Laporan Tugas Kecil 1 Strategi Algoritma If2211

Penyelsaian Cryptarithmetic Dengan Algoritma Bruteforce

Semester II 2020/2021

Disusun oleh:

Randy Zakya Suchrady

NIM 13519061

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

Daftar Isi

BAGIAN I ALGORITMA BRUTE FORCE

Berikut langkah-langkah yang dilakukan di dalam program:

1. Membaca file txt dan ubah menjadi array semua kata di dalamnya
2. Buat array atau string dictionary yang berisi tiap huruf yang muncul di dalam file tersebut (missal : A,B,C,D,E dan tidak boleh muncul dua kali dalam array ini)
3. Hitung jumlah huruf dalam dictionary, hal tersebut akan menentukan nilai awal dan nilai akhir untuk loop (missal : jika jumlah hurufnya ada 4 maka angka awal iterasi adalah 0123 dan angka akhirnya 9876)

   *note : missal ABCD dan iterasinya 0123 maksudnya adalah A = 0 , B = 1 , C = 2 , D = 3

4. Looping dari nilai awal dan akhir, namun lakukan Teknik heuristik pada nilai iterasi yang memiliki kombinasi angka tidak unik menjadi angka unik selanjutnya (missal : 3257520 menjadi 3257601)
5. Setiap nilai iterasi yang memiliki kombinasi angka yang unik, akan dimasukkan ke dalam array yang berisi kemungkinan-kemungkinan
6. Lakukan looping dari indeks awal array kombinasi angka unik sampai akhir, lakukan pencocokan dengan urutan huruf pada kata di array no. 1 , jika cocok dan penjumlahan operand sama hasilnya dengan operator maka masukan elemen tersebut ke array baru yang berisi kombinasi yang sukses
7. Tampilkan seluruh solusi (jika hanya satu maka dimunculkan 1 solusi saja) beserta waktu tempuh dan jumlah percobaan

BAGIAN II SOURCE CODE

Source code ditulis dalam bahasa C++ yaitu sebagai berikut:

