Tugas Kecil 1

IF2211 Strategi Algoritma

Pencarian Solusi dari Permainan *24 Game* dengan Algoritma Brute Force

Disusun oleh :

Muhammad Naufal Nalendra – 13521152

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

1. Permainan 24 Game dan Algoritma Brute Force

Algoritma *brute force* adalah cara mencari penyelesaian dari masalah yang dilakukan secara sederhana dan memiliki kemungkinan tinggi untuk mendapatkan solusi yang dicari. Hal ini karena algoritma *brute force* bekerja dengan cara menemukan semua kemungkinan yang ada dari suatu permasalahan.

Algoritma tersebut juga biasa disebut *complete search* atau *exhaustive search* dikarenakan cara kerjanya yang mencari semua keseluruhan solusi. Kelebihan dari algoritma ini adalah solusi dari permasalahan dapat dipastikan ketemu, tapi algoritma ini juga memiliki kekurangannya. Kekurangan dari algoritma *brute force* adalah waktu proses yang lama, sebuah efek samping dari cara kerjanya yang *complete*. Hal ini semakin terasa terutama saat kemungkinan dari solusi sangat banyak. Oleh karena itu, algoritma *brute force* biasanya digunakan untuk permasalahan skala kecil.

Kali ini, permasalahan yang perlu dipecahkan dengan algoritma *brute force* adalah mengenai mencari solusi dari permainan 24 game. Permainan kartu 24 adalah permainan kartu berbasis aritmatika dengan tujuan permainan mencari operasi yang cocok dari 4 angka agar dapat mencapai nilai 24. Pada tugas kali ini, angka yang diberikan adalah 1 sampai 13 mengikuti jumlah kartu poker. Nilai tiap kartu mengikuti nilai bilangan kartu dengan pengecualian untuk kartu *As, Jack, Queen,* dan *King* dimana *As* bernilai 1, *Jack* bernilai 11, *Queen* bernilai 12, dan *King* bernilai 13.

Data dari keempat kartu akan disimpan sebagai nilai lalu dioperasikan dengan semua variasi operator (tambah,kurang,kali,bagi) serta urutan operasi (penambahan tanda kurung). Nilai kartu akan disimpan sebagai pecahan karena terdapat operasi pembagian. Hal ini agar nilai akhir dapat terjaga keakuratannya dibanding menggunakan floating point.

Jika hasil operasi dari 4 angka adalah 24, maka operasi yang menghasilkan nilai tersebut akan ditampilkan pada layar. Selanjutnya program akan menanyakan pada user jika solusi ingin disimpan atau tidak.

2. Algoritma Brute Force dan Pembagian program

2.1 Algoritma Operasi Matematika

Berikut adalah gambaran singkat mengenai kode yang digunakan untuk operasi matematika pada program :

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

#ifndef #define _OPERATIONS_H
#define _OPERATIONS_H

using namespace std;

typedef struct ···
} number;

// Fungsi untuk cek apa x bisa dibagi oleh y
bool isDivisible(int x, int y) ···

// Fungsi untuk melakukan operasi matematika sesuai operator
number operation(char opr, number num1, number num2) ···

#endif
```

Pada source code diatas, penulis membuat sebuah struct number atau bilangan yang memiliki elemen numerator atau pembilang dan denom atau penyebut. Bentuk struct diatas berguna untuk mengatasi operasi yang mengandung bentuk pecahan, terutama untuk operator perkalian dan pembagian. Fungsi operation akan mengoperasikan dua angka sesuai operatornya sedangkan fungsi isDivisible berguna menentukan jika bilangan satu dapat dibagi dengan bilangan lainnya.

2.2 Algoritma Kartu

Berikut adalah gambaran singkat mengenai kode yang digunakan untuk kartu :

