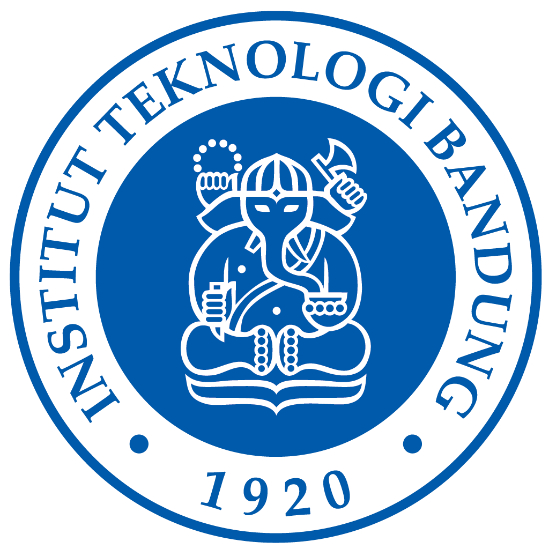
**PENYELESAIAN *CRYPTARITHMETIC* (*CRYPTARITHM*)   
DENGAN ALGORITMA *BRUTE FORCE***

Laporan Tugas Kecil 1 IF 2211 Strategi Algoritma

Semester II Tahun 2020/2021



Oleh:

Dionisius Darryl Hermansyah  
13519058 / Kelas 02

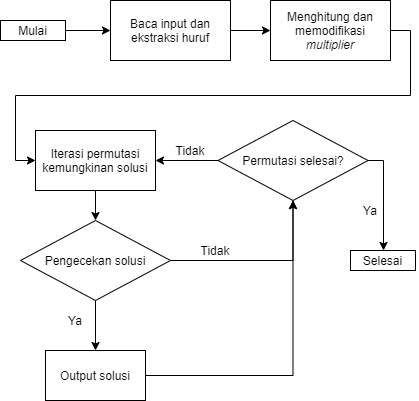
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**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 *cryptarithmetic* 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.

1. 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).

1. 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

1. Seluruh kemungkinan solusi diiterasi dengan menggunakan teknik permutasi dengan mencobakan 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.
2. 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.
3. 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 << "= Cryptoarithmetic 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;

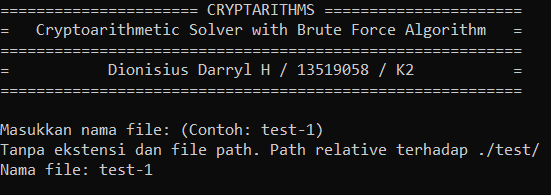
}

return res;

}

**III. INPUT DAN OUTPUT**

Berikut ini merupakan contoh tampilan awal program:



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 |  |
| 2 | COCA  + COLA  ------  OASIS |  |
| 3 | NUMBER  + NUMBER  ------  PUZZLE |  |
| 4 | NO  GUN  + NO  ----  HUNT |  |
| 5 | MEMO  + FROM  -----  HOMER |  |
| 6 | THREE  THREE  TWO  TWO  + ONE  ------  ELEVEN |  |
| 7 | CROSS  + ROADS  ------  DANGER |  |
| 8 | HERE  + SHE  -----  COMES |  |

Tabel 2. Evaluasi progran

|  |  |  |
| --- | --- | --- |
| **Poin** | **Ya** | **Tidak** |
| 1. Program berhasil dikompilasi tanpa kesalahan (*no syntax error*) | V |  |
| 2. Program berhasil *running* | V |  |
| 3. Program dapat membaca file masukan dan menuliskan luaran. | V |  |
| 4. Solusi *cryptarithmetic* hanya benar untuk persoalan *cryptarihtmetic* dengan dua buah *operand*. |  | V |
| 5. Solusi *cryptarithmetic* benar untuk persoalan *cryptarihtmetic* untuk lebih dari dua buah operand. | V |  |

**IV. ALAMAT DRIVE**

Berikut merupakan alamat *repository* GitHub dari *source code* yang digunakan, laporan, program .exe, beserta file test. Pastikan file test berada dalam direktori ./test/ ketika menguji program.

|  |
| --- |
| https://github.com/dionisiusdh/cryptarithmetic-solver |

**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.