LAPORAN
TUGAS KECIL IF2211
STRATEGI ALGORITMA

Permainan 24

Akbar Maulana Ridho 13521093

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022/2023

A. Deskripsi Singkat

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). Yang perlu diperhatikan hanyalah nilai kartu yang didapat (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (×), divisi (/) dan tanda kurung (()). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas.

(dikutip dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf).

B. Alur Program

Berikut adalah alur program:

1. Program menanyakan apakah pengguna ingin mengisi sendiri masukan atau mengacak
    a. Bila diacak, program akan mengisi masukan secara acak.
    b. Bila tidak, program akan meminta masukan dari pengguna. Proses ini terus diulang hingga pengguna mengisi masukan yang benar.
2. Program akan mengiterasi setiap kemungkinan urutan masukan dan urutan operator untuk mencari kombinasi yang menghasilkan 24. Setelah itu, program menampilkan banyaknya solusi yang diperoleh, ekspresi dari setiap kemungkinan, dan lama waktu eksekusi.
3. Program akan menanyakan apakah pengguna ingin menyimpan hasil ke dalam sebuah file.
    a. Bila iya, program akan meminta masukan nama file hasil penyimpanan, lalu program menyimpan hasilnya pada file tersebut.
    b. Bila tidak, lanjut ke langkah berikutnya.
4. Program selesai

C. Algoritma Brute Force

Bagian ini akan menjelaskan detail langkah nomor 2 pada bagian sebelumnya. Berikut adalah alur implementasi algoritma brute force:

1. Lakukan permutasi pada keempat masukan sehingga menghasilkan 4! atau 24 kemungkinan pengurutan masukan. Setiap permutasi bisa duplikat.

2. Buat permutasi tiga buah operator (boleh berulang) sehingga menghasilkan $4 \times 4 \times 4 = 64$ kemungkinan kombinasi operator.
3. Buat kombinasi urutan prioritas ekspresi, yang terdiri atas kombinasi berikut.
    a. A ops B ops C ops D
    b. (A ops B) ops (C ops D)
    c. (A ops B) ops C ops D
    d. A ops B ops (C ops D)
    e. A ops (B ops C) ops D
    f. ((A ops B) ops C) ops D
    g. (A ops (B ops C)) ops D
    h. A ops ((B ops C) ops D)
    i. A ops (B ops (C ops D))
4. Buat kombinasi permutasi masukan, permutasi operator, dan kombinasi prioritas ekspresi sehingga diberoleh $24 \times 64 \times 9 = 13824$ kombinasi.
5. Untuk setiap kombinasi, evaluasi ekspresi tersebut apakah menghasilkan 24 atau tidak. Bila iya, masukkan ekspresinya ke dalam set hasil.
6. Mengingat hasilnya disimpan ke dalam set, maka ekspresi yang sama persis hanya akan dihitung satu kali.

D. Sumber Kode

File main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <set>
#include <limits>
#include <random>
#include <chrono>

using std::set;
using std::string;
using std::vector;
using std::ofstream;
using std::snprintf;
using std::cin;
using std::cout;
using std::cerr;
using std::chrono::high_resolution_clock;
using std::chrono::microseconds;
using std::chrono::duration_cast;

int op_in_ops(const char ops[], int ops_size, char op) {
    int found_idx = -1;
    int i = 0;

    while (found_idx == -1 && i < ops_size) {
        if (ops[i] == op) {
            found_idx = i;
        } else {
            i++;
        }
    }

    return found_idx;
}

double evaluate_arithmetic(double number1, double number2, char op) {
    if (op == '/') {
        if (number2 == 0) {
            return std::numeric_limits<double>::quiet_NaN();
        }
        return number1 / number2;
```