```
#include <bits/stdc++.h>
#ifndef _CARD_H
#define _CARD_H

#include "operations.h"
using namespace std;

char operators[4] = {'+', '-', '*', '/'};
int counter = 0;
vector<string> solutions;

void displaySaved() {···

// Fungsi untuk menyimpan jawaban ke file
void saveAnswer(string input[]) {···

// Fungsi untuk memberi input kartu secara acak
void randomizedCards(string input[])···

// Fungsi untuk mengubah kartu menjadi angka
void changeInteger(string Cards[4], int num[4])···

#endif
```

Pada source code diatas, penulis membuat dua fungsi utama untuk kartu, yaitu randomizedCards dan changeInteger. Fungsi randomizedCards berfungsi memberikan input kartu secara acak menggunakan fungsi bawaan seperti rand dan srand. Sementara itu, fungsi changeInteger berfungsi mengubah input kartu yang berupa string menjadi integer agar dapat dioperasikan.

2.3 Algoritma Pengurutan Operasi (Brute Force)

Berikut adalah gambaran singkat mengenai kode yang digunakan untuk pengurutan operasi :

```
#include "card.h"

#ifndef _ORDERS_H_
#define _ORDERS_H_

// Fungsi untuk cek jika hasil adalah 24 dan menampilkan persamaan ke layar
void Validate(int sol, string str){···

// Fungsi untuk memanggil fungsi Validate
void execValidate(vector<number> card, string str){···
// Prosedur untuk mengoperasikan hasil operasi 2 kartu pertama dengan kartu selanjutnya
void operLeftRight(vector<number> card, string str, int order){···
// Prosedur untuk mengoperasikan hasil operasi kartu 2 dan 3 dengan kartu 1
void operMiddleLeft(vector<number> card, string str)···
// Prosedur untuk mengoperasikan hasil operasi kartu 2 dan 3 dengan kartu 4
void operMiddleRight(vector<number> card, string str)···

// Prosedur untuk mengoperasikan hasil operasi kartu 2 dan 3 dengan kartu 1 dan 4
void operMiddle(vector<number> card, string str)···

// Prosedur untuk mengoperasikan 2 kartu pertama dari 4 kartu
void operFirstTwo(vector<int> c)···

#endif
```

Pada source code diatas, terdapat beberapa fungsi yang berguna untuk mencari solusi dari 24 game secara brute force. Rincian dari algoritma brute force yang digunakan adalah sebagai berikut:

1) Program menentukan 2 kartu pertama yang akan dioperasikan
2) Jika kartu yang dipilih 2 kartu paling kiri, hasil dioperasikan dengan kartu ketiga dan dilanjutkan dengan kartu keempat
3) Jika kartu yang dipilih 2 kartu paling kanan, hasil dioperasikan dengan kartu kedua dan dilanjutkan dengan kartu pertama
4) Jika kartu yang dipilih 2 kartu di tengah, program akan mencoba variasi antara melanjutkan operasi dengan kartu pertama atau kartu keempat.

3. Source Code Program (C++)

3.1 card.h

```
#include <bits/stdc++.h>
#ifndef _CARD_H
#define _CARD_H

#include "operations.h"
using namespace std;

char operators[4] = {'+', '-', '*', '/'};
int counter = 0;
vector<string> solutions;

void displaySaved() {
    cout << endl;
    cout << "-----------------------------------------------------" << endl;
    cout << "| Your solutions have been exported into a .txt file |" << endl;
    cout << "-----------------------------------------------------" << endl;
}

// Fungsi untuk menyimpan jawaban ke file
void saveAnswer(string input[]) {
    ofstream file;
    file.open("../test/solutions_" + input[0] + "_" + input[1] + "_" + input[2] + "_" + input[3]
+ "_" + ".txt");
    if (counter != 0){
        file << "There are " << counter << " solutions!" << endl;
        for (int i = 0; i < solutions.size(); i++){
            file << solutions[i] << endl;
        }
    }
    else {
        file << "No solutions found!" << endl;
```

```
    }
    file.close();

    // Menunjukkan bahwa hasil save berhasil
    displaySaved();

