Tugas Kecil Strategi Algoritma

Laporan Penyelesaian *Cryptarithmetic* dengan Algoritma Brute Force

Oleh:

Kinantan Arya Bagaspati

13519044

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma Brute Force

Algoritma brute force singkatnya ialah cara penyelesaian masalah dengan program yang meninjau semua kasus yang mungkin muncul berdasarkan deskripsi masalah yang diberikan. Meskipun dengan definisi yang diberikan, tetap ada banyak cara berbeda dalam menyelesaikan satu persoalan yang sama dengan metode brute force. Keragaman ini disebabkan pendekatan yang berbeda dalam mengerjakan soal, beberapa observasi yang dapat mengoptimalkan dan membuang kasus yang tidak perlu, atau pemilihan aspek mana yang di-brute force-kan. Secara keseluruhan, brute force merupakan algoritma yang mudah ditemukan dan diimplementasikan, namun membutuhkan waktu dan memori yang banyak.

Dalam menyelesaikan tugas kecil cryptarithmetic yang diberikan, saya menggunakan brute force yang mengutamakan kecepatan program. Terdapat dua hal utama yang berperan pokok dalam optimasi waktu program saya. Yang pertama ialah program ini melakukan prekomputasi sebanyak mungkin agar tiap pengecekan kasus bisa sesingkat mungkin. Kemudian program hanya meninjau permutasi kemungkinan pemetaan huruf yang ada ke 10 kemungkinan digit, bukannya mengecek semua kemungkinan digit per hurufnya. Terakhir, program tidak meninjau kasus permutasi yang sudah dicek atau melanggar aturan solusi cryptarithmetic. Berikut langkah yang dilakukan program saya secara urut.

Pertama, program akan membaca file input baris demi baris. Tiap baris akan dibersihkan karakter ‘ ‘ dan ‘+’ dan akan disimpan kata demi kata dalam array of string operand[] dengan variabel nbOperand dan len masing-masing menyatakan banyak operand terbaca dan panjang kata terpanjang. Pembacaan terus berlangsung hingga terbaca baris dengan karakter pertama ‘-‘, kemudian kata berikutnya akan disimpan dalam operand[nbOperand].

Kemudian disinilah perkomputasi dilakukan. Setiap string operand akan ditinjau karakter per karakter sehingga tiap karakter unik yang ditemukan disimpan dalam array letters. Array of boolean isFirst menyimpan apakah karakter ke-i dalam array menjadi karakter pertama dalam suatu kata atau tidak. Array multiplier menyimpan pengali tiap karakter, dan digunakan saat perbandingan dalam kasus2 yang ditinjau nanti. Terakhir, lokasi/indeks tiap karakter tersimpan dalam letters disimpan dalam array loc sesuai ASCII dari karakter tersebut.

Kemudian dimulailah proses bruteforce dengan semua optimasi yang ada. Pertama program menggunakan array bernama permutasi untuk mensubstitusikan nilai ke tiap karakter dalam letters. Namun program tidak meninjau semua permutasi bila karakter yang lebih kecil dari 10, melainkan program menghitung variabel skip yakni (10-nbKarakter)!, sehingga brute force dilakukan pada permutasi ke - 0, skip, 2*skip, dan seterusnya. Oleh karena itu, terdapat blok kode di awal setiap kasus yang menerima bilangan x dan menghasilkan sebuah permutasi ke-x terkecil secara leksikografis.

Permutasi ini kemudian disubstitusikan ke masing-masing karakter, dicek apakah tidak ada yang melanggar huruf pertama bernilai 0, serta dibandingkan jumlah semua operand dengan hasil menggunakan pengali yang sudah diprekomputasi. Apabila diperoleh kecocokan, hasil akan langsung diprint sesuai format yang ada. Kemudian di akhhir, program akan menampilkan banyak solusi yang diperoleh, pencocokan yang terjadi, serta waktu yang dibutuhkan.