```
    } else if (op == '*') {
        return number1 * number2;
    } else if (op == '+') {
        return number1 + number2;
    } else if (op == '-') {
        return number1 - number2;
    } else {
        cerr << "Invalid ops while evaluating arithmetic\n";
        exit(1);
    }
}

double evaluate_expression(const double original_values[], int values_size, const char
original_ops[], int ops_size) {
    char ops_order[] = {'*', '/', '-', '+'};
    double values[values_size];
    char ops[ops_size];

    for (int j = 0; j < values_size; j++) {
        values[j] = original_values[j];
    }

    for (int j = 0; j < ops_size; j++) {
        ops[j] = original_ops[j];
    }

    while (values_size > 1) {
        double temp_values[4];
        int temp_values_count = 0;
        char temp_ops[3];
        int temp_ops_count = 0;

        double result;

        int idx = -1;
        int i = 0;

        while (idx == -1 && i < 4) {
            idx = op_in_ops(ops, ops_size, ops_order[i]);
            i++;
        }

        if (idx == -1) {
            cerr << "Invalid ops\n";
            exit(1);
        }

        result = evaluate_arithmetic(values[idx], values[idx + 1], ops[idx]);

        for (int j = 0; j < ops_size; j++) {
            if (j != idx) {
                temp_ops[temp_ops_count++] = ops[j];
            }
        }

        for (int j = 0; j < values_size; j++) {
            if (j == idx) {
                temp_values[temp_values_count++] = result;
            } else if (j != idx + 1) {
                temp_values[temp_values_count++] = values[j];
            }
        }

        for (int j = 0; j < temp_values_count; j++) {
            values[j] = temp_values[j];
        }

        for (int j = 0; j < temp_ops_count; j++) {
            ops[j] = temp_ops[j];
        }

        values_size = temp_values_count;
        ops_size = temp_ops_count;
    }

    return values[0];
```

```
}

void get_validated_input(double result[]) {
    bool valid = false;

    while (!valid) {
        cout << "Masukkan kombinasi kartu: \n";

        for (int i = 0; i < 4; i++) {
            double value;
            string input;
            cin >> input;

            if (input == "A") {
                value = 1;
            } else if (input == "2") {
                value = 2;
            } else if (input == "3") {
                value = 3;
            } else if (input == "4") {
                value = 4;
            } else if (input == "5") {
                value = 5;
            } else if (input == "6") {
                value = 6;
            } else if (input == "7") {
                value = 7;
            } else if (input == "8") {
                value = 8;
            } else if (input == "9") {
                value = 9;
            } else if (input == "10") {
                value = 10;
            } else if (input == "J") {
                value = 11;
            } else if (input == "Q") {
                value = 12;
            } else if (input == "K") {
                value = 13;
            } else {
                cout << "Input " << input << " salah. Ulangi!\n";
                cin.clear();
                cin.ignore(INT32_MAX, '\n');
                break;
            }

            result[i] = value;

            if (i == 3) {
                valid = true;
            }
        }
    }
}

string format_string(const string &format, const double values[], const char ops[]) {
    char buffer[100];
    snprintf(buffer, sizeof(buffer), format.c_str(), int(values[0]), ops[0],
int(values[1]), ops[1],
             int(values[2]),
             ops[2], int(values[3]));
    string strbuffer = buffer;

    return strbuffer;
}

void save_to_file(const set<string> &output) {
    string filename;

    cout << "Enter filename: (without .txt)\n";
    cin >> filename;

    filename = filename + ".txt";

    ofstream file(filename);
```

```
    file << "Found " << output.size() << " solutions\n";

    for (auto &i: output) {
        file << i << "\n";
    }

    file.close();

    cout << "File saved as " << filename << "\n";
}

template<typename T>
void swap(T &x, T &y) {
    T temp = x;
    x = y;
    y = temp;
}

void permutations(vector<vector<double> > &res, vector<double> nums, int l, int h) {
    if (l == h) {
        res.push_back(nums);
        return;
    }

    for (int i = l; i <= h; i++) {
        swap(nums[l], nums[i]);
        permutations(res, nums, l + 1, h);
        swap(nums[l], nums[i]);
    }
}

vector<vector<double> > permutate(vector<double> &values) {
    vector<vector<double> > result;
    permutations(result, values, 0, values.size() - 1);
    return result;
}