}

// Fungsi untuk memberi input kartu secara acak
void randomizedCards(string input[])
{
    srand(time(0));
    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 untuk mengubah kartu menjadi angka
void changeInteger(string Cards[4], int num[4])
{
   for (int i = 0; i < 4; i++) {
      if (Cards[i] == "A"){
         num[i] = 1;
      }
      else if (Cards[i] == "J"){
         num[i] = 11;
      }
      else if (Cards[i] == "Q"){
         num[i] = 12;
      }
      else if (Cards[i] == "K"){
         num[i] = 13;
      }
      else if (Cards[i][0] >= '2' && Cards[i][0] <= '9' && Cards[i][1] == '\0'){
         num[i] = stoi(Cards[i]);
      }
      else if (Cards[i][0] == '1' && Cards[i][1] == '0'){
         num[i] = 10;
      }
      else{
         num[i] = -1;
      }
   }
}

#endif
```

3.2 operations.h

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

#ifndef _OPERATIONS_H
#define _OPERATIONS_H

using namespace std;

typedef struct
{
   int numerator;
   int denom;
} number;

// Fungsi untuk cek apa x bisa dibagi oleh y
bool isDivisible(int x, int y)
{
   return (y != 0);
}

// Fungsi untuk melakukan operasi matematika sesuai operator
number operation(char opr, number num1, number num2)
{
   number ans;
   if (opr == '+' || opr == '-'){
      if (num1.denom != num2.denom){
         int lcm = (num1.denom * num2.denom) / __gcd(num1.denom,num2.denom);
         num1.numerator = num1.numerator * lcm / num1.denom;
         num2.numerator = num2.numerator * lcm / num2.denom;
         num1.denom = num2.denom = lcm;
      } else {
         if (opr == '+'){
            ans.numerator = num1.numerator + num2.numerator;
         }
```

```
            else {
                ans.numerator = num1.numerator - num2.numerator;
            }
        }
        ans.denom = num1.denom;
    }

    else if (opr == '*'){
        ans.numerator = num1.numerator * num2.numerator;
        ans.denom = num1.denom * num2.denom;
    }
    else if (opr == '/'){
        ans.numerator = num1.numerator * num2.denom;
        ans.denom = num1.denom * num2.numerator;
    }
    return ans;
}

#endif
```

3.3 ascii.h

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

using namespace std;
#ifndef ASCII_H
#define ASCII_H

void ASCII()
{

cout << " _______  _   ___    _______  _______  __   __  _______ " << endl;
cout << "|       || | |   |  |       ||   _   ||  |_|  ||       |" << endl;
```

```
cout << "|____   || |_|   | |    ___|| |_|  ||      ||    ___|" << endl;
cout << " ____|  ||      | |   | __ |      ||      ||   |___ " << endl;
cout << "| _______||___    | |   ||  ||      ||      ||    ___|" << endl;
cout << "| |_____     |   | |   |_| ||   _   || ||_|| ||   |___ " << endl;
cout << "|________|    |___|  |________||__| |__||_|   |_||________|" << endl;
cout << "--------------------------------------------------------" << endl;

cout << " Permainan kartu 24 menggunakan algoritma brute force" << endl;
cout << "--------------------------------------------------------" << endl;
cout << "           Name : Muhammad Naufal Nalendra          " << endl;
cout << "           NIM  : 13521152                          " << endl;
cout << "--------------------------------------------------------" << endl;
cout << endl;

}

#endif
```

3.4 orders.h

```
#include "card.h"

#ifndef _ORDERS_H_
#define _ORDERS_H_

// Fungsi untuk cek jika hasil adalah 24 dan menampilkan persamaan ke layar
void Validate(int sol, string str){
    if (sol == 24){
        solutions.push_back(str);
        counter++;
    }
}
```