Bisa diakses di https://github.com/rdyzakya/tucil1stima

```
#include <iostream> //input output
#include <vector> //dynamically allocated array
#include <string> //string
#include <fstream> //file open
#include <chrono> //time
#include <algorithm> //bersihin kata dari spasi , garis , +

//directory needs
#include <windows.h>
using namespace std;
using namespace std::chrono;

string getTestDirectoryOnWindows()
{
    const unsigned long maxDir = 260;
    char currentDir[maxDir];
    GetCurrentDirectory(maxDir, currentDir);
    string S = string(currentDir);
    int length = S.size();
    S.erase(length-3,length-1);
    return S + "test\\";
}

//pangkat
int power(int basis, int pow){
        int result = 1;
        for (int i = 0; i < pow; ++i)
```

```
	{
		result *= basis;
	}
	return result;
}

//typecasting string ke int
int stringToInt(string myString){
	int sum = 0;
	for (int i = 0; i < myString.size(); ++i)
	{
		int index = myString.size()-1-i;
		int number = myString[index] - '0';
		sum += number*power(10,i);
	}
	return sum;
}

//typecasting int ke "string" berelemen satu, lupa kenapa ga char
string intToChar(int I){
	switch(I){
		case (0):
			return "0";
		case (1):
			return "1";
		case (2):
			return "2";
		case (3):
			return "3";
		case (4):
			return "4";
		case (5):
			return "5";
```

```
		case (6):
			return "6";
		case (7):
			return "7";
		case (8):
			return "8";
		case (9):
			return "9";
		}
	return "0";
}

//typecasting int ke string
string intToString(int myNumber){
	int num = myNumber;
	string S = "";
	while(num != 0){
		int remainder = num % 10;
		num = num / 10;
		string remainderChar = intToChar(remainder);
		S = remainderChar + S;
	}
	return S;
}

//typecasting long long ke string
string longlongToString(long long myNumber){
	long long num = myNumber;
	string S = "";
	while(num != 0){
		int remainder = num % 10;
		num = num / 10;
		string remainderChar = intToChar(remainder);
```

```
                S = remainderChar + S;
        }
        return S;
}

//typecasting string ke long long
long long stringToLongLong(string myString){
        long long sum = 0;
        for (int i = 0; i < myString.size(); ++i)
        {
                int index = myString.size()-1-i;
                long long number = myString[index] - '0';
                sum += number*power(10,i);
        }
        return sum;
}

//membersihkan line pada file yang ada spasi, garis putus2, dan tanda tambah
vector<string> removeUnnecessary(vector<string> uncleanVector){
        vector<string> cleanVector;
        int length = uncleanVector.size();
        for (int i = 0; i < length; ++i)
        {
                if (i != length-2)
                {
                        string cleanString = uncleanVector[i];
                        string dumpChar = "\n+ ";
                        for(int j = 0; j < dumpChar.size();++j){

        cleanString.erase(remove(cleanString.begin(),cleanString.end(),dumpChar[j]),cleanString
.end());
                        }
                        cleanVector.push_back(cleanString);
                }
```

```
	}
	return cleanVector;
}

bool foundCharInString(char C, string S){
	for (int i = 0; i < S.size(); ++i)
	{
		if (C == S[i])
		{
			return true;
		}
	}
	return false;
}

//melihat apakah komponen pada suatu string setiap karakternya unik
bool isDistinct(string S){
	for (int i = 0; i < S.size(); ++i)
	{
		for (int j = i+1; j < S.size(); ++j)
		{
			if (S[i] == S[j])
			{
				return false;
			}
		}
	}
	return true;
}

//berisi karakter apa saja yang muncul pada sebuah file dengan masing-masing huruf hanya ada satu pada return value
string allCharInArrayUnique(vector<string> arrayOfString){
	string result = "";
```

```
	for (int i = 0; i < arrayOfString.size(); ++i)
	{
		for (int j = 0; j < arrayOfString[i].size(); ++j)
		{
			if (!foundCharInString(arrayOfString[i][j],result))
			{
				result += arrayOfString[i][j];
			}
		}
	}
	return result;
}


//memulai permutasi dengan angka 0123456789 (jika digit atau jumlah huruf ada 10) , mulai dengan 012 jika huruf ada 3
long long initiateNumber(int digit){
	string result = "";
	if (digit <= 10)
	{
		for (int i = 1; i < digit; ++i)
		{
			result += intToChar(i);
		}
		int resultint = stringToInt(result);
		return (long long)resultint;
	}
	return -1;
}


//akhir dari permutasi 9876...0 sesuaikan dengan digit
long long lastNumber(int digit){
	if (digit < 10)
	{
		string result = "";
```

```
		for (int i = 0; i < digit; ++i)
		{
			result += intToChar(9-i);
		}
		return (long long)stringToInt(result);
	}else if (digit == 10)
	{
		return 9876543210;
	}
	return -1;
}

//jika ada string (diaplikasikannya kepada string yang berisi angka), misal 01204 , maka akan mengembalikan
//indeks 0 yang ada pada indeks ke3
int largestNotUniqueIndex(string myNumber){
	for (int i = 0; i < myNumber.size(); ++i)
	{
		for (int j = i+1; j < myNumber.size(); ++j)
		{
			if (myNumber[i] == myNumber[j])
			{
				return j;
			}
		}
	}
	return -1;
}

//ngecek suatu char indeks keberapa pada sebuah string
int whatIndex(char C, string S){
	for (int i = 0; i < S.size(); ++i)
	{
		if (C == S[i])
```

```
		{
			return i;
		}
	}
	return -1;
}

//membuat angka yang tidak unik menjadi unik (tidak ada angka yang berulang seperti 12341)
void findNextUnique(long long * realNumber, int sizebox){
	string stringNumber = longlongToString(*realNumber);
	int longDigit = stringNumber.size();
	if (longDigit < sizebox){ //misal 0123 kan jadi 123
		stringNumber = "0" + stringNumber;
	}
	if (!isDistinct(stringNumber))
	{
		int notUniqueIndex = largestNotUniqueIndex(stringNumber);
		string notUniqueNumber = "";
		notUniqueNumber += stringNumber[notUniqueIndex];
		for (int i = notUniqueIndex+1; i < stringNumber.size(); ++i)
		{
			stringNumber[i] = '0';
		}
		int pow = stringNumber.size()-1- notUniqueIndex;
		long long tenPower = power(10,pow);
		*realNumber = stringToLongLong(stringNumber);
		*realNumber += tenPower;
		findNextUnique(realNumber,sizebox);
	}
}

//generate permutasi
vector<string> generatePossibilities(string allCharInArrayUnique){
```

```
	vector<string> result;
	int digit = allCharInArrayUnique.size();
	if (digit <= 10)
	{
		for (long long i = initiateNumber(digit); i <= lastNumber(digit); ++i)
		{
			findNextUnique(&i,digit);
			if (longlongToString(i).size() < digit)
			{
				result.push_back("0" + longlongToString(i));
			}else{
				result.push_back(longlongToString(i));
			}
		}
	}
	return result;
}

//mencocokan resultan permutasi dengan urutan huruf
string matchNumbersAndChar(string word, string allCharInArrayUnique, string numberString){
	string result = "";
	for (int i = 0; i < word.size(); ++i)
	{
		int index = whatIndex(word[i],allCharInArrayUnique);
		result += numberString[index];
	}
	return result;
}

//true jika digit awal 0
bool isFrontNumberZero(string numberString){
	if (numberString.size() > 1)
	{
```

```
		return numberString[0] == '0';
	}
	return false;
}

//ngecek di sebuah kombinasi angka yang merepresentasikan huruf
//jika ada yang digitnya 0 di awal langsung true
bool isFoundFrontNumberZero(vector<string> arrayOfString, string possibilitiesElmt, string
allCharInArrayUnique){
	for (int i = 0; i < arrayOfString.size(); ++i)
	{
		string numberString =
matchNumbersAndChar(arrayOfString[i],allCharInArrayUnique,possibilitiesElmt);
		if (isFrontNumberZero(numberString))
		{
			return true;
		}
	}
	return false;
}

//memasukkan hasil permutasi yang cocok dengan persoalan (solusinya)
vector<string> solution(vector<string> arrayOfString , long long * tries){
	vector<string> result;
	string arrChar = allCharInArrayUnique(arrayOfString);
	long long trial = 0;
	if (arrChar.size() <= 10)
	{
		vector<string> possibilities = generatePossibilities(arrChar);
		for (int i = 0; i < possibilities.size(); ++i)
		{
			if (!isFoundFrontNumberZero(arrayOfString,possibilities[i],arrChar))
			{
				int sum = 0;
```

```
                                for (int j = 0; j < arrayOfString.size()-1; ++j)
                                {
                                        sum +=
stringToInt(matchNumbersAndChar(arrayOfString[j],arrChar,possibilities[i]));
                                }
                                if (sum ==
stringToInt(matchNumbersAndChar(arrayOfString[arrayOfString.size()-
1],arrChar,possibilities[i])))
                                {
                                        result.push_back(possibilities[i]);
                                }
                                trial = trial + 1;;
                        }
                }
        }
        *tries = trial;
        return result;
}


//output solusi
void printOutSolution(vector<string> uncleanVector,vector<string> arrayOfString, string
allCharInArrayUnique, string resultPossibility){
        for (int i = 0; i < arrayOfString.size(); ++i)
        {
                for (int j = 0; j < uncleanVector[i].size(); ++j)
                {
                        if (uncleanVector[i][j] == ' ')
                        {
                                cout << " ";
                        }
                }
                cout <<
matchNumbersAndChar(arrayOfString[i],allCharInArrayUnique,resultPossibility);
                if (i == arrayOfString.size()-2)
                {
```

