Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh :

Muhammad Zulfiansyah Bayu Pratama

NIM : 13521028

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

1. Penjelasan Algoritma *Brute Force*

Brute force bukan merupakan suatu algoritma yang khusus, tetapi adalah salah satu strategi penyelesaian masalah yang sederhana, serta dapat dipastikan menemukan solusi yang kita harapkan, walaupun pada umumnya memerlukan waktu yang cukup lama. Sebutan lain dari *brute force* adalah *complete search* dan *exhaustive search*. Prinsip dari *brute force* hanya satu, yaitu mencoba semua kemungkinan yang ada.

Brute force menjamin solusi pasti benar karena menelusuri semua kemungkinan. Akibatnya, secara umum *brute force* bekerja dengan lambat, terutama ketika terdapat kemungkinan solusi yang sangat banyak untuk dicoba. Dalam membangkitkan seluruh kemungkinan, kita dapat melakukannya secara iteratif maupun rekursif. Biasanya, *brute force* digunakan untuk permasalahan dengan total pencarian yang kecil. *Brute force* pula menjadi tolak ukur bagi algoritma yang lebih efisien.

Permasalahan yang diangkat pada tugas kecil kali ini adalah bagaimana suatu algoritma dengan pendekatan *brute foce* dapat mencari solusi dari permainan kartu 24. Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Pada tugas kali ini angk yang digunakan adalah angka dari 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.

Untuk menyimpan angka yang direpresentasikan melalui kartu remi agar dapat diproses oleh komputer, maka data tersebut direpresansikan ke dalam bentuk *Abstract Data Type* (ADT). Penulis menggunakan ADT Larik Dinamis dan ADT Pecahan. ADT Larik Dinamis digunakan untuk menyimpan data dari keempat angka. Penulis memilih menggunakan ADT Larik Dinamis karena dapat diisi dan dihapus elemennya sehingga memudahkan penulis untuk mengatur data ketika telah terjadi operasi antar dua bilangan. Pada implementasi ADT Larik Dinamis, penulis menggunakan library *vector* pada *Standard Template Library* (STL) C++. Selain itu, penulis juga menggunakan ADT Pecahan untuk menyimpan dan memproses data bilangan. Penulis memakai ADT Pecahan karena terdapat operasi pembagian antar dua bilangan misal x/y dimana y belum tentu habis membagi x. Untuk meningkatkan keakuratan dalam pembagian desimal, penulis memilih untuk menyimpan data dari suatu bilangan ke dalam bentuk

pecahan daripada memakai tipe data *floating point* yang tingkat keakuratannya lebih rendah dibandingkan bilangan bulat.

Setelah ADT didefinisikan, maka langkah selanjutnya adalah merancang strategi *brute force* menggunakan ADT List Dinamis dan ADT Pecahan. Pengguna akan diminta memasukkan empat kartu remi yang akan dikonversi menjadi angka dan disimpan ke dalam ADT Pecahan. Lalu empat pecahan tersebut dimasukkan ke dalam ADT List Dinamis menggunakan *vector* dari STL C++. Penulis menggunakan tipe data string untuk menyimpan bentuk operasi dari empat bilangan tersebut. Pada awalnya string dan ADT List Dinamis membuat salinannya. Dalam setiap kemungkinan, dua pecahan dalam ADT List Dinamis akan dioperasikan lalu kedua pecahan tersebut dihapus dan digantikan dengan elemen hasil operasi kedua tersebut. Setiap perubahan yang terjadi pada ADT List Dinamis akan dicatat oleh tipe data string. Ketika hasil dari empat bilangan tersebut adalah 24, maka solusi akan dicetak dan disimpan menggunakan ADT List Dinamis yang berbeda dengan tipe data string. Lalu data dari list dinamis akan dikembalikan dengan salinan sebelumnya. Ketika semua kemungkinan telah ditelusuri, program akan menampilkan solusi yang mungkin dan meminta pengguna apakah *output* program tersebut ingin disimpan dalam bentuk file .txt atau tidak.

