TUGAS KECIL 1 IF 2211 STRATEGI ALGORITMA

PENYELESAIAN CRYPTARITHMETIC DENGAN ALGORITMA BRUTE FORCE

LAPORAN

Disusun untuk memenuhi tugas Strategi Algoritma

DISUSUN OLEH:

NAUFAL ALEXANDER SURYASUMIRAT 13519135

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

CEK LIST

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 *cryptarithmetic* dengan dua buah operand		√
5. Solusi *cryptarithmetic* benar untuk persoalan *cryptarithmetic* untuk lebih dari dua buah operand.	√	

BAB I

DESKRIPSI ALGORITMA BRUTE FORCE

Secara umum, algoritma *brute force* yang digunakan merupakan algoritma yang mencoba semua kemungkinan permutasi pasangan tiap karakter yang terdapat pada soal dengan angka 0 sampai dengan 9 agar mendapatkan hasil yang tepat untuk penjumlahan tiap kata tersebut dengan hasilnya.

Lebih rincinya, pertama-tama dibaca file yang dinamakan test.txt (terdapat 13 tes total yang berada pada directory folder test), kemudian tiap kata yang akan dijumlahkan dipisah menjadi elemen yang berbeda pada *List* (proses ini dilakukan pada fungsi *read_file* dengan satu parameter yaitu nama file yang akan dibaca). Kata hasil dipisah menjadi sebuah *String* yang terpisah dari *List* sebelumnya, kemudian dikembalikan menjadi *tuple List* dengan *String*.

Tuple List (berisi kata yang dijumlahkan) dan *String* (kata hasil penjumlahan) kemudian dijadikan parameter fungsi *cryptarithmetic_solve_another*. Fungsi ini menjalankan proses *brute force* dengan tahap sebagai berikut:

- Membuat *List* dengan nama *allwords* yang berisi tiap kata pada file yang dibaca dari tuple yang digunakan.
- Membuat *String* baru bernama *onlywords* yang merupakan gabungan seluruh kata tanpa spasi.
- Membuat *Set* dari *String onlywords* dengan isi tiap karakter tanpa duplikat (Misal AAB akan menjadi *Set* {‘A’, ‘B’}) yang kemudian dijadikan sebuah *String* yang bernama *unique_letters*.
- Jika *String* yang dibuat panjangnya melebihi 10 maka tidak dihitung karena tidak ada solusi untuk persoalan yang memiliki karakter lebih dari 10.
- Setelah itu, dibuat *List* baru bernama *first_letters* untuk menyimpan huruf pertama tiap kata pada soal agar menghindari mendapat solusi dengan huruf pertama bernilai 0.
- Kemudian, dimulailah *brute force* tiap kemungkinan untuk pasangan korespondensi *String unique_letters* dengan permutasi angka 1 sampai dengan 9 tanpa pengulangan.
- *Brute force* menggunakan for while yang mencoba angka 10 pangkat panjang *String unique_letters* dikurangi dengan 1 sampai dengan 10 pangkat panjang *String unique_letters.* Sebelum angka tersebut dicoba pada penjumlahan soal, terlebih dahulu diperiksa jika angka yang digenerasi merupakan angka yang unik (tanpa repetisi angka). Jika tidak unik, maka tidak dihitung dan dilanjutkan ke iterasi angka selanjutnya.
- *String unique_letters* dengan angka yang digenerasi dibuat menjadi dictionary yang merupakan korespondensi tiap huruf dengan angka yang dibuat, sebagai contoh, misal *String “ABCD”* dengan angka 1234, maka *Dictionary* yang dibuat adalah {‘A’: ‘1’, ‘B’: ‘2’, ‘C’: ‘3’, ‘D’: ‘4’}.