```
// Fungsi untuk memanggil fungsi Validate
void execValidate(vector<number> card, string str){
    if (card[0].numerator % card[0].denom == 0)
        Validate(card[0].numerator / card[0].denom, str);
}
// Prosedur untuk mengoperasikan hasil operasi 2 kartu pertama dengan kartu selanjutnya
void operLeftRight(vector<number> card, string str, int order){
    // Kemungkinan
    // 1. hasil operasi kartu 1 dan 2 lalu dilanjutkan kartu 3
    // 2. hasil operasi kartu 3 dan 4 lalu dilanjutkan kartu 2

    string tempStr = str;
    vector<number> operResult = card;

    for (int i = 0; i < 4; i++)
    {
        if (i == 3 && (isDivisible(card[0].numerator, card[1].numerator) == false)){
            break;
        }
        // operasi dengan kartu 2 atau 3
        operResult[1] = operation(operators[i], card[0], card[1]);

        operResult.erase(operResult.begin());

        if (order == 1)
            tempStr = "(" + str + operators[i] + to_string(card[1].numerator) + ")";
        else
                    tempStr = "(" + to_string(card[0].numerator) + operators[i] +
to_string(card[1].numerator) + ")";

        vector<number> backup = operResult;

        // Lanjutkan operasi dengan kartu 4
```

```
        for (int j = 0; j < 4; j++)
        {
            if (j == 3 && (isDivisible(card[0].numerator, card[1].numerator) == false)){
                break;
            }

            operResult[0] = operation(operators[j], operResult[0], card[2]);

            if (order == 1){
                tempStr = tempStr + operators[j] + to_string(card[2].numerator);
                execValidate(operResult,tempStr);
                tempStr = "(" + str + operators[i] + to_string(card[1].numerator) + ")";
                operResult = backup;
            }
            else {
                tempStr = tempStr + operators[j] + str;
                execValidate(operResult,tempStr);
                        tempStr = "(" + to_string(card[0].numerator) + operators[i] +
to_string(card[1].numerator) + ")";
                operResult = backup;
            }
        }
        // reset isi tempStr dan operResult
        tempStr = str;
        operResult = card;
    }
}
// Prosedur untuk mengoperasikan hasil operasi kartu 2 dan 3 dengan kartu 1
void operMiddleLeft(vector<number> card, string str)
{
    string tempStr;
    vector<number> operResult = card;
```

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

    operResult[1] = operation(operators[i], card[0], card[1]);

    operResult.erase(operResult.begin());
    tempStr = "(" + to_string(card[0].numerator) + operators[i] + str + ")";

    vector<number> backup = operResult;

    // Lanjutkan operasi dengan kartu 4
    for (int j = 0; j < 4; j++)
    {
        if (j == 3)
        {
            if (isDivisible(operResult[0].numerator, card[2].numerator) == false)
            {
                break;
            }
        }
        operResult[0] = operation(operators[j], operResult[0], card[2]);

        tempStr = tempStr + operators[j] + to_string(card[2].numerator);
        execValidate(operResult,tempStr);

        tempStr = "(" + to_string(card[0].numerator) + operators[i] + str + ")";
```

```
            operResult = backup;
        }

        // reset isi tempStr dan operResult
        tempStr = str;
        operResult = card;
    }
}
// Prosedur untuk mengoperasikan hasil operasi kartu 2 dan 3 dengan kartu 4
void operMiddleRight(vector<number> card, string str)
{
    string tempStr;
    vector<number> operResult = 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].numerator, card[2].numerator) == false)
            {
                break;
            }
        }

        operResult[1] = operation(operators[i], card[1], card[2]);

        operResult.erase(operResult.begin() + 2);
        tempStr = "(" + str + operators[i] + to_string(card[2].numerator) + ")";

        vector<number> backup = operResult;
```

```
        // Lanjutkan operasi dengan kartu 1
        for (int j = 0; j < 4; j++)
        {
            if (j == 3)
            {
                if (isDivisible(card[0].numerator, operResult[1].numerator) == false)
                {
                    break;
                }
            }
            operResult[0] = operation(operators[j], card[0], operResult[1]);

            tempStr = to_string(card[0].numerator) + operators[j] + tempStr;
            execValidate(operResult,tempStr);



            tempStr = "(" + str + operators[i] + to_string(card[2].numerator) + ")";
            operResult = backup;
        }

        // reset isi tempStr dan operResult
        tempStr = str;
        operResult = card;
    }
}

// Prosedur untuk mengoperasikan hasil operasi kartu 2 dan 3 dengan kartu 1 dan 4
void operMiddle(vector<number> card, string str)
{
    operMiddleLeft(card, str);
    operMiddleRight(card, str);
}
```

```
// Prosedur untuk mengoperasikan 2 kartu pertama dari 4 kartu
void operFirstTwo(vector<int> c)
{
   vector<number> card(4);
   for (int i = 0; i < 4; i++){
      card[i].numerator = c[i];
      card[i].denom = 1;
   }
   vector<number> operResult = card;
   string str;

   for (int i = 0; i < 3; i++){
      int j = i + 1;

      for (int k = 0; k < 4; k++){
          if (operators[k] == '/' && (isDivisible(card[i].numerator, card[j].numerator) == false)){
             break;
          }
          operResult[j] = operation(operators[k], card[i], card[j]);

          operResult.erase(operResult.begin() + i);
                      str = "(" + to_string(card[i].numerator) + operators[k] + to_string(card[j].numerator) + ")";

          if (i == 0){
             operLeftRight(operResult, str, 1);
          }
          else if (i == 2){
             operLeftRight(operResult, str, 2);
          }
          else{
```

```
            operMiddle(operResult, str);
        }

        // Kembalikan string dan operResult ke semula
        str = "";
        operResult = card;
      }
   }
}

#endif
```

