Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Laporan Tugas Kecil 1 IF 2211 Strategi Algoritma

Semester II Tahun 2020/2021

Penyelesaian *Cryptarithmetic* dengan Algoritma Brute Force

Disusun Oleh:

Aulia Adila
13519100

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2020/2021

Pendahuluan

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 persoalan :	Solusi:
SEND	9 5 6 7
+ MORE	+ 1 0 8 5
MONEY	1 0 6 5 2

Dapat disimpulkan bahwa S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2

Algoritma *brute force* yang digunakan

Secara garis besar, solusi yang penulis gunakan untuk menyelesaikan persoalan *cryptarithmetic* dengan *brute force* adalah dengan menerapkan konsep permutasi untuk mencari setiap kemungkinan solusi, kemudian divalidasi dengan algoritma tertentu untuk menentukan apakah solusi yang ditawarkan sudah merupakan solusi yang tepat.

Jika diperinci, terdapat beberapa poin yang dapat menjelaskan langkah algoritma yang digunakan oleh penulis.

No	Langkah algoritma	Nama fungsi/prosedur (jika ada)	Keterangan
1	Membaca masukan berupa file txt	readfile(*filename*)	Fungsi membaca file, menuliskan isi file (sebagai output), dan mengembalikan sebuah array *allstr* berisi masukan per baris
2	Proses *tokenizing* dari seluruh masukan yang	trueword(*allstr*)	Fungsi mengembalikan 2 array : array yang berisi karakter unik dari

	menjadi karakter huruf valid		operand; array yang berisi huruf pertama dari setiap kata.
3	Membuat array dengan karakter huruf unik	uniqueword(*allstr*)	Fungsi mengembalikan sebuah array dengan elemen karakter huruf yang unik, untuk dijadikan acuan karakter dalam persoalan
4	Proses *tokenizing* dari kata yang menjadi operand dalam persoalan	trueword2_quest(*trueword2*) uniqstr_quest(*allstr*)	Mengembalikan array dari kata operand beserta karakternya
5	Menyimpan kata yang menjadi jawaban dari operasi	-	Kata disimpan dalam variabel *answer*
6	Melakukan pencarian karakter dalam array karakter (bersifat generik)	searchuniq(*item, uniqstr*)	Fungsi yang mengembalikan boolean *found* yang menyatakan keberadaan karakter dalam array; serta *i* yang menyatakan indeks
7	Mencari kemungkinan solusi untuk penyelesaian persoalan dengan permutasi	permutation(*array, digit*)	Fungsi yang mengembalikan sebuah array yang terdiri atas array kemungkinan solusi (*array_perm*) dari persoalan. Dari *array* number = [0,1,2,3,4,5,6,7,8,9] akan buat permutasi dengan anggota elemen sejumlah panjang array karakter unik (*uniqstr*)
8	Memilih solusi yang paling tepat dari seluruh kemungkinan solusi	hitung(*array_perm*)	Melakukan iterasi untuk mengambil setiap *array_perm* yang kemudian akan dijadikan parameter fungsi

			hitung. Beberapa validasi yang dilakukan dalam fungsi: • Memeriksa apakah anggota *firstletter* bernilai 0 • Menjadikan karakter huruf pada "operand" dan "jawaban" sebagai sebuah bilangan (Contoh : ABC menjadi 110), kemudian memeriksa apakah hasil penjumlahan "operand" sama dengan nilai "jawaban". Jika kemungkinan solusi berhasil melewati beberapa tahap validasi, maka proses iterasi berhenti; diperoleh solusi terpilih.
9	Mencetak keluaran pada layar (persoalan dan solusi yang sesuai)	-	Untuk mencetak masukan, dilakukan iterasi pada setiap baris dalam array *allstr*; untuk mencetak keluaran, dilakukan iterasi setiap baris dan karakter dalam *allstr*. Setiap nilai (bilangan) yang mewakili sebuah karakter akan dicetak pada layar.
10	Mencetak keluaran pada layar berupa lama waktu eksekusi serta total tes yang dilakukan untuk menemukan substitusi angka yang benar	-	Menggunakan library *time* untuk menghitung waktu; menyimpan variabel *count_test* sebagai variabel global yang menyimpan jumlah test.