Algoritma yang telah dipaparkan dapat dikatakan menjadi realisasi teknik *brute force* karena menulusuri kemungkinan operasi matematika yang sesuai untuk empat bilangan representasi dari kartu remi agar hasilnya sama dengan 24. Meskipun teknik ini sangat menyita waktu, penulis mengoptimasi beberapa bagian pada kemungkinan pada pembagian. Ketika suatu bilangan misal x akan dibagi dengan bilangan lainnya misal y, maka dipastikan y harus tidak sama dengan nol karena suatu bilangan dibagi dengan nol sama dengan tidak terdefinisi.

2. *Souce Code* dengan Bahasa C++

A. *src/main.cpp*

```
// Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force
#include "splash.h"
#include "bruteforce.h"
using namespace std;

typedef long long ll;

void process(string input[]) {
    // Konversi input ke bilangan
    int output[4];
    cardToInt(input, output);

    vector<int> card(4);
    for (int i = 0; i < 4; i++) {
        card[i] = output[i];
    }
    float execution_time = clock();

    // Lakukan permutasi dari card dan cek apakah ada solusi tanpa
menggunakan STL
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            if (j == i) {
                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;
                    }
                    vector<int> temp;
                    temp.push_back(card[i]);
                    temp.push_back(card[j]);
                    temp.push_back(card[k]);
                    temp.push_back(card[l]);
                    bfFirst(temp);
                }
            }
        }
    }

    execution_time = clock() - execution_time;

    // Output jawaban
    cout << endl << "Output : " << endl;
    if (cnt == 0) {
        cout << "No Solution!" << endl;
    } else {
        cout << cnt << " solution found!" << endl;
        for (int i = 0; i < jawaban.size(); i++) {
            cout << jawaban[i] << endl;
        }
    }

    cout << endl;
    string time = "Execution Time = " + to_string(execution_time) + " ms";
    for (int i = 0; i < time.length() + 4; i++) {
        cout << "-";
    }
    cout << endl;
```

```
    cout << "| " << time << " |" << endl;
    for (int i = 0; i < time.length() + 4; i++) {
        cout << "-";
    }

    // Cek apakah output ingin dieksport ke file
    cout << "\n\nApakah anda ingin mengeksport output ke file? (tekan y
untuk YA) : ";
    string inp;
    cin >> inp;
    if (inp == "y") {
        exportAnswer(input);
    }
}

void Input(string pesan_kesalahan)
{
    while (true)
    {
        int output[100];
        string input[1000];
        clearScreen();
        splash();

        cout << pesan_kesalahan;

        cout << "Masukkan input 4 kartu yang dipisahkan dengan spasi/enter
: " << endl;
        cout << "------------------------------------------------------------
-" << endl;
        cout << "Input : " << endl;

        string inputWithSpace;
        getline(cin, inputWithSpace);

        int j = 0;
        int len = inputWithSpace.length();
        for (int i = 0; i < len; i++)
        {
            if (inputWithSpace[i] != ' ')
            {
                input[j] += inputWithSpace[i];
            }
            else
            {
                j++;
            }
        }

        if (j != 3)
        {
            pesan_kesalahan = "Input harus terdiri dari 4 karakter!
Silahkan coba lagi.\n\n";
            continue;
        }

        // Jika salah dua kartu ada yang sama maka input tidak valid
        if (input[0] == input[1] || input[0] == input[2] || input[0] ==
input[3] || input[1] == input[2] || input[1] == input[3] || input[2] ==
input[3])
        {
            pesan_kesalahan = "Input tidak boleh sama! Silahkan coba
lagi.\n\n";
            continue;
        }
        cardToInt(input, output);
```

```
        if (output[0] == -1 || output[1] == -1 || output[2] == -1 ||
output[3] == -1)
        {
            pesan_kesalahan = "Input tidak valid! Silahkan coba lagi.\n\n";
            continue;
        }
        else
        {
            process(input);
            break;
        }
    }


}

void randomInput()
{

    string input[4];
    randomInput(input);

    clearScreen();
    splash();

    cout << "Random Input" << endl;
    cout << "------------" << endl;
    cout << "Input : " << endl;
    for (int i = 0; i < 4; i++)
    {
        cout << input[i] << " ";
    }
    cout << endl;

    process(input);
}

void chooseInput(string pesan_kesalahan)
{
    while (true)
    {
        clearScreen();
        splash();

        cout << pesan_kesalahan;

        cout << "Pilih input : " << endl;
        cout << "------------------------------------------------------------
-" << endl;
        cout << "1. Input manual" << endl;
        cout << "2. Input Random" << endl;
        cout << "3. Keluar" << endl;
        cout << "------------------------------------------------------------
-" << endl;
        cout << "Input : " << endl;

        string input;
        getline(cin, input);

        if (input == "1")
        {
            Input("");
            break;
        }
        else if (input == "2")
        {
            randomInput();
```

```
            break;
        }
        else if (input == "3")
        {
            exit(0);
        }
        else
        {
            pesan_kesalahan = "Input tidak valid! Silahkan coba lagi.\n\n";
            continue;
        }
    }
}
int main()
{
    chooseInput("");
}
```

