

I. ALGORITMA BRUTEFORCE

Langkah yang pertama adalah mempersiapkan array-array yang akan digunakan untuk menyimpan huruf-huruf apa saja yang akan dipakai, array yang menyimpan faktor pengali tiap-tiap huruf, dan array yang menyimpan indeks array huruf yang elemennya berisikan huruf pertama. Langkah berikutnya, file dibaca dan disimpan dalam array baru. Tiap elemen dari array baru tersebut berupa tiap baris dari operasi pertambahan (bisa menjadi operand, garis putus-putus dan hasil)

Selanjutnya, tiap-tiap baris dilakukan penyimpanan huruf-huruf yang dipakai. Jika huruf yang dibaca belum ada di array huruf, tambahkan ke array huruf, tambahkan pula faktor pengali 0 di array faktor pengali yang bersesuaian. Selanjutnya cek, apakah huruf tersebut berada di depan dan belum ada di array huruf depan. Jika ternyata belum ada di array huruf depan, tambahkan. Untuk menghitung faktor pengali, dari operand pertama hingga operand terakhir, faktor pengalinya ditambahkan (Misal FYP, faktor pengali dari F adalah 100, Y adalah 10, dan P adalah 1), sedangkan untuk baris yang menjadi hasil operasi, faktor pengalinya dikurangi. Proses-proses ini dilakukan untuk mencatat huruf-huruf apa saja yang muncul di operasi, faktor pengali tiap huruf yang muncul, serta huruf-huruf apa saja yang menjadi huruf pertama di tiap baris.

Kemudian, semua permutasi yang mungkin terjadi dihitung dan disimpan ke dalam array. Tiap kemungkinannya dicek apakah memenuhi syarat. Jika ada kemungkinan yang menyebabkan salah satu huruf depan bernilai 0, skip ke kemungkinan berikutnya. Jika sudah tidak ada yang menyebabkan huruf depan bernilai 0, jumlahkan semua faktor pengali tiap huruf dikali dengan nilai yang di-assign di huruf tersebut. Jika jumlah operasi tadi bernilai 0, maka kemungkinan memenuhi operasi yang dicari. Sebaliknya, cari kemungkinan selanjutnya yang mungkin memenuhi.

II. SOURCE PROGRAM

```
import time
```

```
#inisialisasi
```

```
digit = [i for i in range(10)]
```

```
letter = []
```

```
summ = []
```

```
firstLetter = []
```

```
dummy = []
```

```
def permutXDigit(arrarray,X):
```

```
    if (X == 0) : return [[]]
```

```
    if (len(arrarray) == 0) : return []
```

```
    if (len(arrarray) == 1) : return [arrarray]
```

```
    l = []
```

```

for i in range (len(array)):

    j = array[i]

    k = array[:i] + array[i+1:]

    for m in permutXDigit(k,X-1):

        l.append([j] + m)

    return l

def checkLet(x):

    i = 0

    found = False

    while (i < len(letter) and not found):

        if (x == letter[i]) : found = True

        i += 1

    return found

def checkFirstLet(x):

    i = 0

    found = False

    while (i < len(firstLetter) and not found):

        if (x == firstLetter[i]) : found = True

        i += 1

    return found

def getPosition(x):

    for i in range(len(letter)):

        if (x == letter[i]) : return i

print("Inputkan nama file uji di folder test : ", end="")

name = input()

name = "../test/" + name

with open(name, 'r') as fd :

    operand = fd.read().splitlines()

start = time.time()

