

PENYELESAIAN CRYPTARITHMETIC DENGAN ALGORITMA BRUTE FORCE

LAPORAN TUGAS KECIL 1

Diajukan sebagai salah satu Tugas Kecil 1

IF2211 Strategi Algoritma Semester II tahun 2020/2021

Oleh:

Mgs. Tabrani (13519122)



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG**

BAGIAN I

ALGORITMA *BRUTE FORCE*

A. Definisi Algoritma *Brute Force*

Algoritma *Brute Force* adalah algoritma yang menggunakan pendekatan yang (*straightforward*) langsung untuk memecahkan suatu persoalan. Algoritma *Brute Force* memecahkan persoalan dengan sangat sederhana, langsung, dan jelas caranya (*obvious way*). Biasanya algoritma *Brute Force* didasarkan pada pernyataan pada persoalan (*problem statement*); definisi/konsep yang dilibatkan. Berikut ini beberapa contoh algoritma *Brute Force* dan langkah-langkah penyelesaiannya.

a. Mencari Elemen Terbesar atau Terkecil

Apabila diberikan sebuah senarai atau *list* yang berisi n buah bilangan bulat (a_1, a_2, \dots, a_n). Algoritma *Brute Force* akan membandingkan setiap elemen senarai atau *list* mulai dari a_1 sampai a_n untuk menemukan elemen terbesar atau terkecil.

b. Pencarian Beruntun (*Sequential Search*)

Apabila diberikan senarai yang berisi n buah bilangan bulat (a_1, a_2, \dots, a_n). Algoritma *Brute Force* akan membandingkan setiap elemen dengan x . Pencarian selesai jika x ditemukan atau seluruh elemen senarai sudah habis diperiksa.

Masih banyak algoritma *Brute Force* yang umum digunakan. Algoritma *Brute Force* ini umumnya tidak “cerdas” dan tidak mangkus, karena ia membutuhkan volume komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Algoritma ini lebih cocok untuk persoalan yang masukannya kecil. Meskipun bukan metode *problem solving* yang mangkus, hampir semua persoalan dapat diselesaikan dengan algoritma *Brute Force*.

B. Algoritma *Brute Force* dalam Menyelesaikan *Cryptarithmic*

Algoritma *Brute Force* dapat digunakan untuk menyelesaikan permasalahan *cryptarithmic*. *Cryptarithmic* adalah sebuah *puzzle* penjumlahan di dalam Matematika, yaitu angka diganti dengan huruf. Setiap angka dipresentasikan dengan huruf yang berbeda.

Penyelesaian *cryptarithmic* dengan pendekatan algoritma *Brute Force* ini memanfaatkan permutasi angka-angka dari 0 sampai 9, kemudian mengecek kombinasi angka-angka tersebut apakah sesuai dengan persoalan *cryptarithmic* yang disajikan. Pertama, program akan membaca masukan *file* yang berisi persoalan *cryptarithmic*, dan menyimpan

setiap operan dan hasil penjumlahan ke dalam variabel. Kemudian program akan menentukan huruf-huruf yang dipakai pada persoalan tersebut dan disimpan ke dalam array. Masing-masing elemen pada array tersebut akan berisi dua nilai, yaitu huruf dan angka yang direpresentasikan oleh huruf tersebut. Lalu, program akan men-*generate* permutasi angka dari 0 sampai 9 dengan urutan tertentu, dan urutan angka-angka tersebut akan disimpan di array yang berisi huruf-huruf yang digunakan dalam persoalan. Namun, apabila kombinasi tersebut diawali dengan 0, kombinasi tidak akan diperiksa. Lalu program akan memeriksa kombinasi angka-angka tersebut dengan persoalan yang diberikan, apabila telah ketemu solusi yang sesuai, program akan berhenti mencari permutasi angka lainnya dan program akan diterminasi menampilkan solusi, waktu eksekusi, dan jumlah kombinasi tes.

