

**PENYELESAIAN *CRYPTARITHMETIC* (*CRYPTARITHM*)
DENGAN ALGORITMA *BRUTE FORCE***

Laporan Tugas Kecil 1 IF 2211 Strategi Algoritma
Semester II Tahun 2020/2021



Oleh:

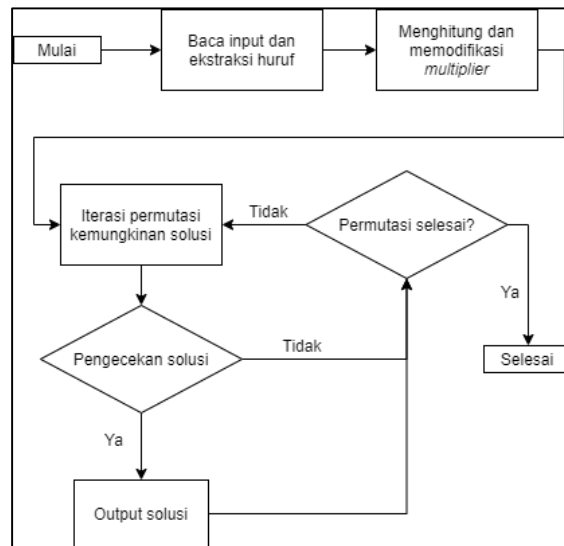
Dionisius Darryl Hermansyah
13519058 / Kelas 02

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

2021

I. ALGORITMA *BRUTE FORCE*

Algoritma *brute force* merupakan sebuah algoritma dengan pendekatan lempang dalam memecahkan sebuah persoalan. Pada tugas kecil kali ini, algoritma *brute force* digunakan untuk menyelesaikan persoalan *cryptarithmic* atau *cryptarithm*, yang merupakan sebuah *puzzle* penjumlahan dalam matematika. Gambar 1 menunjukkan diagram alur algoritma program.



Gambar 1. Diagram alur algoritma program

Berikut merupakan penjelasan lebih lanjut mengenai algoritma penyelesaian *cryptarithm* menggunakan algoritma *brute force* dengan permutasi kemungkinan yang ada:

1. Dari input .txt yang telah dibaca dalam variable string, huruf-huruf alfabet yang ada pada kata-kata input, baik operan maupun hasil *cryptarithm*, diekstraksi dan digabungkan menjadi satu string beralfabet unik. Contohnya:

$$ABC + BCD = CDE$$

maka string yang dihasilkan adalah ABCDE.

2. Mencari *multiplier* atau pengali awal yang akan digunakan dalam algoritma *brute force* dengan menganggap sebuah kata menjadi angka matematik. Contohnya:

$$ABC + BCD = CDE$$

- Pada ABC, nilai A = 100, B = 10, C = 1
- Pada CDE, nilai C = -100, D = -10, E = -1 (Bernilai positif untuk operan dan negatif untuk hasil).

3. Memodifikasi *multiplier* sedemikian rupa dengan aturan, jika operan mengandung huruf yang bersangkutan, maka akan ditambahkan ke *multiplier*, sedangkan jika hasil *cryptoarithm* mengandung huruf yang bersangkutan, maka *multiplier* akan dikurangi. Contohnya:

$$ABC + BCD = CDE$$

nilai *multiplier* keseluruhan adalah

- $A = 100$
 - $B = 10 + 100 = 110$
 - $C = 1 + 10 - 100 = -89$
 - $D = 1 - 10 = -9$
 - $E = -1$
4. Seluruh kemungkinan solusi diiterasi dengan menggunakan teknik permutasi dengan mencoba seluruh kombinasi angka yang mungkin, yakni dari angka 0-9 pada tiap huruf, kecuali huruf yang berada di awal tidak akan diujikan angka 0. Permutasi dilakukan pada list *iterator* yang telah disesuaikan dengan jumlah karakter yang ada.
5. Pada setiap iterasi, seluruh kemungkinan solusi akan diuji kebenarannya dengan aturan, perkalian titik (*dot product*) dari vektor *iterator* dan *multiplier* (dalam hal ini berupa *list*), harus bernilai hasil 0.
6. Melanjutkan iterasi ke tahap 4 hingga seluruh kombinasi permutasi telah selesai diuji.