```

```
for i in range (len(operand)):
```

```
    operand[i] = operand[i].replace(' ', '')
```

```
    operand[i] = operand[i].replace('+', '')
```

```
    dummy.append(operand[i])
```

```
for i in range(len(operand)-2):
```

```
    for j in range(len(operand[i])):
```

```
        if not checkLet(operand[i][j]):
```

```
            letter.append(operand[i][j])
```

```
            summ.append(0)
```

```
            if j == 0 and not checkFirstLet(getPosition(operand[i][j])):
```

```
                firstLetter.append(getPosition(operand[i][j]))
```

```
                summ[getPosition(operand[i][j])] += 10**(len(operand[i])-j-1)
```

```
dum = len(operand)-1
```

```
for j in range(len(operand[dum])):
```

```
    if not checkLet(operand[dum][j]):
```

```
        letter.append(operand[dum][j])
```

```
        summ.append(0)
```

```
        if j == 0 and not checkFirstLet(getPosition(operand[dum][j])):
```

```
            firstLetter.append(getPosition(operand[dum][j]))
```

```
            summ[getPosition(operand[dum][j])] -= 10**(len(operand[dum])-j-1)
```

```
permutation = permutXDigit(digit,len(letter))
```

```
dum = 0
```

```
result = [0 for i in range(len(letter))]
```

```
counter = 0
```

```
dum = 0
```

```
found = False
```

```
while dum != len(permutation) and not found:
```

```

i = 0

zero = False

while i != len(firstLetter) and not zero:

    if permutation[dum][firstLetter[i]] == 0:

        zero = True

    i += 1

if not zero :

    check = 0

    for j in range(len(result)):

        result[j] = permutation[dum][j]

        check += result[j] * summ[j]

    if check == 0 :

        found = True


counter += 1

dum += 1


for i in range(len(dummy)):

    for j in range(len(letter)):

        dummy[i] = dummy[i].replace(letter[j],str(result[j]))


for i in range(len(dummy)):

    if (i == len(dummy)-3):

        print((operand[i]+ "+").ljust(12) + dummy[i] + "+" )

    else:

        print(operand[i].ljust(12) + dummy[i])


end = time.time()

print("jumlah tes yang dilakukan " + str(counter))

```

```
print("waktu yang dihabiskan " + str(end-start))
```

III. SCREENSHOT I/O

Pengujian saya lakukan menggunakan google colab dikarenakan spek laptop saya sudah out of date

SEND	9567	MEMO	8485
MORE+	1085+	FROM+	7358+
-----	-----	-----	-----
MONEY	10652	HOMER	15843
jumlah tes yang dilakukan	1748230	jumlah tes yang dilakukan	128687
waktu yang dihabiskan	15.440460205078125	waktu yang dihabiskan	0.9014971256256104
CROSS	96233	THREE	84611
ROADS+	62513+	THREE	84611
-----	-----	TWO	803
DANGER	158746	TWO	803
jumlah tes yang dilakukan	3519768	ONE+	391+
waktu yang dihabiskan	33.4598388671875	-----	-----
NO	87	ELEVEN	171219
GUN	908	jumlah tes yang dilakukan	3090287
NO+	87+	waktu yang dihabiskan	31.4063503742218
-----	-----	DOUBLE	798064
HUNT	1082	DOUBLE	798064
jumlah tes yang dilakukan	134191	TOIL+	1936+
waktu yang dihabiskan	0.9125006198883057	-----	-----
HERE	9454	TROUBLE	1598064
SHE+	894+	jumlah tes yang dilakukan	2898676
-----	-----	waktu yang dihabiskan	32.33089780807495
COMES	10348	COCA	8186
jumlah tes yang dilakukan	575302	COLA+	8106+
waktu yang dihabiskan	3.451594591140747	-----	-----
CLOCK	90892	OASIS	16292
TICK	6592	jumlah tes yang dilakukan	123695
TOCK+	6892+	waktu yang dihabiskan	0.8757848739624023
-----	-----	NUMBER	201689
PLANET	104376	NUMBER+	201689+
jumlah tes yang dilakukan	3302475	-----	-----
waktu yang dihabiskan	34.51292037963867	PUZZLE	403378
TILES	91542	jumlah tes yang dilakukan	728504
PUZZLES+	3077542+	waktu yang dihabiskan	25.389073610305786
-----	-----	FORTY	29786
PICTURE	3169084	TEN	850
jumlah tes yang dilakukan	3328707	TEN+	850+
waktu yang dihabiskan	34.38673543930054	-----	-----
		SIXTY	31486
		jumlah tes yang dilakukan	1083579
		waktu yang dihabiskan	27.516717672348022

IV. ALAMAT DRIVE SOURCE CODE

<https://github.com/widyaput/TucilSTIMA1>

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