TUGAS KECIL 1
IF2121 STRATEGI ALGORITMA

Penyelesaian Cryptarithmetic dengan Algoritma Brute Force

Reinard Rahardian A. S.

13318056

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

1. Algoritma Brute Force

   Tujuan dari tugas ini adalah mencari pemetaan huruf - angka yang tepat untuk soal penjumlahan dimana angka diganti huruf. Di soal ini ada 2 spesifikasi yang penting untuk merancang solusi:

   (a) semua huruf mewakili angka yang berbeda, dan

   (b) setiap soal maksimal hanya memiliki 10 jenis huruf yang berbeda.

   Jika total jenis huruf unik adalah r , maka kemungkinan solusi ada ${}_{10}P_r$.

   Program diselesaikan dengan langkah berikut:

   (a) Baca file berisi soal input

   (b) Untuk setiap baris, simpan semua karakter alfabet ke suatu vector string. Baris yang tidak memiliki karakter alfabet diabaikan.

   (c) Identifikasi semua karakter unik yang ada di semua string di vector tersebut.

   (d) Simpan karakter ke dalam objek map untuk memetakan nilai yang diwakili karakter tersebut.

   (e) Jalankan fungsi kombinasi untuk mendapatkan kombinasi angka yang mungkin dipakai ke pemetaan, dengan fungsi permutasi ${}_rP_r$ sebagai panggilan baliknya,

   (f) Untuk setiap kombinasi yang ditemukan, jalankan fungsi permutasi ${}_rP_r$ untuk mendapatkan susunan yang mungkin ditemukan dari kombinasi tersebut., dengan fungsi uji pemetaan sebagai panggilan baliknya,

   (g) Dalam fungsi uji, semua string dalam vector dicari nilai angkanya. Semua angka kecuali yang terakhir lalu ditambahkan, dan dilakukan pengecekan bila jumlahnya sama dengan angka terakhir (yang berasal dari kata hasil penjumlahan). Bila pengujian ini lolos, semua proses permutasi dan kombinasi dihentikan, dan hasilnya ditampilkan.