3.5 main.cpp

```
#include <bits/stdc++.h>
#include "ascii.h"
#include "orders.h"

using namespace std;

// Fungsi untuk menampilkan main menu ke layar
void displayMain() {
   cout << "Silahkan pilih cara memberikan input : " << endl;
   cout << "-------------------------------------------------------" << endl;
   cout << "1. Input secara manual" << endl;
   cout << "2. Input secara acak" << endl;
   cout << "3. Keluar" << endl;
   cout << "-------------------------------------------------------" << endl;
   cout << "Input : " << endl;
}
// Fungsi untuk menampilkan input kartu acak ke layar
void displayRandomInput(string input[4]) {
   cout << "Random Input : " << endl;
```

```
    for (int i = 0; i < 4; i++) {
        cout << input[i] << " ";
    }
    cout << endl;
}
// Fungsi untuk menampilkan waktu eksekusi ke layar
void displayExecTime() {
    float execTime = clock();
    execTime = clock() - execTime;

    cout << endl;

    cout << "--------------------------" << endl;
    cout << "Execution time is ";
    cout << fixed << setprecision(2) << execTime;
    cout << " ms" << endl;

}
// Fungsi untuk menampilkan pilihan untuk menyimpan solusi atau tidak
void displaySavePrompt(string input[4]) {
    cout << endl;
    cout << "Apakah anda ingin menyimpan solusi ke file? (tekan Y/y jika ingin) : ";
    string inp;
    cin >> inp;
    if (inp == "y" || inp == "Y") {
        saveAnswer(input);
    }
}
// Fungsi untuk menampilkan solusi dari 24 game
void displaySolutions() {
    // Output solutions
    cout << endl << "Output : " << endl;
```

```
    if (counter == 0)
    {
        cout << "No Solutions found!" << endl;
    }
    else
    {
        cout << endl << "There are " << counter << " solutions!" << endl << endl;
        for (int i = 0; i < solutions.size(); i++)
        {
            cout << solutions[i] << endl;
        }
    }
}

// Fungsi untuk menjalankan program dengan input dari user
void userInput(string message)
{
    int output[100];
    string input[100];
    vector<int> card(4);

    while (true)
    {
        ASCII();
        string input_4;
        int j = 0;

        cout << message;

        cout << "Masukkan input 4 kartu yang dipisahkan dengan spasi : " << endl;
        cout << "Petunjuk --> (A = 1, J = 11, Q = 12, K = 13)" << endl;
        cout << "--------------------------------------------------" << endl;
```

```
    cout << "Input : " << endl;

    getline(cin, input_4);

    for (int i = 0; i < input_4.length(); i++){
        if (input_4[i] == ' '){
            j++;
        }
        else{
            input[j] += input_4[i];
        }
    }

    if (j != 3) {
        message = "Input Anda wajib memiliki 4 karakter! Silahkan coba lagi\n";
        continue;
    }

    changeInteger(input, output);

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

for (int i = 0; i < 4; i++){
    card[i] = output[i];
}
```

```
    while (next_permutation(card.begin(), card.end()))
    {
        operFirstTwo(card);
    }

    // tampilan solusi 24 game
    displaySolutions();
    // tampilan waktu eksekusi
    displayExecTime();
    // tampilan pilihan saving
    displaySavePrompt(input);
}
// Fungsi untuk menjalankan program dengan input secara acak
void randomInput() {
    int output[4];
    string input[4];
    vector<int> card(4);

    // Memberi input kartu secara acak
    randomizedCards(input);

    // Menampilkan input yang diacak ke layar dan mengubah card ke integer
    ASCII();
    displayRandomInput(input);
    changeInteger(input, output);

    for (int i = 0; i < 4; i++){
        card[i] = output[i];
    }

    while (next_permutation(card.begin(), card.end()))
    {
        operFirstTwo(card);
```

