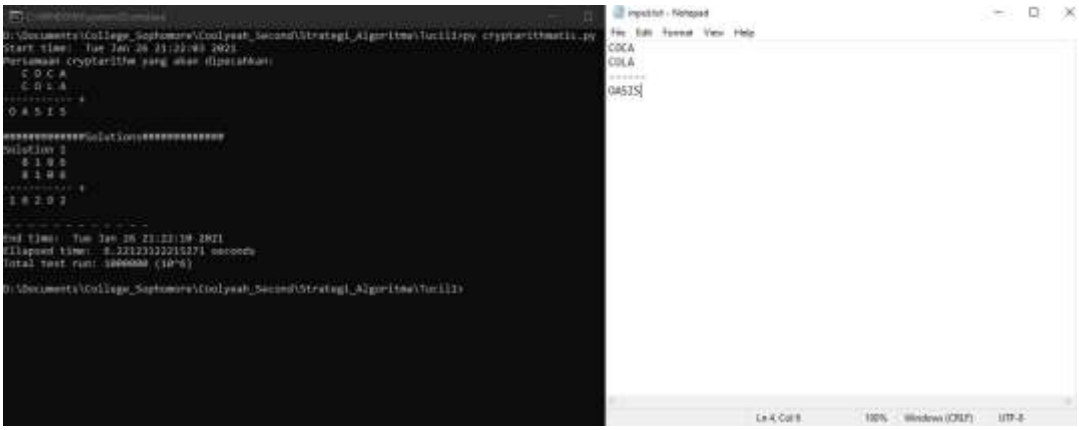


6) ALIAS + ASNI = AINUN

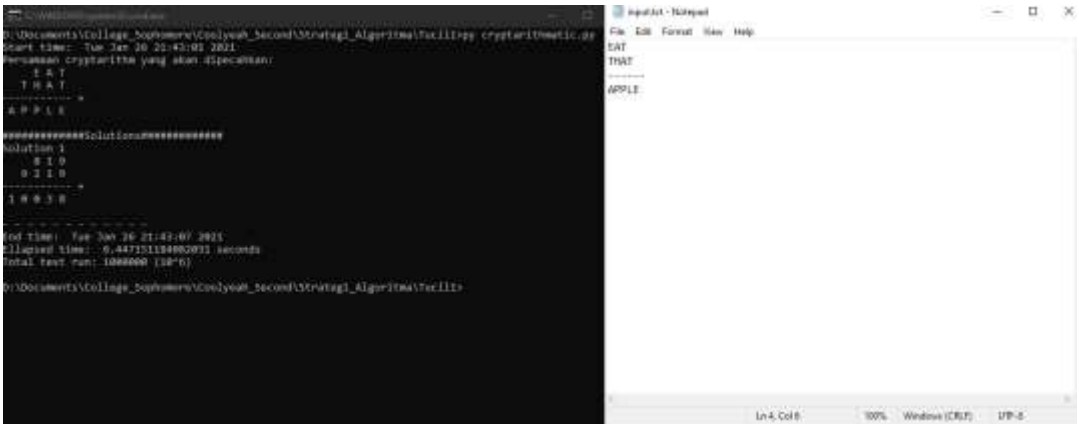




7) COCA + COLA = OASIS



8) EAT + THAT = APPLE



4. Source Code

```
import time

def InitializeVarDict(wordList):
  # Menginisialisasi dictionary yang berisi pasangan seluruh variabel pada equati
on dan nilainya
  temp = {}
  for word in wordList:
    for char in word:
      temp.update({char: 0})
  return temp

def InterpretWord (word, varDict):
  # Menentukan nilai dari suatu Word berdasarkan kombinasi nilai variabel pada va
rDict
  # word    : string
  # varDict : dictionary dengan pasangan <Var, Value>
  value = 0
  temp = varDict.copy()
  for currentChar in word:
    value *= 10
    value += temp.get(currentChar)
  return value

def IncrementVarConfiguration(varDict):
  # Meng-increment konfigurasi nilai variabel-variabel pada varDict
  tempVarDict = varDict.copy()
  for Var in tempVarDict:
    currentVal = tempVarDict.get(Var)
    if (currentVal == 9):
      tempVarDict.update({Var: 0})
    else:
      currentVal += 1
      tempVarDict.update({Var: currentVal})
      break
  return tempVarDict

def IsEquationValid(wordList, varDict):
  # Menentukan apakah suatu konfigurasi nilai variabel-variabel memenuhi equation
  tempVarDict = varDict.copy()
  tempSum = 0
  curWordVal = 0   # akan berguna saat menyimpan nilai dari word terakhir (hasil
penjumlahan dalam persamaan)
  for word in wordList:
```

```
    curWordVal = InterpretWord(word, tempVarDict)
    tempSum += curWordVal
  # curWordVal bernilai value dari word terakhir, tempSum bernilai hasil penjumla
han semua word dalam persamaan
  return (tempSum == 2*curWordVal)

def HasZeroAsInitial(wordInitials, varDict):
  # Menentukan apakah huruf inisial sebuah kata yang bernilai 0
  tempVarDict = varDict.copy()
  for Var in tempVarDict:
    if (tempVarDict.get(Var) == 0):
      if (Var in wordInitials):
        return True
  return False

def IsVarUnique(varDict):
  # Menentukan apakah nilai semua variabel unik
  tempVarDict = varDict.copy()
  unique = True
  tempValList = []
  for Var in tempVarDict:
    tempValList.append(tempVarDict.get(Var))
  # cek jika ada kemunculan lebih dari sekali
  return not(any([tempValList.count(element) > 1 for element in tempValList]))

def PrintEquation(wordList, varDict, interpret):
  # Mencetak persamaan dalam format penjumlahan ke bawah
  # Contoh:
  #    S E N D
  #    M O R E
  # ----------- +
  #  M O N E Y

  numWord = len(wordList)
  i = numWord + 1    # banyak baris
  maxWordLength = max(len(word) for word in wordList)
  j = 2*maxWordLength + 3         # banyak kolom

  nthWord = 0
  for row in range(i):
    if (row == numWord-1):  # baris garis
      for col in range(2*maxWordLength+1):
        print("-", end='')
      print(" ", end='')
      print("+", end='')
```

```
    else:                   # baris berisi word