- Kemudian tiap kata yang telah dibaca pada soal dikonversikan ke angka yang berkoresponden pada *Dictionary* yang dibuat, kemudian diperiksa ketika dijumlahkan jika solusinya ditemukan atau tidak. Jika ditemukan, *Dictionary* yang telah dibuat dikembalikan dalam bentuk *Tuple* bersama dengan jumlah percobaan yang telah dilakukan.
- Hasil tersebut kemudian diperlihatkan pada *command line* beserta dengan panjang waktu yang digunakan dalam satuan detik.

BAB II

SOURCE PROGRAM DALAM BAHASA PYTHON

Fungsi **read_file** (membaca file dan mengembalikan soal dalam bentuk tuple *List* dan *String*). *List* berisi kata yang dijumlahkan, *String* merupakan kata hasil penjumlahan.

```
# Membaca file (parameter filename yaitu test.txt)
# Mengembalikan tuple yang berisi kata yang akan digunakan untuk Cryptarithmetic dalam bentuk tuple
# Tuple merupakan sebuah List dengan sebuah string
# Contoh tuple: (['SEND', 'MORE'], 'MONEY')
def read_file(filename):
    local_list = []
    # Seluruh kata (dalam file) pada awalnya dimasukkan ke dalam local_list
    with open ("../test/" + filename, 'r') as file:
        for line in file:
            for word in line.split():
                word_append = word.replace("+", "").replace("-", "")
                # Menghilangkan tanda + dan -
                local_list.append(word_append)
                # Memasukkan tiap kata (tiap line berisi satu kata pada file)
                # ke dalam local_list
    word_list = local_list[:-2]
        # Mengambil seluruh elemen dalam list local_list kecuali kedua terakhir
    result = local_list[len(local_list) - 1]
        # Menbamgil elemen terakhir list local_list
    return word_list, result # Mengembalikan dalam bentuk tuple ([List], 'string')
```

```python
# Menggabungkan seluruh elemen dalam list of kata
def join_words(list_of_words):
    to_return = ""
    for word in list_of_words:
        to_return += word
    return to_return

# Mengembalikan huruf pertama tiap kata dalam bentuk list
def get_firstletters(list_of_words):
    to_return = []
    for word in list_of_words:
        to_return.append(word[0])
    return to_return

# Mengkonversikan huruf ke angka korespondensinya sesuai dictionary
# Mengembalikan value ber-tipe integer
def convert(word, dictionary):
    to_return = ""
    for character in word:
        to_return += dictionary.get(character) # Mengambil value dari karakter pada dictionary
    return int(to_return)

# Menghitung jika hasil penjumlahan cryptarithmetic dengan skema brute force benar atau tidak
# Mengembalikan value True or False
def calculate(word_list, result, dictionary):
    calculated = 0
    for word in word_list:
        calculated += convert(word, dictionary)
    return calculated == convert(result, dictionary)
```

Fungsi **join_words** untuk menjadikan tiap elemen dalam *List* yang berisi *String*, menjadi satu *String*.

Fungsi **get_firstletters** untuk mencatat huruf pertama tiap kata, dan menyimpannya dalam bentuk *List*.

Fungsi **convert** untuk mengkonversi *String* (kata pada soal Cryptarithmetic) menjadi angka yang berkorespondensi sesuai *Dictionary* yang telah dibuat.

