PENYELESAIAN *CRYPTARITHMETIC* DENGAN ALGORITMA *BRUTE FORCE*

LAPORAN TUGAS KECIL

Diajukan Untuk Memenuhi Tugas IF 2211 Strategi Algoritma

Semester II 2020/2021

Disusun oleh

Reihan Andhika Putra (13519043)

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

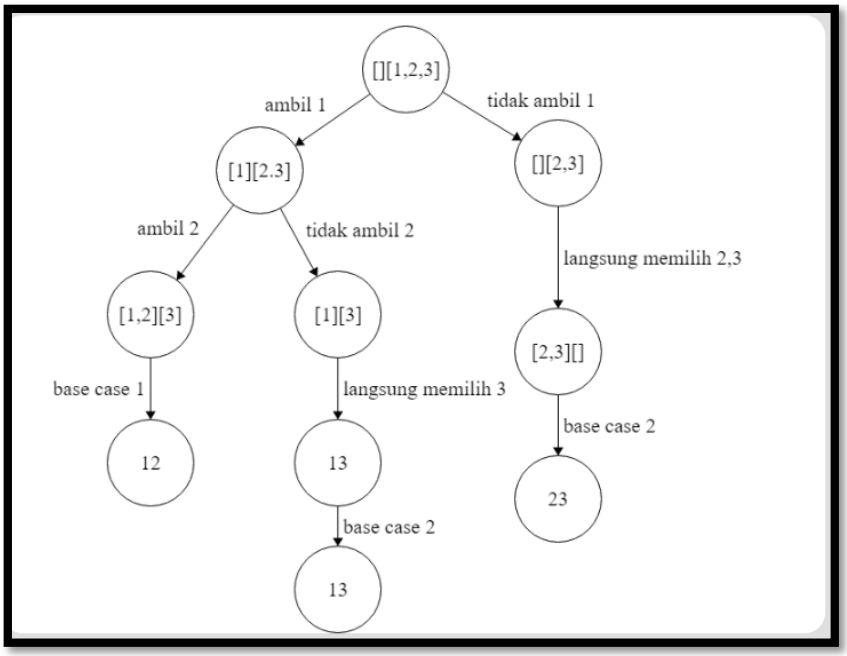
2020

BAB I
ALGORITMA BRUTE-FORCE

Algoritma yang saya gunakan untuk menyelesaikan permasalahan *cryptarithmetic* Brute Force – Exhaustive Search. Saya memilih algoritma ini karena solusi dari sebuah persoalan *cryptarithmetic* tidaklah unik sehingga tidak ada cara lain selain mengecek seluruh kombinasi angka yang mungkin. Tentunya kombinasi angka yang berulang dan leading zero diabaikan. Secara singkat, Brute Force Exhaustive Search adalah algortima penyelesaian persoalan kombinatorika dengan mencari semua kemungkinan solusi penyelesaian dan mencoba solusi tersebut satu-persatu. Pada kasus *cryptarithmetic* saya mencari semua permutasi angka dari 0-9 sebanyak 'n' dengan n adalah banyaknya huruf unik pada persoalan dan solusi. Berikut langkah-langkah dari algoritma dan penyelesaian yang saya buat di program saya:

1. Membaca input dari file "problem.txt".
2. Mulai mencatat waktu dimulai dengan library time.
3. Memisahkan operan dan solusi dari input yang dibaca. Kumpulan operan ditampung di list.
4. Mencari huruf unik dari list berisi operan dan string solusi, kemudian di tampung ke dalam list huruf unik.
5. Melakukan kombinasi dari angka 0-9 sebanyak 'n' angka dengan n adalah banyaknya huruf unik. Hasil kombinasi ditampung di list kombinasi.