B. Source Program

tucil1 > cryptarithms.cpp > ...

```cpp
#include<bits/stdc++.h>
#include <fstream>
using namespace std;

const int mxOperand = 100000;
string operand[mxOperand];

int main () {
    clock_t start, end;
    string line, add = " ";
    ifstream input ("input.txt");
    if (input.is_open()){
        //Membaca file dan menyimpan semua operand
        int nbOperand = 0, len = 0;
        bool isOperand = true;
        while(getline(input, line)){
            if(isOperand){
                if(line[0] == '-'){
                    isOperand = false;
                }else{
                    operand[nbOperand] = "";
                    for(int i=0; i<line.length(); i++){
                        if(line[i]!=' ' && line[i]!='+'){
                            add[0] = line[i];
                            operand[nbOperand] += add;
                        }
                    }
                    len = max(len, (int) operand[nbOperand].length());
                    nbOperand++;
                }
            }else{
                operand[nbOperand] = "";
                for(int i=0; i<line.length(); i++){
                    if(line[i]!=' '){
                        add[0] = line[i];
                        operand[nbOperand] += add;
                    }
                }
                len = max(len, (int) operand[nbOperand].length());
            }
        }
```

```
        //Prekomputasi tiap karakter disimpan dalam letters,
        //karakter pertama atau tidak disimpan dalam isFirst,
        //pengali tiap karakter dalam multiplier,
        //serta lokasi penyimpanan (idx) tiap karakter dalam letters disimpan dalam loc sesuai ASCII
        start = clock();
        char letters[10];
        bool isFirst[10], found;
        int nbLetters = 0, curOperand = 0, slot, multiplier[11], mul, loc[256];
        for(int i=0; i<10; i++){
            isFirst[i] = false;
            multiplier[i] = 0;
        }
        while(nbLetters <= 10 && curOperand<=nbOperand){
            int mul = 1;
            for(int i=operand[curOperand].length()-1; i>0 && nbLetters<=10; i--){
                found = false;
                slot = 0;
                while(!found && slot<nbLetters){
                    if(letters[slot] == operand[curOperand][i]){
                        found = true;
                    }else{
                        slot++;
                    }
                }
                if(found){
                    multiplier[slot] += mul;
                }else{
                    if(slot<10){
                        letters[slot] = operand[curOperand][i];
                        loc[operand[curOperand][i]] = slot;
                        multiplier[slot] += mul;
                    }
                    nbLetters++;
                }
                mul*=10;
            }
            found = false;
            slot = 0;
            while(!found && slot<nbLetters){
                if(letters[slot] == operand[curOperand][0]){
                    found = true;
                }else{
                    slot++;
                }
            }
            if(found){
                multiplier[slot] += mul;
                isFirst[slot] = true;
            }else{
                if(slot<10){
                    letters[slot] = operand[curOperand][0];
                    loc[operand[curOperand][0]] = slot;
                    multiplier[slot] += mul;
                    isFirst[slot] = true;
                }
                nbLetters++;
            }
            mul*=10;
            curOperand++;
        }
```

```
104         //Penulisan persamaan kembali dalam terminal secara rata kanan
105         for(int i=0; i<nbOperand; i++){
106             cout << endl;
107             int curlen = operand[i].length();
108             for(int j=0; j<len - curlen; j++){
109                 cout << " ";
110             }
111             cout << operand[i];
112         }
113         cout << "+" << endl;
114         for(int i=0; i<=len; i++){
115             cout << "-";
116         }
117         cout << endl;
