

LAPORAN TUGAS KECIL I IF2211 STRATEGI ALGORITMA  
SEMESTER II TAHUN 2020/2021  
PENYELESAIAN *CRYPTARITHMETIC* DENGAN ALGORITMA *BRUTE  
FORCE*



OLEH  
RAYHAN ALGHIFARI FAUZTA (13519039)

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

## I. Algoritma Brute Force

Tugas kecil ini meminta mahasiswa untuk memecahkan permasalahan *cryptarithmic* dengan membuat program komputer yang memanfaatkan algoritma *brute force*. Pada tugas ini, program yang saya buat menggunakan bahasa pemrograman Python 3 dikarenakan lebih familiar dibanding pilihan bahasa lainnya (C/C++/Java).

Alur program adalah sebagai berikut:

1. Program dimulai dengan meminta masukan file test case yang akan diuji dari pengguna.
2. Mulai menghitung waktu dan mencetak problem yang dibaca dari test case.
3. Inisialisasi parameter-parameter yang dibutuhkan untuk fungsi `solve()`.
4. Mulai mencari solusi dengan fungsi `solve()` yang juga memanfaatkan fungsi `permutations`. Kedua fungsi ini akan dijelaskan lebih lanjut.
5. Ketika solusi ditemukan, program mencetak solusi dengan format sesuai dengan test case. Langkah ini dilakukan di dalam fungsi `solve()`.
6. Terakhir, program akan mencetak waktu eksekusi program dihitung sejak selesai meminta input pengguna hingga solusi berhasil ditemukan. Jumlah total tes yang dilakukan untuk mendapatkan solusi juga akan dicetak setelah dihitung di fungsi `solve()`.
7. Program berakhir.

Fungsi menarik pertama yang dijelaskan adalah fungsi `permutations()`. Fungsi ini memiliki 2 parameter masukan `list_num` (list angka dari 0 sampai 9) dan `r` (panjang elemen permutasi dari `list_num`) digunakan untuk mencari seluruh permutasi dari input list angka untuk kemudian diuji dengan karakter-karakter yang ada. Sebagian besar algoritmanya diambil dari dokumentasi resmi Python 3 ([itertools — Functions creating iterators for efficient looping — Python 3.9.1 documentation](#)). Modifikasi yang dilakukan adalah membuat parameter `r` menjadi wajib ada (tidak boleh None). Cara kerja fungsi ini secara garis besar adalah menyimpan permutasi saat ini pada variabel `index` yang akan diupdate oleh variabel `cycle` (dari prinsip permutasi siklik). Misal elemen `cycle` di indeks ke-*i* saat itu adalah *j*, maka elemen `index` ke-*i* dari kiri akan ditukar dengan elemen ke-*j* dari kanan. Hal ini terus dilakukan hingga seluruh elemen telah ditemukan, tentunya dengan memperhatikan beberapa kasus seperti jika `cycle` mencapai 0.

Fungsi `permutations()` akan digunakan untuk memecahkan permasalahan dalam fungsi `solve()`. Cara kerja fungsi `solve()` adalah meminta beberapa masukan yaitu operands (*op*), hasil (*res*), karakter unik (*char*), dan list angka 0-9 (`list_num`). Cara kerja fungsi secara garis besar adalah untuk setiap elemen permutasi *perm*, akan dilakukan penjumlahan terhadap angka yang merepresentasikan operands dan result. Jumlah operand kemudian akan dicocokkan dengan jumlah result. Jika jumlah sesuai dan jumlah digit result sesuai dengan jumlah huruf pada stringnya, maka solusi akan diprint dan keluar dari fungsi. Setiap iterasinya akan dicatat pada variabel `count` untuk ditampilkan di akhir program.

## II. Source Code

1/27/2021

stima-cryptarithm/main.py at main · rayfazt/stima-cryptarithm

rayfazt / **stima-cryptarithm** (Private)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#)

main

...

stima-cryptarithm / src / main.py / [Code](#) Jump to



rayfazt minor changes

[History](#)

1 contributor

Raw

Blame



125 lines (109 sloc) | 3.74 KB

```
1 import time
2
3 # read from file
4 def read_file(filename):
5     f = open("../test/" + filename, "r")
6     lines = [line.rstrip() for line in f]
7     f.close()
8     return lines
9
10 # get operands and result string from problem
11 def get_op_and_res(input_file):
12     op = []
13     res = [input_file[-1]]
14     for string in input_file[:-2]:
15         op.append(string.replace("+", ""))
16     return op, res
17
18 # get first letter of operands
19 def get_first_letter(list_string):
20     first_char = []
21     for i in range(len(list_string) - 2):
22         letter = list_string[i]
23         first_char.append(letter[0])
24     return first_char
25
26 # create list of unique characters without + and -
27 def unique_char(list_string):
```

