

PENYELESAIAN CRYPTARITHMETIC DENGAN ALGORITMA BRUTE FORCE

LAPORAN

Sebagai Bagian dari Tugas Kecil 1 mata kuliah Strategi Algoritma IF2211 pada
semester II Tahun Akademik 2020/2021



Oleh

Isabella Handayani Sumantri

13519081

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021**

DAFTAR ISI

DAFTAR ISI	1
BAB I	2
1.1. Cryptarithmic	2
1.2. Brute force	2
1.3. Penyelesaian Cryptarithmic dengan Algoritma Brute Force	2
BAB II	4
2.1. Variabel Global	4
2.2. Fungsi Analyse	4
2.3. Fungsi searchIdx	5
2.4. Fungsi Cek	5
2.5. Fungsi Valid	6
2.6. Fungsi Permutation	6
2.7. Prosedur Clean	7
2.8. Program Utama	7
BAB III	9
3.1. Uji 1	9
3.2. Uji 2	9
3.3. Uji 3	9
3.4. Uji 4	9
3.5. Uji 5	10
3.6. Uji 6	10
3.7. Uji 7	10
3.8. Uji 8	11
3.9. Uji 9	11
BAB IV	12

BAB I

Algoritma *Brute Force*

1.1. *Cryptarithmic*

Cryptarithmic 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.

1.2. *Brute force*

Algoritma *brute force* adalah pendekatan yang *straightforward* untuk menyelesaikan suatu masalah. Salah satu teknik pencarian solusi yang dapat digunakan bersama algoritma *brute force* adalah teknik *exhaustive search*. Teknik ini digunakan untuk memecahkan persoalan kombinatorik, yaitu persoalan di antara objek kombinatorik.

Algoritma *brute force* umumnya adalah algoritma yang tidak mangkus. Untuk itu, digunakan teknik heuristik untuk mempercepat pencarian solusi dengan mengurangi kemungkinan solusi tanpa harus mengeksplorasinya terlebih dahulu. Selain itu, metode teknik heuristik yang digunakan juga dapat didasarkan dari pengalaman.

1.3. Penyelesaian *Cryptarithmic* dengan Algoritma *Brute Force*

Untuk menyelesaikan *cryptarithmic*, program akan mencari terlebih dahulu daftar huruf unik dan daftar huruf pertama dari operand dan hasil *input*. Kemudian, penyelesaian *cryptarithmic* dengan algoritma *brute force* dilakukan dengan melakukan enumerasi setiap kemungkinan solusi. Kemungkinan solusi dibangkitkan melalui algoritma permutasi. Agar algoritma semakin mangkus, terdapat beberapa metode teknik heuristik yang digunakan,

1. Membatasi parameter perulangan yang menghasilkan kemungkinan solusi sehingga hasil dari list yang menyimpan kemungkinan solusi sama dengan banyaknya daftar huruf unik.
2. Berdasarkan pengalaman ditemukan bahwa algoritma akan lebih mangkus jika permutasi dimulai dari angka yang paling besar.

Berikut adalah potongan kode yang menunjukkan implementasi kedua teknik heuristik,

```
for i in range(10**(len(letter)), (10**(len(letter) - 1)), - 1):  
    output = [int(d) for d in str(i)]
```

Setelah kemungkinan solusi dibangkitkan, akan dilakukan evaluasi terhadap kemungkinan solusi tersebut satu persatu. Proses akan dihentikan jika evaluasi berhasil menemukan penjumlahan semua operand sama dengan hasil dan merupakan kemungkinan solusi yang valid. Kemungkinan solusi yang valid adalah ketika semua hurufnya unik dan huruf pertama tidaklah dilambangkan dengan angka 0. Kemudian, solusi yang diperoleh akan ditampilkan pada layar bersama dengan jumlah tes yang dilakukan dan waktu yang dihabiskan.

