

**LAPORAN TUGAS KECIL 1**  
**IF2211 STRATEGI ALGORITMA**  
**SEMESTER II TAHUN 2020/2021**  
**PENYELESAIAN *CRYPTARITHMETIC* DENGAN ALGORITMA**  
***BRUTE FORCE***



Disusun oleh:  
Ruhayah Faradishi Widiaputri  
13519034

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2021**

## DAFTAR ISI

<b>BAB 1 ALGORITMA BRUTE FORCE .....</b>	<b>2</b>
<b>BAB 2 SOURCE PROGRAM.....</b>	<b>3</b>
<b>BAB 3 HASIL PERCOBAAN.....</b>	<b>9</b>
<b>BAB 4 EVALUASI DAN LINK KODE PROGRAM.....</b>	<b>17</b>

## BAB 1

### ALGORITMA BRUTE FORCE

Langkah-langkah yang dilakukan dalam penyelesaian *cryptarithmic* dengan algoritma *brute force* yang kami buat adalah sebagai berikut.

1. Membaca *directory* file txt yang berisi persoalan *cryptarithmic* yang dimasukkan oleh pengguna.
2. Membaca file txt yang berisi persoalan *cryptarithmic* yang ingin diselesaikan. Masukan yang sudah dibaca (berupa string perbaris) disimpan dalam tipe data list berkait yang elemennya bertipe data kata. Kemudian waktu mulai disimpan.
3. Persoalan yang terdapat di file txt dicetak ke layar
4. Menghitung ada berapa banyak huruf yang ada di persoalan
5. Mencari solusi persoalan
  - a. Jika banyak huruf yang ada di dalam persoalan  $> 10$  maka persoalan tersebut tidak dapat diselesaikan. Akan tercetak di layar “Tidak ada solusi yang mungkin”
  - b. Jika banyak huruf yang ada di persoalan lebih kecil atau sama dengan 10 maka program akan melakukan langkah sebagai berikut.
    - i. Menyimpan huruf apa saja yang ada di persoalan
    - ii. Menginisialisasi array 10 elemen yang menyimpan angka apa saja yang sudah dipakai dalam kombinasi. Dalam inisialisasi awal ini semua elemen array diisi sama dengan nol yang artinya belum dipakai.
    - iii. Menyimpan huruf apa saja yang nilainya tidak boleh sama dengan nol, yaitu huruf pertama di setiap baris
    - iv. Mencari permutasi angka yang mungkin untuk tiap-tiap huruf.
    - v. Setiap permutasi kemudian dihitung apakah hasilnya memenuhi sebagaimana penyelesaian *cryptarithmic*.
    - vi. Jika hasilnya ditemukan maka solusi *cryptarithmic* tersebut ditampilkan ke layar.
    - vii. Proses pencarian permutasi yang angka yang cocok dilakukan sampai semua kemungkinannya sudah dicek satu-persatu, jadi bisa jadi ada persoalan yang solusinya lebih dari 1.
6. Setelah selesai mencari solusi-solusi yang mungkin waktu akhir disimpan. Waktu eksekusi kemudian dihitung.
7. Menampilkan ke layar total waktu yang dibutuhkan
8. Menampilkan total tes yang dilakukan

## BAB 2

### SOURCE PROGRAM

Adapun source program dalam bahasa C yang kami gunakan dalam menyelesaikan persoalan cryptarithmic adalah sebagai berikut.

