LAPORAN

TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

24 GAME SOLVER DENGAN ALGORITMA BRUTE FORCE

Oleh:

Hidayatullah Wildan Ghaly Buchary (13521015)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG BANDUNG

2022/2023

DAFTAR ISI

BAB I

PENDAHULUAN

Permainan kartu merupakan jenis permainan yang sangat populer di dunia. Berbagai permainan kartu seperti poker, canasta, blackjack, casino, solitaire, bridge, dan sebagainya dimainkan di berbagai belahan dunia. Salah satu permainan kartu yang tergolong unik adalah permainan 24. Permainan ini bisa dikatakan unik karena cara bermainnya mirip seperti menyelesaikan teka-teki. Cara mainnya sendiri adalah dengan menyiapkan empat buah kartu lalu mencoba untuk mendapatkan hasil 24 dengan mengoperasikan setiap kartu dengan aritmatika.

Permainan 24 menarik banyak peminat, terutama anak-anak, karena dapat meningkatkan kemampuan berhitung dan mengasah otak sehingga dapat berpikir cepat dan akurat. Permainan 24 kartu biasa dimainkan dengan kartu remi. Dek terdiri dari 52 kartu yang dibagi menjadi empat jenis (sekop, hati, klub dan wajik), yang masing-masing terdiri dari 13 kartu yaitu Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King.

Selain mencoba untuk mencari jawabannya secara manual, permainan 24 ini bisa juga diselesaikan dengan menggunakan algoritma. Algoritma adalah metode terbatas, sekumpulan instruksi yang didefinisikan untuk menyelesaikan suatu masalah. Algoritma memiliki kriteria tertentu pada kondisi awal sebelum algoritma dijalankan. Algoritma berakhir ketika semua kondisi awal memenuhi kriteria. Dimulai dengan nilai awal, serangkaian instruksi dijalankan untuk memproses kondisi yang ditentukan untuk menghasilkan keluaran dan untuk menentukan kondisi akhir.

Salah satu algoritma yang dapat mendapatkan semua jawaban dari permainan 24 ini adalah algoritma brute force. Brute force adalah pendekatan langsung untuk memecahkan masalah, biasanya didasarkan pada menyatakan masalah dan mendefinisikan konsep. Algoritme brute force memecahkan masalah dengan cara yang sangat sederhana, langsung, dan jelas.

BAB II

LANGKAH JALANNYA PROGRAM

Berikut ini adalah langkah-langkah yang dilakukan program dengan menggunakan algoritma brute force dalam menjalankan program 24 game solver:

1. Program akan menampilkan perintah untuk menerima input kartu atau memilih random card.
2. Jika user memilih input kartu maka program akan melakukan looping sampai input kartu valid yaitu 4 kartu dipisahkan dengan spasi dan kartu terdaftar (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). Jika user memilih random card maka program akan melakukan random terhadap 4 kartu sebagai input.
3. Setelah program mendapatkan 4 kartu, program akan menampilkan kartu yang telah terpilih dan menyimpan waktu di saat itu.
4. Setelah itu, program akan mendapatkan semua kemungkinan untuk posisi kartu dengan menggunakan brute force (posisi kartu ditukar sampai semua kemungkinan didapatkan). Apabila ada kartu yang sama, maka program tidak akan menimpannya lebih dari satu kali (misal 1 2 4 4 lalu kartu ketiga ditukar dengan kartu keempat). Kemungkinan posisi kartu ini ada 4! yaitu 24 kemungkinan.
5. Setelah itu, program juga akan melakukan brute force terhadap operators yang valid (+, -, *, /). Operator yang digunakan dalam satu operasi hanyalah tiga dan boleh sama sehingga ada 4 x 4 x 4 yaitu 64 kemungkinan.
6. Selain operator, program juga harus memerhatikan tentang penempatan tanda kurung. Untuk itu, akan dilakukan percobaan terhadap semua kemungkinan tanda kurung yaitu 5 kemungkinan:
    - ((card _ card) _ card) _ card
    - ((card _ (card _ card) _ card
    - (card _ card) _ (card _ card)
    - card _ ((card _ card) _ card)
    - card _ (card _ (card _ card))
7. Setelah seluruh kemungkinan didapatkan, maka program bisa melakukan brute force dengan mencoba semuanya dan menyimpannya ke dalam array jika hasilnya adalah 24.
8. Program menampilkan jawaban dan waktu eksekusi lalu meminta input jika jawaban ingin disimpan ke dalam file.