Source program dalam bahasa pemrograman yang dipilih (Python)

```python
import time
t0= time.perf_counter()

notprint = [" ","-","+"]
count_test = 0

def readfile(filename):
    f = open(filename, "r")
    allstr = f.read().splitlines()
    for i in allstr:
        print(i)
    return allstr

def trueword(allstr):
    truestr = []
    first_letter = []
    for i in allstr:
        count = 0
        word = []
        for j in i:
            if (j not in notprint):
                count+=1
                word.append(j)
                if count == 1 and j not in first_letter:
                    first_letter.append(j)
        if count != 0:
            truestr.append(word)
    return truestr, first_letter
```

```python
def uniqueword(allstr):
    uniqstr = []
    for word in allstr: #akses baris
        for char in word: #akses kata
            if (char in notprint) or (char in uniqstr):
                continue #continue to next iteration
            else:
                uniqstr.append(char)
    return uniqstr

filename = input("Enter the filename (include ./test/): ")
print("\nPROBLEMS:")
allstr = readfile(filename)
uniqstr = uniqueword(allstr)
trueword2, firstletter = trueword(allstr)
answer = trueword2[len(trueword2)-1]


def searchuniq(item, uniqstr):
    found = False
    i = 0
    while (not found) and (i < len(uniqstr)):
        if item == uniqstr[i]:
            found = True
        else:
            i += 1
    return found, i
```

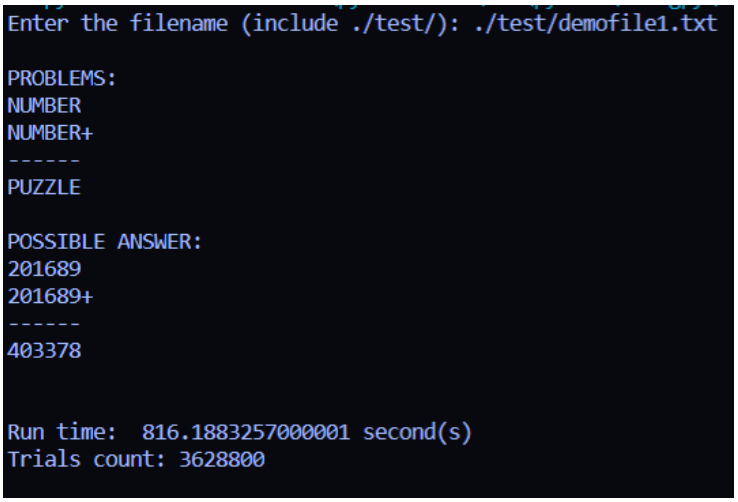
```python
def searchuniq(item, uniqstr):
    found = False
    i = 0
    while (not found) and (i < len(uniqstr)):
        if item == uniqstr[i]:
            found = True
        else:
            i += 1
    return found, i


def trueword2_quest(trueword2):
    ques = []
    i = 0
    while i<(len(trueword2)-1):
        ques.append(trueword2[i])
        i+=1
    return ques

def uniqstr_quest(allstr):
    ques = []
    i = 0
    while i<(len(allstr)-1):
        for j in allstr[i]:
            if (j not in notprint) and (j not in ques):
                ques.append(j)
        i += 1
    return ques
```

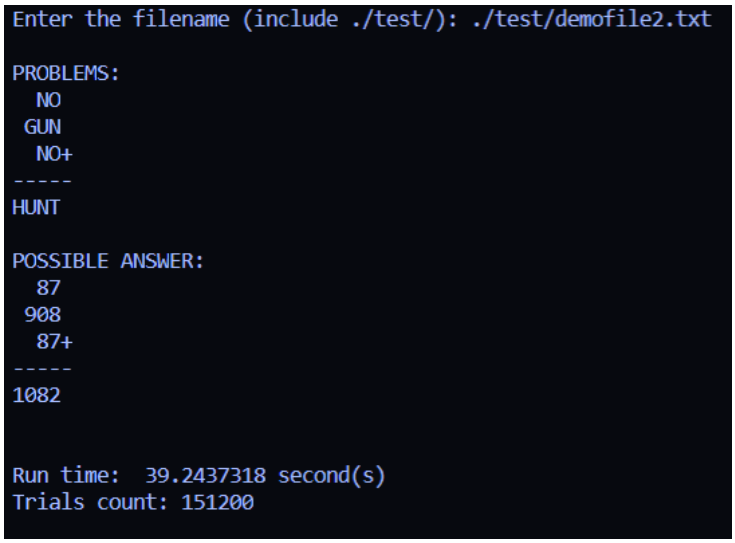
```python
uniqstr_ques = uniqstr_quest(allstr)
trueword2_ques = trueword2_quest(trueword2)


def hitung(array_perm):
    match = False

    for i in firstletter:
        found,idx = searchuniq(i,uniqstr)
        if array_perm[idx] == 0:
            return match


    sum = 0
    for i in trueword2_ques:
        for j in range (len(i)):
            power = len(i)-j-1
            found_char,index_char = searchuniq(i[j],uniqstr)
            sum = sum + (10**power)*array_perm[index_char]


    sum_hasil = 0
    for j in range (len(answer)):
        power = len(answer)-j-1
        found_char,index_char = searchuniq(answer[j],uniqstr)
        sum_hasil = sum_hasil + (10**power)*array_perm[index_char]

    if sum == sum_hasil:
        match = True

    return match
```

