

LAPORAN TUGAS KECIL 1
PENYELESAIAN *CRYPTARITHMETIC* DENGAN ALGORITMA *BRUTE*
FORCE

IF2211 Strategi Algoritma



Oleh:

Alif Bhadrika Parikesit

13519186

Kelas:

K-04

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2021

BAB I

ALGORITMA *BRUTE FORCE* PADA PENYELESAIAN *CRYPTARITHMETIC*

Algoritma *Brute Force* merupakan suatu pendekatan *straightforward* untuk memecahkan suatu persoalan, biasanya didasarkan pada pernyataan pada persoalan (*problem statement*) atau definisi konsep yang dilibatkan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung dan jelas (*in obvious way*). Hampir tidak ada persoalan yang tidak dapat diselesaikan melalui pendekatan algoritma *brute force*. Pada kesempatan penelitian ini, penulis akan mencoba menyelesaikan persoalan *cryptarithmic puzzles* dengan pendekatan algoritma *brute force*.

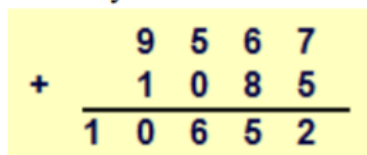
Cryptarithmic (atau cryptarithm) adalah sebuah puzzle penjumlahan di dalam matematika dimana angka diganti dengan huruf. Setiap angka dipresentasikan dengan huruf yang berbeda. Deskripsi permainan ini adalah: diberikan sebuah penjumlahan huruf, carilah angka yang merepresentasikan huruf-huruf tersebut.

Contoh:

$$\begin{array}{r} \text{S E N D} \\ + \text{M O R E} \\ \hline \text{M O N E Y} \end{array}$$

Gambar 1. Contoh persoalan *cryptarithmic puzzle*

Memiliki solusi:


$$\begin{array}{r} 9567 \\ + 1085 \\ \hline 10652 \end{array}$$

Gambar 2. Solusi *cryptarithmic puzzle* dari gambar 1

Sehingga solusi dari *puzzle* pada gambar 1 adalah $S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2$.

Untuk menyelesaikan *cryparitmetic puzzle* dapat dilakukan dengan melalui berbagai pendekatan. Namun, pada kesempatan penelitian kali ini, penulis akan menggunakan pendekatan algoritma *brute force*. Berikut adalah langkah-langkah penyelesaian *cryparithmetic puzzles* dengan pendekatan algoritma *brute force*:

1. Mengidentifikasi setiap karakter *unique* yang terdapat pada persamaan.

Pada contoh gambar 1, dari kata “SEND”, “MORE”, “MONEY”, karakter *unique* yang teridentifikasi adalah

2. Berdasarkan karakter yang sudah teridentifikasi, beri nilai setiap karakter dengan angka 0-9, dengan catatan setiap karakter *unique* memiliki nilai yang berbeda. Cara yang tepat untuk melakukan aksi ini adalah dengan melakukan permutasi dari 10 buah objek (0-9) ke dalam banyaknya karakter *unique*.

Sebagai contoh, $\{S = 0, E = 1, N = 2, D = 3, M = 4, O = 5, R = 6, Y = 7\}$. Permutasi dilakukan dari 10 buah objek ke dalam 8 buah karakter *unique*.

3. Melakukan pengecekan pada nilai untuk karakter pertama pada hasil penjumlahan, apabila nilainya 0 maka pencarian dilanjutkan lagi.

Sebagai contoh, $SEND + MORE = MONEY$, huruf M tidak boleh bernilai 0 karena digit pertama pada hasil tidak mungkin 0.

4. Melakukan substitusi nilai setiap angka ke persamaan *cryparithmetic*.
5. Mengulang langkah 2 hingga menemukan kombinasi angka yang tepat, yaitu kombinasi yang memenuhi persamaan *cryparithmetic*.

Sebagai contoh, kombinasi yang tepat untuk persoalan gambar 1 adalah $\{S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2\}$, karena memenuhi $9567 + 1085 = 10652$.

BAB II

SOURCE CODE PROGRAM CRYPARITHMETIC SOLVER

Source code program selengkapnya dapat diakses melalui tautan berikut:

<https://github.com/alifbhadrika/cryparithmetic-solver>

```
import time

def main():
    print("--- CRYPARITHMETIC PUZZLEZ ---")

    print()
    inputfile = input("ENTER inputFilename.txt: ")
    print()

    print("----- PROBLEM: -----\\n")
    with open('../test/'+inputfile,'r') as file:
        listInput = []
        for line in file:
            print(line.rstrip('\\n'))
            if line.startswith('-'):
                continue
            else:
                listInput.append(line.replace('+','').rstrip('\\n').lower())
    )

    print("\\n----- SOLUTION: -----\\n")
    start = time.time()

    solve(listInput)

    end = time.time()
    print("EXECUTION TIME: {:.6f} seconds".format(end-start))

def wordValue(word, letterValue):

    wordValue = 0
    n = len(word)
    factor = 1
    for _ in range (n-1):
        factor *= 10
```

```

    for letter in word:
        wordValue += factor * letterValue[letter]
        factor = int (factor/10)
    return wordValue

def printSolution(operands, value, letterValue):

    operands_letter = list('/'.join(operands))
    value_letter = list('').join(value)
    for op_letter in operands_letter:
        if op_letter == '/':
            print()
            continue
        print("{}".format(letterValue[op_letter]), end="")
    print("+")
    print("-----")
    for val_letter in value_letter:
        print("{}".format(letterValue[val_letter]), end="")

def permutation(listInput, n, r):
    '''
    permutasi dengan mengadopsi algoritma heap
    '''
    if n == 1:
        yield listInput[:r]
    else:
        for i in range(n-1):
            for perm in permutation(listInput, n-1, r):
                yield perm
            if (n % 2) == 1:
                j=0
            else:
                j=i
            listInput[j], listInput[n-1] = listInput[n-1], listInput[j]
        for perm in permutation(listInput, n-1, r):
            yield perm

def solve(words):

    letters_from_words = ''.join(words)
    letters = [] # ordered list of letters
    for letter in letters_from_words:
        if letter not in letters:
            letters.append(letter)