BAB III
SOURCE CODE PROGRAM

Berikut ini adalah source code program yang telah dibuat dalam bahasa C++ untuk 24 game solver dengan memanfaatkan algoritma brute force

1. Source code dalam file input.cpp

```
/* Nama file    : input.cpp */
/* Fungsi       : Pengoperasian input dari user */

#include <iostream>
#include <time.h>
#include <string>
#include <vector>
#include <array>
#include <fstream>
#include <chrono>
using namespace std;

string cards[] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q",
"K"};
char ops[] = {'*', '/', '+', '-'};

void userInput(double arr[4]){
/* Prosedur digunakan untuk mendapatkan input dari user */
    string card[4];
    while (true){
        cout << "1. Input manual\n2. Dipilih secara random\n>> ";
        int N;
        cin >> N;
        cin.ignore();

        /* Validasi input user */
        if ((N < 1) || (N > 2)){
            cout << "Masukan tidak valid. Harap ulangi!\n";
            continue;
        }

        if (N == 1){
            /* Pilihan untuk input kartu dari keyboard */
            while (true){
                bool isValidInput = true;
                cout << "Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J,
Q, K) pisahkan dengan spasi:\nContoh: 10 A K 8\n>> ";
```

```
                for (int i = 0; i < 4; i++){
                    cin >> card[i];
                }
                cin.ignore();
                for (int i = 0; i < 4; i++){
                    bool isValid = false;
                    for (int j = 0; !isValid && j < 13; j++){
                        if (card[i] == cards[j]){
                            arr[i] = (double)(j + 1);
                            isValid = true;
                        }
                    }
                    if (!isValid){
                        cout << "Masukan tidak valid. Harap ulangi!\n";
                        isValidInput = false;
                        break;
                    }
                }
                if (isValidInput){
                    break;
                }
            }
        } else {
            /* Pilihan kartu random dari user */
            for (int i = 0; i < 4; i++){
                int random_card = rand() % 13 + 1;
                arr[i] = random_card;
                card[i] = cards[random_card - 1];
            }
        }
        cout <<"================================\n";
        printf("Kartu yang dipilih: %s %s %s %s\n", card[0].c_str(),
card[1].c_str(), card[2].c_str(), card[3].c_str());
        cout <<"================================\n";
        break;
    }
}
```

2. Source code dalam file brackets.cpp

```
/* Nama file    : brackets.cpp */
/* Fungsi       : Melakukan pengoperasian pada letak tanda kurung */

#include "input.cpp"

double operation(double a, double b, char op){
    switch (op){
```

```
        case '+': return a + b;
        case '-': return a - b;
        case '*': return a * b;
        default: return a / b;
    }
}

void allPossibleKurung(vector<string> *solution, double nums[4], char ops[3]){
/* Operasi pada kartu dipengaruhi juga dengan letak tanda kurung */
    int bracket[5][2][2] = {{{0, 1}, {2, 3}},
                            {{0, 1}, {0, 2}},
                            {{1, 2}, {0, 2}},
                            {{1, 2}, {1, 3}},
                            {{2, 3}, {1, 3}}};

    for (int i = 0; i < 5; i++){
        /* Operasi hanya dilakukan 3x */
        int *op1 = bracket[i][0];
        int *op2 = bracket[i][1];
        int opDex[3];
        opDex[0] = op1[1] - 1;

        if (op1[1] == op2[1]){
            opDex[1] = op2[0];
        } else {
            opDex[1] = op2[1] - 1;
        }
        opDex[2] = 3 - opDex[0] - opDex[1];

        /* First operation */
        double result1 = operation(nums[op1[0]], nums[op1[1]], ops[opDex[0]]);
        bool isFirst = false, isSecond = false;

        double next1[3];
        for (int i = 0; i < 3; i++){
            if (i < op1[0]){
                next1[i] = nums[i];
            } else if (i < op1[1]) {
                next1[i] = result1;
            } else {
                next1[i] = nums[i + 1];
            }
        }

        if (op2[0] >= op1[1]){
            op2[0]--;
            isFirst = true;
        }
```

```
        if (op2[1] >= op1[1]){
            op2[1]--;
            isSecond = true;
        }

        /* Second operation */
        double result2 = operation(next1[op2[0]], next1[op2[1]],
ops[opDex[1]]);
        double next2[2];
        for (int i = 0; i < 2; i++){
            if (i < op2[0]){
                next2[i] = next1[i];
            } else if (i < op2[1]){
                next2[i] = result2;
            } else {
                next2[i] = next1[i + 1];
            }
        }
        if (isFirst) {
            op2[0]++;
        } if (isSecond) {
            op2[1]++;
        }

        /* Final operation */
        double res3 = operation(next2[0], next2[1], ops[opDex[2]]);

        /* Get answer list */
        if (res3 == 24){
            string ans = "";
            for (int i = 0; i < 4; i++){
                if (i == op1[0]){
                    ans += "(";
                }
                if (i == op2[0]){
                    ans += "(";
                }
                ans += to_string((int)nums[i]);
                if (i == op1[1]){
                    ans += ")";
                }
                if (i == op2[1]){
                    ans += ")";
                }
                if (i < 3){
                    ans += " " + string(1, ops[i]) + " ";
                }
            }
```

```
            solution -> push_back(ans);
        }

    }

}
```

