# Laporan
# Tugas Kecil 1 IF2211 Strategi Algoritma
# Penyelesaian *Cryptarithmetic* dengan Algoritma *Brute Force*


oleh

Andres Jerriel Sinabutar / 13519218

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG


2018

## 1. Algoritma Brute-Force

Langkah-langkah algoritma *brute force* yang dilakukan:

1. Pertama, masukan dari file .txt dibaca dan dimasukkan ke dalam sebuah string
2. String tersebut di-*passing* ke dalam fungsi *solve* yang di dalamnya akan dilakukan proses *brute force* untuk menyelesaikan persoalan yang diterima dalam bentuk string
3. Lalu, string akan diiterasi untuk mengambil satu per satu alfabet yang terkandung di dalamnya dan setiap alfabet akan diganti dengan *digit* angka yang berbeda antara 0 sampai 9
4. Angka-angka tersebut dibuat sebagai sebuah *array of boolean* dengan indeks 0 sampai 9 dimana jika nilainya bernilai *false* berarti angka dengan indeks tersebut belum diambil, dan jika bernilai *true* berarti angka sudah diambil
5. Setelah semua alfabet sudah diganti menjadi *digit* angka, maka string akan di-*tokenize* menjadi sebuah ekpresi matematika untuk menguji apakah kombinasi angka yang sudah dipilih sudah memenuhi dan benar secara matematis
6. Jika kombinasi angka tidak memenuhi, langkah ketiga sampai kelima akan diulangi terus-menerus dengan kombinasi angka yang berbeda-beda sampai ditemukan kombinasi angka yang menghasilkan ekspresi matematika yang benar dan sesuai.
7. Jika solusi ditemukan, perulangan dihentikan dan solusinya akan ditampilkan pada *command line interface*. Sebaliknya, jika seluruh kombinasi angka tidak ada yang memenuhi, maka akan ditampilkan pesan *"Puzzle cannot be solved"* untuk memberitahukan bahwa persoalan *cryptarithmetic* tidak bisa diselesaikan.

## 2. *Source Code* dalam Bahasa Pemrograman Java

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Scanner;
import java.util.StringTokenizer;

import static java.lang.Character.isAlphabetic;
import static java.lang.Character.isDigit;

public class Cryptarithmetic {
    public static int totalTest = 0;

    // Method ini digunakan untuk membaca isi dari sebuah file .txt dan
mengembalikannya sebagai sebuah string
    public static String readFileAsString(String fileName) {
        String text = "";

        try {
            text = new String(Files.readAllBytes(Paths.get(fileName)));
        } catch (IOException e) {
            e.printStackTrace();
        }

        return text;
    }
```

```
    // Method ini digunakan untuk melakukan tokenization terhadap string menjadi
sebuah ekspresi penjumlahan matematika
    // dan mengembalikan nilai hasil penjumlahannya
    static int check(String exp) {
        exp = exp.replaceAll("\\+\n", "");
        int val = 0;
        StringTokenizer tokenizer = new StringTokenizer(exp, "\n", true);

        while (tokenizer.hasMoreTokens()) {
            String nextToken = tokenizer.nextToken().trim();
            if (nextToken.equals("\n") || nextToken.equals("")) {
                val += Integer.parseInt(tokenizer.nextToken().trim());
            } else {
                val = Integer.parseInt(nextToken);
            }
        }
        return val;
    }

    // Method ini digunakan untuk melakukan permutasi terhadap problem cryptarithmetic
dan mengembalikan solusi dalam
    // bentuk string jika ada, dan mengembalikan string kosong jika tidak ada solusi
    static String solve(String puzzle, String delimiter) {
        char cc = 0; // Current character

        // Melakukan iterasi ke dalam string dan menyimpan alfabet ke dalam cc
        for (int i = 0; i < puzzle.length(); i++) {
            if (isAlphabetic(puzzle.charAt(i))) {
                cc = puzzle.charAt(i);
                break;
            }
        }

        if (cc == 0) {
            // Jika seluruh karakter dalam string sudah diganti dengan digit angka,
maka akan dilakukan pengujian
            // terhadap kombinasi angka

            totalTest++;    // Menghitung total tes yang dilakukan untuk menemukan
kombinasi angka yang benar
            String[] operands = puzzle.split(delimiter);
            int op1 = check(operands[0]);
            int op2 = check(operands[1]);
            if (op1 == op2) {
                return puzzle;
            } else {
                return "";
            }
        } else {
            // Buat array of numbers [0..9] untuk menyimpan angka-angka yang sudah
digunakan
            boolean[] numbers = new boolean[10];