BAB II

Source Program

2.1. Variabel Global

```
import time

# Input
text = []
output = []
letter = []
first = []
```

Pada bagian program ini dilakukan inisialisasi terhadap list berikut,

1. *text* : menyimpan data dari file input
2. *output* : menyimpan kemungkinan solusi yang dibangkitkan permutasi
3. *letter* : menyimpan daftar huruf unik dari operand dan hasil
4. *first* : menyimpan semua huruf pertama dari operand dan hasil

2.2. Fungsi Analyse

```
def analyse():
    operand = True
    cnt = 0
    result = False
    for x in text:
        front = True
        for i in range(len(x)):
            if x[i] == "-":
                operand = False
                break

            if x[i] != "\n" and x[i] != "+" and x[i] != " ":
                if x[i] not in letter:
                    letter.append(x[i])
                if front:
                    first.append(x[i])
                    front = False

            if not operand and i == (len(x)-1):
                result = True

        cnt += 1
        if result:
            break
    return cnt
```

Fungsi *analyse* digunakan untuk menyimpan huruf-huruf unik pada operand dan hasil. Kemudian, fungsi tersebut akan mengembalikan jumlah dari operand pada operasi ditambah dengan sebuah hasil.

2.3. Fungsi searchIdx

```
def searchIdx(x):  
    i = 0  
    found = False  
    while i < len(letter) and not found:  
        if letter[i] == x:  
            found = True  
        else:  
            i += 1  
    return i
```

Fungsi *searchIdx* akan mengembalikan index pada array letter yang *character*-nya sama dengan x.

2.4. Fungsi Cek

```
def cek():  
    operand = True  
    totalsum = 0  
    for x in text:  
        sum = 0  
        if operand:  
            i = 0  
            while i < len(x) and operand:  
                if x[i] == "-":  
                    operand = False  
                elif x[i] != "\n" and x[i] != " " and x[i] != "+":  
                    sum = 10*sum + output[searchIdx(x[i])]  
                    i += 1  
            if operand:  
                totalsum += sum  
        else:  
            for i in range(len(x)):  
                if x[i] != "\n" and x[i] != " " and x[i] != "+":  
                    sum = 10*sum + output[searchIdx(x[i])]  
            return sum == totalsum
```

Fungsi *cek* digunakan untuk memeriksa apakah kemungkinan bilangan yang dimiliki huruf dan disimpan pada *list output* akan memenuhi persamaan yang diberikan. Jika hasil dari penjumlahan semua operand sama dengan hasil makan fungsi cek akan mengembalikan *True*.

2.5. Fungsi Valid

```
def valid():
    i = 0
    valid = True
    while i < len(first) and valid:
        if(output[searchIdx(first[i])] == 0):
            valid = False
        else:
            i += 1
    return valid
```

Fungsi *valid* digunakan untuk mengecek huruf pertama adalah angka 0 atau tidak. Jika 0 fungsi tersebut akan mengembalikan *False*.

2.6. Fungsi Permutation

```
def permutation():
    global output
    count = 0
    found = False
    for i in range(10**(len(letter)), (10**(len(letter) - 1)), - 1):
        output = [int(d) for d in str(i)]
        count += 1
        if len(set(output)) == len(output) and cek() and valid():
            found = True
            break
    return (count, found)
```

Fungsi *permutation* akan menyimpan semua kemungkinan permutasi ke dalam array output. Permutasi dibangkitkan dengan melakukan perulangan dengan pengurangan sebanyak 1 kemudian mengubahnya menjadi sebuah array, sebagai contoh bilangan 12345 akan menghasilkan output = [1,2,3,4,5]. Kemudian, akan dilakukan pemeriksaan apakah semua setiap angka di array output menyatakan angka yang unik. Selanjutnya, akan dilakukan pemanggilan kepada fungsi *cek* dan *valid*. Terakhir, fungsi akan mengembalikan jumlah testing yang dilakukan dan apakah solusi berhasil ditemukan.