int main() {
    std::default_random_engine generator{std::random_device{}()};
    std::uniform_int_distribution<int> distribution(1, 13);

    const double target = 24;
    double input[4];
    char ops[] = {'/', '*', '+', '-'};

    cout << "Apakah ingin mengacak input? [ya/tidak]\n";
    string random;
    cin >> random;

    if (random == "ya") {
        for (double &i: input) {
            i = double(distribution(generator));
        }
    } else {
        get_validated_input(input);
    }

    set<string> valid_combination;
    vector<double> initial_vec = {input[0], input[1], input[2], input[3]};

    auto start = high_resolution_clock::now();
    auto permutations = permutate(initial_vec);

    for (const auto &combination: permutations) {
        for (int i = 0; i < 4; i++) {
            input[i] = combination[i]; // ini jelek tapi lg butuh cepet :(
        }

        for (char &op1: ops) {
            for (char &op2: ops) {
                for (char &op3: ops) {
                    string expression;
                    double result, num1, num2;

                    // case A ops B ops C ops D
                    char ops_combination[] = {op1, op2, op3};
```

```
                    result = evaluate_expression(input, 4, ops_combination, 3);

                    if (result == target) {
                        valid_combination.insert(format_string("%d %c %d %c %d %c %d",
input, ops_combination));
                    }

                    // case (A ops B) ops (C ops D)
                    num1 = evaluate_arithmetic(input[0], input[1], op1);
                    num2 = evaluate_arithmetic(input[2], input[3], op3);
                    result = evaluate_arithmetic(num1, num2, op2);

                    if (result == target) {
                        valid_combination.insert(format_string("(%d %c %d) %c (%d %c %d)",
input, ops_combination));
                    }

                    // case (A ops B) ops C ops D
                    num1 = evaluate_arithmetic(input[0], input[1], op1);
                    double new_input1[] = {num1, input[2], input[3]};
                    char new_ops1[] = {op2, op3};
                    result = evaluate_expression(new_input1, 3, new_ops1, 2);

                    if (result == target) {
                        valid_combination.insert(format_string("(%d %c %d) %c %d %c %d",
input, ops_combination));
                    }

                    // case A ops B ops (C ops D)
                    num2 = evaluate_arithmetic(input[2], input[3], op3);
                    double new_input2[] = {input[0], input[1], num2};
                    char new_ops2[] = {op1, op2};
                    result = evaluate_expression(new_input2, 3, new_ops2, 2);

                    if (result == target) {
                        valid combination.insert(format string("%d %c %d %c (%d %c %d)",
input, ops_combination));
                    }

                    // case A ops (B ops C) ops D
                    num1 = evaluate arithmetic(input[1], input[2], op2);
                    double new_input3[] = {input[0], num1, input[3]};
                    char new_ops3[] = {op1, op3};
                    result = evaluate_expression(new_input3, 3, new_ops3, 2);

                    if (result == target) {
                        valid combination.insert(format string("%d %c (%d %c %d) %c %d",
input, ops_combination));
                    }

                    // case ((A ops B) ops C) ops D
                    num1 = evaluate_arithmetic(input[0], input[1], op1);
                    num2 = evaluate_arithmetic(num1, input[2], op2);
                    result = evaluate_arithmetic(num2, input[3], op3);

                    if (result == target) {
                        valid_combination.insert(format_string("((%d %c %d) %c %d) %c %d",
input, ops_combination));
                    }

                    // case (A ops (B ops C)) ops D
                    num2 = evaluate_arithmetic(input[1], input[2], op2);
                    num1 = evaluate_arithmetic(input[0], num2, op1);
                    result = evaluate_arithmetic(num1, input[3], op3);

                    if (result == target) {
                        valid_combination.insert(format_string("(%d %c (%d %c %d)) %c %d",
input, ops_combination));
                    }

