

TUGAS KECIL 1
PENYELESAIAN CRYPTARITHMETIC DENGAN ALGORITMA BRUTE
FORCE

IF2211 STRATEGI ALGORITMA
SEMESTER II

Oleh

Muhammad Fawwaz Naabigh - 13519206



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2020/2021

Algoritma *brute force*

1. Mula-mula setiap kata dibaca, lalu dicari setiap huruf unik dalam semua kata tersebut
2. Kemudian dicari permutasi dari angka 0-9 sebanyak jumlah huruf unik ($n = 10$, $r = \text{jumlah huruf unik}$)
3. Selanjutnya, dari hasil permutasi tersebut, dilakukan iterasi untuk mencari permutasi yang memenuhi operasi penjumlahan pada masukan. Berikut langkah-langkah dalam iterasi tersebut
 - a. Hasil dari permutasi akan disimpan ke dalam sebuah variabel secara berurutan sesuai dengan huruf unik. Jadi, setiap huruf unik akan berisi satu angka sesuai urutan permutasi.
 - b. Selanjutnya, dilakukan iterasi untuk tiap kata (operan), yang kemudian akan dibuat variabel baru dengan nama yang sesuai dengan nama kata tersebut. Variabel tersebut akan berisi angka sesuai dengan susunan dari huruf unik pada proses sebelumnya.
 - c. Kemudian setiap operan akan dijumlah, bila perhitungan tersebut sesuai dengan hasil penjumlahan yang sebenarnya, maka akan ditampilkan solusi beserta banyak percobaan dan waktu yang dibutuhkan.
 - d. Bila, perhitungan tidak sesuai dengan hasil penjumlahan sebenarnya, maka akan dilakukan iterasi lagi untuk hasil permutasi selanjutnya.

Source Program

```
import re, time, os

# Membaca masukan dari file.txt
# dan mengembalikan list yang berisi setiap kata dari masukan
def bacaFile():
    directory = os.path.abspath("./test/cryptarith.txt")
    f = open(directory, "r")
    lines = f.readlines()
    clean_lines = []
    for line in lines:
        x = re.sub(r'^\w\s', '', line)
        y = re.sub(r'\n', '', x)
        if y!="":
            clean_lines.append(y)
    return clean_lines

# Mengembalikan huruf unik dari masukan
def toUniqueLetters(wordsArray):
    huruf = []
    for words in wordsArray:
        abjad = re.findall(r'[A-Z]', words)
        for a in abjad:
            if a not in huruf:
                huruf.append(a)
    return huruf

words = bacaFile()
huruf = toUniqueLetters(words)
banyakHurufUnik = len(huruf)

banyakPercobaan = 0
t0 = time.time()
```

```

perm = []

# Menampung hasil permutasi ke perm
def insertPerm(numbers, n):
    for i in range(0, n, n):
        perm.append(numbers[:n])

# Mencari permutasi
def permut(numbers, length, r):
    if (length == 1):
        insertPerm(numbers, r)
        return
    for i in range(length):
        permut(numbers, length-1, r)
        # jika length ganjil, numbers indeks terakhir ditukar dengan
        indeks pertama
        if length%2==1:
            tmp = numbers[0]
            numbers[0] = numbers[length-1]
            numbers[length-1] = tmp
        else:
            tmp = numbers[i]
            numbers[i] = numbers[length-1]
            numbers[length-1] = tmp

permut([9, 8, 7, 6, 5, 4, 3, 2, 1, 0], 10, banyakHurufUnique)

for permutasi in perm:
    # Menyimpan hasil permutasi ke huruf unik secara berurutan
    for i in range(len(huruf)):
        globals()[huruf[i]] = permutasi[i]
    # Menghitung nilai setiap kata, sesuai dengan urutan huruf unik yang
    dimiliki
    for word in words:
        panjangWord = len(word)
        globals()[word.lower()] = 0
        for j in range(panjangWord):
            globals()[word.lower()] += ((10**(panjangWord-j-1)) *
            globals()[word[j]])
        # Berisi digit pertama untuk setiap kata
        digitPertama = []

        # Menghitung hasil penjumlahan dari tiap operan
        jumlah = 0
        for i in range(len(words)-1):
            jumlah += globals()[words[i].lower()]
            globals()["words"+str(i)] = globals()[words[i].lower()]
            # Menambahkan digit pertama tiap operan
            digitPertama.append(int(str(globals()[words[i].lower()])[0]))

        # Berisi nilai dari "penjumlahan"/berisi nilai dari kata terakhir
        hasil = globals()[words[-1].lower()]
        digitPertama.append(int(str(globals()[words[-1].lower()])[0]))