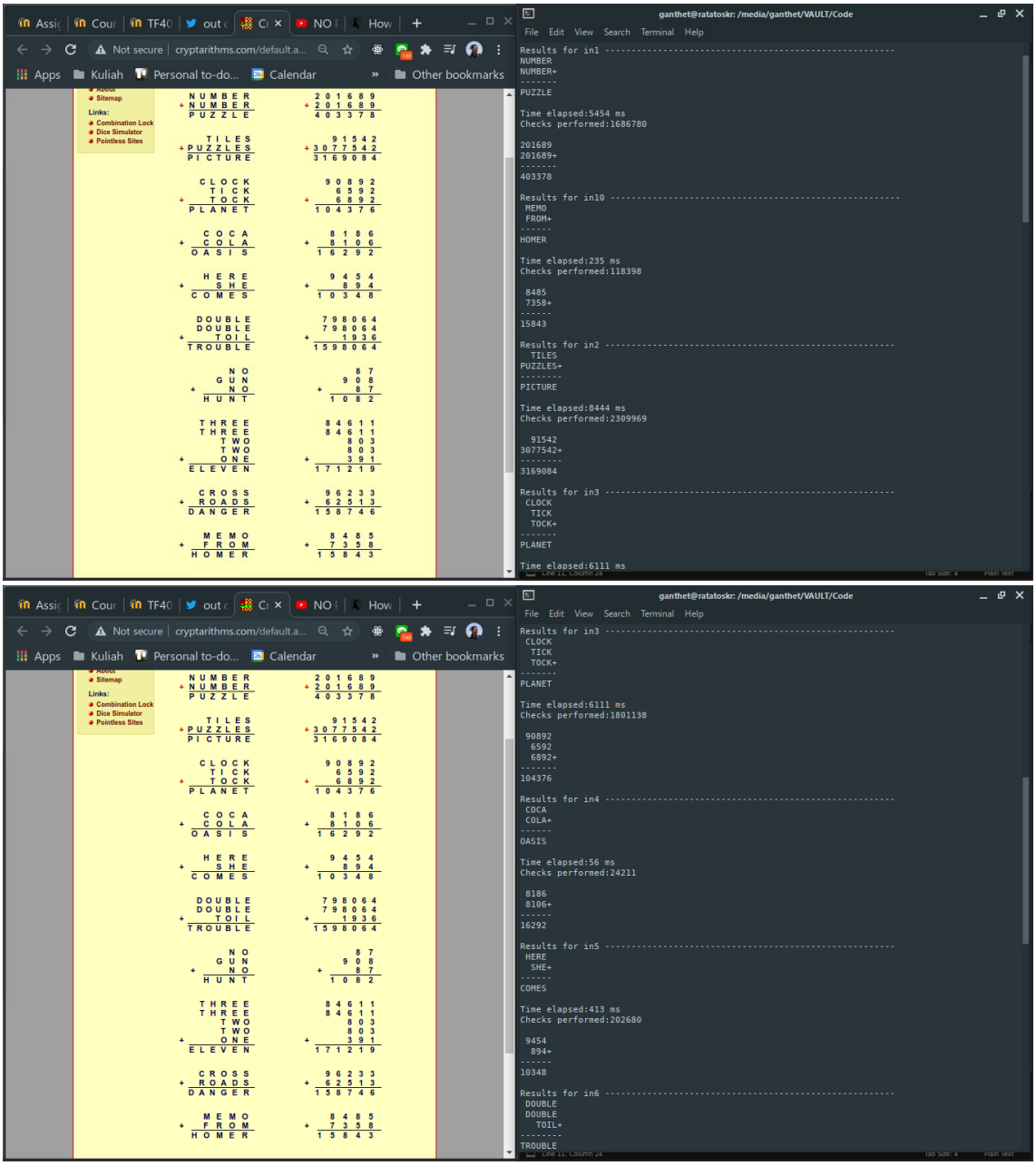
2. Kode

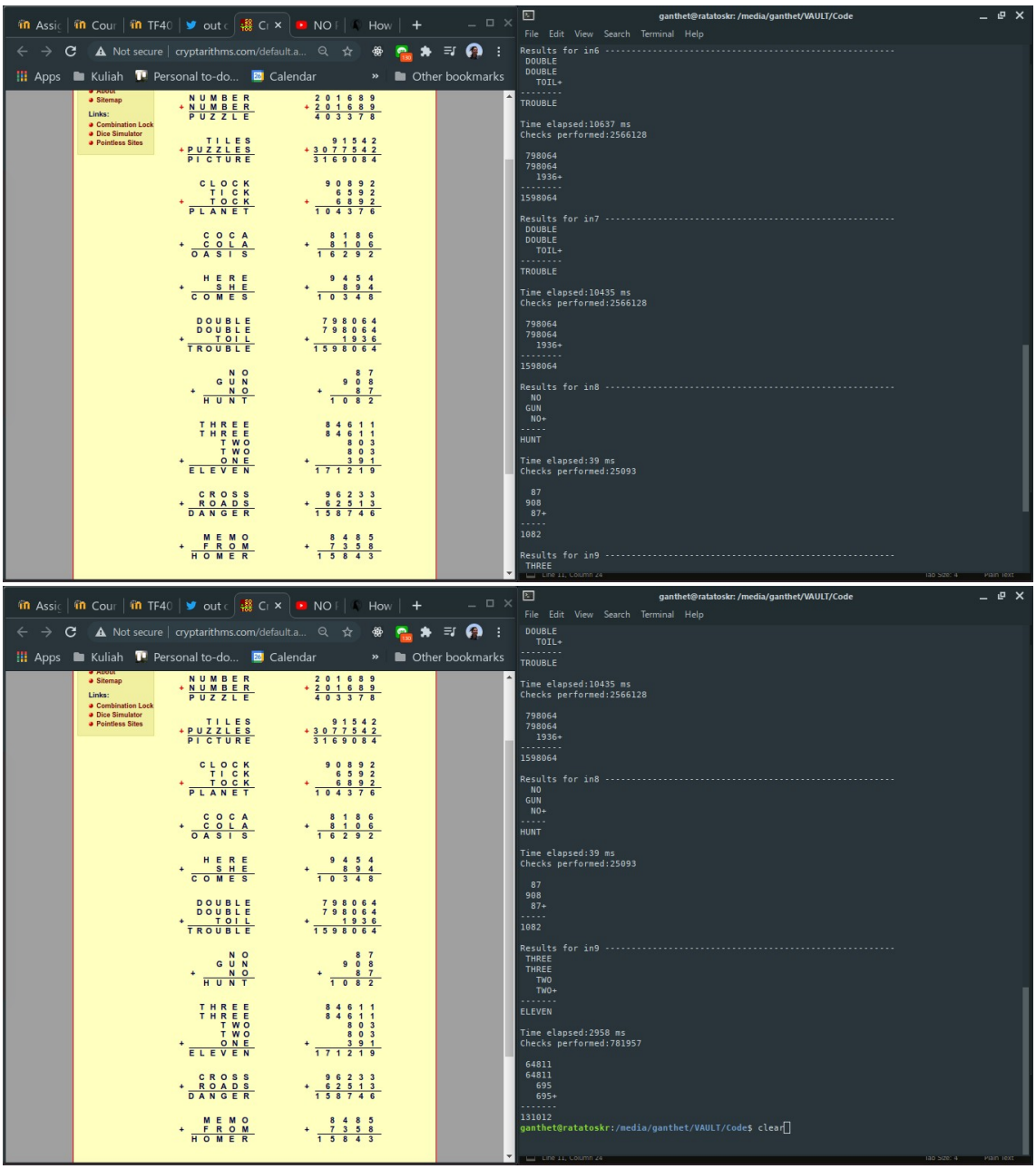
```
#include <iostream>
#include <fstream>
#include <map>
#include <string>
#include <vector>
#include <utility>
#include <algorithm>
#include <chrono>

using namespace std;

// Fungsi untuk membuat pemetaan huruf -> nilai. Semua huruf akan diberi nilai awal 0.
map<const char, short> create_dict(vector<string> &words){
	map<const char, short> dict;

	// Iterasi semua huruf di kata yang diberikan.
	// std::map otomatis membuat key-value pair baru
	// jika diberikan key yang belum tercatat sebelumnya.
	for(string &s : words)
	for(const char &c : s)
		dict[c] = 0;

	return dict;
}

// Terjemahkan kata menjadi angka.
int get_word_value(const string &word, map<const char, short> &dict){
	int sum = 0;
	int mul = 1;

	// Iterasi dari belakang
	for(int i = word.size() - 1; i >= 0; i--){
	sum += mul * dict[ word[i] ];
	mul *= 10;
	}

	return sum;
}

// Cek apakah pemetaan angka -> nilai dari pemetaan sesuai.
bool check_guess(vector<string> &words, map<const char, short> &dict){
	// Last element in vector is the sum result
	int result_sum = get_word_value(words.back(), dict);

	// Get all word sum result, minus 1 because last is result word
	int words_sum = 0;
	for(int i = 0; i < words.size() - 1; i++){
	words_sum += get_word_value(words[i], dict);
	}

	return result_sum == words_sum;
}

// Validity check for next try: First letters are not 0
bool first_not_zero(vector<string> &words, map<const char, short> &dict){
	for (string &s : words){
	if(dict[s[0]] == 0){
		return false;
	}
	}

	return true;
}
```

```
// Recursion tree-style permutation algorithm
// Reference: https://www.geeksforgeeks.org/write-a-c-program-to-print-all-permutations-
of-a-given-string/
void permute(int offset, vector<short*> &dict_idx, auto cb, bool &found){
	if (offset == dict_idx.size() - 1 ) {
	found = cb();
	return;
	}

	for(int i = offset; i < dict_idx.size(); ++i){
	int tmp;

	// swap the value
	tmp = *dict_idx[i];
	*dict_idx[i] = *dict_idx[offset];
	*dict_idx[offset] = tmp;

	permute(offset + 1, dict_idx, cb, found);
	if(found) return; // Stop prematurely if match found

	// swap back
	tmp = *dict_idx[i];
	*dict_idx[i] = *dict_idx[offset];