                    // case A ops ((B ops C) ops D)
                    num1 = evaluate_arithmetic(input[1], input[2], op2);
                    num2 = evaluate_arithmetic(num1, input[3], op3);
                    result = evaluate_arithmetic(input[0], num2, op1);

                    if (result == target) {
```

```
                        valid_combination.insert(format_string("%d %c ((%d %c %d) %c %d)",
input, ops_combination));
                    }
                    // case A ops (B ops (C ops D))
                    num2 = evaluate_arithmetic(input[2], input[3], op3);
                    num1 = evaluate_arithmetic(input[1], num2, op2);
                    result = evaluate_arithmetic(input[0], num1, op1);

                    if (result == target) {
                        valid_combination.insert(format_string("%d %c (%d %c (%d %c %d))",
input, ops_combination));
                    }
                }
            }
        }
    }

    auto end = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end - start);

    if (valid_combination.empty()) {
        cout << "No solution found\n";
        cout << "Execution done in " << duration.count() << " microseconds\n";
    } else {
        cout << "Found " << valid_combination.size() << " solutions\n";

        for (auto &i: valid_combination) {
            cout << i << "\n";
        }

        cout << "Execution done in " << duration.count() << " microseconds\n";

        cout << "Apakah ingin menyimpan solusi? [ya/tidak]\n";
        string answer;
        cin >> answer;

        if (answer == "ya") {
            save_to_file(valid_combination);
        }
    }

    return 0;
}
```

E. Kasus Uji

1. Masukan 8 7 2 9

```
barcode@LAPTOP-1ARP10R0:~/Tucil1_13521093$ bin/game24
Apakah ingin mengacak input? [ya/tidak]
tidak
Masukkan kombinasi kartu:
8 7 2 9
Found 8 solutions
((7 + 9) * 2) - 8
((9 + 7) * 2) - 8
(2 * (7 + 9)) - 8
(2 * (9 + 7)) - 8
(7 + 9) * 2 - 8
(9 + 7) * 2 - 8
2 * (7 + 9) - 8
2 * (9 + 7) - 8
Execution done in 778 microseconds
Apakah ingin menyimpan solusi? [ya/tidak]
tidak
```

2. Masukan A 9 8 Q

```
Masukkan kombinasi kartu:
A 9 8 Q
Found 56 solutions
((1 * 12) - 9) * 8
((1 + 9) - 8) * 12
((1 - 8) + 9) * 12
((12 * 1) - 9) * 8
((12 - 9) * 1) * 8
((12 - 9) * 8) * 1
((12 - 9) * 8) / 1
((12 - 9) / 1) * 8
((12 / 1) - 9) * 8
((9 + 1) - 8) * 12
((9 - 8) + 1) * 12
(1 * (12 - 9)) * 8
(1 * 8) * (12 - 9)
(1 + (9 - 8)) * 12
(1 - (8 - 9)) * 12
(12 - (1 * 9)) * 8
(12 - (9 * 1)) * 8
(12 - (9 / 1)) * 8
(12 - 9) * (1 * 8)
(12 - 9) * (8 * 1)
(12 - 9) * (8 / 1)
(12 - 9) * 1 * 8
(12 - 9) * 8 * 1
```

```
(12 - 9) * 8 / 1
(12 - 9) / (1 / 8)
(8 * (12 - 9)) * 1
(8 * (12 - 9)) / 1
(8 * 1) * (12 - 9)
(8 / 1) * (12 - 9)
(9 + (1 - 8)) * 12
(9 - (8 - 1)) * 12
1 * ((12 - 9) * 8)
1 * (12 - 9) * 8
1 * (8 * (12 - 9))
1 * 8 * (12 - 9)
12 * ((1 + 9) - 8)
12 * ((1 - 8) + 9)
12 * ((9 + 1) - 8)
12 * ((9 - 8) + 1)
12 * (1 + (9 - 8))
12 * (1 - (8 - 9))
12 * (9 + (1 - 8))
12 * (9 - (8 - 1))
8 * ((1 * 12) - 9)
8 * ((12 * 1) - 9)
8 * ((12 - 9) * 1)
8 * ((12 - 9) / 1)
```