B. *src/bilangan.h*

```
#include <bits/stdc++.h>

#ifndef _BILANGAN_H
#define _BILANGAN_H

typedef struct
{
    int pembilang;
    int penyebut;
} bil;

int lcm(int x, int y)
{
    return (x * y) / __gcd(x, y);
}

void simpleFraction(bil *a)
{
    int fpb = __gcd((*a).pembilang, (*a).penyebut);
    (*a).pembilang /= fpb;
    (*a).penyebut /= fpb;
}
```

```
bil operasi(char opr, bil angka1, bil angka2)
{
    bil ans;

    if (opr == '*')
    {
        ans.pembilang = angka1.pembilang * angka2.pembilang;
        ans.penyebut = angka1.penyebut * angka2.penyebut;
    }
    else if (opr == '/')
    {
        ans.pembilang = angka1.pembilang * angka2.penyebut;
        ans.penyebut = angka1.penyebut * angka2.pembilang;
    }
    else
    {
        if (angka1.penyebut != angka2.penyebut)
        {
            int kpk = lcm(angka1.penyebut, angka2.penyebut);
            angka1.pembilang = angka1.pembilang * kpk / angka1.penyebut;
            angka2.pembilang = angka2.pembilang * kpk / angka2.penyebut;
            angka1.penyebut = kpk;
            angka2.penyebut = kpk;
        }

        if (opr == '+')
            ans.pembilang = angka1.pembilang + angka2.pembilang;
        else
            ans.pembilang = angka1.pembilang - angka2.pembilang;
        ans.penyebut = angka1.penyebut;
    }
    return ans;
}

#endif
```