II. SOURCE CODE PROGRAM

Program ini dibuat menggunakan bahasa C++. *Source code* yang dilampirkan telah disesuaikan kode, komentar, dan indentasinya agar lebih rapi dan singkat. Berikut merupakan *source code* dari program:

```
/* ***** IMPORT LIBRARIES ***** */
#include <iostream>
#include <fstream>
#include <time.h>
#include <string.h>

using namespace std;

/* ***** HEADER FUNGSI ***** */
string getWord(string x);
// Mengekstrak kata dalam bentuk alfabet dari sebuah string alphanumerik
```

```

bool isCharValid(char x);
// Mereturn True apabila sebuah karakter valid
// Karakter valid adalah karakter antara 'A' dan 'Z'
// Karakter yang dianggap 'tak valid' adalah angka atau simbol

bool isCharInWord(char a, string b);
// Mereturn True apabila ada karakter a dalam string b

int findCharIdx(char a, string b);
// Mencari index suatu karakter a pada string b
// Jika a tidak ada dalam b, direturn -1

string eliminateDuplicateChar(string x);
// Menghapus karater yang tidak unik pada string
// Mereturn string dengan karakter unik

bool isAllElUnique(int arr[10]);
// Mengecek apakah semua elemen pada sebuah array of int unik

bool isSolution(int itr[10], int mul[10]);
// Mengecek apakah sebuah kombinasi iterator dan multiplier merupakan solusi
cryptarithms
// Kombinasi dikatakan sebagai solusi, jika total sum perkalian iterator dan
multiplier adalah 0

int power(int x, int y);
// Menghitung x pangkat y dengan brute force

/* ***** MAIN PROGRAM ***** */
int main(){
    bool exit = false;          // Status exit

    cout << "===== CRYPTARITHMS =====" << endl;
    cout << "=   Cryptoarithmic Solver with Brute Force Algorithm   =" << endl;
    cout << "===== " << endl;
    cout << "=               Dionisius Darryl H / 13519058 / K2               =" << endl;
    cout << "===== " << endl;

    while (!exit) {
        /* Inisiasi Variabel */
        string filename;        // Nama file

        int nOp = 0;            // Jumlah operan
        int nSol = 0;            // Jumlah solusi
        int nTest = 0;           // Jumlah test yang dilakukan untuk suatu huruf

        int stt[10];            // Start
        int end[10];            // End

        int itr[10];            // Iterator
        int mul[10];            // Multiplier

        string reader;
        string c_opr[10];        // Operands cryptarithms
        string c_res;            // Hasil cryptarithms
        string c_char;           // Kumpulan huruf dalam cryparithms
        string c_all;            // All string dari cryptarithms

        string exit_option;      // Keluar program atau tidak

        clock_t t_awal;          // Perhitungan waktu awal
        clock_t t_akhir;         // Perhitungan waktu akhir

```

```

/* Read File .txt */
cout << endl << "Masukkan nama file: (Contoh: test-1)" << endl;
cin >> filename;

    filename = "test/" + filename + ".txt";
ifstream file (filename.c_str());

if (file.is_open()){
    while (reader[0] != '+'){
        getline(file, reader);
        c_opr[nOp] = reader;
        c_all += reader + "\n";
        nOp += 1;
    }

    getline (file, reader);    // Operan setelah tanda '+'
    c_all += reader + "\n";

    getline (file, reader);    // Hasil cryparithm
    c_res = reader;
    c_all += reader + "\n";

    file.close();
} else {
    cout << "File tidak ditemukan.";
}

cout << endl << "Masukkan anda: " << endl << c_all;

/* Mulai Perhitungan Waktu */
t_awal = clock();

/* Pengekstrakan Seluruh Alfabet ke dalam String */
for (int i=0; i<nOp; i++){
    c_char += getWord(c_opr[i]);
}

c_char = getWord(c_res)+c_char;
c_char = eliminateDuplicateChar(c_char);
int nChar = c_char.length(); // Banyaknya jumlah karakter

/* Inisiasi nilai array Start, End, Multiplier, dan Iterator */
for (int i=0; i<10; i++){
    stt[i]=0;
    end[i]=0;
    mul[i]=0;
    itr[i]=0;
}

/* Menghitung Multiplier */
for (int i=10-nChar; i<10; i++){
    stt[i] = 0;
    end[i] = 10;
    mul[i] = 0;

    // Proses operands
    string curr_oper;
    int curr_len;

    for (int j=0; j<nOp; j++){
        curr_oper = getWord(c_opr[j]);
        curr_len = curr_oper.length();
    }
}