	*dict_idx[offset] = tmp;
	}
}

// Concatenation-style combination algorithm
// Reference: https://stackoverflow.com/questions/12991758/creating-all-possible-k-
combinations-of-n-items-in-c/28698654
void combine(int offset, int depth, vector<short*> &dict_idx, auto cb, bool &found) {
	if (depth == dict_idx.size()) {
	permute(0, dict_idx, cb, found);
	return;
	}

	for (int i = offset; i <= 10 - (dict_idx.size() - depth); ++i) {
	*dict_idx[depth] = i;

	combine(i+1, depth+1, dict_idx, cb, found);
	if(found) return; // Stop prematurely if match found
	}
}
```

```
int main(int argc, char* argv[]){
	chrono::steady_clock::time_point begin, end;
	unsigned long long int checks = 0;
	vector<string> words;
	vector<string> original;
	map<const char, short> dict;
	vector<short*> dict_idx; // hack to enable dict number access via order index

	// Read input
	ifstream file(argv[1]);
	string s;
	while (getline(file, s)) {
	original.push_back(s);
	}

	// Extract alphabetical character from input string. Sorry, man's lazy.
	for(string s : original){
	cout << s << "\n";
	s.erase(remove_if(s.begin(), s.end(), [](char c) { return !isalpha(c); } ),
s.end());
	if(s != ""){
		words.push_back(s);
	}
	}

	// Initialize letter: value dictionary and indexed version (needs so since not
available normally)
	dict = create_dict(words);
	for (pair<const char, short> &p : dict){
	dict_idx.push_back(&(p.second));
	}

	// Callback to feed for permutation
	auto cb = [&] {
	checks ++;

	if(first_not_zero(words, dict))
		return check_guess(words, dict);
	return false;
	};

	begin = std::chrono::steady_clock::now();

	bool found = false;
	combine(0, 0, dict_idx, cb, found);

	end = std::chrono::steady_clock::now();

	// Print result
	cerr << "\nTime elapsed:" << chrono::duration_cast<std::chrono::milliseconds>
(end - begin).count() << " ms\n";
	cerr << "Checks performed:" << checks << "\n\n";
	for(string &s : original){
	string output;

	for(const char c : s){
		if (isalpha(c)){
			output += to_string(dict[c]);
		}
		else
			output += c;
	}

	cout << output << "\n";
	}
}
```

3. Hasil tangkapan layar





Kode ada di https://github.com/Retorikal/stima_bruteforce

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