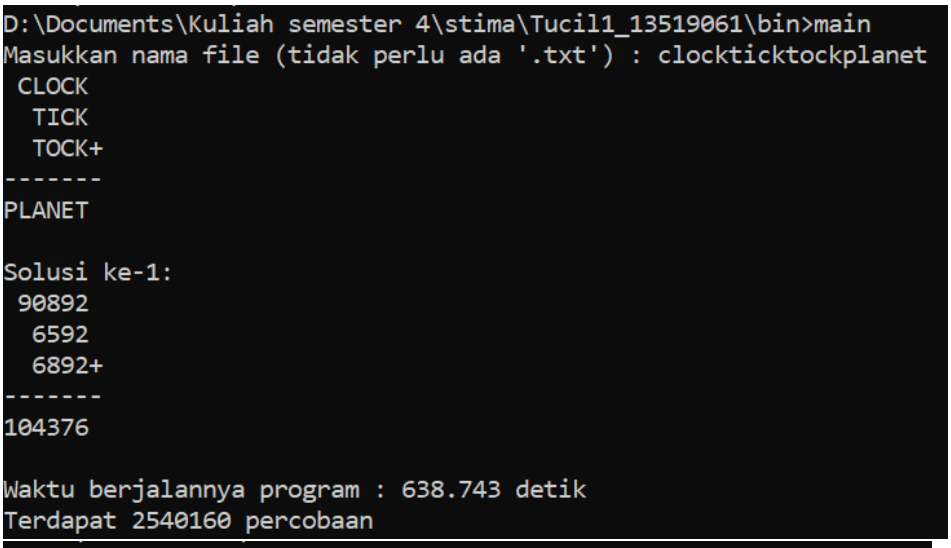
```
                        cout << "+" << endl << uncleanVector[uncleanVector.size()-2];
                }
                cout << endl;
        }
}


int main() {
        cout << "Masukkan nama file (tidak perlu ada '.txt') : ";
        string namaFile;
        cin >> namaFile;
        ifstream file( getTestDirectoryOnWindows() + namaFile + ".txt");
        auto start = high_resolution_clock::now();
        string lines;
        vector<string> arraySemuaLine;
        while(getline(file,lines)){
                arraySemuaLine.push_back(lines);
        }
        file.close();
        vector<string> arr = removeUnnecessary(arraySemuaLine);
        long long tries;
        vector<string> result = solution(arr,&tries);
        for (int j = 0; j < arraySemuaLine.size(); ++j)
        {
                cout << arraySemuaLine[j] << endl;
        }
        cout << endl;
        for (int i = 0; i < result.size(); ++i)
        {
                cout << "Solusi ke-" << i+1 << ":" << endl;
                printOutSolution(arraySemuaLine,arr,allCharInArrayUnique(arr),result[i]);
                cout << endl;
        }
```

