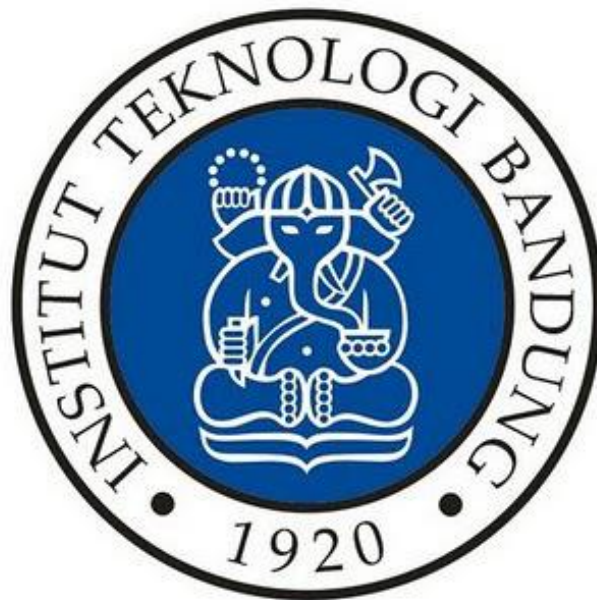


## **Tugas Kecil 1 IF2211 Strategi Algoritma**

*Penyelesaian Cryptarithmic dengan Algoritma Brute Force*

**Allief Nuriman  
13519221**



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

## Daftar Isi

Daftar Isi .....	2
A. Algoritma Brute Force .....	3
Algoritma Permutasi Selanjutnya (Next Permutation) .....	4
B. Source Code .....	4
C. Contoh Masukan & Keluaran .....	8
D. Checklist .....	9
E. Reposit Program .....	10

## A. Algoritma Brute Force

Penyelesaian masalah tersebut secara garis besar mengikuti langkah-langkah berikut:

1. Baca semua operan lalu simpan di dalam senarai berisi sekumpulan karakter, senarai tersebut bernama “arrAbjad” (Prekondisinya total karakter maksimumnya adalah 10 sehingga tidak dilakukan pengecekan).
2. Pembacaan file input selesai, jalankan *counter* waktu
3. Didefinisikan sebuah senarai yang akan dikaitkan dengan senarai “arrAbjad”, senarai tersebut bernama “arrVal”, pada awalnya, senarai tersebut akan diatur sebagai berikut (Akan berubah-ubah karena akan dipermutasikan):

Val	0	1	2	3	4	5	6	7	8	9
Idx	0	1	2	3	4	5	6	7	8	9

4. Proses permutasi pada “arrVal” dilakukan
5. Dilakukan pengecekan melalui fungsi “test”, pada dasarnya mengambil baris pada file, kemudian mengaitkan karakter-karakter pada baris tersebut. Misalnya pada kata “SEND” (Soal pertama, SEND + MORE = MONEY), diambil huruf ‘S’, kemudian dicari indeks yang merepresentasikan ‘S’ pada “arrAbjad”, kemudian indeksnya dikaitkan dengan “arrVal”. Ilustrasinya sebagai berikut:

arrVal	Val	0	1	2	3	4	5	6	7	8	9
	Idx	0	1	2	3	4	5	6	7	8	9
	Arah	↑	↑	↑	↑	↑	↑	↑	↑		
arrAbjad	Val	‘S’	‘E’	‘N’	‘D’	‘M’	‘O’	‘R’	‘Y’		
	Idx	0	1	2	3	4	5	6	7		

Begitu juga dengan huruf kedua, ketiga, dan keempatnya, yaitu E, N, dan D, sehingga membentuk 0123. Kemudian, disimpan ke dalam variabel “sum1”