BAGIAN II

SOURCE PROGRAM

Program ini dibuat dengan menggunakan bahasa Python. *Source code* dari program ini terdiri dari 3 *file* .py, yaitu main.py yang digunakan sebagai *file* utama, function.py yang berisi fungsi dan prosedur yang digunakan dalam program, dan input.py berisi perintah masukan dan variabel-variabel global yang digunakan program. Berikut ini *source code* masing-masing *file*.

a. main.py

```
#Import all function
from function import *

#Import time library to count time execution
import timeit
start = timeit.default_timer()

#ALGORITMA
putLetters()      #Put all letters used in letterUsed array
initNumber()      #Initialize all element in letterUsed with 0
heapPermutation() #Generate the combination and check the answer

#Counting time execution
stop = timeit.default_timer()
timeExecution = stop - start

#Display the solution
print(21*" ", "SOLUTION")
if(not found[0]):
    print("No solution")
else:
    displaySolution()

#Show the time execution
print(50*"=")
print(18*" ", "TIME EXECUTION")
print("Time Execution =", timeExecution, "second")
```

```

#Show total test
print(50*"=")
print(20*" ", "TOTAL TEST")
print("The total test =", countTest[0])

```

b. function.py

```

#Import variables from input.py
from input import operand,result,letterUsed,countTest,found

#Put all letter used in file to letterUsed array
def putLetters():
    for element in operand+[result]:
        for letter in element:
            isUnique = True
            for j in range(len(letterUsed)):
                if(letter == letterUsed[j]):
                    isUnique = False
            if(isUnique and letter != " "):
                letterUsed.append(letter)

#Add each element at letterUsed array with 0
def initNumber():
    for i in range(len(letterUsed)):
        letterUsed[i] += "0"

#Change a number in letterUsed element with particular number
def changeNumber(letter, number):
    letter = list(letter)
    letter[1] = str(number)
    return letter[0]+letter[1]

#Check the solution by seeing letterUsed array
def checkAnswer(letterUsed):
    operandAnswer = [" " for i in range(len(operand))]
    resultAnswer = " "
    for i in range(len(operand)):

```

```

        for j in range(len(operand[i])):
            for usedLetter in letterUsed:
                if(operand[i][j] == usedLetter[0]):
                    operandAnswer[i] += usedLetter[1]

    for i in range(len(result)):
        for usedLetter in letterUsed:
            if(result[i] == usedLetter[0]):
                resultAnswer += usedLetter[1]

    isValid = True
    for answer in operandAnswer:
        if(answer[0] == "0"):
            isValid = False

    if(resultAnswer[0] == "0"):
        isValid = False

    if(not isValid):
        countTest[0] -= 1

    sum = 0
    for number in operandAnswer:
        sum += int(number)

    return (sum == int(resultAnswer) and isValid)

#Using heap permutation to generate permutation of possibility
def heapPermutation():
    n = 10
    combination = list(range(10))
    cons = [0 for i in range(n)]
    i = 0
    while(i < n and not found[0]):
        if(cons[i] < i):
            if (i % 2 == 0):
                combination[0],combination[i] =
combination[i],combination[0]
            else:
                combination[cons[i]],combination[i] =
combination[i],combination[cons[i]]
            if(combination[0] != 0):

```

```

        for k in range(len(letterUsed)):
            letterUsed[k] =
changeNumber(letterUsed[k],combination[k])
            countTest[0] += 1
            if(checkAnswer(letterUsed)):
                found[0] = True
            cons[i] += 1
            i = 0
    else:
        cons[i] = 0
        i += 1

#Display the solution
def displaySolution():
    for operands in operand[:len(operand)-1]:
        for letter in operands:
            for number in letterUsed:
                if(number[0] == letter):
                    print(number[1], end="")
            if(letter == " "):
                print(" ",end="")
        print()
    for letter in operand[len(operand)-1]:
        for number in letterUsed:
            if(number[0] == letter):
                print(number[1], end="")
        if(letter == " "):
            print(" ", end="")
    print("+")
    print((len(result)+1)*"-")
    for letter in result:
        for number in letterUsed:
            if(letter == number[0]):
                print(number[1],end="")
    print()

```

c. input.py

```
#Input and read file
inputFile = input("Input file: ")
inputFile = open(inputFile, "r")
operand = []

#Print input problem
print(50*"=")
print(21*" ", "PROBLEM")
read = inputFile.readline()
print(read, end="")
while(read[0] != "-"):
    operand.append(read)
    read = inputFile.readline()
    print(read, end="")
result = inputFile.readline()
print(result)
print(50*"=")

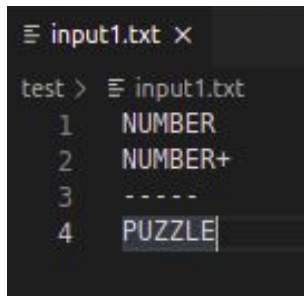
#Delete new line and + from operand and result
for i in range(len(operand)-1):
    operand[i] = operand[i][:len(operand[i])-1]
operand[len(operand)-1] =
operand[len(operand)-1][:len(operand[len(operand)-2])]

#Initial variable
letterUsed = [] #For storing all the letter used
countTest = [0] #For counting total combination checked
found = [False] #For determining weather there is solution
```