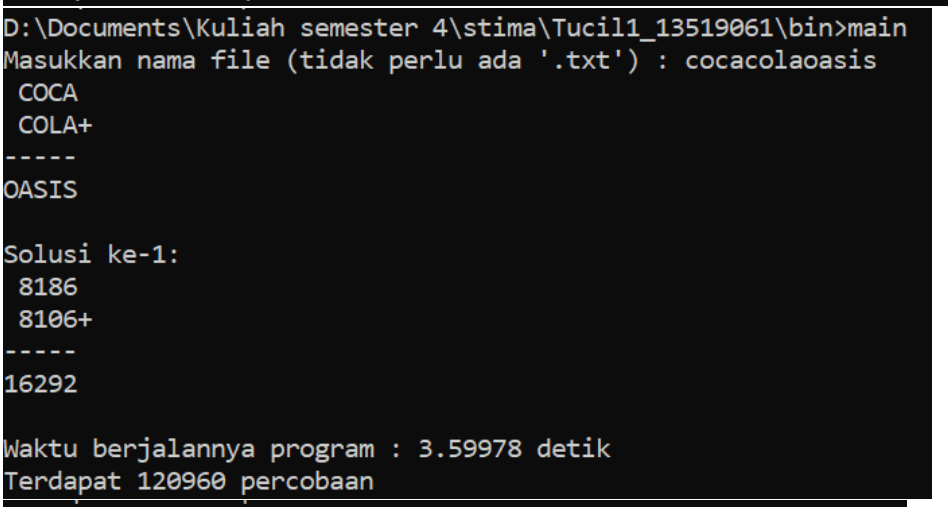
```
        auto stop = high_resolution_clock::now();
        auto duration = duration_cast<microseconds>(stop-start);
        cout << "Waktu berjalannya program : " << double(duration.count())/double(1000000);
        cout << " detik" << endl;
        cout << "Terdapat " << tries << " percobaan";
        return 0;
}
```