6. Langkah keempat diulangi lagi tetapi untuk penanganan operan lain, hanya saja, dengan penanganan khusus jika ada tanda ‘+’, yaitu mengembalikan 0 saja (Tidak berpengaruh terhadap nilai) karena ‘+’ tidak relevan.
7. Setelah semua operan dijumlahkan, bagian yang bertanda ‘-’ akan dilewati, kemudian akan diproses huruf yang merepresentasikan hasil dengan skema seperti di langkah 4. Nilainya disimpan ke dalam variabel “sum2”
8. Akan dilakukan penyamaan antara sum1 & sum2, jika sama, akan dicek lagi apakah huruf pertamanya 0 atau tidak. Jika nol, maka proses diulang dari langkah 4 sampai memenuhi kondisi  $sum1 == sum2$  dan tidak ada huruf pertama yang bernilai nol. Proses diulang selama permutasi selanjutnya masih ada **atau**  $sum1 == sum2$  & huruf pertama pada tiap kata tidak nol.
9. Dua kemungkinan: Soal terselesaikan, atau tidak terjawab. Sebagai solusi, disiapkan variabel “Found” sebagai tanda apabila ditemukan permutasi yang memenuhi, bernilai “True” jika ada, dan “False” jika tidak.

### Algoritma Permutasi Selanjutnya (Next Permutation)

Algoritma ini menggunakan langkah sebagai berikut:

1. Temukan indeks terbesar  $k$  sehingga  $a[k] < a[k+1]$ . Jika indeks tersebut ada, maka permutasi itu adalah permutasi terakhir.
2. Temukan indeks terbesar  $l$  yang lebih besar dari  $k$  sehingga  $a[k] < a[l]$ .
3. Tukar nilai  $a[k]$  dengan  $a[l]$ .
4. Balikkan sekuens dari  $a[k+1]$  sampai ke elemen terakhir  $a[n]$

#### Ilustrasi:

Perhatikan senarai [1, 2, 3, 4]

1. Indeks  $k = 2$ , karena 3 berada pada indeks yang memenuhi kondisi indeks terbesar yang masih lebih kecil daripada  $a[k+1]$  yaitu 4.
2. Indeks  $l = 3$ , karena 4 hanyalah satu-satunya nilai pada senarai yang lebih besar dari 3 untuk memenuhi kondisi ini.
3. Nilai  $a[2]$  dan  $a[3]$  ditukar untuk membentuk senarai [1,2,4,3].
4. Senarai setelah  $a[k+1]$  ( $k = 2$ ) sampai elemen terakhir dibalikkan. Karena hanya ada satu nilai, yaitu 3, senarainya tidak berubah pada langkah ini.

### B. Source Code

```
main.py

# NIM: 13519221
# Nama: Allief Nuriman
# Tanggal: Rabu, 27 Januari 2021

# Program CryptarithmCalc
# Membaca input dari suatu file, yang
# menyatakan suatu cryptarithm, kemudian
# mengeluarkan persoalan & solusi, waktu
# eksekusi program (Tidak termasuk waktu
# pembacaan file input), dan jumlah total
# tes yang dilakukan untuk menemukan
# substitusi angka yang benar untuk setiap
# huruf

from time import perf_counter

def next_permutation(L):
    n = len(L)
    # Step 1: find rightmost position i such that L[i] < L[i+1]
    i = n - 2
    while i >= 0 and L[i] >= L[i+1]:
        i -= 1
    if i == -1:
        return False
    # Step 2: find rightmost position j to the right of i such that L[j] > L[i]
```

```

    j = i + 1
    while j < n and L[j] > L[i]:
        j += 1
    j -= 1
    # Step 3: swap L[i] and L[j]
    L[i], L[j] = L[j], L[i]
    # Step 4: reverse everything to the right of i
    left = i + 1
    right = n - 1
    while left < right:
        L[left], L[right] = L[right], L[left]
        left += 1
        right -= 1
    return True

def getVal(arrAbj, arrVal, x):
    ketemu = False
    i = 0
    if (x == '+'):
        return 0
    while (not (ketemu)):
        if (arrAbj[i] == x):
            ketemu = True
        else:
            i = i + 1
    return arrVal[i]

def test(arrAbj, arrVal, cc): # Fungsi ini digunakan untuk melakukan testing apakah sudah benar solusinya
    i = 0
    global sum2
    sum1 = 0 # Jumlah antara operan-operan
    sum2 = 0 # Jumlah hasil

    for i in range(len(cc)-2):
        for j in range(len(cc[i])):
            if (cc[i][len(cc[i])-1] == '+' and cc[i][j] != '+'):
                sum1 = sum1 + getVal(arrAbj, arrVal, cc[i][j])*(10**(len(cc[i])-2-j))
            else:
                sum1 = sum1 + getVal(arrAbj, arrVal, cc[i][j])*(10**(len(cc[i])-1-j))

        i = i + 2

    for j in range(len(cc[i])):
        sum2 = sum2 + getVal(arrAbj, arrVal, cc[i][j])*(10**(len(cc[i])-1-j))
    return (sum1 == sum2)

def hurufpertamanol(arrAbj, arrVal, cc):
    i = 0
    while (i < len(cc)):