Fungsi **calculate** untuk menghitung jika penjumlahan kata (setelah dikonversi ke integer sesuai *Dictionary*) dan mengembalikan boolean yang menunjukkan jika solusi didapatkan atau tidak.

```python
# Algoritma kedua lebih cepat untuk mendapatkan hasilnya
def cryptarithmetic_solve_another(local_tuple):
    return_dictionary = {}
    return_count = 0
    words = local_tuple[0]
    result = local_tuple[1]
    allwords = []
    for element in words:
        allwords.append(element)
    allwords.append(result)
        # Menggabungkan tuple menjadi sebuah list untuk digunakan selanjutnya
    onlywords = join_words(allwords)
        # Menggabungkan tiap elemen dalam list allwords menjadi satu string
        # Misal : ['SEND', 'MORE', 'MONEY']
        # Menjadi : 'SENDMOREMONEY'
    unique_letters = join_words(set(onlywords))
        # Membuat string onlywords menjadi sebuah set dengan elemen unik dan kemudian
        # dijadikan string untuk dibuat dictionary
        # Misal : 'SENDMOREMONEY'
        # Menjadi : 'EODSYMNR'
    if len(unique_letters) > 10 :
        return None
    first_letters = get_firstletters(allwords)
    first_letters = list(dict.fromkeys(first_letters))
        # Menghilangkan duplikat karakter pertama tiap kata pada soal
        # Misal Soal : Send + More = Money
        # Menjadi : ['S', 'M']
        # Membuat list yang berisi karakter pertama tiap kata (agar tidak 0)
```

```python
for permute in range(10**(len(unique_letters) - 1), 10**len(unique_letters)):
    # Mencoba kemungkinan kombinasi angka yang mungkin sesuai banyaknya huruf yang unik pada persoalan
    # For loop ini mencoba untuk mereplikasi permutasi angka 0 sampai dengan 9 tanpa repetisi sesuai banyak huruf yang unik pada soal Cryptarithmetic
    if len(set(str(permute))) == len(unique_letters):
        # Mengecek jika integer yang di-generate merupakan angka yang unik tanpa repetisi angka
        return_count += 1
        zero_first_letter = False # Indikator jika huruf pertama pada sebuah kata bernilai 0
        dictionary = dict(zip(unique_letters, str(permute)))
            # Menghasilkan dictionary korespondensi tiap karakter dengan angka
            # hasil dari fungsi permutation, yaitu permutations_list
            # Misal dua string yaitu "STR" dan "123"
            # Maka hasil dictionarynya adalah: {'S' : '1', 'T' = '2', 'R' = '3'}
            # Tiap key dan tiap valuenya ber-tipe str
        for letter in first_letters:
            if int(dictionary.get(letter)) == 0:
                zero_first_letter = True
        if zero_first_letter == True:
            continue
        else:
            found_solution = calculate(words, result, dictionary)
                # Menghitung tiap hasil fungsi permutation untuk soal
                # Menghasilkan True or False
        if found_solution == True: # Jika solusi ditemukan
            return_dictionary = dictionary
            # return_dictionary menjadi dictionary yang berisi solusi dari soal Cryptarithmetic
            break
    else:
        continue
if found_solution == False:
    return None
else:
    return return_dictionary, return_count # Tuple (Dictionary, Integer)
    # Mengembalikan dalam bentuk Tuple seperti di atas ^
```

Fungsi **cryptarithmetic_solve_another**, algoritma *brute force* seperti yang telah dijelaskan pada bagian atas laporan.

```python
# Fungsi untuk output solusi dari soal Cryptarithmetic
def print_solution(words, result, dictionary, count, time):
    word_list = []
    for word in words:
        word_list.append(convert(word, dictionary))
    print("Solusi:")
    print(dictionary.__str__().replace("{", "").replace("}", "").replace("'", ""))
        # Format hasil output dictionary agar menghilangkan "{" dan "}"
    print(*word_list, sep = " + ", end = " = ")
    print(convert(result, dictionary))
    print("Jumlah percobaan :", count, "kali")
    print("Waktu percobaan  :", time, "detik")
```

```python
## Main Program

print("List nama file berada pada folder test")
print("List nama file : test, test2, test3, test4, test5, test6, test7, test8, test9, test10, test11, testdummy, testdummy2")
print("Silahkan input nama file Cryptarithmetic yang akan diselesaikan")

nama_file = input()
problem = read_file(nama_file + '.txt') # Pembacaan file
    # Untuk mengganti soal yang akan dikerjakan, jalankan program dan tuliskan nama file yang ingin diselesaikan
    # List nama file: test, test2, test3, test4, test5, test6, test7, test8, test9, test10, test11 (semua dalam format .txt)
    # testdummy.txt = Contoh soal tanpa solusi, testdummy2.txt = Contoh soal dengan lebih dari 10 karakter unik

print("Soal:")
words = problem[0]
result = problem[1]
print(*words, sep = " + ", end = " = ")
    # Meng-output list dengan separator tiap elemen " + " dan tanpa newline
print(result)
    # Hasil format : Kata1 + Kata2 + ... = KataHasil
print("")

starttime = time.time() # Waktu mulai
solution = cryptarithmetic_solve_another(problem)
endtime = time.time() # Waktu selesai
runtime = endtime - starttime # Menghitung runtime program yang dijalankan (Algoritma brute force)
    # Tidak termasuk membaca input dari file
if solution != None:
    print_solution(words, result, solution[0], solution[1], runtime)
else:
    print("Tidak ada solusi untuk persoalan Cryptarithmetic di atas")
```

