**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 *CRYPARITHMETIC***

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 *cryparithmetic puzzles* dengan pendekatan algoritma *brute force*.

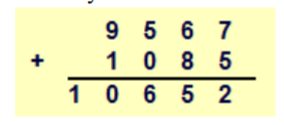
Cryptarithmetic (atau cryptarithm) adalah sebuah puzzle penjumlahan di dalam matematika dimana angka diganti dengan huruf. Setiap angka dipresentasikan dengan huruf yang berbeda. Deskripsi permainan ini adalah: diberikan sebuah penjumlahan huruf, carilah angka yang merepresentasikan huruf-huruf tersebut.

Contoh:



**Gambar 1.** Contoh persoalan c*ryparithmetic puzzle*

Memiliki solusi:



**Gambar 2.** Solusi c*ryparithmetic 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. Mengidentifiaksi setiap karakter *unique* yang terdapat pada persamaan.

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

1. 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*.

1. 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.

1. Melakukan substitusi nilai setiap angka ke persamaan *cryparithmatic*.
2. 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 |  |  |
| 2.txt |  |  |
| 3.txt |  |  |
| 4.txt |  |  |
| 5.xt5.txt |  |  |
| 6.txt |  |  |
| 7.txt |  |  |
| 8.txt |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **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 cryptarithmetic hanya benar untuk persoalan cryptarihtmetic dengan dua buah operand | **V** |  |
| 5. | Solusi cryptarithmetic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand. | **V** |  |

**TABEL PENILAIAN**

**DAFTAR PUSTAKA**

Cryparithms.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> . 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.