```

```

        if (cc[i][0] == '-'):
            i = i + 1
        else:
            if (getVal(arrAbj, arrVal, cc[i][0]) == 0):
                return True
            i = i + 1
    return False

def printjumlahTiapOp(arrAbj, arrVal, cc):
    for i in range(len(cc)):
        sum = 0
        if (cc[i][0] == '-'):
            print(cc[i])
            outfile.write(str(cc[i]))
            outfile.write("\n")
        elif (cc[i][len(cc[i])-1] == '+'):
            for j in range(len(cc[i])-1):
                if (cc[i][j] != '+'):
                    sum = sum + getVal(arrAbj, arrVal, cc[i][j])*(10**(len(cc[i])-2-j))
            print(sum, end='')
            outfile.write(str(sum)+"")
            outfile.write("\n")
            print("")
        elif (i != len(cc)-1):
            for j in range(len(cc[i])):
                sum = sum + getVal(arrAbj, arrVal, cc[i][j])*(10**(len(cc[i])-1-j))
            print(sum)
            outfile.write(str(sum))
            outfile.write("\n")
        else:
            print(sum2)
            outfile.write(str(sum2))
            outfile.write("\n")
            outfile.write("\n")

# ALGORITMA
print("Untuk keluar, tulis exit")
print("Masukkan nama file: ", end='')
filename = input()

while (filename != "exit"): # Filename valid & bukan tanda keluar
    arrAbjad = [] # Dipakai untuk menyimpan alfabet
    arrVal = [0,1,2,3,4,5,6,7,8,9] # Indeks arrAbjad menunjuk ke sini, ini yang akan diperm
    utasikan
    berkas = open(filename, 'r') # Buka file testing untuk read-
    only, asumsikan file valid & berisi input yang diharapkan
    cc = berkas.read().splitlines() # Baca baris pertama (operan pertama)

    # Masukkan cc ke arrAbjad

```

```

for i in range(len(cc)):
    if (cc[i][0] != '-'):
        for j in range(len(cc[i])):
            unik = True
            k = 0
            while (k < len(arrAbjad) and unik):
                if (arrAbjad[k] == cc[i][j]):
                    unik = False
                else:
                    k = k + 1
            if (unik and cc[i][j] != '+'):
                arrAbjad.append(cc[i][j])

t1_start = perf_counter()

for i in range(len(cc)):
    print(cc[i])

totalaksi = 1

Found = False
while (next_permutation(arrVal) and not (Found)):
    if (test(arrAbjad, arrVal, cc)):
        if (not (hurufpertamanol(arrAbjad, arrVal, cc))):
            Found = True
            totalaksi = totalaksi + 1

t1_stop = perf_counter()

outfile = open("output.txt", 'w')

if (not Found):
    print("Tidak ditemukan kombinasi yang memenuhi")
    outfile.write("Tidak ditemukan kombinasi yang memenuhi")
    outfile.write("\n")
else:
    print()
    printjumlahTiapOp(arrAbjad, arrVal, cc)
    print()

print("Waktu pencarian (dalam sekon) adalah ", end='')
outfile.write("Waktu pencarian (dalam sekon) adalah "+str(t1_stop-t1_start))
outfile.write("\n")
outfile.write("Jumlah tes yang dilakukan adalah "+str(totalaksi))
outfile.write("\n")
outfile.close()
print(t1_stop-t1_start)
print("Jumlah tes yang dilakukan adalah ", end='')
print(totalaksi)
print()

