Laporan Tugas Kecil 1

Cryptarithmetric

Oleh :

Denilsen Axel Candiasa / 13519059

Algoritma Brute Force :

1. Nyatakan huruf unik yang muncul dalam persamaan *cryptarithmetic* sebagai himpunan
2. Kemudian, buat sebuah list baru yang berisi angka dari 0 sampai *len(list huruf)* yang tiap indeks elemennya berkorespondensi dengan indeks list huruf yang muncul dalam persamaan *cryptarithmetic*, contoh :
   List Huruf : [‘A’, ‘B’, ‘C’]
   List Angka : [0, 1, 2]
   Penjelasan : A = 0, B = 1, dan C = 2.
3. Kemudian, hitung apakah list angka sekarang merupakan solusi dari persamaan *cryptarithmetic* yang diberikan, dengan mengganti huruf dengan angka yang berkorenspondensi pada list angka.
4. Apabila list angka tadi bukan merupakan, ubah list angka menjadi permutasi berikutnya dengan memanfaatkan *lexicographical order* untuk menentukan permutasi berikutnya dari list angka yang sebelumnya, contoh :
   [0, 1, 2] -> [0, 2, 1] -> [1, 0, 2] -> [1, 2, 0] -> [2, 0, 1] -> [2, 1, 0]
   Perhatikan bahwa urutan list angka diatas terurut secara *lexicographic.*
5. Apabila permutasi terakhir dari list angka yang diberikan masih bukan solusi, cari kombinasi list angka lain yang memungkinkan, pastikan pula angka dalam list angka tersbut terurut menaik, hal ini agar pencarian permutasi dari list angka berikutnya dapat lengkap, dalam artian semua kemungkinan permutasi dites tepat sekali, contoh :
   List angka yang sebelumnya [0, 1, 2], diubah menjadi [0, 1, 3], apabila permutasi terakhir dari [0, 1, 3] masih bukan solusi, ubah list angka menjadi [0, 1, 4], begini terus sampai list angka [7, 8, 9].
6. Ulangi langkah 3 sampai 5 sampai solusi persamaan *cryptarithmetric* diperoleh. Solus persamaan bisa saja tidak memiliki solusi, dan hal ini terjadi apabila seluruh permutasi yang mungkin dari angka 0 hingga 9 sudah dites dan tidak ditemukan satupun solusi.

Pada *worst case*, algoritma diatas mengecek sebanyak P_n^{10} kemungkinan list angka, dan melakukan sebanyak m kali operasi substitusi untuk mengganti huruf dengan angka yang berkorespondensi, sehingga algoritma ini memliki komplekstias $O(m \cdot P_n^{10})$, dimana m adalah total jumlah huruf dalam persamaan *cryptarithmetic*, dan n merupakan jumlah huruf unik yang ada dalam persamaan *cryptarithmetric*.

Source Code Program (dalam Python) :

Berikut *screenshoot* dari source code program dalam bahasa Python :

```python
import time


def findIndex(elmt, list):
    i = 0
    found = False
    while (i < len(list) and not(found)):
        if(list[i] == elmt):
            found = True
        i += 1

    if(not(found)):
        return -1
    else:
        return i-1


def readFile(fileName):
    f_input = open(fileName, "r")
    words_input = f_input.read().splitlines()
    f_input.close()
    return words_input


```

```python
def makeAlphabetList(wordList):
    alphabetList = ""
    for words in wordList:
        if(words[0] == '-'):
            continue
        else:
            for alphabet in words:
                if((findIndex(alphabet, alphabetList) != -1)):
                    continue
                else:
                    alphabetList += alphabet

    return alphabetList