            // Melakukan iterasi ke dalam string untuk menandai angka-angka yang sudah
digunakan
            for (int i = 0; i < puzzle.length(); i++)
                if (isDigit(puzzle.charAt(i)))
                    numbers[puzzle.charAt(i) - '0'] = true;

            for (int i = 0; i < 10; i++) {
```

```
                if (!numbers[i]) {

                    // Melakukan substitusi alfabet dengan angka sampai ketemu
kombinasi angka yang sesuai
                    String solution = solve(puzzle.replaceAll(String.valueOf(cc),
                            String.valueOf(i)), delimiter);

                    // Jika kombinasi angka sudah teruji benar secara matematis,
                    // kita akan cek apakah ada angka 0 di depan
                    if (!solution.isEmpty()) {
                        String[] split = solution.split("\n");
                        boolean zeroAtLeft = false;

                        for (int j = 0; j < split.length; j++){
                            split[j] = split[j].trim();
                            if (split[j].charAt(0) == '0'){
                                zeroAtLeft = true;
                                break;
                            }
                        }
                        // Jika tidak ada angka 0 di depan, solusi ditemukan dan di-
return ke main
                        if (!zeroAtLeft) {
                            return solution;
                        }
                    }
                }
            }
        }
        return "";
    }

    public static void main(String[] args) {
        System.out.println("Welcome to the Cryptarithmetic Solver Program");
        System.out.println("Enter your file name with directory (ex.
../test/test1.txt): ");
        Scanner scanner = new Scanner(System.in);
        String filename = scanner.nextLine();

        String data = readFileAsString(filename);
        String[] split = data.split("\n");
        String delimiter = split[split.length - 2] + "\n";

        long startTime = System.nanoTime();
        String result = solve(data, delimiter);

        if (result.isEmpty()) {
            System.out.println("Puzzle cannot be solved");
            return;
        }

        String[] result_split = result.split("\n");
        System.out.println("\nSolution: ");

        for (int i = 0; i < result_split.length; i++) {
            if ((i == result_split.length - 3) || (i == result_split.length - 2)) {
                System.out.println(split[i] + "        " + result_split[i]);
            } else {
                System.out.println(split[i] + "         " + result_split[i]);
            }
        }
```

```
        long endTime = System.nanoTime();
        long elapsedTime = (endTime - startTime);

        System.out.println("\nTotal tests: " + totalTest);
        System.out.println("Execution time: " + ((float) elapsedTime / 1000000000) + "
seconds");
    }
}
```

### 3. Input & Output

| Isi file teks (.txt) | Output |
|---|---|
| SEND<br>MORE+<br>------<br>MONEY | Welcome to the Cryptarithmetic Solver Program<br>Enter your file name with directory (ex. ../test/test1.txt):<br>../test/test1.txt<br><br>Solution:<br>SEND 9567<br>MORE+ 1085+<br>------ ------<br>MONEY 10652<br><br>Total tests: 1748230<br>Execution time: 3.716142 seconds |
| NUMBER<br>NUMBER+<br>-------<br>PUZZLE | Welcome to the Cryptarithmetic Solver Program<br>Enter your file name with directory (ex. ../test/test1.txt):<br>../test/test2.txt<br><br>Solution:<br>NUMBER 201689<br>NUMBER+ 201689+<br>------- -------<br>PUZZLE 403378<br><br>Total tests: 728504<br>Execution time: 2.3238392 seconds |

```
  TILES
PUZZLES+
--------
PICTURE
```

```
Welcome to the Cryptarithmetic Solver Program
Enter your file name with directory (ex. ../test/test1.txt):
../test/test3.txt

Solution:
  TILES           91542
PUZZLES+        3077542+
--------        --------
PICTURE         3169084

Total tests: 3328707
Execution time: 10.457681 seconds
```

```
 CLOCK
  TICK
  TOCK+
-------
PLANET
```

```
Enter your file name with directory (ex. ../test/test1.txt):
../test/test4.txt

Solution:
 CLOCK          90892
  TICK           6592
  TOCK+          6892+
-------        -------
PLANET         104376

Total tests: 3302475
Execution time: 11.215711 seconds
```

```
 COCA
 COLA+
------
OASIS
```

```
Welcome to the Cryptarithmetic Solver Program
Enter your file name with directory (ex. ../test/test1.txt):
../test/test5.txt

Solution:
 COCA           8186
 COLA+          8106+
------         ------
OASIS          16292