1. prog.c (berisi program utama)

```
// RUHIYAH FARADISHI WIDIAPUTRI - 13519034

#include<stdio.h>
#include<stdlib.h>
#include"time.h"
#include "boolean.h"
#include"katachar.h"

char line[20];
combin node;
int angkapakai[10];
int totaltes,totalkombinasi;
char taknol[10]; // char mana saja yang nilainya tidak boleh sama dengan nol
boolean siap;

address Alokasi (char line[]){
    address P;
    P = (address) malloc (1*sizeof(listring));
    if (P != NULL){
        P->word = CharToKata(line);
        P->next = NULL;
    }
}

void InsLast (List *L, char line[]){
    address P, last;
    if (First(*L)==NULL){
        P = Alokasi(line);
        if (P != NULL){
            P->next = L->First;
            L->First = P;
        }
    }
    else{
        P = Alokasi(line);
        if (P != NULL){
            last = L->First;
            while (last->next != NULL){
                last = last->next;
            }
            last->next = P;
            (last->next)->next = NULL;
        }
    }
}
```

```

boolean hurufawal (int idx){
    boolean found = false;
    int i = 0;
    while (!found && i <= 10 ){
        if (node.hu[idx].hur == taknol[i]){
            found = true;
        }
        else{
            i = i+1;
        }
    }
    return found;
}

int carinilai(char c){
    int j = 0;
    int hasil;
    boolean found;
    found = false;
    while (!found && j < 10){
        if (c == node.hu[j].hur){
            hasil = node.hu[j].nilainya;
            found = true;
        }
        else{
            j++;
        }
    }
    return hasil;
}

boolean ketemu(List L){
    int hitung1, hitung2;
    hitung1 = 0;
    hitung2 = 0;
    address P;
    P = First(L);
    while (P->word.TabKata[0] != '-'){
        int pengali = 1;
        for (int i = P->word.Length-1; i >= 0; i--){
            if (P->word.TabKata[i] != '+' && P->word.TabKata[i] != ' '){
                int n = carinilai(P->word.TabKata[i]);
                hitung1 = hitung1 + (pengali * n);
                pengali = pengali * 10;
            }
        }
        P = Next(P);
    }
    P = Next(P);
    int pengali = 1;
    for (int i = P->word.Length-1; i >= 0; i--){
        if (P->word.TabKata[i] != ' '){
            int n = carinilai(P->word.TabKata[i]);
            hitung2 = hitung2 + (pengali * n);
            pengali = pengali * 10;
        }
    }
    return (hitung1 == hitung2);
}

```

```

void salinhasil(List L){
    printf("HASIL YANG DITEMUKAN:\n");
    address P;
    int i, nilai;
    P = L.First;
    while (P != NULL){
        for (i=0; i<P->word.Length; i++){
            if (P->word.TabKata[i] != '-' && P->word.TabKata[i] != '+' && P->word.TabKata[i] != ' '){
                nilai = carinilai(P->word.TabKata[i]);
                printf("%d",nilai);
            }
            else{
                printf("%c",P->word.TabKata[i]);
            }
        }
        printf("\n");
        P = P->next;
    }
    printf("\n");
}

```

```

boolean permutasi (List L, int idx){
    if (idx == node.length - 1){
        for (int a = 0; a < 10; a++){
            if (angkapakai[a] == 0 && (a != 0 || (!hurufawal(idx) && a == 0))){
                node.hu[idx].nilainya = a;
                if (ketemu(L)){
                    salinhasil(L);
                    // return true;
                }
                totaltes++;
            }
        }
        return false;
    }
    for (int b=0; b<10; b++){
        if (angkapakai[b] == 0 && (b != 0 || (!hurufawal(idx) && b == 0))){
            node.hu[idx].nilainya = b;
            angkapakai[b] = 1;
            if (permutasi(L,idx+1)){
                return true;
            }
            angkapakai[b] = 0;
        }
    }
    return false;
}

```

```

int main(){
    int i=0;
    listring s;
    address P;
    List L;
    char namafile[50];

    First(L) = NULL;
    totaltes = 0;
}

```

```

// Membaca dari txt dan menyimpannya
printf("Masukkan directory file soal (misalnya ../test/tes1.txt)\n");
scanf("%s",namafile);
FILE *teks = fopen(namafile,"r");
while (fgets(line,sizeof(line),teks) != NULL){
    |   InsLast(&L,line);
}

// mulai hitung waktu
clock_t start, end;
start = clock();

// mencetak soal ke layar
printf("PERSOALAN CRYPTARITHMETIC: \n");
P = L.First;
while (P != NULL){
    |   for (i=0; i<P->word.Length; i++){
    |       |   printf("%c",P->word.TabKata[i]);
    |   }
    |   printf("\n");
    |   P = P->next;
}
printf("\n");