<https://github.com/rayfazt/stima-cryptarithm/blob/main/src/main.py>

1/4

```

28     char = []
29     for string in list_string:
30         for letter in string:
31             if letter not in char and letter != "-" and letter != "+":
32                 char.append(letter)
33     return char
34
35 # convert character to integer
36 def to_integer(string, dict_sol):
37     total = ''
38     for char in string:
39         total += str(dict_sol[char])
40     return int(total)
41
42 # algorithm taken from python docs (https://docs.python.org/3/library/itertools.html#itertools.permutations)
43 # Parameters: list_num (list of numbers), r (length of permutation elements)
44 # DISCLAIMER: I don't know if this classify as brute force or not, tried to use backtracking and
45 #             it somehow performed much worse than this "\_('▽')_"
46 def permutations(list_num, r):
47     pool = tuple(list_num)
48     n = len(pool)
49     if r > n:
50         return
51     index = list(range(n))
52     cycle = list(range(n, n-r, -1))
53     yield tuple(pool[i] for i in index[:r])
54     while n:
55         for i in reversed(range(r)):
56             cycle[i] -= 1
57             if cycle[i] == 0:
58                 index[i:] = index[i+1:] + index[i:i+1]
59                 cycle[i] = n - i
60             else:
61                 j = cycle[i]
62                 index[i], index[-j] = index[-j], index[i]
63                 yield tuple(pool[i] for i in index[:r])
64                 break
65         else:
66             return
67
68 # solver
69 # note: can only search for one solution
70 def solve(op, res, char, first, list_num):
71     count = 0
72
73     for perm in permutations(list_num, len(char)):
74         dict_sol = dict(zip(char, perm))
75         count += 1
76         for num in first:

```

```

76         if dict_sol[num] != 0:      # first digit of operands can't be 0
77             total_op = 0
78             total_res = 0
79             for string in op:
80                 total_op += to_integer(string, dict_sol)
81             for string in res:
82                 total_res += to_integer(string, dict_sol)
83
84             # when solution is found, print it
85             if total_op == total_res and len(str(total_res)) == len(res[0]):
86                 print("SOLUTION:")
87                 for string in op[:-1]:
88                     print(to_integer(string, dict_sol))
89                 print(to_integer(op[-1], dict_sol), "+", sep='')
90                 print("-----")
91                 print(total_res)
92                 return count
93
94
95 # main program
96 if __name__ == "__main__":
97     filename = input("Masukkan test case (dengan .txt): ")
98
99     # begin time
100    start = time.time()
101
102    # print problem
103    f = open("../test/" + filename, 'r')
104    print()
105    print("PROBLEM:")
106    print(f.read())
107    print()
108    f.close()
109
110    # initialize parameters
111    input_file = read_file(filename)
112    op, res = get_op_and_res(input_file)
113    char = unique_char(input_file)
114    first = get_first_letter(input_file)
115    list_num = [0,1,2,3,4,5,6,7,8,9]
116
117    # solve
118    count = solve(op, res, char, first, list_num)
119
120    # print performance
121    print("\nWaktu Eksekusi Program: {:.2f}".format(time.time() - start), "detik")
122    print("Jumlah Total Tes:", count)
123

```

### III. Testing

No. Test Case	Hasil
1	<pre>ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test1.txt  PROBLEM: SEND MORE+ ----- MONEY  SOLUTION: 9567 1085+ ----- 10652  Waktu Eksekusi Program: 59.135982275009155 detik Jumlah Total Tes: 1748230  Press enter to exit  </pre>
2	<pre>ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test2.txt  PROBLEM: JUNE JULY+ ----- APRIL  SOLUTION: 5486 5437+ ----- 10923  Waktu Eksekusi Program: 27.45241665840149 detik Jumlah Total Tes: 2009240  Press enter to exit  </pre>

3	<pre> ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test3.txt  PROBLEM: FORTY TEN TEN+ ----- SIXTY  SOLUTION: 29786 850 850+ ----- 31486  Waktu Eksekusi Program: 58.91765880584717 detik Jumlah Total Tes: 1083579  Press enter to exit   </pre>
4	<pre> ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test4.txt  PROBLEM: HERE SHE+ ----- COMES  SOLUTION: 9454 894+ ----- 10348  Waktu Eksekusi Program: 6.438483476638794 detik Jumlah Total Tes: 575302  Press enter to exit   </pre>
5	<pre> ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test5.txt  PROBLEM: MEMO FROM+ ----- HOMER  SOLUTION: 8485 7358+ ----- 15843  Waktu Eksekusi Program: 1.3154842853546143 detik Jumlah Total Tes: 128687  Press enter to exit   </pre>

6	<pre> ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test6.txt  PROBLEM: NO GUN NO+ ----- HUNT  SOLUTION: 87 908 87+ ----- 1082  Waktu Eksekusi Program: 1.9815704822540283 detik Jumlah Total Tes: 134191  Press enter to exit </pre>
7	<pre> ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/tucil 1/src \$ python main.py Masukkan test case (dengan .txt): test7.txt  PROBLEM: NUMBER NUMBER+ ----- PUZZLE  SOLUTION: 201689 201689+ ----- 403378  Waktu Eksekusi Program: 27.207191228866577 detik Jumlah Total Tes: 728504  Press enter to exit </pre>
8	<pre> ASUS@LAPTOP-GQF2KGUJ MINGW64 ~/Documents/college/stima/tucil/stima-cryptarithm/src \$ python main.py Masukkan test case (dengan .txt): test8.txt  PROBLEM: COCA COLA+ ----- OASIS  SOLUTION: 8186 8106+ ----- 16292  Waktu Eksekusi Program: 4.01 detik Jumlah Total Tes: 123695  Press enter to exit </pre>



#### IV. Alamat Source Code & Checklist

Kode dapat diakses di tautan Google Drive berikut:

<https://drive.google.com/drive/u/0/folders/1wb5hmqUxUIWa5SahqD7oxCDjFZB-GyF3>

Atau dapat juga diakses di repositori GitHub:

<https://github.com/rayfazt/stima-cryptarithm>

#### Checklist

No	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 <i>cryptarithmic</i> hanya benar untuk persoalan <i>cryptarithmic</i> dengan dua buah operand		✓
5	Solusi <i>cryptarithmic</i> benar untuk persoalan <i>cryptarithmic</i> untuk lebih dari dua buah operand	✓	