C. *src/bruteforce.h*

```
#include "fitur.h"

#ifndef _BRUTEFORCE_H_
#define _BRUTEFORCE_H_

// Prosedur ketika kartu pertama dan kedua sudah dioperasikan
void bfSecond_1(vector<bil> card, string s)
{
    // Buat string dan vector sementara
    string tempS = s;
    vector<bil> tempCard = card;

    for (int i = 0; i < 4; i++)
    {
        if (i == 3)
        {
            if (isDivisible(card[0].pembilang, card[1].pembilang) == false)
            {
                break;
            }
        }
        tempCard[1] = operasi(op[i], card[0], card[1]);

        tempCard.erase(tempCard.begin());
        tempS = "(" + s + op[i] + to_string(card[1].pembilang) + ")";
```

```
        tempS = "(" + s + op[i] + to_string(card[1].pembilang) + ")";

        vector<bil> tempCard2 = tempCard;

        // Sekarang operasikan kartu hasil operasi barusan dengan kartu
keempat
        for (int j = 0; j < 4; j++)
        {
            if (j == 3)
            {
                if (isDivisible(tempCard[0].pembilang, card[2].pembilang)
== false)
                {
                    break;
                }
            }

            tempCard[0] = operasi(op[j], tempCard[0], card[2]);

            tempS = tempS + op[j] + to_string(card[2].pembilang);
            if (tempCard[0].pembilang % tempCard[0].penyebut == 0)
                cekValid(tempCard[0].pembilang / tempCard[0].penyebut,
tempS);

            // Kembalikan string dan tempCard ke semula
            tempS = "(" + s + op[i] + to_string(card[1].pembilang) + ")";
            tempCard = tempCard2;
        }

        // Kembalikan string dan tempCard ke semula
        tempS = s;
        tempCard = card;
    }
}

// Prosedur ketika kartu ketiga dan keempat sudah dioperasikan
void bfSecond_2(vector<bil> card, string s)
{
    // Buat string dan vector sementara
    string tempS;
    vector<bil> tempCard = card;

    for (int i = 0; i < 4; i++)
    {
        // Kemungkinan semua operator
        if (i == 3)
        {
            if (isDivisible(card[0].pembilang, card[1].pembilang) == false)
            {
                break;
            }
        }

        tempCard[1] = operasi(op[i], card[0], card[1]);

        tempCard.erase(tempCard.begin());
        tempS = "(" + to_string(card[0].pembilang) + op[i] +
to_string(card[1].pembilang) + ")";
        vector<bil> tempCard2 = tempCard;

        for (int j = 0; j < 4; j++)
        {
            if (j == 3)
            {
                if (isDivisible(tempCard[0].pembilang, card[2].pembilang)
== false)
                {
```

```
                break;
            }
        }

        tempCard[1] = operasi(op[i], card[0], card[1]);

        tempCard.erase(tempCard.begin());
        tempS = "(" + to_string(card[0].pembilang) + op[i] +
to_string(card[1].pembilang) + ")";
        vector<bil> tempCard2 = tempCard;

        for (int j = 0; j < 4; j++)
        {
            if (j == 3)
            {
                if (isDivisible(tempCard[0].pembilang, card[2].pembilang)
== false)
                {
                    break;
                }
            }

            tempCard[0] = operasi(op[j], tempCard[0], card[2]);

            tempS = tempS + op[j] + s;
            if (tempCard[0].pembilang % tempCard[0].penyebut == 0)
                cekValid(tempCard[0].pembilang / tempCard[0].penyebut,
tempS);
            // cekValid(tempCard[0].pembilang / tempCard[0].penyebut,
tempS);

            // Kembalikan string dan tempCard ke semula
            tempS = "(" + to_string(card[0].pembilang) + op[i] +
to_string(card[1].pembilang) + ")";
            tempCard = tempCard2;
        }

        // Kembalikan tempCard ke semula
        tempCard = card;
    }
}

void bfThird_1(vector<bil> card, string s)
{
    string tempS;
    vector<bil> tempCard = card;

    // Operasikan kartu pertama dengan hasil kartu dari operasi pertama
lalu operasikan dengan kartu keempat
    for (int i = 0; i < 4; i++)
    {
        if (i == 3)
        {
            if (isDivisible(card[0].pembilang, card[1].pembilang) == false)
            {
                break;
            }
        }

        tempCard[1] = operasi(op[i], card[0], card[1]);

        tempCard.erase(tempCard.begin());
        tempS = "(" + to_string(card[0].pembilang) + op[i] + s + ")";
        // cout << "\t\t" + tempS << endl;