3. Source code dalam display.cpp

```
/* Nama file    : display.cpp */
/* Fungsi       : Melakukan display, penyimpanan file, dan pengoperasian
dengan brute force */

#include "brackets.cpp"

void start(){
/* Opening program */
    cout << "=========================================\n";
    cout << "=== Selamat datang di '24 Game Solver' ===\n";
    cout << "=========================================\n";
}

void finishing(){
/* Closing program */
    cout << "=============================================================\n";
    cout << "=== Terima kasih karena telah mencoba program ini :) ===\n";
    cout << "=============================================================\n";
}

void displayAnswer(double cards[4]){
    auto timeStart = chrono::high_resolution_clock::now();
    vector<string> allSol;              /* Menyimpan solusi */
    vector<array<char, 3>> allOps;      /* Operators yang legal */
    vector<array<int, 4>> allPossib;
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < 4; j++){
            if (i == j)
                continue;
            for (int k = 0; k < 4; k++){
                if (k == i || k == j)
                    continue;
                for (int l = 0; l < 4; l++){
                    if (l == i || l == j || l == k)
                        continue;
                    array<int, 4> arr = {i, j, k, l};
                    bool isValid = true;
                    for (int i = 0; i < 3 && isValid; i++){
```

```
                        for (int j = i + 1; j < 4 && isValid; j++){
                            if (arr[i] > arr[j] && cards[arr[i]] ==
cards[arr[j]]){
                                isValid = false;
                            }
                        }
                    }
                    if (isValid)
                        allPossib.push_back(arr);
                }
            }
        }
    }

    /* Brute force operators */
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < 4; j++){
            for (int k = 0; k < 4; k++){
                array<char, 3> arr = {ops[i], ops[j], ops[k]};
                allOps.push_back(arr);
            }
        }
    }

    /* Brute force perhitungan */
    for (int i = 0; i < allPossib.size(); i++){
        for (int j = 0; j < allOps.size(); j++){
            double cur_nums[4] = {cards[allPossib[i][0]],
                                  cards[allPossib[i][1]],
                                  cards[allPossib[i][2]],
                                  cards[allPossib[i][3]]};
            char cur_ops[3] =    {allOps[j][0],
                                  allOps[j][1],
                                  allOps[j][2]};
            allPossibleKurung(&allSol, cur_nums, cur_ops);
        }
    }

    /* Perhitungan waktu eksekusi */
    auto exeDone = chrono::high_resolution_clock::now();
    auto durasi = chrono::duration_cast<chrono::microseconds>(exeDone -
timeStart);
    double executionTime = (double)durasi.count() / 1000000;

    /* Finalisasi */
    if (allSol.size() == 0){
        cout << "Tidak ditemukan solusi\nWaktu eksekusi program: " <<
executionTime << " detik\n";
```

```
        return;
    }
    cout << allSol.size() << " solusi ditemukan\n";
    for (int i = 0; i < allSol.size(); i++){
        cout << allSol[i] << endl;
    }
    cout << "Waktu eksekusi program: " << executionTime << " detik\n\n";

    while(true){
        char ans;
        cout << "Apakah Anda ingin menyimpan solusi (y/n)?\n>> ";
        cin >> ans;
        cin.ignore();
        if ((ans == 'y') || (ans == 'Y')){
            /* Menyimpan solusi ke dalam file */
            string name;
            cout << "Masukkan nama file: ";
            getline(cin, name);
            ofstream file("../test/" + name);
            file << "Kartu dipilih  : " << cards[0] << " " << cards[1] << " "
<< cards[2] << " " << cards[3] << endl;
            file << "Jumlah solusi  : " << allSol.size() << "\n";
            file << "Waktu eksekusi : " << executionTime << " detik\n\n";
            for (int i = 0; i < allSol.size(); i++){
                file << allSol[i] << endl;
            }
            file.close();
            cout << "Solusi berhasil disimpan ke file test/" << name << endl;
            finishing();
            break;
        } else if ((ans == 'n') || (ans == 'N')){
            finishing();
            break;
        } else {
            cout << "Input tidak valid!\n";
        }
    }
}
```