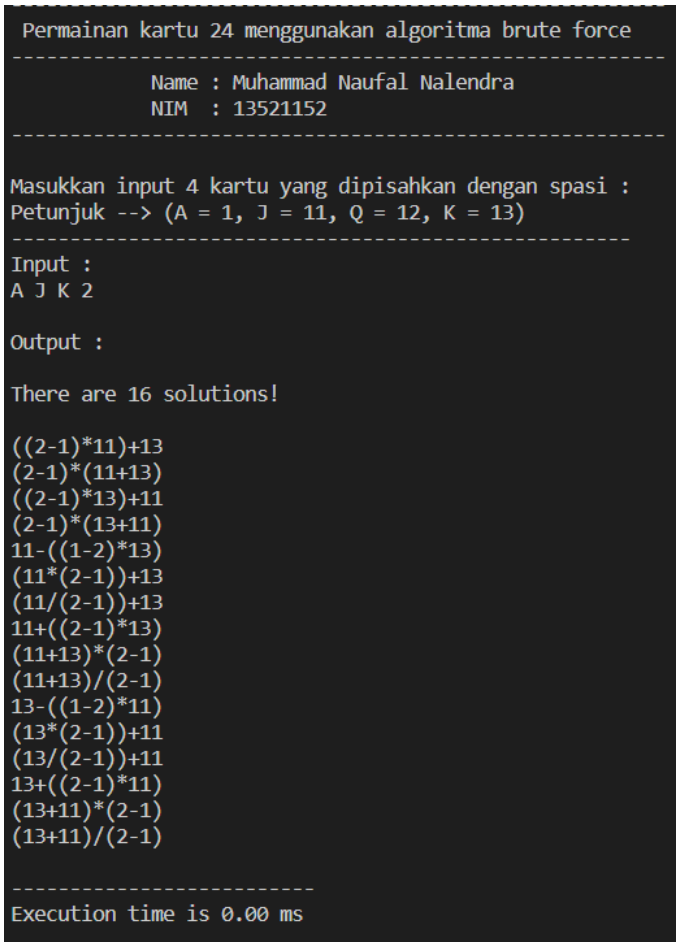
```
    }

    // tampilan solusi 24 game
    displaySolutions();
    // tampilan waktu eksekusi
    displayExecTime();
    // tampilan pilihan saving
    displaySavePrompt(input);

}
// Fungsi untuk menampilkan main menu ke layar
void displayMenu(string message) {
    while (true)
    {
        ASCII();
        string input;

        cout << message;

        displayMain();

        getline(cin, input);

        if (input == "1"){
            userInput("");
            break;
        }
        else if (input == "2"){
            randomInput();
            break;
        }
        else if (input == "3"){
            exit(0);
```