```

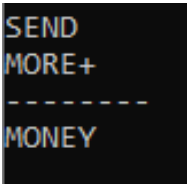
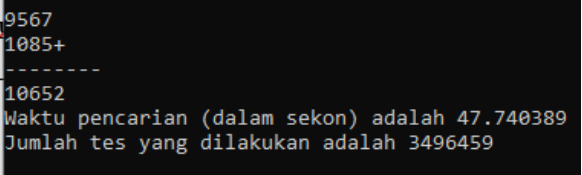
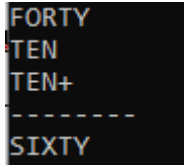
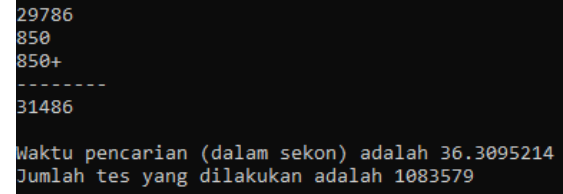
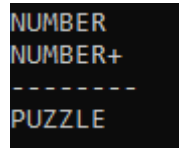
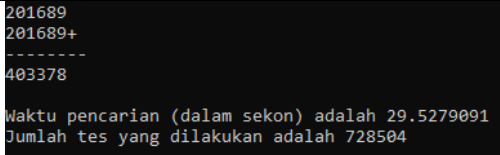
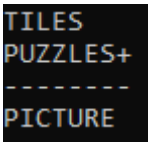
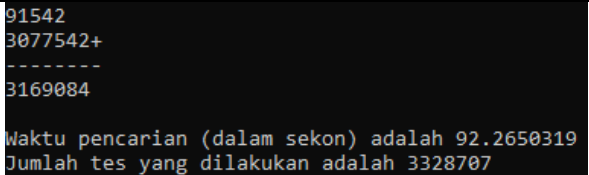

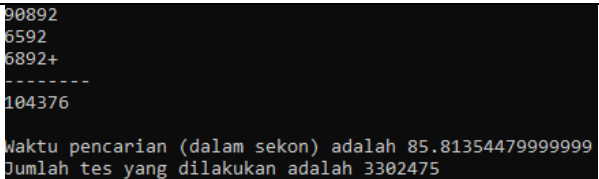
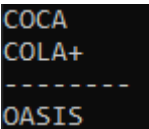
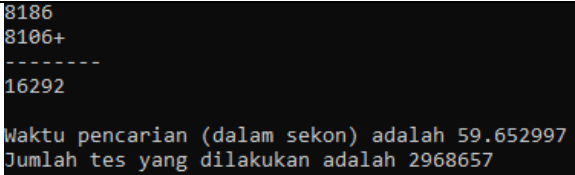
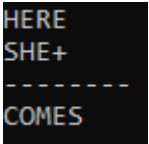
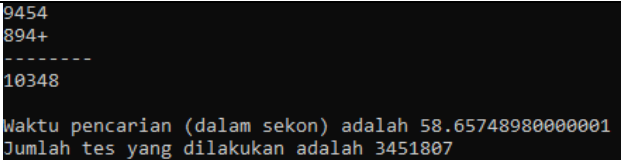
```

berkas.close()
outfile.close()

print("Untuk keluar, tulis exit")
print("Masukkan nama file: ",end='')
filename = input()

```

### C. Contoh Masukan & Keluaran

Masukan	Keluaran
	
	
	
	
	
	
	



THREE THREE TWO TWO ONE+ ----- ELEVEN	84611 84611 803 803 391+ ----- 171219  Waktu pencarian (dalam sekon) adalah 116.6536552 Jumlah tes yang dilakukan adalah 3090287
DOUBLE DOUBLE TOIL+ ----- TROUBLE	798064 798064 1956+ ----- 1598064  Waktu pencarian (dalam sekon) adalah 111.1077469 Jumlah tes yang dilakukan adalah 2898676
NO GUN NO+ ----- HUNT	87 908 87+ ----- 1082  Waktu pencarian (dalam sekon) adalah 71.3234544 Jumlah tes yang dilakukan adalah 3220561
MEMO FROM+ ----- HOMER	8485 7358+ ----- 15843  Waktu pencarian (dalam sekon) adalah 78.18176649999998 Jumlah tes yang dilakukan adalah 3088465
CROSS ROADS+ ----- DANGER	96233 62543+ ----- 158746  Waktu pencarian (dalam sekon) adalah 57.764503500000046 Jumlah tes yang dilakukan adalah 3519768

#### D. Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan ( <i>no syntax error</i> )	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat membaca file masukan dan menuliskan luaran.	✓	
4. Solusi cryptarithmic 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> .	✓	
--	---	--

#### E. Reposit Program

Dapat diakses pada [tautan berikut ini](#).