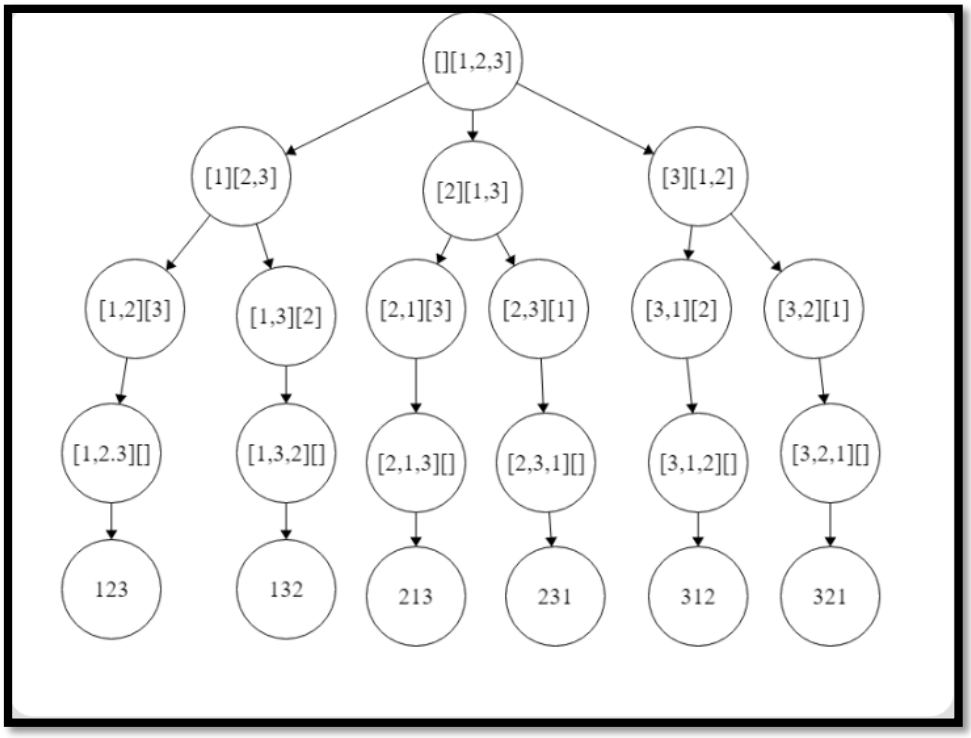
Algoritma Kombinasi



Algoritma untuk kombinasi yang saya gunakan menggunakan konsep rekursif. Konsepnya digambarkan seperti diagram diatas. Ada dua list yaitu list accumulator dan list angka yang tersedia untuk diambil. Untuk setiap index ada pilihan untuk mengambil angka di index tersebut atau tidak mengambil angka di index tersebut. Jika jumlah angka yang diambil (di dalam list accumulator) sudah sebanyak n angka yang diminta maka yang ada di list accumulator akan dimasukkan ke list kombinasi (base case 1). Jika jumlah angka yang belum terpilih sama dengan jumlah angka yang diminta maka yang ada di list accumulator akan dimasukkan ke list kombinasi (base case 2).

6. Melakukan permutasi untuk setiap kombinasi yang ada di list kombinasi. Hasil permutasi di tampung di list permutasi.

Algoritma Permutasi



Algoritma untuk permutasi yang saya gunakan menggunakan konsep rekursif. Konsepnya digambarkan seperti di atas. Ada dua list yaitu list accumulator dan list angka yang bisa di ambil. Untuk setiap tahap akan dipilih elemen ke-i lalu dimasukkan ke accumulator dan elemen tersebut dikeluarkan dari list angka yang bisa diambil. Hal tersebut dilakukan hingga angka yang bisa diambil habis.

7. Setiap hasil permutasi akan menjadi aturan binding terhadap list huruf unik. Misal permutasi (3.1,2) dan list char unik (a, b, c) maka a => 3, b => 1, c => 2. Untuk setiap hasil permutasi, huruf akan di binding dengan aturan permutasi tersebut (dikonversi ke integer dengan aturan tersebut).

8. Semua operan yang sudah dikonvert dijumlahkan. Lalu integer hasil penjumlahannya akan dikonvert kembali ke string dan dicek apakah string hasil konvertnya sama dengan string hasil.
9. Jika string nya sama maka akan diprint bentuk penjumlahannya, jumlah substitusi hingga sekarang, dan waktu yang dibutuhkan dari membaca file hingga menemukan solusi tersebut. Jika string nya berbeda maka lanjut mengecek permutasi selanjutnya hingga habis.