Fungsi **print_solution** menampilkan solusi dari soal Cryptarithmetic. Disertakan dengan program utama di bawah fungsi tersebut.

BAB III

SCREENSHOT INPUT DAN OUTPUT

Seluruh file input ber-format ‘.txt’ dan mengikuti format yang telah diberikan pada spesifikasi, tiap file input dinamakan test.txt dengan angka yang berbeda untuk tiap soal (test.txt sampai dengan test11.txt), terdapat dua test untuk menguji jika program diberikan soal yang tidak memiliki solusi bernama testdummy.txt dan testdummy2.txt.

Seluruh screenshot tiap soal disertai input diikuti dengan outputnya pada *terminal VSCode*.

Soal 1: SEND + MORE = MONEY

```
 SEND
MORE+
------
MONEY
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\s
tima.py"
Soal:
SEND + MORE = MONEY

Solusi:
'Y': '2', 'R': '8', 'D': '7', 'M': '1', 'N': '6', 'O': '0', 'E': '5', 'S': '9'
9567 + 1085 = 10652
Jumlah percobaan : 338289 kali
Waktu percobaan  : 11.488458156585693 detik
```

Soal 2: THREE + THREE + TWO + TWO + ONE = ELEVEN

```
THREE
THREE
TWO
TWO
ONE+
-----
ELEVEN
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
THREE + THREE + TWO + TWO + ONE = ELEVEN

Solusi:
'H': '4', 'E': '1', 'O': '3', 'W': '0', 'V': '2', 'T': '8', 'N': '9', 'L': '7', 'R': '6'
84611 + 84611 + 803 + 803 + 391 = 171219
Jumlah percobaan : 1139136 kali
Waktu percobaan  : 174.49032998085022 detik
```

Soal 3: NUMBER + NUMBER = PUZZLE

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

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
NUMBER + NUMBER = PUZZLE

Solusi:
'M': '1', 'B': '6', 'P': '4', 'N': '2', 'U': '0', 'L': '7', 'E': '8', 'R': '9', 'Z': '3'
201689 + 201689 = 403378
Jumlah percobaan : 217505 kali
Waktu percobaan  : 36.41587042808533 detik
```

Soal 4: TILES + PUZZLES = PICTURE

```
TILES
PUZZLES+
--------
PICTURE
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
TILES + PUZZLES = PICTURE

Solusi:
'Z': '7', 'T': '9', 'L': '5', 'R': '8', 'E': '4', 'U': '0', 'P': '3', 'S': '2', 'I': '1', 'C': '6'
91542 + 3077542 = 3169084
Jumlah percobaan : 2529855 kali
Waktu percobaan  : 4386.285050153732 detik
```

Soal 5: CLOCK + TICK + TOCK = PLANET

```
CLOCK
TICK
TOCK+
-----
PLANET
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
CLOCK + TICK + TOCK = PLANET

Solusi:
'K': '2', 'L': '0', 'A': '4', 'N': '3', 'T': '6', 'O': '8', 'I': '5', 'C': '9', 'E': '7', 'P': '1'
90892 + 6592 + 6892 = 104376
Jumlah percobaan : 374004 kali
Waktu percobaan  : 678.0012831687927 detik
```

Soal 6: COCA + COLA = OASIS