118         for(int i=0; i<len - operand[nbOperand].length(); i++){
119             cout << " ";
120         }
121         cout << operand[nbOperand] << endl;
122 
123         if(nbLetters>10){
124             cout << endl << "More than 10 letters included, hence no solution" << endl;
125         }else{
126             //Inisialisasi permutasi, nilai-nilai faktorial, banyak solusi yang diperoleh
127             int permutation[10], used[10], factorial[11], sum, result, value[256], curlen, solutions = 0;
128             bool violate, useCheck;
129             factorial[0] = 1;
130             for(int i=1; i<=10; i++){
131                 factorial[i] = factorial[i-1]*i;
132                 used[i-1] = false;
133             }
134             int skip = factorial[10-nbLetters], counter, countUse, currentUse, comparations = 0;
135             for(int brute=0; brute < factorial[10]; brute += skip){
136                 //Blok kode dibawah men-generate permutasi ke-brute terkecil (leksikografis)
137                 useCheck = used[0];
138                 counter = brute;
139                 for(int i=9; i>=0; i--){
140                     countUse = counter/factorial[i];
141                     currentUse = 0;
142                     while((used[currentUse] != useCheck) || countUse>0){
143                         if(used[currentUse] == useCheck){
144                             countUse--;
145                         }
146                         currentUse++;
147                     }
148                     used[currentUse] = !useCheck;
149                     permutation[9-i] = currentUse;
150                     counter %= factorial[i];
151                 }
152                 //sum dan result akan dibandingkan, sambil melihat pelanggaran huruf pertama
153                 sum = 0;
154                 violate = false;
155                 for(int i=0; i<nbLetters; i++){
156                     sum += permutation[i]*multiplier[i];
157                     value[letters[i]] = permutation[i];
158                     if(permutation[i] == 0 && isFirst[i]){
159                         violate = true;
160                     }
161                 }
162                 result = 0;
163                 for(int i=0; i<operand[nbOperand].length(); i++){
164                     result *=10;
165                     result += value[operand[nbOperand][i]];
166                 }
```

```
                if(sum == 2*result && !violate){
                    //Print solusi
                    for(int i=0; i<nbOperand; i++){
                        cout << endl;
                        curlen = operand[i].length();
                        for(int j=0; j<len - curlen; j++){
                            cout << " ";
                        }
                        for(int j=0; j<curlen; j++){
                            cout << value[operand[i][j]];
                        }
                    }
                    cout << "+" << endl;
                    for(int i=0; i<=len; i++){
                        cout << "-";
                    }
                    cout << endl;
                    curlen = operand[nbOperand].length();
                    for(int i=0; i<len - curlen; i++){
                        cout << " ";
                    }
                    for(int i=0; i<curlen; i++){
                        cout << value[operand[nbOperand][i]];
                    }
                    cout << endl;
                    solutions++;
                }
                if(!violate){
                    comparations++;
                }
            }
            cout << endl << "Total: " << solutions << " solution(s) acquired" << endl;
            cout << "Total comparations: " << comparations << endl;
        }
        end = clock();
        double time_taken = double(end - start) / double(CLOCKS_PER_SEC);
        cout << "Time taken by program is : " << fixed
            << time_taken << setprecision(5);
        cout << " sec " << endl;
        input.close();
    }else{
        cout << "Unable to open file";
    }
    return 0;
}
```