```

```

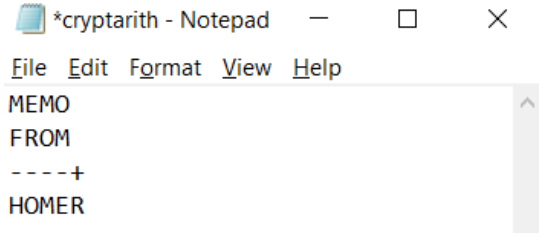
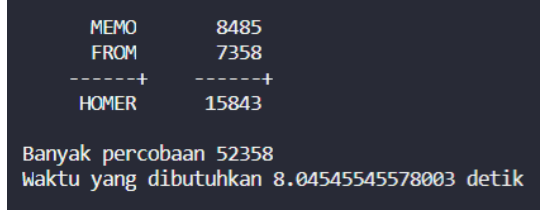
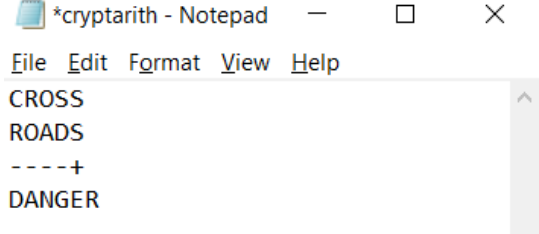
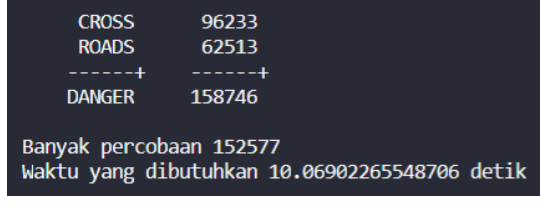
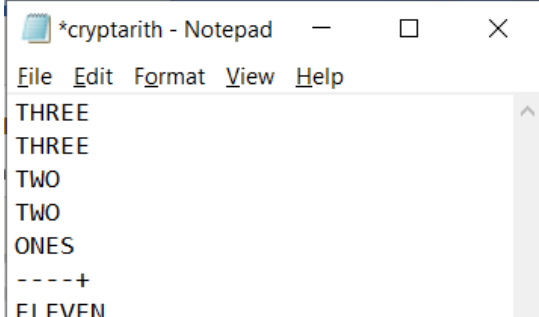
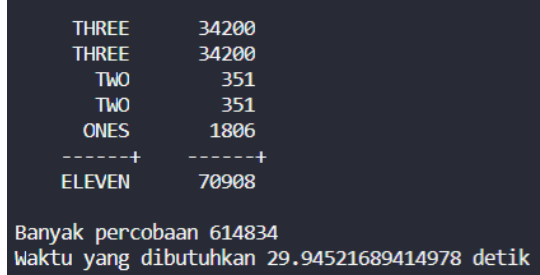
    banyakPercobaan+=1






    # Jika hasil penjumlahan sesuai dengan nilai kata terakhir hasil
    akan ditulis
    if((jumlah == hasil) and not(0 in digitPertama)):
        print()
        for i in range(len(words)-1):
            print(words[i].rjust(10),
str(globals()[words[i].lower()]).rjust(10))
            print("-----".rjust(10)+ "+" + "-----".rjust(10)+ "+")
            print(words[-1].rjust(10), str(globals()[words[-
1].lower()]).rjust(10))

        t1 = time.time()
        waktu = t1-t0
        print("\nBanyak percobaan", banyakPercobaan)
        print("Waktu yang dibutuhkan", waktu, "detik")
        break