        vector<bil> tempCard2 = tempCard;

        // Sekarang operasikan kartu hasil operasi barusan dengan kartu
```

```
        for (int j = 0; j < 4; j++)
        {
            if (j == 3)
            {
                if (isDivisible(tempCard[0].pembilang, card[2].pembilang)
== false)
                {
                    break;
                }
            }
            tempCard[0] = operasi(op[j], tempCard[0], card[2]);

            tempS = tempS + op[j] + to_string(card[2].pembilang);
            if (tempCard[0].pembilang % tempCard[0].penyebut == 0)
                cekValid(tempCard[0].pembilang / tempCard[0].penyebut,
tempS);

            // Kembalikan string dan tempCard ke semula
            tempS = "(" + to_string(card[0].pembilang) + op[i] + s + ")";
            tempCard = tempCard2;
        }

        // Kembalikan string dan tempCard ke semula
        tempS = s;
        tempCard = card;
    }
}

void bfThird_2(vector<bil> card, string s)
{
    string tempS;
    vector<bil> tempCard = card;

    // Operasikan kartu keempat dengan hasil kartu dari operasi pertama
lalu operasikan dengan kartu pertama
    for (int i = 0; i < 4; i++)
    {
        // Kemungkinan semua operator
        if (i == 3)
        {
            if (isDivisible(card[1].pembilang, card[2].pembilang) == false)
            {
                break;
            }
        }

        tempCard[1] = operasi(op[i], card[1], card[2]);

        tempCard.erase(tempCard.begin() + 2);
        tempS = "(" + s + op[i] + to_string(card[2].pembilang) + ")";
        // cout << "\t\t" + tempS << endl;
        vector<bil> tempCard2 = tempCard;

        // Sekarang operasikan kartu hasil operasi barusan dengan kartu
pertama
        for (int j = 0; j < 4; j++)
        {
            if (j == 3)
            {
                if (isDivisible(card[0].pembilang, tempCard[1].pembilang)
== false)
                {
                    break;
                }
            }
            tempCard[0] = operasi(op[j], card[0], tempCard[1]);
```

```
            tempS = to_string(card[0].pembilang) + op[j] + tempS;
            if (tempCard[0].pembilang % tempCard[0].penyebut == 0)
                cekValid(tempCard[0].pembilang / tempCard[0].penyebut,
tempS);

            // Kembalikan string dan tempCard ke semula
            tempS = "(" + s + op[i] + to_string(card[2].pembilang) + ")";
            tempCard = tempCard2;
        }

        // Kembalikan string dan tempCard ke semula
        tempS = s;
        tempCard = card;
    }
}

// Prosedur ketika kartu kedua dan ketiga sudah dioperasikan
void bfSecond_3(vector<bil> card, string s)
{
    // Ada dua cabang dalam operasi ini
    // 1. Operasikan kartu pertama dengan hasil kartu dari operasi pertama
lalu operasikan dengan kartu keempat
    // 2. Operasikan kartu keempat dengan hasil kartu dari operasi pertama
lalu operasikan dengan kartu pertama

    // Operasi 1
    bfThird_1(card, s);
    bfThird_2(card, s);
}

// Prosedur untuk menentukan kartu mana yang harus dioperasikan terlebih
dahulu
void bfFirst(vector<int> c)
{
    vector<bil> card(4);
    for (int i = 0; i < 4; i++)
    {
        card[i].pembilang = c[i];
        card[i].penyebut = 1;
    }
    vector<bil> tempCard = card;
    string s;

    for (int i = 0; i < 3; i++)
    {

        int j = i + 1;

        for (int k = 0; k < 4; k++)
        {
            if (op[k] == '/')
            {
                if (isDivisible(card[i].pembilang, card[j].pembilang) ==
false)
                {
                    break;
                }
            }
            tempCard[j] = operasi(op[k], card[i], card[j]);

            // cout << "\t"
            //      << "operasi = " << op[k] << endl;
            tempCard.erase(tempCard.begin() + i);
            s = "(" + to_string(card[i].pembilang) + op[k] +
to_string(card[j].pembilang) + ")";

            if (i == 0)
            {
```

```
                bfSecond_1(tempCard, s);
            }
            else if (i == 2)
            {
                bfSecond_2(tempCard, s);
            }
            else
            {
                bfSecond_3(tempCard, s);
            }

            // Kembalikan string dan tempCard ke semula
            s = "";
            tempCard = card;
        }
    }
}

#endif
```