C. Input dan Output

tucil1 > input.txt

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

```

```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

NUMBER
NUMBER+
-------
PUZZLE

201689
201689+
-------
403378

Total: 1 solution(s) acquired
Total comparations: 2903040
Time taken by program is : 1.204000 sec
```

tucil1 > input.txt

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

```

```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

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

  91542
3077542+
--------
3169084

Total: 1 solution(s) acquired
Total comparations: 2903040
Time taken by program is : 1.203000 sec
```

tucil1 > input.txt

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

```

```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

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

 90892
  6592
  6892+
-------
104376

Total: 1 solution(s) acquired
Total comparations: 2540160
Time taken by program is : 1.205000 sec
```

tucil1 > input.txt

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

```

```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

 COCA
 COLA+
------
OASIS

 8186
 8106+
------
16292

Total: 1 solution(s) acquired
Total comparations: 120960
Time taken by program is : 0.076000 sec
```

tucil1 > input.txt

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

```

```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

 HERE
  SHE+
------
COMES

 9454
  894+
------
10348

Total: 1 solution(s) acquired
Total comparations: 423360
Time taken by program is : 0.225000 sec
```

tucil1 > input.txt

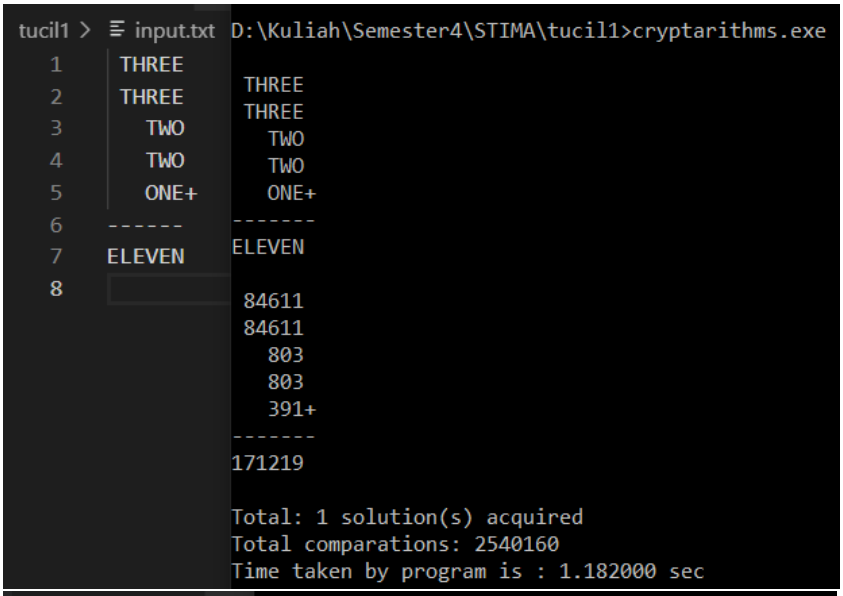
```
DOUBLE
DOUBLE
  TOIL+
--------
TROUBLE

```

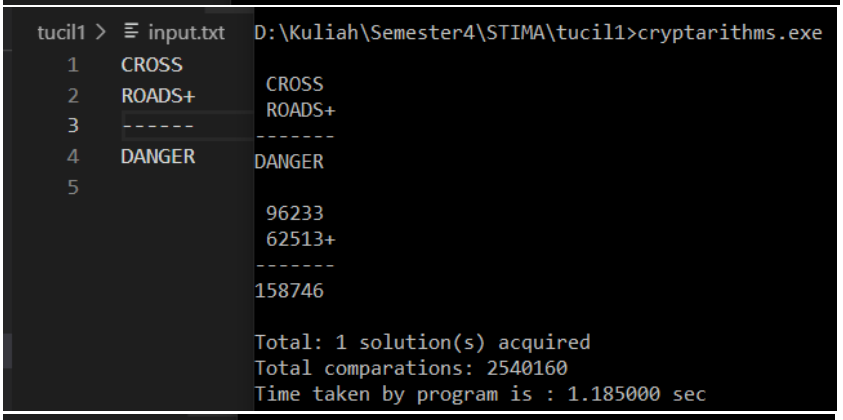
```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

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

 798064
 798064
   1936+
--------
1598064

Total: 1 solution(s) acquired
Total comparations: 2903040
Time taken by program is : 1.232000 sec
```

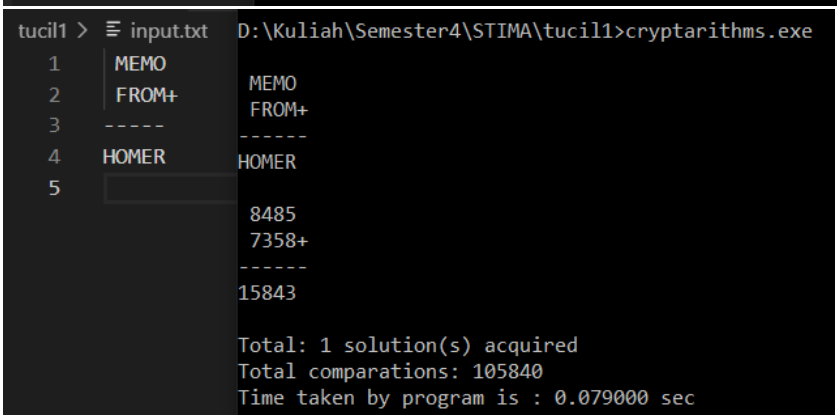
tucil1 > input.txt

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

```

```
D:\Kuliah\Semester4\STIMA\tucil1>cryptarithms.exe

  NO
 GUN
  NO+
-----
HUNT

  87
 908
  87+
-----
1082

Total: 1 solution(s) acquired
Total comparations: 105840
Time taken by program is : 0.079000 sec
```







D.Alamat Drive

https://drive.google.com/drive/u/1/folders/1j6dxA34jl_ooia1Cx7bE-1LFLIj6dMq_

E. Tabel Ceklist

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	√	
Program berhasil running	√	
Program dapat membaca file masukan dan menuliskan luaran	√	
Solusi cryptarithmetic hanya benar untuk persoalan cryptarithmetic dengan dua buah operand.		√
Solusi cryptarithmetic benar untuk persoalan cryptarihtmetic untuk lebih dari dua buah operand.	√	