```
COCA
COLA+
-----
OASIS
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
COCA + COLA = OASIS

Solusi:
'C': '8', 'O': '1', 'I': '9', 'L': '0', 'A': '6', 'S': '2'
8186 + 8106 = 16292
Jumlah percobaan : 109011 kali
Waktu percobaan  : 0.7357959747314453 detik
```

Soal 7: HERE + SHE = COMES

```
HERE
SHE+
----
COMES
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
HERE + SHE = COMES

Solusi:
'E': '4', 'S': '8', 'O': '0', 'C': '1', 'R': '5', 'M': '3', 'H': '9'
9454 + 894 = 10348
Jumlah percobaan : 228528 kali
Waktu percobaan  : 2.9700088500976562 detik
```

Soal 8: DOUBLE + DOUBLE + TOIL = TROUBLE

```
DOUBLE
DOUBLE
TOIL+
------
TROUBLE
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
DOUBLE + DOUBLE + TOIL = TROUBLE

Solusi:
'R': '5', 'O': '9', 'B': '0', 'E': '4', 'T': '1', 'L': '6', 'D': '7', 'U': '8', 'I': '3'
798064 + 798064 + 1936 = 1598064
Jumlah percobaan : 1776306 kali
Waktu percobaan  : 274.2717146873474 detik
```

Soal 9: NO + GUN + NO = HUNT

```
  NO
 GUN
 NO+
 ----
HUNT
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
NO + GUN + NO = HUNT

Solusi:
'N': '8', 'U': '0', 'T': '2', 'O': '7', 'G': '9', 'H': '1'
87 + 908 + 87 = 1082
Jumlah percobaan : 106226 kali
Waktu percobaan  : 0.8165683746337891 detik
```

Soal 10: CROSS + ROADS = DANGER

```
 CROSS
 ROADS+
 ------
DANGER
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
CROSS + ROADS = DANGER

Solusi:
'E': '4', 'S': '3', 'G': '7', 'N': '8', 'A': '5', 'C': '9', 'D': '1', 'R': '6', 'O': '2'
96233 + 62513 = 158746
Jumlah percobaan : 1238868 kali
Waktu percobaan  : 192.6882140636444 detik
```

Soal 11: MEMO + FROM = HOMER

```
 MEMO
 FROM+
 -----
HOMER
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
MEMO + FROM = HOMER

Solusi:
'O': '5', 'E': '4', 'M': '8', 'H': '1', 'R': '3', 'F': '7'
8485 + 7358 = 15843
Jumlah percobaan : 68504 kali
Waktu percobaan  : 0.4719388484954834 detik
```

Soal Tes 1: ABC + CBA = AB (Soal tanpa solusi)

```
ABC
CBA+
----
AB
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
ABC + CBA = AB

Tidak ada solusi untuk persoalan Cryptarithmetic di atas
```

Soal Tes 2: ABC…IJK + LMNO…UVW = XYZ (Soal lebih dari 10 huruf unik)

```
ABCDEFGHIJK
LMNOPQRSTUVW+
--------------
XYZ
```

```
C:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma>python -u "c:\Users\Noler\Documents\Tugas Kecil Strategi Algoritma\stima.py"
Soal:
ABCDEFGHIJK + LMNOPQRSTUVW = XYZ

Tidak ada solusi untuk persoalan Cryptarithmetic di atas
```

ALAMAT DRIVE

https://github.com/naufalsuryasumirat/Tucil1_13519135

Alamat github (Dibuat public setelah deadline)

https://drive.google.com/drive/folders/1naiQ6xns1XqoFHQJUZUIZqw8wJSv-2Wz?usp=sharing

Alamat drive alternatif (berisi zip)