```
8 * ((12 / 1) - 9)
8 * (1 * (12 - 9))
8 * (12 - (1 * 9))
8 * (12 - (9 * 1))
8 * (12 - (9 / 1))
8 * (12 - 9) * 1
8 * (12 - 9) / 1
8 * 1 * (12 - 9)
8 / (1 / (12 - 9))
Execution done in 729 microseconds
Apakah ingin menyimpan solusi? [ya/tidak]
```

3. Masukan A A A A

```
Masukkan kombinasi kartu:
A A A A
No solution found
Execution done in 1424 microseconds
```

4. Masukan A 8 9 Q

```
barcode@LAPTOP-1ARP10R0:~
Apakah ingin mengacak inpu
tidak
Masukkan kombinasi kartu:
A 8 9 Q
Found 56 solutions
((1 * 12) - 9) * 8
((1 + 9) - 8) * 12
((1 - 8) + 9) * 12
((12 * 1) - 9) * 8
((12 - 9) * 1) * 8
((12 - 9) * 8) * 1
((12 - 9) * 8) / 1
((12 - 9) / 1) * 8
((12 / 1) - 9) * 8
((9 + 1) - 8) * 12
((9 - 8) + 1) * 12
(1 * (12 - 9)) * 8
(1 * 8) * (12 - 9)
(1 + (9 - 8)) * 12
(1 - (8 - 9)) * 12
(12 - (1 * 9)) * 8
(12 - (9 * 1)) * 8
(12 - (9 / 1)) * 8
(12 - 9) * (1 * 8)
(12 - 9) * (8 * 1)
```

```
(12 - 9) * (8 / 1)
(12 - 9) * 1 * 8
(12 - 9) * 8 * 1
(12 - 9) * 8 / 1
(12 - 9) / (1 / 8)
(8 * (12 - 9)) * 1
(8 * (12 - 9)) / 1
(8 * 1) * (12 - 9)
(8 / 1) * (12 - 9)
(9 + (1 - 8)) * 12
(9 - (8 - 1)) * 12
1 * ((12 - 9) * 8)
1 * (12 - 9) * 8
1 * (8 * (12 - 9))
1 * 8 * (12 - 9)
12 * ((1 + 9) - 8)
12 * ((1 - 8) + 9)
12 * ((9 + 1) - 8)
12 * ((9 - 8) + 1)
12 * (1 + (9 - 8))
12 * (1 - (8 - 9))
12 * (9 + (1 - 8))
12 * (9 - (8 - 1))
8 * ((1 * 12) - 9)
```

```
8 * ((12 * 1) - 9)
8 * ((12 - 9) * 1)
8 * ((12 - 9) / 1)
8 * ((12 / 1) - 9)
8 * (1 * (12 - 9))
8 * (12 - (1 * 9))
8 * (12 - (9 * 1))
8 * (12 - (9 / 1))
8 * (12 - 9) * 1
8 * (12 - 9) / 1
8 * 1 * (12 - 9)
8 / (1 / (12 - 9))
Execution done in 739 microseconds
Apakah ingin menyimpan solusi? [ya/tidak]
tidak
```

5. Masukan 7 5 8 3

```
Apakah ingin mengacak input?
tidak
Masukkan kombinasi kartu:
7 8 5 3
Found 70 solutions
((3 * 7) + 8) - 5
((3 * 7) - 5) + 8
((5 * 7) - 3) - 8
((5 * 7) - 8) - 3
((7 * 3) + 8) - 5
((7 * 3) - 5) + 8
((7 * 5) - 3) - 8
((7 * 5) - 8) - 3
((8 - 5) * 7) + 3
(3 * 7) + (8 - 5)
(3 * 7) + 8 - 5
(3 * 7) - (5 - 8)
(3 * 7) - 5 + 8
(5 * 7) - (3 + 8)
(5 * 7) - (8 + 3)
(5 * 7) - 3 - 8
(5 * 7) - 8 - 3
(7 * (8 - 5)) + 3
(7 * 3) + (8 - 5)
(7 * 3) + 8 - 5
(7 * 3) - (5 - 8)
```