2.7. Prosedur Clean

```
def clean(cnt):
    global output, first, letter, text
    i = 0
    while i < (cnt+1) and len(text) > 0:
        text.pop(0)
        i += 1
    output = []
    first = []
    letter = []
```

Prosedur *clean* akan menghapus isi *list output*, *first*, *letter*, dan operand beserta hasil yang telah ditemukan solusinya pada *list text*. Penghapusan dilakukan dengan cara melakukan *pop* terhadap list *text* sebanyak *cnt*. Variabel *cnt* menunjukkan jumlah baris yang digunakan sebagai masukan pada operasi sebelumnya.

2.8. Program Utama

```
def main():
    global text
    filename = input("Masukkan nama file: ")
    try :
        file = open(f"../test/{filename}", "r")
        text = list(file)
        while len(text) > 0:
            start = time.time()
            cnt = analyse()
            tes = permutation()
            if(tes[1]):
                for i in range(cnt):
                    inNumber = ""
                    for j in range(len(text[i])):
                        if text[i][j] != "\n" and text[i][j] != " " and text[i][j] != "-" and text[i][j] != "+":
                            inNumber = inNumber + str(output[searchIdx(text[i][j])])
                        else:
                            inNumber = inNumber + text[i][j]
                    print(text[i].rstrip('\n'), inNumber.rstrip('\n'))
            else:
                print("No solution found")
            end = time.time()
            print("Banyaknya tes :", tes[0])
            print(f"Waktu : {end-start} detik")
            clean(cnt)
        file.close()
    except:
        print("No File Found")

    print("Enter to close..")
    input()
```

Pada program utama, pengguna diminta memasukkan *file* berisi data uji. Kemudian, *file* tersebut akan dibuka dan isinya dijadikan sebuah list. Proses pencarian solusi dilakukan dalam sebuah perulangan selama isi dari *list text* tidaklah kosong. Dilakukan pencarian daftar huruf unik melalui pemanggilan fungsi *analyse*. Selanjutnya, dilakukan enumerasi kemungkinan solusi melalui fungsi *permutation*. Jika menemukan solusinya,

akan dilakukan pencetakan *input* dan solusi secara berdampingan. Terakhir, dilakukan pencetakan waktu yang dihabiskan dan banyaknya tes yang dilakukan.

BAB III

Screenshot

3.1. Uji 1

```
SEND 9567
MORE+ 1085+
-----
MONEY 10652
Banyaknya tes : 4328919
Waktu : 9.863404750823975 detik
```

3.2. Uji 2

```
COCA 8186
COLA+ 8106+
-----
OASIS 16292
Banyaknya tes : 183972
Waktu : 0.5792570114135742 detik
```

3.3. Uji 3

```
HERE 9454
SHE+ 894+
-----
COMES 10348
Banyaknya tes : 541898
Waktu : 1.383965015411377 detik
```

3.4. Uji 4

```
NO 87
GUN 908
NO+ 87+
-----
HUNT 1082
Banyaknya tes : 120989
Waktu : 0.49399256706237793 detik
```

3.5. Uji 5

```
MEMO 8485
FROM+ 7358+
-----
HOMER 15843
Banyaknya tes : 154270
Waktu : 0.8167078495025635 detik
```

3.6. Uji 6

```
DOUBLE 798064
DOUBLE 798064
TOIL+ 1936+
-----
TROUBLE 1598064
Banyaknya tes : 201935866
Waktu : 941.3847103118896 detik
```

3.7. Uji 7

```
THREE 84611
THREE 84611
TWO 803
TWO 803
ONE+ 391+
-----
ELEVEN 171219
Banyaknya tes : 153896029
Waktu : 899.6202237606049 detik
```

3.8. Uji 8

```
CROSS    96233
ROADS+   62513+
-----
DANGER 158746
Banyaknya tes : 37648127
Waktu : 139.13661694526672 detik
```

3.9. Uji 9





```
JUNE     9257
JULY+    9203+
-----
APRIL    18460
Banyaknya tes : 742968155
Waktu : 3001.1392719745636 detik
```

BAB IV LAMPIRAN

4.1. Alamat Repository

https://github.com/isabellahandayani/Tucil1_13519081

4.2. Checklist

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> .	