D. *src/fitur.h*

```
#ifndef _FITUR_H
#define _FITUR_H
#include "bilangan.h"

int cnt = 0;
vector<string> jawaban;

char op[4] = {'+', '-', '*', '/'};

void clearScreen() {
    cout << "\033[2J" << "\033[H";
}

// Fungsi mengekspor jawaban ke file
void exportAnswer(string input[]) {
    ofstream file;
    file.open("../test/answer_" + input[0] + "_" + input[1] + "_" +
input[2] + "_" + input[3] + "_" + ".txt");
    if (cnt == 0) {
        file << "No Solution!" << endl;
    } else {
        file << cnt << " solution found!" << endl;
        for (int i = 0; i < jawaban.size(); i++) {
            file << jawaban[i] << endl;
        }
    }
    file.close();
    cout << endl;
    cout << "----------------------------------------" << endl;
    cout << "| Jawaban diexport ke file jawaban.txt |" << endl;
    cout << "----------------------------------------" << endl;
}

// Fungsi memilih kartu random
void randomInput(string input[])
{
    for (int i = 0; i < 4; i++)
    {
        int random = rand() % 13 + 1;
        if (random == 1)
        {
            input[i] = "A";
```

```
        }
        else if (random == 11)
        {
            input[i] = "J";
        }
        else if (random == 12)
        {
            input[i] = "Q";
        }
        else if (random == 13)
        {
            input[i] = "K";
        }
        else
        {
            input[i] = to_string(random);
        }
    }

}

// Fungsi cek apakah bilangan bisa dibagi
bool isDivisible(int a, int b)
{
    if (b != 0)
    {
        return true;
    }
    return false;
}

// Mengubah kartu menjadi angka
void cardToInt(string input[4], int output[4])
{
    for (int i = 0; i < 4; i++)
    {
        if (input[i] == "A")
        {
            output[i] = 1;
        }
        else if (input[i] == "J")
        {
            output[i] = 11;
        }
        else if (input[i] == "Q")
        {
            output[i] = 12;
        }
        else if (input[i] == "K")
        {
            output[i] = 13;
        }
        else
        // Cek apakah input bukan merupakan angka
        if (input[i][0] >= '2' && input[i][0] <= '9' && input[i][1] ==
'\0')
        {
            output[i] = stoi(input[i]);
        }
        else
        if (input[i][0] == '1' && input[i][1] == '0')
        {
            output[i] = 10;
        }
        else
        {
            output[i] = -1;
```

```
            }
        }
}

void cekValid(int ans, string s)
{
    if (ans == 24)
    {
        cnt++;
        jawaban.push_back(s);
        // cout << "\t\t\t" << s << endl;
    }
}

#endif
```