4. Source code dalam main.cpp

```
/* Nama file    : main.cpp */
/* Fungsi       : Program utama yang dijalankan */

#include "display.cpp"

int main(){
```

```
    double cards[4];
    start();
    srand(time(NULL));
    userInput(cards);
    displayAnswer(cards);
    return 0;
}
```

BAB IV

INPUT DAN OUTPUT

1. test01.txt

```
========================================
=== Selamat datang di '24 Game Solver' ===
========================================
1. Input manual
2. Dipilih secara random
>> 2
==============================
Kartu yang dipilih: 3 7 8 K
==============================
16 solusi ditemukan
(3 * 13) - (7 + 8)
((3 * 13) - 7) - 8
(3 * 13) - (8 + 7)
((3 * 13) - 8) - 7
8 * (13 - (3 + 7))
8 * ((13 - 3) - 7)
8 * (13 - (7 + 3))
8 * ((13 - 7) - 3)
(13 * 3) - (7 + 8)
((13 * 3) - 7) - 8
(13 - (3 + 7)) * 8
((13 - 3) - 7) * 8
(13 * 3) - (8 + 7)
((13 * 3) - 8) - 7
(13 - (7 + 3)) * 8
((13 - 7) - 3) * 8
Waktu eksekusi program: 0.000458 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test01.txt
Solusi berhasil disimpan ke file test/test01.txt
========================================================
=== Terima kasih karena telah mencoba program ini :) ===
========================================================
```

test > test01.txt

```
1   Kartu dipilih  : 3 7 8 13
2   Jumlah solusi  : 16
3   Waktu eksekusi : 0.000458 detik
4
5   (3 * 13) - (7 + 8)
6   ((3 * 13) - 7) - 8
7   (3 * 13) - (8 + 7)
8   ((3 * 13) - 8) - 7
9   8 * (13 - (3 + 7))
10  8 * ((13 - 3) - 7)
11  8 * (13 - (7 + 3))
12  8 * ((13 - 7) - 3)
13  (13 * 3) - (7 + 8)
14  ((13 * 3) - 7) - 8
15  (13 - (3 + 7)) * 8
16  ((13 - 3) - 7) * 8
17  (13 * 3) - (8 + 7)
18  ((13 * 3) - 8) - 7
19  (13 - (7 + 3)) * 8
20  ((13 - 7) - 3) * 8
21
```

2. test02.txt

```
========================================
=== Selamat datang di '24 Game Solver' ===
========================================
1. Input manual
2. Dipilih secara random
>> 1
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi:
Contoh: 10 A K 8
>> 6 6 10 9
==============================
Kartu yang dipilih: 6 6 10 9
==============================
18 solusi ditemukan
((6 + 10) / 6) * 9
(6 + 10) / (6 / 9)
(6 + 10) * (9 / 6)
((6 + 10) * 9) / 6
((10 + 6) / 6) * 9
(10 + 6) / (6 / 9)
(10 + 6) * (9 / 6)
((10 + 6) * 9) / 6
(10 * (9 - 6)) - 6
(9 / 6) * (6 + 10)
9 / (6 / (6 + 10))
(9 * (6 + 10)) / 6
9 * ((6 + 10) / 6)
(9 / 6) * (10 + 6)
9 / (6 / (10 + 6))
((9 - 6) * 10) - 6
(9 * (10 + 6)) / 6
9 * ((10 + 6) / 6)
Waktu eksekusi program: 0.000228 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test02.txt
Solusi berhasil disimpan ke file test/test02.txt
========================================================
=== Terima kasih karena telah mencoba program ini :) ===
========================================================
```