BAGIAN III

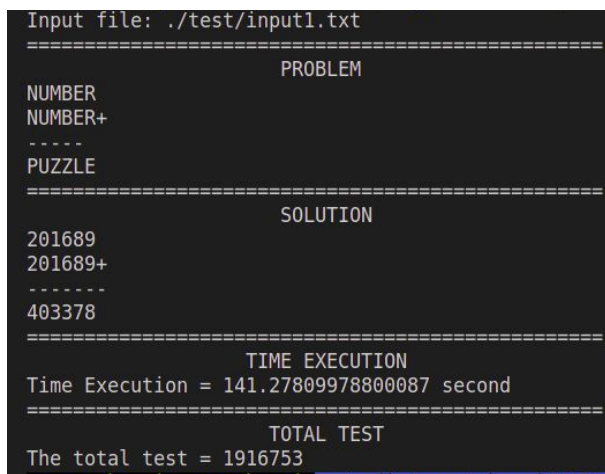
SKRINSHUT

a. Input 1



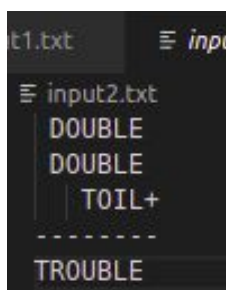
```
input1.txt X
test > input1.txt
1  NUMBER
2  NUMBER+
3  -----
4  PUZZLE
```

b. Output 1



```
Input file: ./test/input1.txt
=====
                                PROBLEM
NUMBER
NUMBER+
-----
PUZZLE
=====
                                SOLUTION
201689
201689+
-----
403378
=====
                                TIME EXECUTION
Time Execution = 141.27809978800087 second
=====
                                TOTAL TEST
The total test = 1916753
```

c. Input 2



```
t1.txt  input2.txt
input2.txt
DOUBLE
DOUBLE
TOIL+
-----
TROUBLE
```

d. Output 2

[illegible]

e. Input 3

```
input1.txt  ≡ i
≡ input3.txt
  |
  |  TILES
  | PUZZLES+
  | -----
  | PICTURE|
```

f. Output 3

```
stracegi -t/gz/./tmpd/cagaz/tact/cryptan1emmette/shc
Input file: ./test/input3.txt
=====
PROBLEM
TILES
PUZZLES+
-----
PICTURE
=====
SOLUTION
91542
3077542+
-----
3169084
=====
TIME EXECUTION
Time Execution = 44.14924651899946 second
=====
TOTAL TEST
The total test = 499375
```

g. Input 4

```
out1.txt  ≡ in
> ≡ input4.txt
  NO
  GUN
  NO+
  ----
  HUNT
```

h. Output 4

```
=====
                                PROBLEM
  NO
  GUN
  NO+
  ----
  HUNT
=====
                                SOLUTION
  87
  908
  87+
  ----
  1082
=====
                                TIME EXECUTION
Time Execution = 70.00311281199993 second
=====
                                TOTAL TEST
The total test = 1003301
```

i. Input 5

```
t5.txt  ×
≡ input5.txt
  CLOCK
  TICK
  TOCK+
  -----
  PLANET
```

j. Output 5

```
Input file: ./test/input5.txt
=====
                                PROBLEM
CLOCK
TICK
TOCK+
-----
PLANET
=====
                                SOLUTION
90892
6592
6892+
-----
104376
=====
                                TIME EXECUTION
Time Execution = 227.20117172499977 second
=====
                                TOTAL TEST
The total test = 2010201
```

k. Input 6

```
ut1.txt  ≡ input
≡ input6.txt
THREE
THREE
  TWO
  TWO
  ONE+
-----
ELEVEN
```

l. Output 6

```
cryptarithmic/src/main.py
Input file: ./test/input6.txt
=====
                                PROBLEM
THREE
THREE
  TWO
  TWO
  ONE+
-----
ELEVEN
=====
                                SOLUTION
84611
84611
  803
  803
  391+
-----
171219
=====
                                TIME EXECUTION
Time Execution = 221.38640173000022 second
=====
                                TOTAL TEST
The total test = 1539439
mgs_tahrani@mgs_tahrani: /media/mgs_tahrani/DATA/2020
```