```

```

        for (int k=curr_len-1; k>=0; k--){
            if (curr_oper[k] == c_char[i-(10-nChar)]){
                mul[i] += power(10, curr_len-k-1);
                if (k == 0){
                    stt[i] = 1;
                }
            }
        }
    }

    // Proses hasil
    curr_oper = getWord(c_res);
    curr_len = curr_oper.length();

    for (int j=curr_len-1; j>=0; j--){
        if (curr_oper[j] == c_char[i-(10-nChar)]){
            mul[i] -= power(10, curr_len-j-1);
            if (j == 0){
                stt[i] = 1;
            }
        }
    }
}

/* Manipulasi Array Start dan End untuk Iterasi*/
for (int i=0; i<10-nChar; i++){
    stt[i] = 10+i;
    end[i] = stt[i]+1;
}

cout << endl << "Menghitung solusi..." << endl;

/* Permutasi Seluruh Kemungkinan Solusi */
for(itr[0] = stt[0]; itr[0]<end[0]; itr[0]++){
    nTest = 0;
    for(itr[1] = stt[1]; itr[1]<end[1]; itr[1]++){
        for(itr[2] = stt[2]; itr[2]<end[2]; itr[2]++){
            for(itr[3] = stt[3]; itr[3]<end[3]; itr[3]++){
                for(itr[4] = stt[4]; itr[4]<end[4]; itr[4]++){
                    for(itr[5] = stt[5]; itr[5]<end[5]; itr[5]++){
                        for(itr[6] = stt[6]; itr[6]<end[6]; itr[6]++){
                            for(itr[7] = stt[7]; itr[7]<end[7]; itr[7]++){
                                for(itr[8] = stt[8]; itr[8]<end[8]; itr[8]++){
                                    for(itr[9] = stt[9]; itr[9]<end[9]; itr[9]++){
                                        if (isAllElUnique(itr)){
                                            if (isSolution(itr, mul)){
                                                nSol += 1;
                                                cout << endl << "Solusi ke-" << nSol << endl;
                                                for (int i=0; i<c_all.length(); i++){
                                                    if (isCharValid(c_all[i])){
                                                        cout << c_all[i];
                                                    }
                                                    else {
                                                        cout << itr[findCharIdx(c_all[i],c_char)+10-nChar];
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    cout << "Diperlukan " << nTest << " kali percobaan untuk menemukan solusi.";
    } else {
        nTest += 1;
    } else {
        nTest += 1;
    }
}
}

```

```

    }
    }
    }
    }
    }
    }
    }
    }

    if (nSol == 0) {
        cout << "Tidak ada solusi yang memenuhi." << endl;
    }

    /* Perhitungan Waktu Eksekusi Program */
    t_akhir = clock();
    cout << "Waktu eksekusi" << (double)(t_akhir-t_awal)/CLOCKS_PER_SEC;

    /* Permintaan Exit Program */
    cout << endl << "Apakah anda ingin memproses file lain? (Y/N)" << endl;
    cin >> exit_option;

    if (exit_option == "N" || exit_option == "n"){
        exit = true;
    }
    cout << endl;
}
cout << "Terima kasih!";
return 0;
}

/* ***** REALISASI FUNGSI ***** */
string getWord(string x){
    int n = 0;
    int len = x.length();
    char x_word[len];

    for (int i=0; i<len; i++){
        if (x[i] >= 65 && x[i] <= 90){
            x_word[n] = x[i];
            n += 1;
        }
    }
    x_word[n] = '\0';
    return x_word;
}

bool isCharValid(char x){
    return x < 'A' || x > 'Z';
}

bool isCharInWord(char a, string b){
    int n = 0;
    bool found = false;

    while (n < b.length() && !found){
        if (b[n] == a){
            found = true;
        }
        n += 1;
    }
    return found;
}

```

```

}

int findCharIdx(char a, string b){
    int n = 0;

    if (!isCharInWord(a, b)){
        return -1;
    }
    while (n < b.length()){
        if (b[n] == a){
            return n;
        }
        n += 1;
    }
}

string eliminateDuplicateChar(string x){
    int n = 0;
    int len = x.length();
    char res[len];

    for (int i=0; i<len; i++){
        if(!isCharInWord(x[i], res)){
            res[n] = x[i];
            n++;
        }
    }
    res[n] = '\0';
    return res;
}

bool isAllElUnique(int arr[10]){
    bool unique = true;

    for (int i=0; i<10; i++){
        for (int j=i+1; j<10; j++){
            if (arr[i] == arr[j]){
                unique = false;
                break;
            }
        }
    }
    return unique;
}

bool isSolution(int itr[10], int mul[10]){
    int sum = 0;

    for (int i=0; i<10; i++){
        sum += itr[i]*mul[i];
    }
    return sum == 0;
}

int power(int x, int y){
    int res = 1;
    for (int i=0; i<y; i++){
        res *= x;
    }
    return res;
}