```

```

// mengecek ada berapa huruf
int hurufapa[26];
for (i=0; i<26; i++){
    |   hurufapa[i] = 0;
}
P = First(L);
while (P != NULL){
    |   for (i=0; i<P->word.Length; i++){
    |       |   if (P->word.TabKata[i] != '+' && P->word.TabKata[i] != '-' && P->word.TabKata[i] != ' '){
    |           |   int indeks = P->word.TabKata[i] - 'A';
    |           |   if (hurufapa[indeks]==0){
    |               |   hurufapa[indeks] = 1;
    |           }
    |       }
    |   }
    |   P = Next(P);
}

int count=0;
for (i=0; i<26; i++){
    |   if (hurufapa[i] != 0){
    |       |   count++;
    |   }
}

```

```

// mencari permutasi yang cocok
if (count > 10){
    printf("Tidak ada solusi yang mungkin\n");
}
else{
    // -- mendata huruf yang ada
    int j = 0;
    for (i=0; i < 26; i++){
        if (hurufapa[i] != 0){
            node.hu[j].hur = (char) (i+'A');
            j++;
        }
    }
    node.length = count;
    // -- membuat angka yang dipakai apa saja hehe
    for (i=0; i<10; i++){
        angkapakai[i] = 0;
    }
    // -- mencatat huruf mana saja yang nilainya tidak boleh 0
    for (int j=0; j<10; j++){
        taknol[j] = '?';
    }
    P = First(L);
    i = 0;
    while (P != NULL){
        if (P->word.TabKata[0] != '-' ){
            int j=0;
            while ((P->word.TabKata[j] == '+' || P->word.TabKata[j] == ' ') && j < P->word.Length){
                j=j+1;
            }
            taknol[i] = P->word.TabKata[j];
            i++;
        }
        P = Next(P);
    }
    // -- mencari permutasi yang cocok
    siap = permutasi(L,0);
    end = clock();
    double time_taken = (double)(end-start)/(double)(CLOCKS_PER_SEC);
    printf("Waktu yang dibutuhkan: %.10lf sekon\n",time_taken);
    printf("Total tes yang dilakukan: %d\n",totaltes);
}
return 0;
}

```

## 2. boolean.h

```

/* Definisi type boolean */

#ifndef _BOOLEAN_h
#define _BOOLEAN_h

#define boolean unsigned char
#define true 1
#define false 0

#endif

```

## 3. katachar.h



```

#ifndef __KATA_CHAR_H_
#define __KATA_CHAR_H_

#define NMax 50

typedef struct {
    char TabKata[NMax]; /* container penyimpan kata, indeks yang dipakai [0..NMax-1] */
    int Length;
} Kata;
typedef struct listr *address;
typedef struct listr{
    Kata word;
    address next;
} listring;
typedef struct {
    address First;
}List;
#define Info(P) (P)->info
#define Next(P) (P)->next
#define First(L) ((L).First)

typedef struct nilaihuruf{
    char hur;
    int nilainya;
} nilaihuruf;
typedef struct combi{
    nilaihuruf hu[10];
    int length;
} combin;

Kata CharToKata(char Line[]);

#endif

```

#### 4. katachar.c

```

#include<stdio.h>
#include"katachar.h"

Kata CharToKata(char Line[]){
    Kata currkata;
    int i = 0;

    while (Line[i] != '\n' && Line[i] != '\0'){
        currkata.TabKata[i] = Line[i];
        i++;
    }
    currkata.Length = i;

    return currkata;
}

```

## BAB 3

### HASIL PERCOBAAN

1. Percobaan ke-1

a. Input (tes1.txt)

```
NUMBER
NUMBER +
-----
PUZZLE
```

b. Output

```
Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes1.txt
PERSOALAN CRYPTARITHMETIC:
NUMBER
NUMBER +
-----
PUZZLE

HASIL YANG DITEMUKAN:
201689
201689 +
-----
403378

Waktu yang dibutuhkan: 0.9360000000 sekon
Total tes yang dilakukan: 2903040
```

2. Percobaan ke-2

a. Input (tes2.txt)

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

b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes2.txt
PERSOALAN CRYPTARITHMETIC:
  TILES
+PUZZLES
-----
  PICTURE

HASIL YANG DITEMUKAN:
   91542
+3077542
-----
 3169084

Waktu yang dibutuhkan: 1.0300000000 sekon
Total tes yang dilakukan: 2903040

```

3. Percobaan ke-3

- a. Input (tes3.txt)