BAB II

SOURCE CODE PROGRAM

```
#----------------**********   Fungsi dan Prosedur   **********--------------#
#############     Kelompok Kerja Enkripsi, Dekripsi dan Formatting     #############
def decrypt_StringToInteger(stringCheck,selectedPermutation):
    # I.S string yang dicek dan aturan yang digunakan valid
    # F.S Mengirimkan string yang sudah dikonvert ke integer dengan aturan 'selected
Permutation'
    value_int = 0
    for char in stringCheck:
        value_int = value_int*10 + selectedPermutation[uniqueChar.index(char)]
    return value_int

def encrypt_IntegerToString(integerCheck,selectedPermutation):
    # I.S string yang dicek dan aturan yang digunakan valid
    # F.S Mengirimkan integer yang sudah dikonvert ke string dengan aturan 'selected
Permutation'. String ini akan digunakan untuk validasi.
    # Jika aturan 'selectedPermutation' tidak bisa menerjemahkan balik maka akan men
girimkan string yang salah.
    value_str = ""
    integerCheck = str(integerCheck)
    counter = 0
    for char in integerCheck:
        try:
            value_str += (uniqueChar[selectedPermutation.index(int(char))])
            counter += 1
        except:
            if counter < len(solution):
                if solution[counter] in uniqueChar :
                    break
                else :
                    value_str += solution[counter]
                    counter += 1
    return value_str

def formatInteger(integerResult):
    # I.S Integer yang diformat valid
    # F.S Mengirimkan integer yang sudah diformat menjadi string dan ditambahkan spa
si antar elemennya
    str_integerResult = str(integerResult)
    value_str = ""
    first = True
    for char in str_integerResult:
        if (first):
            value_str += char
            first = False
```

```
        else:
            value_str = value_str + ' ' + char
    return value_str
#############################################################################################

###################      Kelompok Kerja Permutasi dan Kombinasi     ################
def bruteForcePermutation(accumulator,nowList):
    # Melakukan permutasi secara brute force yang dioptimisasi dengan rekursif
    # I.S accumulator mulanya adalah list kosong dan nowLIst adalah list yang akan d
ipermutasi
    # F.S accumulator berisikan salah satu permutasi dari "nowList", nowList menjadi
 kosong dan "allPermutation" akan berisikan kumpulan dari accumulator

    # Base Case => jika list yang akan dipermutasi kosong maka accumulator ditambahk
an ke list permutasi
    if (nowList == []):
        allPermutation.append(accumulator)
    else:
        # Recursive Case => jika list yang akan dipermutasi belum kosong maka ambil
salah satu elemennya
        # Lakukan rekursif hingga list yang akan dipermutasi kosong
        for integer in nowList:
            newAccumulator = accumulator.copy()
            newAccumulator.append(integer)
            newList = nowList.copy()
            newList.remove(integer)
            bruteForcePermutation(newAccumulator,newList)

def initPermutation(uniqueInteger):
    # Merupakan fungsi pemicu permutasi
    bruteForcePermutation([], uniqueInteger)

def bruteForceCombination(accumulator,nowList,nowIndex,numOfElement):
    # Melakukan kombinasi secara brute force yang dioptimisasi dengan rekursif
    # I.S Awalnya accumulator adalah list kosong, nowList berisi list yang akan diko
mbinasi, nowIndex adalah 0 , dan numOfElemen jumlah elemen yang akan di kombinasi
    # F.S Saat prosedur ini berakhir maka "allCombination" akan berisikan kombinasi
dari "nowList" sebanyak "numOfElement" ( kombinasi didapat dari accumulator )

    # Base Case 1 => jika mengkombinasi 0 elemen maka tambahkan accumulator
    if (numOfElement==0):
        allCombination.append(accumulator)
    # Base Case 2 => jika jumlah angka yang mau dikombinasi == angka yang tersisa un
tuk dipilih maka angka sisanya langsung ditambahkan
    elif (nowIndex == len(nowList)-numOfElement):
        for i in range(nowIndex,len(nowList)):
            accumulator.append(nowList[i])
        allCombination.append(accumulator)
```

```
    # Recursion Case => Saat ada di index ke-
0 anda dapat memilih untuk mengambil angka tersebut atau tidak dst. Sehingga ada 2 r
ekursif seperti pohon biner
    else:
        newAccumulator1 = accumulator.copy()
        newAccumulator2 = accumulator.copy()
        newAccumulator1.append(nowList[nowIndex])
        bruteForceCombination(newAccumulator1,nowList,nowIndex+1,numOfElement-1)
        bruteForceCombination(newAccumulator2,nowList,nowIndex+1,numOfElement)

def initCombination(uniqueInteger, numOfElement):
    # Merupakan fungsi pemicu kombinasi
    bruteForceCombination([],uniqueInteger,0,numOfElement)
#############################################################################################
#-----------**********-----------------**********-----------#

#------------**********   Main Program   **********------------#
# Important Notes
print("Waktu eksekusi dan jumlah pengecekan adalah untuk tiap solusi")

# Import library
import time

# Deklarasi Variabel-Variabel Penting
problems = []           # Kumpulan string operan
solution = []           # String hasil
uniqueChar = []         # Char unik dari solusi dan operan
allPermutation = []     # Semua permutasi sebanyak char unik dari 10 angka
allCombination = []     # Semua kombinasi sebanyak char unik dari 10 angka

#  Membaca input dari file txt dan menulisnya ke layar
filename = input("Masukkan nama file (tanpa ekstensi): ")
inputs = open('test/'+ filename +'.txt','r').read().split('\n')
print("Soal")
for input in inputs:
    print(input)
print("")

# Mencatat waktu dimulai setelah membaca file
startTime = time.perf_counter()

# Parsing input ( Membagi mana yang soal(operan) dan mana yang solusi (hasil penjuml
ahan) )
for input in inputs:
    if '-' in input:
        break
    else:
        if '+' in input:
```

```
            input = input.replace('+','')
        input = input.replace(' ', '')
        problems.append(input)
BanyakProblem = len(problems)
solution = inputs[len(inputs)-1].replace(' ', '')

# Parsing input (Menambahkan semua char unik ke dalam list "uniqueChar")
for problem in problems:
    for char in problem:
        if char not in uniqueChar:
            uniqueChar.append(char)
for char in solution:
    if char not in uniqueChar:
        uniqueChar.append(char)

# Mencari semua kemungkinan permutasi dengan mengkombinasikan huruf sebanyak uniqueC
har dan mempermutasikannya satu-satu
initCombination([1,2,3,4,5,6,7,8,9,0],len(uniqueChar))
for combination in allCombination:
    initPermutation(combination)

# Algoritma Brute Force dengan Pengecekan Satu-Satu
print("Jawaban")
jumlahPengecekan = 0
solusiKe = 0
for permutation in allPermutation:
    jumlahPengecekan += 1
    integerResult = 0
    for problem in problems:
        integerResult += decrypt_StringToInteger(problem,permutation)
    # Pengecekan apakah dengan aturan yang sekarang integer hasil penjumlahan operan
 akan menjadi string solusi jika dikonvert
    stringResult = encrypt_IntegerToString(integerResult,permutation)
    if (stringResult == solution):
        solusiKe += 1
        print("Solusi ke-",solusiKe)
        print("Jumlah substitusi :", jumlahPengecekan, "substitusi")
        print("Waktu eksekusi : ", time.perf_counter() - startTime, " detik")
        formattedResult = formatInteger(integerResult)
        for i in range(BanyakProblem):
            formattedInteger = formatInteger(decrypt_StringToInteger(problems[i],per
mutation))
            if (i == BanyakProblem-1):
                print("+ "+" "*(len(formattedResult)-
len(formattedInteger))+ formattedInteger)
            else:
                print("  "+" "*(len(formattedResult)-
len(formattedInteger))+formattedInteger)
```