```


III. INPUT DAN OUTPUT

Berikut ini merupakan contoh tampilan awal program:

```
===== CRYPTARITHMS =====
=   Cryptoarithmic Solver with Brute Force Algorithm   =
=====
=   Dionisius Darryl H / 13519058 / K2                 =
=====

Masukkan nama file: (Contoh: test-1)
Tanpa ekstensi dan file path. Path relative terhadap ./test/
Nama file: test-1
```

Gambar 2. Tampilan awal program

Tabel 1. menunjukkan hasil test case sebanyak 8 kasus *cryptarithms* yang diambil dari website <http://www.cryptarithms.com/default.asp?pg=1>.

Tabel 1. Input dan output program

No	Input	Output
1	DOUBLE DOUBLE + TOIL ----- TROUBLE	Masukkan anda: DOUBLE DOUBLE + TOIL ----- TROUBLE Menghitung solusi... Solusi ke-1 798064 798064 + 1936 ----- 1598064 Diperlukan 53825823 kali percobaan untuk menemukan solusi. Waktu eksekusi program 70.15 detik.
2	COCA	

	+ COLA ----- OASIS	Masukkan anda: COCA + COLA ----- OASIS Menghitung solusi... Solusi ke-1 8186 + 8106 ----- 16292 Diperlukan 56680 kali percobaan untuk menemukan solusi. Waktu eksekusi program 0.079 detik.
3	NUMBER + NUMBER ----- PUZZLE	Masukkan anda: NUMBER + NUMBER ----- PUZZLE Menghitung solusi... Solusi ke-1 201689 + 201689 ----- 403378 Diperlukan 273403169 kali percobaan untuk menemukan solusi. Waktu eksekusi program 78.373 detik.
4	NO GUN + NO ----- HUNT	

		<pre> Masukkan anda: NO GUN + NO ---- HUNT Menghitung solusi... Solusi ke-1 87 908 + 87 ---- 1082 Diperlukan 6551 kali percobaan untuk menemukan solusi. Waktu eksekusi program 0.074 detik. </pre>
5	<pre> MEMO + FROM ----- HOMER </pre>	<pre> Masukkan anda: MEMO + FROM ----- HOMER Menghitung solusi... Solusi ke-1 8485 + 7358 ----- 15843 Diperlukan 47193 kali percobaan untuk menemukan solusi. Waktu eksekusi program 0.077 detik. </pre>
6	<pre> THREE THREE TWO TWO + ONE ----- ELEVEN </pre>	

		<pre> Masukkan anda: THREE THREE TWO TWO + ONE ----- ELEVEN Menghitung solusi... Solusi ke-1 84611 84611 803 803 + 391 ----- 171219 Diperlukan 59116142 kali percobaan untuk menemukan solusi. Waktu eksekusi program 63.43 detik. </pre>
7	<pre> CROSS + ROADS ----- DANGER </pre>	<pre> Masukkan anda: CROSS + ROADS ----- DANGER Menghitung solusi... Solusi ke-1 96233 + 62513 ----- 158746 Diperlukan 47584723 kali percobaan untuk menemukan solusi. Waktu eksekusi program 60.955 detik. </pre>
8	<pre> HERE + SHE ----- COMES </pre>	

		<pre> Masukkan anda: HERE + SHE ----- COMES Menghitung solusi... Solusi ke-1 9454 + 894 ----- 10348 Diperlukan 28255 kali percobaan untuk menemukan solusi. Waktu eksekusi program 0.614 detik. </pre>
--	--	---

Tabel 2. Evaluasi program

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

IV. ALAMAT DRIVE

Berikut merupakan alamat *google drive* dari *source code* yang digunakan berserta file test. Pastikan file test berada dalam directory `./test/` ketika menguji program.

https://drive.google.com/drive/folders/1LZcPuN2k_PeEdLiv8igHOGpAEQeoq6q7?usp=sharing

V. DAFTAR PUSTAKA

- [1] Munir, R. 2021. *Algoritma Brute Force (Bagian 1)*. Bandung: Institut Teknologi Bandung.
- [2] N.N. N.D. *Cryptarithms: Alphametics – Examples*, dilansir dari www.cryptarithms.com.
- [3] N.N. N.D. *Cryptarithm*, dilansir dari www.basic-mathematics.com.