m. Input 7

```
input7.txt
input7.txt
COCA
COLA+
-----
OASIS
```

n. Output 7

```
Input file: ./test/input7.txt
=====
PROBLEM
COCA
COLA+
-----
OASIS
=====
SOLUTION
8186
8106+
-----
16292
=====
TIME EXECUTION
Time Execution = 60.519871552999575 second
=====
TOTAL TEST
The total test = 1166861
```

o. Input 8

```
input8.txt
input8.txt
CROSS
ROADS+
-----
DANGER
```

p. Output 8

```
Input file: ./test/input8.txt
=====
                                PROBLEM
CROSS
ROADS+
-----
DANGER
=====
                                SOLUTION
96233
62513+
-----
158746
=====
                                TIME EXECUTION
Time Execution = 180.92134464799983 second
=====
                                TOTAL TEST
The total test = 2253926
ms_tabrani@ms_tabrani: /media/ms_tabrani/DAT
```

q. Input 9

```
1.txt
input9.txt
HERE
SHE+
-----
COMES
```

r. Output 9

```
cryptan1@mmccic:/src/main.py
Input file: ./test/input9.txt
=====
                                PROBLEM
HERE
SHE+
-----
COMES
=====
                                SOLUTION
9454
894+
-----
10348
=====
                                TIME EXECUTION
Time Execution = 58.08634223899935 second
=====
                                TOTAL TEST
The total test = 912833
ms_tabrani@ms_tabrani: /media/ms_tabrani/DAT
```

s. Input 10

```
t1.txt
input10.txt
MEMO
FROM+
-----
HOMER
```

t. Output 10

```
cryptarithmic/src/main.py
Input file: ./test/input10.txt
=====
PROBLEM
MEMO
FROM+
-----
HOMER
=====
SOLUTION
8485
7358+
-----
15843
=====
TIME EXECUTION
Time Execution = 6.804820124000798 second
=====
TOTAL TEST
The total test = 112870
ms_taheri@ms_taheri: /media/ms_taheri/DATA
```

u. Input 11

```
t11.txt X
input11.txt
JUNE
JULY+
-----
APRIL
```

v. Output 11

```
Input file: ./test/input11.txt
=====
PROBLEM
JUNE
JULY+
-----
APRIL
=====
SOLUTION
8532
8564+
-----
17096
=====
TIME EXECUTION
Time Execution = 19.48341985300067 second
=====
TOTAL TEST
The total test = 279900
ms_taheri@ms_taheri: /media/ms_taheri/DATA
```

w. Input 12

```
input12.txt X
≡ input12.
SEND
MORE+
-----
MONEY
```

x. Output 12

```
Input file: ./test/input12.txt
=====
PROBLEM
SEND
MORE+
-----
MONEY
=====
SOLUTION
9567
1085+
-----
10652
=====
TIME EXECUTION
Time Execution = 72.00834349599972 second
=====
TOTAL TEST
The total test = 1215020
```

y. Input 13

```
input13.txt X
≡ input13.t
FORTY
TEN
TEN+
-----
SIXTY
```

z. Output 13

```
Input file: ./test/input13.txt
=====
PROBLEM
FORTY
TEN
TEN+
-----
SIXTY
=====
SOLUTION
29786
850
850+
-----
31486
=====
TIME EXECUTION
Time Execution = 128.72116313300012 second
=====
TOTAL TEST
The total test = 1327171
```


BAGIAN IV
ALAMAT DRIVE

<https://github.com/mgstabrani/cryptarithmic>

REFERENSI

<https://www.basic-mathematics.com/cryptarithms.html> (diakses pada 21 Januari 2021).

<https://koding.alza.web.id/mengukur-lama-eksekusi-kode-program/> (diakses pada 23 Januari 2021).

<https://www.xspdf.com/resolution/50569606.html#:~:text=Heap's%20algorithm%20non%20recursive&text=Heap's%20algorithm%20generates%20all%20possible,2%20elements%20are%20not%20disturbed>. (diakses pada 23 Januari 2021).

<http://www.cryptarithms.com/default.asp?pg=1> (diakses pada 25 Januari 2021).

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-\(2021\)Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)Bag1.pdf) (diakses pada 26 Januari 2021).

LAMPIRAN

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 cryptarithmic hanya benar untuk persoalan cryptarihtmetic dengan dua buah operand.		✓
5. Solusi cryptarithmic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand.	✓	