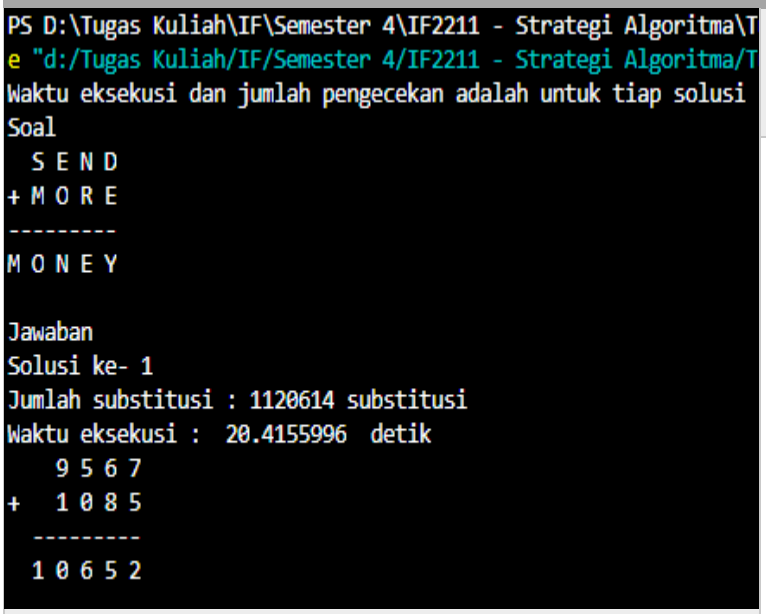
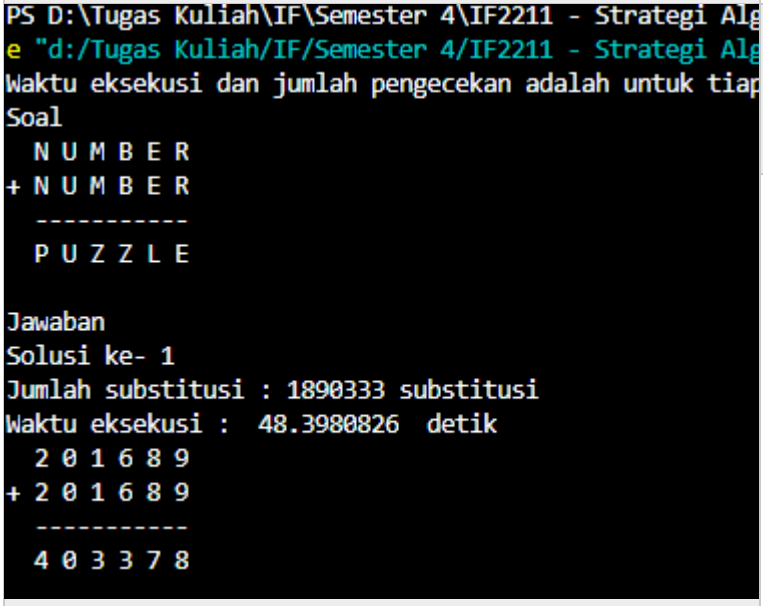
```
        print('   ' + '-'* len(formattedResult))
        print("   "+ formattedResult)
        print("")

if (solusiKe == 0):
    print("Tidak ada solusi yang memenuhi")
#-----------********************----------------********************----------#
```