E. *src/splash.h*

```
#include <bits/stdc++.h>
using namespace std;
#ifndef SPLASH_H
#define SPLASH_H

void splash()
{

    cout << "
<< endl;
    cout << "
<< endl;
    cout << "
<<endl;
    cout << "
<< endl;
    cout << "
<< endl;
    cout << "
<< endl << endl;
    cout << "          TUGAS KECIL 1" << endl;
    cout << "          " << endl;
endl;
    cout << "   Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force" <<
endl;
    cout << "           ------------------------------------------" << endl;
    cout << "           | Nama : Muhammad Zulfiansyah Bayu Pratama |" << endl;
    cout << "           | NIM  : 13521028                          |" << endl;
    cout << "           ------------------------------------------" << endl <<
endl;

}

#endif
```

3. *Screenshoot* Masukan dan Keluaran Program

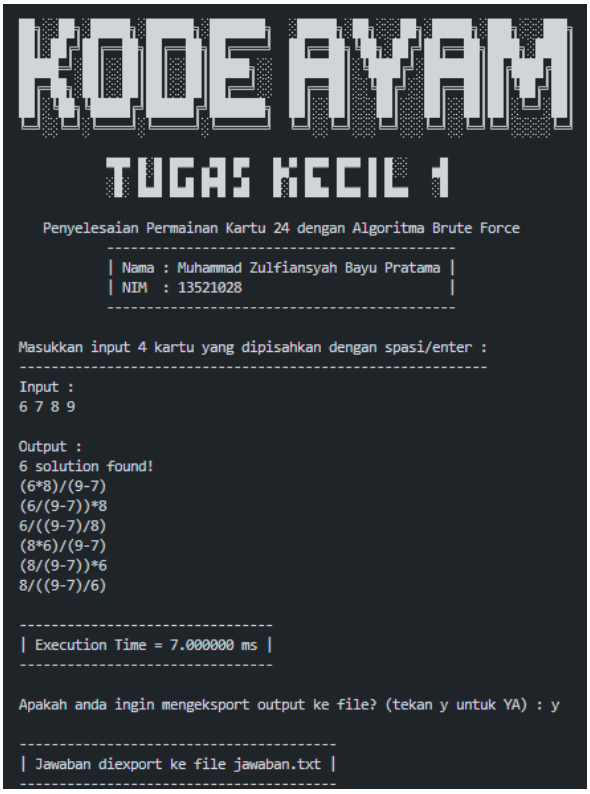
A. Testcase #1

```
Masukkan input 4 kartu yang dipisahkan dengan spasi/enter :
------------------------------------------------------------
Input :
A 8 9 Q

Output :
39 solution found!
((1-8)+9)*12
(1-(8-9))*12
(1*8)*(12-9)
((1+9)-8)*12
(1+(9-8))*12
((1*12)-9)*8
(1*(12-9))*8
1*((12-9)*8)
8*((1*12)-9)
(8*1)*(12-9)
(8/1)*(12-9)
8*((12*1)-9)
8*((12/1)-9)
(8*(12-9))*1
(8*(12-9))/1
8*((12-9)*1)
8*((12-9)/1)
8/((12/9)-1)
((9+1)-8)*12
(9+(1-8))*12
((9-8)+1)*12
(9-(8-1))*12
12*((1-8)+9)
((12*1)-9)*8
((12/1)-9)*8
12*((1+9)-8)
(12-(1*9))*8
((12-9)*1)*8
((12-9)/1)*8
12*((9+1)-8)
(12-(9*1))*8
(12-(9/1))*8
(12-9)*(1*8)
(12-9)/(1/8)
((12-9)*8)*1
((12-9)*8)/1
12*((9-8)+1)
(12-9)*(8*1)
(12-9)*(8/1)
```

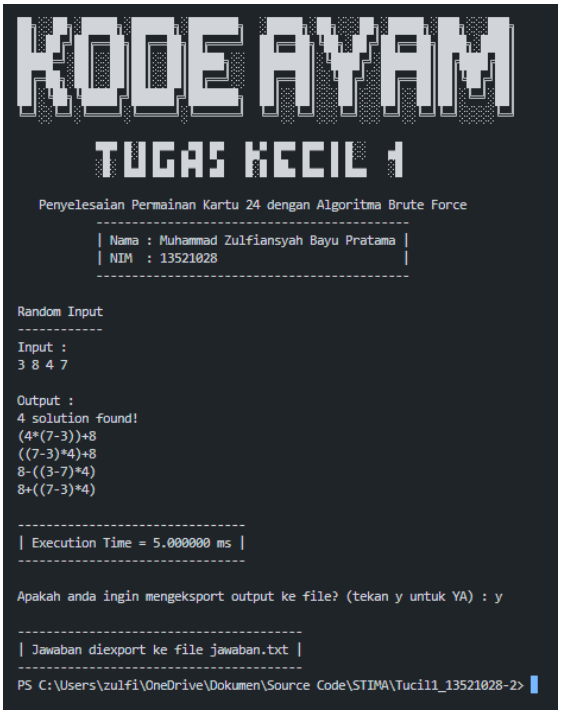
B. Testcase #2

```
Masukkan input 4 kartu yang dipisahkan dengan spasi/enter :
------------------------------------------------------------
Input :
2 6 7 8

Output :
16 solution found!
((2-6)+7)*8
(2-(6-7))*8
((2+7)-6)*8
(2+(7-6))*8
(2*(7+8))-6
(2*(8+7))-6
((7+2)-6)*8
(7+(2-6))*8
((7-6)+2)*8
(7-(6-2))*8
((7+8)*2)-6
8*((2-6)+7)
8*((2+7)-6)
((8+7)*2)-6
8*((7+2)-6)
8*((7-6)+2)

--------------------------------
| Execution Time = 6.000000 ms |
--------------------------------

Apakah anda ingin mengeksport output ke file? (tekan y untuk YA) : y
```