      curWord = wordList[nthWord]
      curWordLength = len(curWord)
      nthChar = 0
      for col in range(maxWordLength):
        print(" ", end='')
        if (col >= maxWordLength - curWordLength):
          if (interpret):
            print(varDict.get(curWord[nthChar]), end = '')
          else:
            print(curWord[nthChar], end = '')
          nthChar += 1
        else:
          print(" ", end = '')
      nthWord += 1
    print()

def UpdateWordDict(wordDict, varDict):
  # Memperbarui value dari setiap word dalam wordDict
  tempVarDict = varDict.copy()
  tempWordDict = wordDict.copy()
  for word in tempWordDict:
    newVal = InterpretWord(word, tempVarDict)
    tempWordDict.update({word: newVal})
  return tempWordDict

# Kamus
# wordList    List "kata-kata" apa saja yang ada dalam persamaan
# wordDict    Dictionary berisi pasangan key-
value <Word, Value>, Value bergantung pada interpretasi setiap karakter pada varD
ict
# varDict     Dictionary berisi semua variabel pada persamaan dan kombinasi nilai
nya
# solutions   List of dictionary kombinasi pasangan <Var, Value> yang memenuhi pe
rseamaan


# Enumerasi "kata" dan inisial kata yang ada pada persamaan
wordList = []
wordInitials = []

# Pembacaan file
with open("input.txt") as file:
  for row in file:
    if (row[0] != '-'):
```

```
      if (row[-1] == '\n'):
        wordList.append(row[:-1])
      else:
        wordList.append(row)
file.close()

# Waktu Awal
start = time.time()
print("Start time: ", time.ctime(start))

# Membuat list huruf inisial
wordInitials = [word[0] for word in wordList if word[0] not in wordInitials]

# Inisialisasi Dictionary <Word,Value>
wordDict = {}
wordDict = wordDict.fromkeys(wordList, 0)

# Inisialisasi Dictionary <Var, Val>
varDict = InitializeVarDict(wordList)

# Persamaan
print("Persamaan cryptarithm yang akan dipecahkan:")
PrintEquation(wordList, varDict, False)
print()

# Iterasi untuk setiap kombinasi nilai variabel yang mungkin (Brute Force part)
# Karena menggunakan Brute Force, jumlah kemungkinan yang dicoba itu fixed

solutions = []
varLength = len(varDict.keys()) # menentukan banyaknya iterasi yang harus dilakuk
an

for _ in range(10**varLength):
  if (not(HasZeroAsInitial(wordInitials, varDict)) and IsVarUnique(varDict)):
    if (IsEquationValid(wordList, varDict)):
      solutions.append(varDict)
  varDict = IncrementVarConfiguration(varDict)


print("##############Solutions##############")
i = 1
#print(solutions)
for item in solutions:
  #print("item", item)
  print("Solution", i)
```

```
  PrintEquation(wordList, item, True)
  print()
  i += 1
  print("- - - - - - - - - - - -")
if not any(solutions):
  print("No solutions")

# Waktu Akhir
end = time.time()
print("End time: ", time.ctime(end))
print("Ellapsed time: ", (end-start), "seconds")

print("Total test run:", 10**varLength, "(10^" + str(varLength) + ")" )
```