BAGIAN III HASIL PERCOBAAN

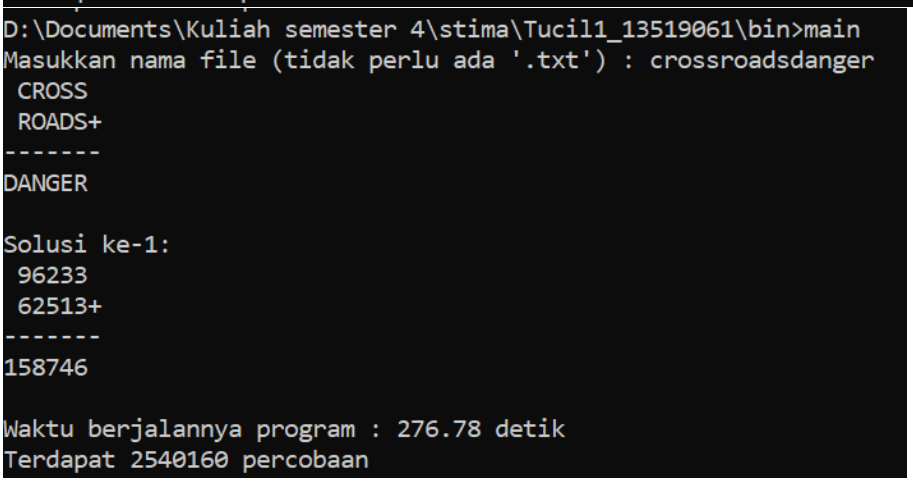
Berikut hasil percobaan dari beberapa testcase :

1. 
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : clocktictockplanet
 CLOCK
  TICK
  TOCK+
-------
PLANET

Solusi ke-1:
 90892
  6592
  6892+
-------
104376

Waktu berjalannya program : 638.743 detik
Terdapat 2540160 percobaan
```
2. 
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : cocacolaoasis
 COCA
 COLA+
-----
OASIS

Solusi ke-1:
 8186
 8106+
-----
16292

Waktu berjalannya program : 3.59978 detik
Terdapat 120960 percobaan
```
3. 
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : crossroadsdanger
 CROSS
 ROADS+
-------
DANGER

Solusi ke-1:
 96233
 62513+
-------
158746

Waktu berjalannya program : 276.78 detik
Terdapat 2540160 percobaan
```

4.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : doubledoubletoiltrouble
 DOUBLE
 DOUBLE
   TOIL+
--------
TROUBLE

Solusi ke-1:
 798064
 798064
   1936+
--------
1598064

Waktu berjalannya program : 322.371 detik
Terdapat 2903040 percobaan
```

5.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : hereshecomes
 HERE
  SHE+
------
COMES

Solusi ke-1:
 9454
  894+
------
10348

Waktu berjalannya program : 17.7442 detik
Terdapat 423360 percobaan
```

6.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : memofromhomer
 MEMO
 FROM+
------
HOMER

Solusi ke-1:
 8485
 7358+
------
15843

Waktu berjalannya program : 4.15624 detik
Terdapat 105840 percobaan
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>
```

7.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : numbernumberpuzzle
NUMBER
NUMBER+
-------
PUZZLE

Solusi ke-1:
201689
201689+
-------
403378

Waktu berjalannya program : 407.492 detik
Terdapat 2903040 percobaan
```

8.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : threethreetwotwooneeleven
 THREE
 THREE
   TWO
   TWO
   ONE+
-------
ELEVEN

Solusi ke-1:
 84611
 84611
   803
   803
   391+
-------
171219

Waktu berjalannya program : 426.531 detik
Terdapat 2540160 percobaan
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>
```

9.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : tilespuzzlespicture
  TILES
PUZZLES+
--------
PICTURE

Solusi ke-1:
  91542
3077542+
--------
3169084

Waktu berjalannya program : 649.525 detik
Terdapat 2903040 percobaan
```

10.
```
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>main
Masukkan nama file (tidak perlu ada '.txt') : nogunnohunt
  NO
 GUN
  NO+
-----
HUNT

Solusi ke-1:
  87
 908
  87+
-----
1082

Waktu berjalannya program : 4.30001 detik
Terdapat 105840 percobaan
D:\Documents\Kuliah semester 4\stima\Tucil1_13519061\bin>
```

TABEL PENILAIAN

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 *cryptarihtmetic* dengan dua buah *operand*.		✓
5. Solusi *cryptarithmetic* benar untuk persoalan *cryptarihtmetic* untuk lebih dari dua buah operand.	✓	