C. Testcase #3



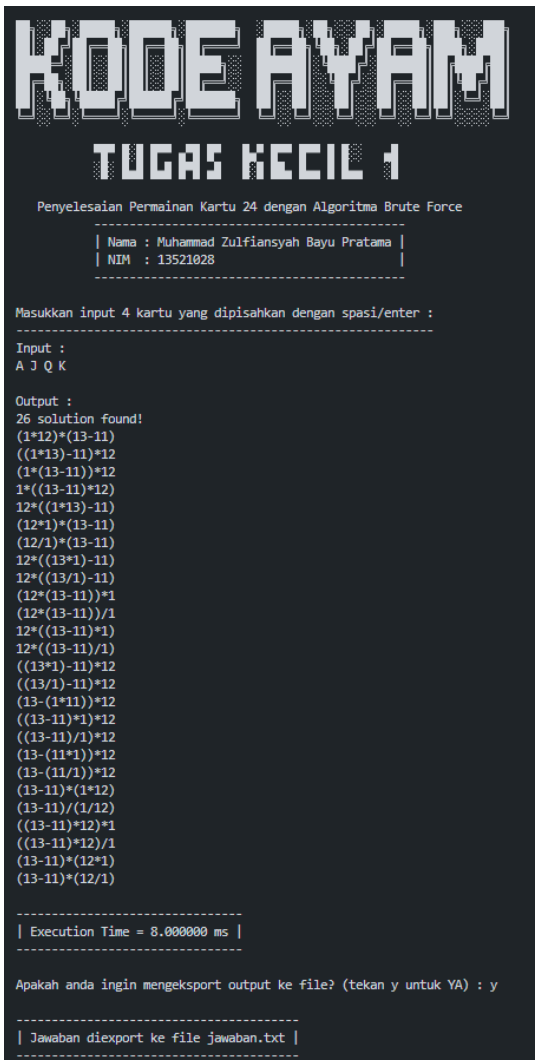
D. Testcase #4



E. Testcase #5


KODE AYAM
TUGAS KECIL 1
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force
| Nama : Muhammad Zulfiansyah Bayu Pratama |
| NIM : 13521028 |
Masukkan input 4 kartu yang dipisahkan dengan spasi/enter :
Input :
8 9 10 J
Output :
No Solution!
| Execution Time = 10.000000 ms |
Apakah anda ingin mengeksport output ke file? (tekan y untuk YA) : y
| Jawaban diexport ke file jawaban.txt |
PS C:\Users\zulfi\OneDrive\Dokumen\Source Code\STIMA\Tucil1_13521028-2>

F. Testcase #6


KODE AYAM
TUGAS KECIL 1
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force
| Nama : Muhammad Zulfiansyah Bayu Pratama |
| NIM : 13521028 |
Masukkan input 4 kartu yang dipisahkan dengan spasi/enter :
Input :
A J Q K
Output :
26 solution found!
(1*12)*(13-11)
((1*13)-11)*12
(1*(13-11))*12
1*((13-11)*12)
12*((1*13)-11)
(12*1)*(13-11)
(12/1)*(13-11)
12*((13*1)-11)
12*((13/1)-11)
(12*(13-11))*1
(12*(13-11))/1
12*((13-11)*1)
12*((13-11)/1)
((13*1)-11)*12
((13/1)-11)*12
(13-(1*11))*12
((13-11)*1)*12
((13-11)/1)*12
(13-(11*1))*12
(13-(11/1))*12
(13-11)*(1*12)
(13-11)/(1/12)
((13-11)*12)*1
((13-11)*12)/1
(13-11)*(12*1)
(13-11)*(12/1)
| Execution Time = 8.000000 ms |
Apakah anda ingin mengeksport output ke file? (tekan y untuk YA) : y
| Jawaban diexport ke file jawaban.txt |

4. Link Github

https://github.com/zulfiansyah404/Tucil1_13521028

5. *Checklist*

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	√	