test > test02.txt

```
1   Kartu dipilih  : 6 6 10 9
2   Jumlah solusi  : 18
3   Waktu eksekusi : 0.000228 detik
4
5   ((6 + 10) / 6) * 9
6   (6 + 10) / (6 / 9)
7   (6 + 10) * (9 / 6)
8   ((6 + 10) * 9) / 6
9   ((10 + 6) / 6) * 9
10  (10 + 6) / (6 / 9)
11  (10 + 6) * (9 / 6)
12  ((10 + 6) * 9) / 6
13  (10 * (9 - 6)) - 6
14  (9 / 6) * (6 + 10)
15  9 / (6 / (6 + 10))
16  (9 * (6 + 10)) / 6
17  9 * ((6 + 10) / 6)
18  (9 / 6) * (10 + 6)
19  9 / (6 / (10 + 6))
20  ((9 - 6) * 10) - 6
21  (9 * (10 + 6)) / 6
22  9 * ((10 + 6) / 6)
```

3. test03.txt

```
==========================================
=== Selamat datang di '24 Game Solver' ===
==========================================
1. Input manual
2. Dipilih secara random
>> 2
==============================
Kartu yang dipilih: J K 6 Q
==============================
6 solusi ditemukan
((13 - 11) * 6) + 12
(6 * (13 - 11)) + 12
12 - ((11 - 13) * 6)
12 + ((13 - 11) * 6)
12 - (6 * (11 - 13))
12 + (6 * (13 - 11))
Waktu eksekusi program: 0.000342 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test03.txt
Solusi berhasil disimpan ke file test/test03.txt
============================================================
=== Terima kasih karena telah mencoba program ini :) ===
============================================================
```

test > test03.txt

```
1   Kartu dipilih  : 11 13 6 12
2   Jumlah solusi  : 6
3   Waktu eksekusi : 0.000342 detik
4
5   ((13 - 11) * 6) + 12
6   (6 * (13 - 11)) + 12
7   12 - ((11 - 13) * 6)
8   12 + ((13 - 11) * 6)
9   12 - (6 * (11 - 13))
10  12 + (6 * (13 - 11))
```

4. test04.txt

```
======================================
=== Selamat datang di '24 Game Solver' ===
======================================
1. Input manual
2. Dipilih secara random
>> 1
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi
Contoh: 10 A K 8
>> 5 5 6 6
============================
Kartu yang dipilih: 5 5 6 6
============================
11 solusi ditemukan
(5 * 5) - (6 / 6)
((5 + 5) - 6) * 6
(5 + (5 - 6)) * 6
((5 - 6) + 5) * 6
(5 - (6 - 5)) * 6
5 * (6 - (6 / 5))
6 * ((5 + 5) - 6)
6 * (5 + (5 - 6))
6 * ((5 - 6) + 5)
6 * (5 - (6 - 5))
(6 - (6 / 5)) * 5
Waktu eksekusi program: 0.000128 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test04.txt
Solusi berhasil disimpan ke file test/test04.txt
==================================================
=== Terima kasih karena telah mencoba program ini :) ===
==================================================
```

test > test04.txt

```
1   Kartu dipilih  : 5 5 6 6
2   Jumlah solusi  : 11
3   Waktu eksekusi : 0.000128 detik
4
5   (5 * 5) - (6 / 6)
6   ((5 + 5) - 6) * 6
7   (5 + (5 - 6)) * 6
8   ((5 - 6) + 5) * 6
9   (5 - (6 - 5)) * 6
10  5 * (6 - (6 / 5))
11  6 * ((5 + 5) - 6)
12  6 * (5 + (5 - 6))
13  6 * ((5 - 6) + 5)
14  6 * (5 - (6 - 5))
15  (6 - (6 / 5)) * 5
```