```

```python
def isAnswer(wordList, alphabetList, numList):
    ans = 0
    num = ""

    if(numList[0] == 0):
        return False

    # Calculating operands
    for words in wordList:
        if(words[0] == '-'):
            break
        else:
            num = ""
            for alphabet in words:
                if(findIndex(alphabet, alphabetList) == -1):
                    continue
                else:
                    num += str(numList[findIndex(alphabet, alphabetList)])
            ans += int(num)

    # Check if equation is true
    num = ""
    for alphabet in wordList[-1]:
        num += str(numList[findIndex(alphabet, alphabetList)])
        if(num[0] == '0'):
            return False

    return int(num) == ans


```

```
70  def printAnswer(wordList, alphabetList, numList):
71      blank = 0
72      for alphabet in alphabetList:
73          print("%s : %d" %
74                (alphabet, numList[findIndex(alphabet, alphabetList)]))
75      print()
76
77      for words in wordList:
78          if(words[0] == '-'):
79              continue
80          if(len(words) > blank):
81              blank = len(words)
82
83      print("SOLUTION :")
84      print()
85
86      for words in wordList:
87          if(words[0] == '-'):
88              print('-'*blank, end="")
89              print('+')
90          else:
91              print(" "*(blank-len(words)), end="")
92              print(words)
93      print()
94
95      for words in wordList:
96          if(words[0] == '-'):
97              print('-'*blank, end="")
98              print('+')
99          else:
100             print(" "*(blank-len(words)), end="")
```

```
101             for i in range(len(words)):
102                 print(numList[findIndex(words[i], alphabetList)], end="")
103             print()
104     print()
105
106
107 def nextPermutation(numList):
108     ans = [0 for i in range(len(numList))]
109     for i in range(len(numList)):
110         ans[i] = numList[i]
111
112     i = len(ans) - 1
113     while i > 0 and ans[i-1] >= ans[i]:
114         i = i - 1
115     if i <= 0:
116         return []
117
118     j = len(ans) - 1
119     while ans[j] <= ans[i - 1]:
120         j = j - 1
121     ans[i-1], ans[j] = ans[j], ans[i-1]
122
123     ans[i:] = ans[len(ans) - 1: i-1: -1]
124
125     return ans
126
127
```

```
def cryptBruteForce(wordList, alphabetList):
    numList = [0 for i in range(len(alphabetList))]
    test = 1
    noSolution = False

    for i in range(len(alphabetList)):
        numList[i] = i

    startTime = time.time()
    currPermutation = numList

    while(not(isAnswer(wordList, alphabetList, currPermutation)) and not(noSolution)):
        test += 1
        currPermutation = nextPermutation(currPermutation)

        if(len(currPermutation) == 0):
            if(len(alphabetList) == 10):
                noSolution = True

            else:
                index = len(alphabetList) - 1
                if(numList[index] == 9):
                    while(numList[index] == 10 - len(alphabetList) + index):
                        index -= 1

                    numList[index] += 1

                    if(numList[0] > 10 - len(alphabetList)):
                        noSolution = True

                    index += 1
```

```

                    while(index < len(alphabetList)):
                        numList[index] = numList[index-1] + 1
                        index += 1

                else:
                    numList[index] += 1

            currPermutation = numList

    if(noSolution):
        print("NO SOLUTION!!")
        input("Tekan enter untuk menutup program!!")
    else:
        printAnswer(wordList, alphabetList, currPermutation)
        print("Jumlah tes :", test)
        print("Waktu pemrosesan : %s detik" % (time.time() - startTime))
        print()


ongoing = True
while(ongoing):
    fileName = input("Masukkan nama file : ")
    root = "../test/" + fileName
    wordList = readFile(root)
    alphabetList = makeAlphabetList(wordList)
    cryptBruteForce(wordList, alphabetList)

```

```python
    cmd = input("Apakah anda masih ingin menghitung persamaan lain? (Y/N): ")

    if(cmd == 'Y'):
        ongoing = True
    elif(cmd == 'N'):
        ongoing = False

```

Screenshoot Input dan Output Program :