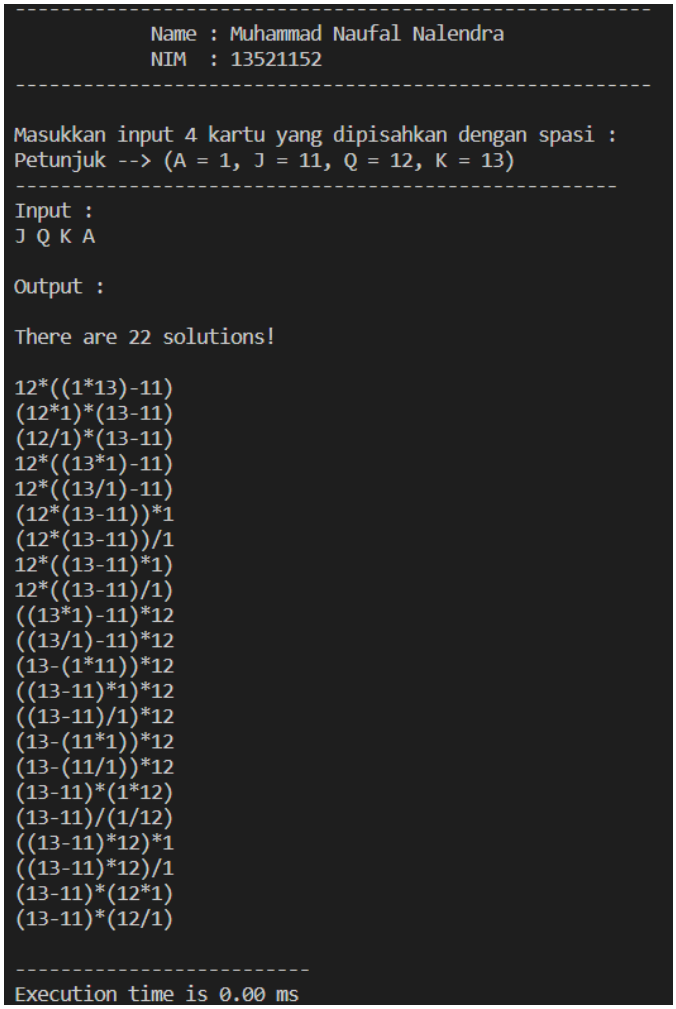
```
        }
        else{
            message = "Input Anda tidak valid, silahkan coba lagi!.\n";
            continue;
        }
    }
}

int main() {
    displayMenu("");
}
```