5. test05.txt

```
========================================
=== Selamat datang di '24 Game Solver' ===
========================================
1. Input manual
2. Dipilih secara random
>> 1
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi
Contoh: 10 A K 8
>> 5 6 5 5
==============================
Kartu yang dipilih: 5 6 5 5
==============================
10 solusi ditemukan
(5 - 6) + (5 * 5)
5 - (6 - (5 * 5))
((5 * 5) - 6) + 5
(5 * 5) - (6 - 5)
(5 * 5) + (5 - 6)
((5 * 5) + 5) - 6
(5 + (5 * 5)) - 6
5 + ((5 * 5) - 6)
(5 - (5 / 5)) * 6
6 * (5 - (5 / 5))
Waktu eksekusi program: 0.000102 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test05.txt
Solusi berhasil disimpan ke file test/test05.txt
========================================================
=== Terima kasih karena telah mencoba program ini :) ===
========================================================
```

test05.txt

```
1   Kartu dipilih  : 5 6 5 5
2   Jumlah solusi  : 10
3   Waktu eksekusi : 0.000102 detik
4
5   (5 - 6) + (5 * 5)
6   5 - (6 - (5 * 5))
7   ((5 * 5) - 6) + 5
8   (5 * 5) - (6 - 5)
9   (5 * 5) + (5 - 6)
10  ((5 * 5) + 5) - 6
11  (5 + (5 * 5)) - 6
12  5 + ((5 * 5) - 6)
13  (5 - (5 / 5)) * 6
14  6 * (5 - (5 / 5))
```

6. test06.txt

```
========================================
=== Selamat datang di '24 Game Solver' ===
========================================
1. Input manual
2. Dipilih secara random
>> 3
Masukan tidak valid. Harap ulangi!
1. Input manual
2. Dipilih secara random
>> 1
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi
Contoh: 10 A K 8
>> 2 5 2 2
==============================
Kartu yang dipilih: 2 5 2 2
==============================
8 solusi ditemukan
2 * ((5 * 2) + 2)
((2 * 5) + 2) * 2
(2 + (5 * 2)) * 2
2 * ((2 * 5) + 2)
2 * (2 + (5 * 2))
(2 + (2 * 5)) * 2
2 * (2 + (2 * 5))
((5 * 2) + 2) * 2
Waktu eksekusi program: 8.4e-05 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test06.txt
Solusi berhasil disimpan ke file test/test06.txt
========================================================
=== Terima kasih karena telah mencoba program ini :) ===
========================================================
```

test06.txt

```
1   Kartu dipilih  : 2 5 2 2
2   Jumlah solusi  : 8
3   Waktu eksekusi : 8.4e-05 detik
4
5   2 * ((5 * 2) + 2)
6   ((2 * 5) + 2) * 2
7   (2 + (5 * 2)) * 2
8   2 * ((2 * 5) + 2)
9   2 * (2 + (5 * 2))
10  (2 + (2 * 5)) * 2
11  2 * (2 + (2 * 5))
12  ((5 * 2) + 2) * 2
```

7. test07.txt

```
=========================================
=== Selamat datang di '24 Game Solver' ===
=========================================
1. Input manual
2. Dipilih secara random
>> 2
============================
Kartu yang dipilih: 6 Q 9 7
============================
22 solusi ditemukan
6 * ((9 - 12) + 7)
6 * (9 - (12 - 7))
6 * ((9 + 7) - 12)
6 * (9 + (7 - 12))
(6 * (9 - 7)) + 12
6 * ((7 - 12) + 9)
6 * (7 - (12 - 9))
6 * ((7 + 9) - 12)
6 * (7 + (9 - 12))
12 + (6 * (9 - 7))
12 - (6 * (7 - 9))
12 + ((9 - 7) * 6)
12 - ((7 - 9) * 6)
((9 - 12) + 7) * 6
(9 - (12 - 7)) * 6
((9 - 7) * 6) + 12
((9 + 7) - 12) * 6
(9 + (7 - 12)) * 6
((7 - 12) + 9) * 6
(7 - (12 - 9)) * 6
((7 + 9) - 12) * 6
(7 + (9 - 12)) * 6
Waktu eksekusi program: 0.000356 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> y
Masukkan nama file: test07.txt
Solusi berhasil disimpan ke file test/test07.txt
=====================================================
=== Terima kasih karena telah mencoba program ini :) ===
=====================================================
```