BAB III

SKRINSHUT DAN PENJELASAN

Note : Pada saat melakukan percobaan berulang kadang terdapat perbedaan waktu eksekusi dari 2-8 detik lebih cepat/lebih lama (banyakan lebih cepat tapi). Waktu yang saya skrinshut dibawah bukanlah yang tecepat maupun terlama, namun yang pertama saya skrinshut.

Skrinshut	Penjelasan
	**Huruf yang berkorelasi** S => 9 | E => 5 | N => 6 | D => 7 M => 1 | O => 0 | R => 8 | Y => 2 **Jumlah Solusi:** 1 buah **Hasil Tebakan:** Sudah benar. **Waktu Eksekusi:** Terdapat 8 huruf yang unik, waktu eksekusi 20.4 detik cukup cepat untuk menyelesaikan soal ini. (Jika dibanding pada testcase lain yang ada disini)
	**Huruf yang berkorelasi** N => 2 | U => 0 | M => 1 | B => 6 E => 8 | R => 9 | P => 4 | Z => 3 L => 7 **Jumlah Solusi:** 1 buah **Hasil Tebakan:** Sudah benar. **Waktu Eksekusi:** Terdapat 9 huruf yang unik, waktu eksekusi 48.39 detik normal untuk menyelesaikan soal ini (Jika dibanding pada testcase lain yang ada disini). Hal ini terjadi karena terdapat 9 buah permutasi angka yang harus dicek

```
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi
e "d:/Tugas Kuliah/IF/Semester 4/IF2211 - Strategi
Waktu eksekusi dan jumlah pengecekan adalah untuk
Soal
     T I L E S
+ P U Z Z L E S
  ------------
  P I C T U R E

Jawaban
Solusi ke- 1
Jumlah substitusi : 2919621 substitusi
Waktu eksekusi :  68.57192  detik
     9 1 5 4 2
+ 3 0 7 7 5 4 2
  ------------
  3 1 6 9 0 8 4
```