```

CLOCK
  TICK
+ TOCK
-----
PLANET

```

- b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes3.txt
PERSOALAN CRYPTARITHMETIC:
  CLOCK
  TICK
+ TOCK
-----
PLANET

HASIL YANG DITEMUKAN:
  90892
  6592
+ 6892
-----
104376

Waktu yang dibutuhkan: 0.9570000000 sekon
Total tes yang dilakukan: 2540160

```

4. Percobaan ke-4

a. Input (tes4.txt)

```

  COCA
+ COLA
-----
  OASIS

```

b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes4.txt
PERSOALAN CRYPTARITHMETIC:
  COCA
+ COLA
-----
  OASIS

HASIL YANG DITEMUKAN:
  8186
+ 8106
-----
 16292

Waktu yang dibutuhkan: 0.0390000000 sekon
Total tes yang dilakukan: 120960

```

5. Percobaan ke-5

a. Input (tes5.txt)

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

b. Output

```
Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes5.txt
PERSOALAN CRYPTARITHMETIC:
HERE
SHE +
-----
COMES

HASIL YANG DITEMUKAN:
9454
894 +
-----
10348

Waktu yang dibutuhkan: 0.1290000000 sekon
Total tes yang dilakukan: 423360
```

6. Percobaan ke-6

a. Input (tes6.txt)

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

b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes6.txt
PERSOALAN CRYPTARITHMETIC:
  DOUBLE
  DOUBLE
    TOIL +
  -----
TROUBLE

HASIL YANG DITEMUKAN:
  798064
  798064
    1936 +
  -----
1598064

Waktu yang dibutuhkan: 1.1380000000 sekon
Total tes yang dilakukan: 2903040

```

7. Percobaan ke-7
  - a. Input (tes7.txt)

```

  NO
  GUN
  NO +
  ----
HUNT

```

- b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes7.txt
PERSOALAN CRYPTARITHMETIC:
  NO
  GUN
  NO +
----
HUNT

HASIL YANG DITEMUKAN:
  87
  908
  87 +
----
1082

Waktu yang dibutuhkan: 0.0370000000 sekon
Total tes yang dilakukan: 105840

```

8. Percobaan ke-8
  - a. Input (tes8.txt)

```

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

```

- b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes8.txt
PERSOALAN CRYPTARITHMETIC:
  THREE
  THREE
    TWO
    TWO
+  ONE
-----
ELEVEN

HASIL YANG DITEMUKAN:
  84611
  84611
    803
    803
+  391
-----
171219

Waktu yang dibutuhkan: 0.9310000000 sekon
Total tes yang dilakukan: 2540160

```

9. Percobaan ke-9

a. Input (tes9.txt)

```

  CROSS
  ROADS +
-----
  DANGER

```

b. Output



```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes9.txt
PERSOALAN CRYPTARITHMETIC:
  CROSS
  ROADS +
-----
DANGER

HASIL YANG DITEMUKAN:
  96233
  62513 +
-----
158746

Waktu yang dibutuhkan: 0.7910000000 sekon
Total tes yang dilakukan: 2540160

```

10. Percobaan ke-10

- a. Input (tes10.txt)

```

      MEMO
+   FROM
-----
      HOMER

```

- b. Output

```

Masukkan directory file soal (misalnya ../test/tes1.txt)
../test/tes10.txt
PERSOALAN CRYPTARITHMETIC:
  MEMO
+ FROM
-----
  HOMER

HASIL YANG DITEMUKAN:
  8485
+ 7358
-----
 15843

Waktu yang dibutuhkan: 0.0350000000 sekon
Total tes yang dilakukan: 105840

```

## BAB 4

### EVALUASI DAN LINK KODE PROGRAM

#### 1. Evaluasi

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	✓	
2. Program berhasil <i>running</i>	✓	
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	✓	

#### 2. Link *source code*

Github: [https://github.com/ruhiyahfw/stima\\_tucil1.git](https://github.com/ruhiyahfw/stima_tucil1.git)

Drive: <https://drive.google.com/drive/folders/1ALydglMZO-HuzFqkASeIOPwtZhWrdQ-9?usp=sharing>