```

Screenshot hasil percobaan

Masukan	Luaran
 <pre> *cryptarith - Notepad File Edit Format View Help MEMO FROM ----+ HOMER </pre>	 <pre> MEMO 8485 FROM 7358 ----+----+ HOMER 15843 Banyak percobaan 52358 Waktu yang dibutuhkan 8.04545545578003 detik </pre>
 <pre> *cryptarith - Notepad File Edit Format View Help CROSS ROADS ----+ DANGER </pre>	 <pre> CROSS 96233 ROADS 62513 ----+----+ DANGER 158746 Banyak percobaan 152577 Waktu yang dibutuhkan 10.06902265548706 detik </pre>
 <pre> *cryptarith - Notepad File Edit Format View Help THREE THREE TWO TWO ONES ----+ ELEVEN </pre>	 <pre> THREE 34200 THREE 34200 TWO 351 TWO 351 ONES 1806 ----+----+ ELEVEN 70908 Banyak percobaan 614834 Waktu yang dibutuhkan 29.94521689414978 detik </pre>

 *cryptarith - Notepad — □ ✕ File Edit Format View Help HERE SHE ----+ COMES	<table> <tr><td>HERE</td><td>2858</td></tr> <tr><td>SHE</td><td>628</td></tr> <tr><td>----+</td><td>----+</td></tr> <tr><td>COMES</td><td>3486</td></tr> </table> <p>Banyak percobaan 378901 Waktu yang dibutuhkan 14.165547370910645 detik</p>	HERE	2858	SHE	628	----+	----+	COMES	3486		
HERE	2858										
SHE	628										
----+	----+										
COMES	3486										
 cryptarith - Notepad — □ ✕ File Edit Format View Help COCA COLA ----+ OASIS	<table> <tr><td>COCA</td><td>8186</td></tr> <tr><td>COLA</td><td>8106</td></tr> <tr><td>----+</td><td>----+</td></tr> <tr><td>OASIS</td><td>16292</td></tr> </table> <p>Banyak percobaan 1169850 Waktu yang dibutuhkan 30.21542453765869 detik</p>	COCA	8186	COLA	8106	----+	----+	OASIS	16292		
COCA	8186										
COLA	8106										
----+	----+										
OASIS	16292										
 *cryptarith - Notepad — □ ✕ File Edit Format View Help CLOCK TICK TOCK ----+ PLANET	<table> <tr><td>CLOCK</td><td>90892</td></tr> <tr><td>TICK</td><td>6592</td></tr> <tr><td>TOCK</td><td>6892</td></tr> <tr><td>----+</td><td>----+</td></tr> <tr><td>PLANET</td><td>104376</td></tr> </table> <p>Banyak percobaan 1173492 Waktu yang dibutuhkan 58.89901852607727 detik</p>	CLOCK	90892	TICK	6592	TOCK	6892	----+	----+	PLANET	104376
CLOCK	90892										
TICK	6592										
TOCK	6892										
----+	----+										
PLANET	104376										
 *cryptarith - Notepad — □ ✕ File Edit Format View Help TILES PUZZLES ----+ PICTURE	<table> <tr><td>TILES</td><td>91542</td></tr> <tr><td>PUZZLES</td><td>3077542</td></tr> <tr><td>----+</td><td>----+</td></tr> <tr><td>PICTURE</td><td>3169084</td></tr> </table> <p>Banyak percobaan 928562 Waktu yang dibutuhkan 29.763248682022095 detik</p>	TILES	91542	PUZZLES	3077542	----+	----+	PICTURE	3169084		
TILES	91542										
PUZZLES	3077542										
----+	----+										
PICTURE	3169084										
 *cryptarith - Notepad — □ ✕ File Edit Format View Help NUMBER NUMBER ----+ PUZZLE	<table> <tr><td>NUMBER</td><td>201689</td></tr> <tr><td>NUMBER</td><td>201689</td></tr> <tr><td>----+</td><td>----+</td></tr> <tr><td>PUZZLE</td><td>403378</td></tr> </table> <p>Banyak percobaan 2034488 Waktu yang dibutuhkan 59.13001465797424 detik</p>	NUMBER	201689	NUMBER	201689	----+	----+	PUZZLE	403378		
NUMBER	201689										
NUMBER	201689										
----+	----+										
PUZZLE	403378										

Link Source Code

https://github.com/fwznbg/strategiAlgoritma/tree/main/Tucil%201_13519206

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat membaca file masukan dan menuliskan luaran.	√	
4. Solusi <i>cryptarithmic</i> hanya benar untuk persoalan <i>cryptarithmic</i> dengan dua buah <i>operand</i> .	√	
5. Solusi <i>cryptarithmic</i> benar untuk persoalan <i>cryptarithmic</i> untuk lebih dari dua buah <i>operand</i>	√	