Huruf yang berkorelasi
T => 9 | I => 1 | L => 5 | E => 4
S => 2 | P => 3 | U => 0 | Z => 7
C => 6 | R => 8

Jumlah Solusi:
1 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 10 huruf yang unik, waktu eksekusi 68.57 detik cukup untuk menyelesaikan soal ini (Jika dibanding pada testcase lain yang ada disini). Hal ini terjadi karena terdapat 10 buah permutasi angka yang harus dicek

```
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Alg
e "d:/Tugas Kuliah/IF/Semester 4/IF2211 - Strategi Alg
Waktu eksekusi dan jumlah pengecekan adalah untuk tiap
Soal
    C L O C K
      T I C K
+     T O C K
  ----------
  P L A N E T

Jawaban
Solusi ke- 1
Jumlah substitusi : 3262155 substitusi
Waktu eksekusi :  69.3760679  detik
    9 0 8 9 2
      6 5 9 2
+     6 8 9 2
  ----------
  1 0 4 3 7 6
```

Huruf yang berkorelasi
C => 9 | L => 0 | O => 8 | K => 2
T => 6 | I => 5 | P => 1 | A => 4
N => 3 | E => 7

Jumlah Solusi:
1 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 10 huruf yang unik, waktu eksekusi 69.37 detik cukup untuk menyelesaikan soal ini (Jika dibanding pada testcase lain yang ada disini). Hal ini terjadi karena terdapat 10 buah permutasi angka yang harus dicek Program dapat menyelesaikan kasus diatas 2 operan.

```
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Al
e "d:/Tugas Kuliah/IF/Semester 4/IF2211 - Strategi Al
Waktu eksekusi dan jumlah pengecekan adalah untuk tia
Soal
    C O L A
+   C O L A
  ---------
  O A S I S

Jawaban
Solusi ke- 1
Jumlah substitusi : 18609 substitusi
Waktu eksekusi :  0.725711  detik
    8 1 3 6
+   8 1 3 6
  ---------
  1 6 2 7 2

Solusi ke- 2
Jumlah substitusi : 35049 substitusi
Waktu eksekusi :  0.9444798  detik
    8 1 4 6
+   8 1 4 6
  ---------
  1 6 2 9 2
```

Huruf yang berkorelasi
C => 9 | O => 0 | L => 8 | A => 2
S => 6 | I => 5

Jumlah Solusi:
2 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 6 huruf yang unik, waktu eksekusi dibawah satu detik untuk kedua kasus ini sangat cepat (Jika dibanding pada testcase lain yang ada disini). Hal ini terjadi karena jumlah huruf yang dicek jauh lebih sedikit

```
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Alg
e "d:/Tugas Kuliah/IF/Semester 4/IF2211 - Strategi Alg
Waktu eksekusi dan jumlah pengecekan adalah untuk tiap
Soal
    H E R E
+     S H E
  ---------
  C O M E S

Jawaban
Solusi ke- 1
Jumlah substitusi : 331502 substitusi
Waktu eksekusi :  7.0923991  detik
    9 4 5 4
+     8 9 4
  ---------
  1 0 3 4 8
```

Huruf yang berkorelasi
H => 9 | E => 0 | R => 8 | S => 2
C => 6 | O => 5 | M =>

Jumlah Solusi:
1 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 7 huruf yang unik, waktu eksekusi 7 detik untuk kedua kasus ini cukup cepat (Jika dibanding pada testcase lain yang ada disini).

```
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Alg
e "d:/Tugas Kuliah/IF/Semester 4/IF2211 - Strategi Alg
Waktu eksekusi dan jumlah pengecekan adalah untuk tiap
Soal
   T H R E E
   T H R E E
      T W O
      T W O
+     O N E
  ----------
  E L E V E N

Jawaban
Solusi ke- 1
Jumlah substitusi : 2073708 substitusi
Waktu eksekusi :  63.8793052  detik
    8 4 6 1 1
    8 4 6 1 1
        8 0 3
        8 0 3
+       3 9 1
  -----------
  1 7 1 2 1 9
```