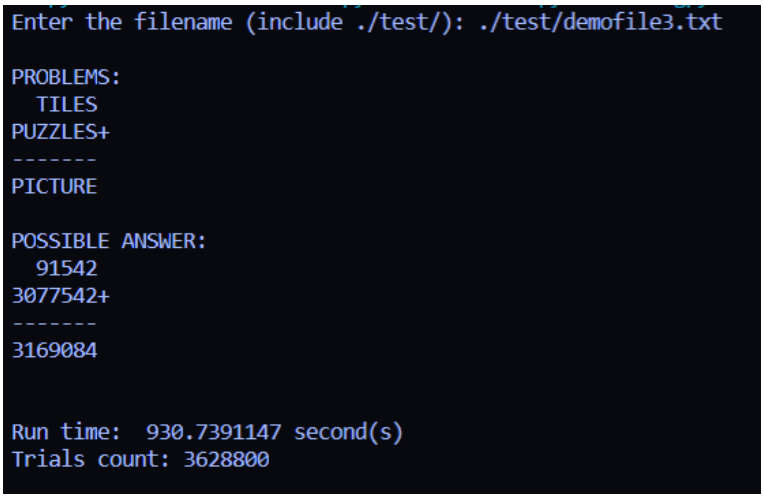
```
def permutation(array,digit):
    if (digit==0):
        return [[]]
    perm_array = []
    for i in range(len(array)):
        remaining = array[:i] + array[i+1:]
        current = array[i]
        temp = permutation(remaining,digit-1)
        for tail in temp:
            perm_array.append([current]+tail)
    return perm_array

number = [0,1,2,3,4,5,6,7,8,9]
hasil = permutation(number,len(uniqstr))

final_result = []
for i in hasil:
    count_test += 1
    match = hitung(i)
    if (match):
        final_result = i
```

```
print("\nPOSSIBLE ANSWER:")

tes = allstr
for i in tes:
    for j in i:
        found, idx = searchuniq(j,uniqstr)
        if found:
            j = str(final_result[idx])

column = len(allstr[0])
row = len(allstr)
matrix = [[0 for i in range(column)] for j in range(row)]


for i in allstr:
    for j in range (len(i)):
        found, idx = searchuniq(i[j],uniqstr)
        if found:
            print(final_result[idx],end="")
        else:
            print(i[j], end="")
    print("")
print("\n")


t1 = time.perf_counter() - t0
print("Run time: ", t1,"second(s)")
print("Trials count:",count_test)
print("\n")
```

Hasil *screenshots* dari masukan dan keluaran







```
Enter the filename (include ./test/): ./test/demofile4.txt

PROBLEMS:
 MEMO
 FROM+
-----
HOMER

POSSIBLE ANSWER:
 8485
 7358+
-----
15843


Run time:  201.6035732 second(s)
Trials count: 151200
```

```
Enter the filename (include ./test/): ./test/demofile5.txt

PROBLEMS:
 COCA
 COLA+
-----
OASIS

POSSIBLE ANSWER:
 8186
 8106+
-----
16292


Run time:  35.5001718 second(s)
Trials count: 151200
```

```
Enter the filename (include ./test/): ./test/demofile6.txt

PROBLEMS:
 HERE
  SHE+
-------
COMES

POSSIBLE ANSWER:
 9454
  894+
-------
10348


Run time:  96.8148699 second(s)
Trials count: 604800
```

```
Enter the filename (include ./test/): ./test/demofile7.txt

PROBLEMS:
 CROSS
 ROADS+
------
DANGER

POSSIBLE ANSWER:
 96233
 62513+
------
158746


Run time:  726.5465231000001 second(s)
Trials count: 3628800
```

```
Enter the filename (include ./test/): ./test/demofile8.txt

PROBLEMS:
  CLOCK
   TICK
   TOCK+
------
PLANET

POSSIBLE ANSWER:
  90892
   6592
   6892+
------
104376


Run time:  855.7904586999999 second(s)
Trials count: 3628800
```

Alamat drive yang berisi kode program (eksekusi program jika diperlukan oleh asisten)

https://drive.google.com/drive/u/0/folders/1y7u0LIIha5tV8R6sr9vOnMFQqVzRVUf2

Checklists

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