```

```

value = words[-1]                                # hasil penjumlahan
words.pop()
operands = words                                # operand penjumlahan
first_letter = value[0]                        # char 1st dari value penjmlhn

tescount = 0
digits = [dig for dig in range(10)]
for perm in permutation(digits, len(digits), len(letters)):
    letterValue = dict(zip(letters, perm))
    tescount += 1

    if letterValue[first_letter] == 0:
        continue

    sum_of_operands = 0
    for word in operands:
        sum_of_operands += wordValue(word, letterValue)

    if sum_of_operands == wordValue(value, letterValue):
        break

printSolution(operands, value, letterValue)
print("\n\nNUMBER OF TEST: {}".format(tescount))

if __name__ == '__main__':
    main()

```

BAB III

PENGUJIAN PROGRAM

NAMA FILE	INPUT	OUTPUT
1.txt	SEND MORE+ ----- MONEY	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 1.txt ----- PROBLEM: ----- SEND MORE+ ----- MONEY ----- SOLUTION: ----- 9567 1085+ ----- 10652 NUMBER OF TEST: 1565900 EXECUTION TIME: 8.825850 seconds </pre>
2.txt	NUMBER NUMBER+ ----- PUZZLE	<pre> --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 2.txt ----- PROBLEM: ----- NUMBER NUMBER+ ----- PUZZLE ----- SOLUTION: ----- 201689 201689+ ----- 403378 NUMBER OF TEST: 2475622 EXECUTION TIME: 17.672194 seconds </pre>

3.txt	TILES PUZZLES+ ----- PICTURE	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 3.txt ----- PROBLEM: ----- TILES PUZZLES+ ----- PICTURE ----- SOLUTION: ----- 91542 3077542+ ----- 3169084 NUMBER OF TEST: 638575 EXECUTION TIME: 4.835645 seconds </pre>
4.txt	CLOCK TICK TOCK+ ----- PLANET	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 4.txt ----- PROBLEM: ----- CLOCK TICK TOCK+ ----- PLANET ----- SOLUTION: ----- 90892 6592 6892+ ----- 104376 NUMBER OF TEST: 3022838 EXECUTION TIME: 24.582628 seconds </pre>
5.txt5.txt	COCA COLA+ ----- OASIS	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 5.txt ----- PROBLEM: ----- COCA COLA+ ----- OASIS ----- SOLUTION: ----- 8186 8106+ ----- 16292 NUMBER OF TEST: 1500845 EXECUTION TIME: 8.209514 seconds </pre>

6.txt	<p>HERE SHE+ ----- COMES</p>	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 6.txt ----- PROBLEM: ----- HERE SHE+ ----- COMES ----- SOLUTION: ----- 9454 894+ ----- 10348 NUMBER OF TEST: 1363481 EXECUTION TIME: 7.756580 seconds </pre>
7.txt	<p>DOUBLE DOUBLE TOIL+ ----- TROUBLE</p>	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 7.txt ----- PROBLEM: ----- DOUBLE DOUBLE TOIL+ ----- TROUBLE ----- SOLUTION: ----- 798064 798064 1936+ ----- 1598064 NUMBER OF TEST: 1157529 EXECUTION TIME: 9.167154 seconds </pre>
8.txt	<p>THREE THREE TWO TWO ONE+ ----- ELEVEN</p>	<pre> C:\Users\ASUS\Documents\cryparithmetic-solver\src>py main.py --- CRYPARITHMETIC PUZZLEZ --- ENTER inputFilename.txt: 8.txt ----- PROBLEM: ----- THREE THREE TWO TWO ONE+ ----- ELEVEN ----- SOLUTION: ----- 84611 84611 803 803 391+ ----- 171219 NUMBER OF TEST: 2315647 EXECUTION TIME: 21.037426 seconds </pre>

TABEL PENILAIAN

Poin	Kriteria	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 cryptarithmic hanya benar untuk persoalan cryptarihtmetic dengan dua buah operand	V	
5.	Solusi cryptarithmic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand.	V	

DAFTAR PUSTAKA

Cryptarithms.com. “*Alphametics – Example*”. <http://www.cryptarithms.com/> . Diakses *online* pada 24 Januari 2020.

Munir, Rinaldi. Slide kuliah “*Algoritma Brute Force*”.
[http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-\(2016\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf) . Diakses *online* pada 24 Januari 2020.

Munir, Rinaldi. Slide kuliah “*Tugas Kecil 1 IF2211*”.
[http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-1-\(2021\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-1-(2021).pdf) . Diakses *online* pada 24 Januari 2020.

Wang TC. “*Heaps Algorithm Fun Observation*”. <https://medium.com/sodalabs/heaps-algorithm-fun-observation-4986a188a80>. Diakses *online* 25 Januari 2020.