Total tests: 123695
Execution time: 0.5738983 seconds
```

```
 HERE
  SHE+
------
COMES
```

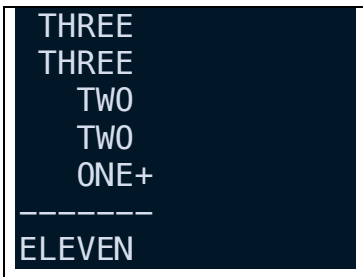
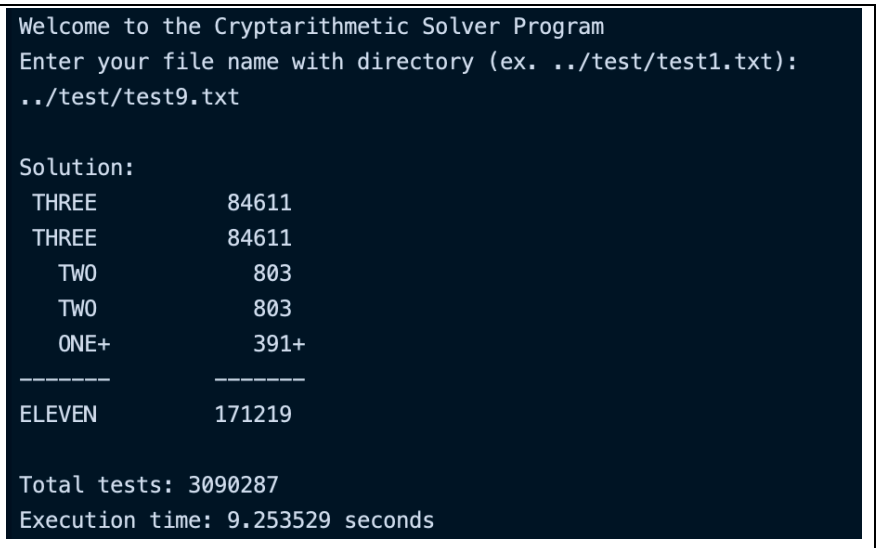
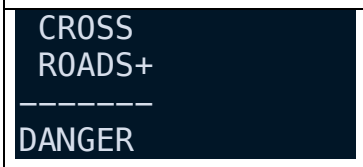
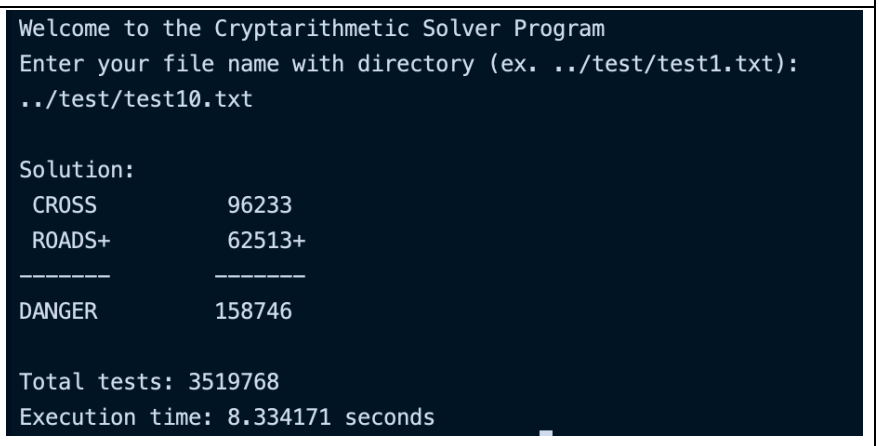
```
Welcome to the Cryptarithmetic Solver Program
Enter your file name with directory (ex. ../test/test1.txt):
../test/test6.txt

Solution:
 HERE          9454
  SHE+          894+
------        ------
COMES         10348

Total tests: 575302
Execution time: 1.3850785 seconds
```

```
 DOUBLE
 DOUBLE
   TOIL+
--------
TROUBLE
```

```
Welcome to the Cryptarithmetic Solver Program
Enter your file name with directory (ex. ../test/test1.txt):
../test/test7.txt

Solution:
 DOUBLE          798064
 DOUBLE          798064
   TOIL+           1936+
--------        --------
TROUBLE         1598064

Total tests: 2898676
Execution time: 7.7527494 seconds
```

```
  NO
 GUN
  NO+
-----
HUNT
```

```
Welcome to the Cryptarithmetic Solver Program
Enter your file name with directory (ex. ../test/test1.txt):
../test/test8.txt

Solution:
  NO            87
 GUN           908
  NO+           87+
-----         -----
HUNT          1082

Total tests: 134191
Execution time: 0.92772776 seconds
```

| | |
|---|---|
|  |  |
|  |  |

### 4. Alamat Github Kode Program

Berikut adalah pranala akses menuju repositori dari kode program ini:
https://github.com/andresjerriels/Tucil1Stima

### 5. *Check List* Program

| 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* hanya benar untuk persoalan *cryptarithmetic* dengan dua buah *operand* | ✓ | |