```
(7 * 3) - 5 + 8
(7 * 5) - (3 + 8)
(7 * 5) - (8 + 3)
(7 * 5) - 3 - 8
(7 * 5) - 8 - 3
(8 + (3 * 7)) - 5
(8 + (7 * 3)) - 5
(8 - 5) * 7 + 3
(8 - 5) + (3 * 7)
(8 - 5) + (7 * 3)
(8 - 5) + 3 * 7
(8 - 5) + 7 * 3
3 * 7 + (8 - 5)
3 * 7 + 8 - 5
3 * 7 - (5 - 8)
3 * 7 - 5 + 8
3 + ((8 - 5) * 7)
3 + (7 * (8 - 5))
3 + (8 - 5) * 7
3 + 7 * (8 - 5)
3 - ((5 - 8) * 7)
3 - (5 - 8) * 7
3 - (7 * (5 - 8))
3 - 7 * (5 - 8)
```

```
5 * 7 - (3 + 8)
5 * 7 - (8 + 3)
5 * 7 - 3 - 8
5 * 7 - 8 - 3
7 * (8 - 5) + 3
7 * 3 + (8 - 5)
7 * 3 + 8 - 5
7 * 3 - (5 - 8)
7 * 3 - 5 + 8
7 * 5 - (3 + 8)
7 * 5 - (8 + 3)
7 * 5 - 3 - 8
7 * 5 - 8 - 3
8 + ((3 * 7) - 5)
8 + ((7 * 3) - 5)
8 + (3 * 7) - 5
8 + (7 * 3) - 5
8 + 3 * 7 - 5
8 + 7 * 3 - 5
8 - (5 - (3 * 7))
8 - (5 - (7 * 3))
8 - 5 + (3 * 7)
8 - 5 + (7 * 3)
8 - 5 + 3 * 7
8 - 5 + 7 * 3
Execution done in 731 microse
Apakah ingin menyimpan solusi
```

6. Masukan A J Q K

```
barcode@LAPTOP-1ARP10R0:~/Tu
Apakah ingin mengacak input?
tidak
Masukkan kombinasi kartu:
A J Q K
Found 40 solutions
((1 * 13) - 11) * 12
((13 * 1) - 11) * 12
((13 - 11) * 1) * 12
((13 - 11) * 12) * 1
((13 - 11) * 12) / 1
((13 - 11) / 1) * 12
((13 / 1) - 11) * 12
(1 * (13 - 11)) * 12
(1 * 12) * (13 - 11)
(12 * (13 - 11)) * 1
(12 * (13 - 11)) / 1
(12 * 1) * (13 - 11)
(12 / 1) * (13 - 11)
(13 - (1 * 11)) * 12
(13 - (11 * 1)) * 12
(13 - (11 / 1)) * 12
(13 - 11) * (1 * 12)
(13 - 11) * (12 * 1)
(13 - 11) * (12 / 1)
(13 - 11) * 1 * 12
(13 - 11) * 12 * 1
(13 - 11) * 12 / 1
(13 - 11) / (1 / 12)
```

```
1 * ((13 - 11) * 12)
1 * (12 * (13 - 11))
1 * (13 - 11) * 12
1 * 12 * (13 - 11)
12 * ((1 * 13) - 11)
12 * ((13 * 1) - 11)
12 * ((13 - 11) * 1)
12 * ((13 - 11) / 1)
12 * ((13 / 1) - 11)
12 * (1 * (13 - 11))
12 * (13 - (1 * 11))
12 * (13 - (11 * 1))
12 * (13 - (11 / 1))
12 * (13 - 11) * 1
12 * (13 - 11) / 1
12 * 1 * (13 - 11)
12 / (1 / (13 - 11))
Execution done in 1046 microseconds
Apakah ingin menyimpan solusi? [ya/tidak]
```

F. Tabel Isian

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil running	✓	
Program dapat membaca input/ generate sendiri dan memberikan luaran	✓	
Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
Program dapat menyimpan solusi dalam file teks	✓	

G. Lain-lain

Berikut adalah taunan menuju repository tugas kecil ini:
https://github.com/akbarmridho/Tucil1_13521093