test > test07.txt

```
1   Kartu dipilih  : 6 12 9 7
2   Jumlah solusi  : 22
3   Waktu eksekusi : 0.000356 detik
4
5   6 * ((9 - 12) + 7)
6   6 * (9 - (12 - 7))
7   6 * ((9 + 7) - 12)
8   6 * (9 + (7 - 12))
9   (6 * (9 - 7)) + 12
10  6 * ((7 - 12) + 9)
11  6 * (7 - (12 - 9))
12  6 * ((7 + 9) - 12)
13  6 * (7 + (9 - 12))
14  12 + (6 * (9 - 7))
15  12 - (6 * (7 - 9))
16  12 + ((9 - 7) * 6)
17  12 - ((7 - 9) * 6)
18  ((9 - 12) + 7) * 6
19  (9 - (12 - 7)) * 6
20  ((9 - 7) * 6) + 12
21  ((9 + 7) - 12) * 6
22  (9 + (7 - 12)) * 6
23  ((7 - 12) + 9) * 6
24  (7 - (12 - 9)) * 6
25  ((7 + 9) - 12) * 6
26  (7 + (9 - 12)) * 6
```

8. Tidak ditemukan solusi

```
=========================================
=== Selamat datang di '24 Game Solver' ===
=========================================
1. Input manual
2. Dipilih secara random
>> 2
============================
Kartu yang dipilih: 6 Q 7 7
============================
Tidak ditemukan solusi
Waktu eksekusi program: 0.000189 detik
```

9. Semua kartu sama

```
========================================
=== Selamat datang di '24 Game Solver' ===
========================================
1. Input manual
2. Dipilih secara random
>> 1
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi:
Contoh: 10 A K 8
>> 4 4 4 4
==============================
Kartu yang dipilih: 4 4 4 4
==============================
6 solusi ditemukan
(4 * 4) + (4 + 4)
((4 * 4) + 4) + 4
(4 + (4 * 4)) + 4
4 + ((4 * 4) + 4)
(4 + 4) + (4 * 4)
4 + (4 + (4 * 4))
Waktu eksekusi program: 3.8e-05 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> k
Input tidak valid!
Apakah Anda ingin menyimpan solusi (y/n)?
>> n
========================================================
=== Terima kasih karena telah mencoba program ini :) ===
========================================================
```

10. Input tidak sesuai

```
========================================
=== Selamat datang di '24 Game Solver' ===
========================================
1. Input manual
2. Dipilih secara random
>> 1
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi:
Contoh: 10 A K 8
>> wrong input confirmed yes
Masukan tidak valid. Harap ulangi!
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi:
Contoh: 10 A K 8
>> a b c d
Masukan tidak valid. Harap ulangi!
Masukkan 4 input kartu (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) pisahkan dengan spasi:
Contoh: 10 A K 8
>> 6 6 J 6
==============================
Kartu yang dipilih: 6 6 J 6
==============================
2 solusi ditemukan
(6 * (11 - 6)) - 6
((11 - 6) * 6) - 6
Waktu eksekusi program: 0.0001 detik

Apakah Anda ingin menyimpan solusi (y/n)?
>> k
Input tidak valid!
Apakah Anda ingin menyimpan solusi (y/n)?
>> n
========================================================
=== Terima kasih karena telah mencoba program ini :) ===
========================================================
```

BAB V

KESIMPULAN

Program yang telah dibuat oleh penulis merupakan program yang bisa mendapatkan semua solusi yang mungkin untuk menyelesaikan permainan 24. Program yang telah dibuat ini melakukan seluruh output kemungkinan dengan adanya validasi agar tidak ada lebih dari satu output yang **sama persis**. Output dari program mungkin bisa menghasilkan hasil yang mirip tetapi tidak sama persis. Dengan adanya program ini, maka telah terbukti bahwa algoritma brute force bisa menyelesaikan permainan 24 dengan waktu yang tidak lama.

BAB VI

LAMPIRAN

Link repository github: https://github.com/WildanGhaly/Tucil1_13521015

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