Huruf yang berkorelasi
T => 8 | H => 4 | R => 6 | E => 1
W => 0 | O => 3 | N => 9 | L => 7
V => 2

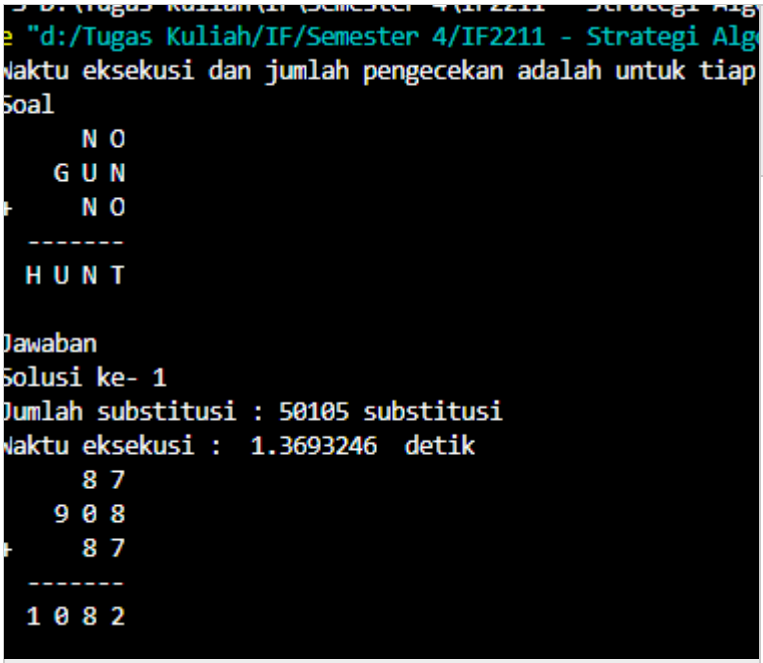
Jumlah Solusi:
1 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 9 huruf yang unik, waktu eksekusi 63.8 detik untuk kasus ini cukup lama (Jika dibanding pada testcase lain yang ada disini). Hal ini terjadi karena terdapat 9 buah permutasi angka yang harus dicek dan 5 operan yang harus dijumlahkan. Program dapat menyelesaikan kasus diatas 2 operan.

```
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Al
e "d:/Tugas Kuliah/IF/Semester 4/IF2211 - Strategi Al
Waktu eksekusi dan jumlah pengecekan adalah untuk tia
Soal
    D O U B L E
    D O U B L E
+       T O I L
  -------------
  T R O U B L E

Jawaban
Solusi ke- 1
Jumlah substitusi : 3139189 substitusi
Waktu eksekusi :  79.62741390000001  detik
    7 9 8 0 6 4
    7 9 8 0 6 4
+       1 9 3 6
  -------------
  1 5 9 8 0 6 4
```

Huruf yang berkorelasi
D => 7 | O => 9 | U => 8 | B => 0
L => 6 | E => 4 | T => 1 | I => 3
R => 5

Jumlah Solusi:
1 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 9 huruf yang unik, waktu eksekusi 79.6 detik untuk kasus ini cukup lama (Jika dibanding pada testcase lain yang ada disini). Hal ini terjadi karena terdapat 9 buah permutasi angka yang harus dicek dan 3 operan yang dijumlahkan Program dapat menyelesaikan kasus diatas 2 operan.



Huruf yang berkorelasi
N => 8 | O => 7 | G => 9 | U => 0
H => 1 | T => 2

Jumlah Solusi:
1 buah
Hasil Tebakan:
Sudah benar.
Waktu Eksekusi:
Terdapat 6 huruf yang unik, waktu eksekusi 1.39 detik untuk kasus ini sangat cepat (Jika dibanding pada testcase lain yang ada disini). Program dapat menyelesaikan kasus diatas 2 operan.

BAB IV

LAMPIRAN

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

Link Drive
https://drive.google.com/drive/folders/1dn29tPtcQHrIrWUzwtST9DKmIBUx8bwk?usp=sharing