1. Input :

```
JUNE
JULY
------+
APRIL
```

Output :

```
Masukkan nama file : input.txt
J : 5
U : 4
N : 8
E : 6
L : 3
Y : 7
A : 1
P : 0
R : 9
I : 2

SOLUTION :

 JUNE
 JULY
-----+
APRIL

 5486
 5437
-----+
10923

Jumlah tes : 2009240
Waktu pemrosesan : 53.56144046783447 detik
```

2. Input :

```
1    SEND
2    MORE
3    ------+
4    MONEY
```

Output :

```
Masukkan nama file : input.txt
S : 9
E : 5
N : 6
D : 7
M : 1
O : 0
R : 8
Y : 2

SOLUTION :

 SEND
 MORE
-----+
MONEY

 9567
 1085
-----+
10652

Jumlah tes : 844280
Waktu pemrosesan : 26.97899603843689 detik
```

3. Input :

```
THREE
THREE
TWO
TWO
ONE
------+
ELEVEN
```

Output :

```
Masukkan nama file : input.txt
T : 8
H : 4
R : 6
E : 1
W : 0
O : 3
N : 9
L : 7
V : 2

SOLUTION :

 THREE
 THREE
   TWO
   TWO
------+
ELEVEN

 84611
 84611
   803
   803
   391
------+
171219

Jumlah tes : 1756932
Waktu pemrosesan : 100.75398325920105 detik
```

4. Input :

input.txt

```
1   NUMBER
2   NUMBER
3   ------+
4   PUZZLE
```

Output :

```
Masukkan nama file : input.txt
N : 2
U : 0
M : 1
B : 6
E : 8
R : 9
P : 4
Z : 3
L : 7

SOLUTION :

NUMBER
NUMBER
------+
PUZZLE

201689
201689
------+
403378

Jumlah tes : 1532493
Waktu pemrosesan : 66.23589611053467 detik
```

5. Input :

input.txt

```
1   TILES
2   PUZZLES
3   ------+
4   PICTURE
```

Output :

```
Masukkan nama file : input.txt
T : 9
I : 1
L : 5
E : 4
S : 2
P : 3
U : 0
Z : 7
C : 6
R : 8

SOLUTION :

  TILES
PUZZLES
-------+
PICTURE

  91542
3077542
-------+
3169084

Jumlah tes : 3328707
Waktu pemrosesan : 159.1497721672058 detik
```

6. Input :

input.txt

```
1    CLOCK
2    TICK
3    TOCK
4    ------+
5    PLANET
```

Output :

```
Masukkan nama file : input.txt
C : 9
L : 0
O : 8
K : 2
T : 6
I : 5
P : 1
A : 4
N : 3
E : 7

SOLUTION :

 CLOCK
  TICK
  TOCK
------+
PLANET

 90892
  6592
  6892
------+
104376

Jumlah tes : 3302475
Waktu pemrosesan : 141.6630232334137 detik
```

7. Input :

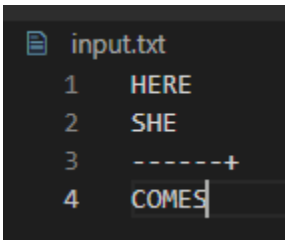
input.txt

```
1    COCA
2    COLA
3    ------+
4    OASIS
```

Output :

```
Masukkan nama file : input.txt
C : 8
O : 1
A : 6
L : 0
S : 2
I : 9

SOLUTION :

 COCA
 COLA
-----+
OASIS

 8186
 8106
-----+
16292

Jumlah tes : 24277
Waktu pemrosesan : 0.5116288661956787 detik
```

8. Input :



Output :

```
Masukkan nama file : input.txt
H : 9
E : 4
R : 5
S : 8
C : 1
O : 0
M : 3

SOLUTION :

 HERE
  SHE
-----+
COMES

 9454
  894
-----+
10348

Jumlah tes : 206373
Waktu pemrosesan : 4.9001617431640625 detik
```

Alamat Drive :

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

Dalam folder Tucil1_13519059

Tabel Ceklis :

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 *operand*	√	