4. Test Case Manual dan Random

4.1 Test Case Manual

4.1.1 Test Input Benar





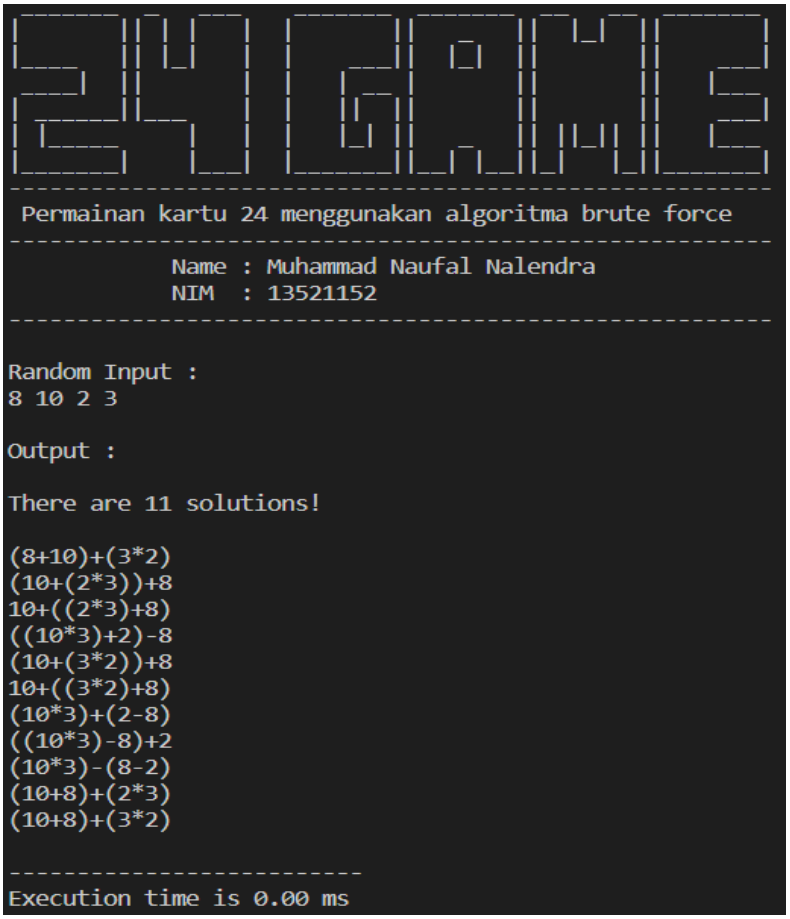
4.1.2 Test Input Tidak 4 Character

4.1.3 Test Input Tidak Valid

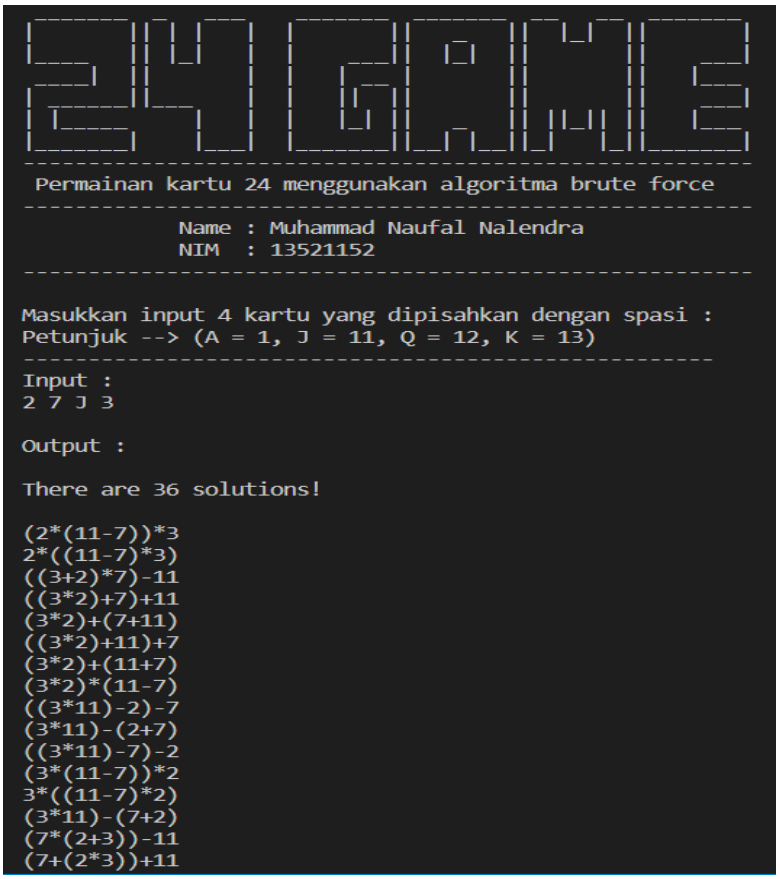
4.2 Test Case Random

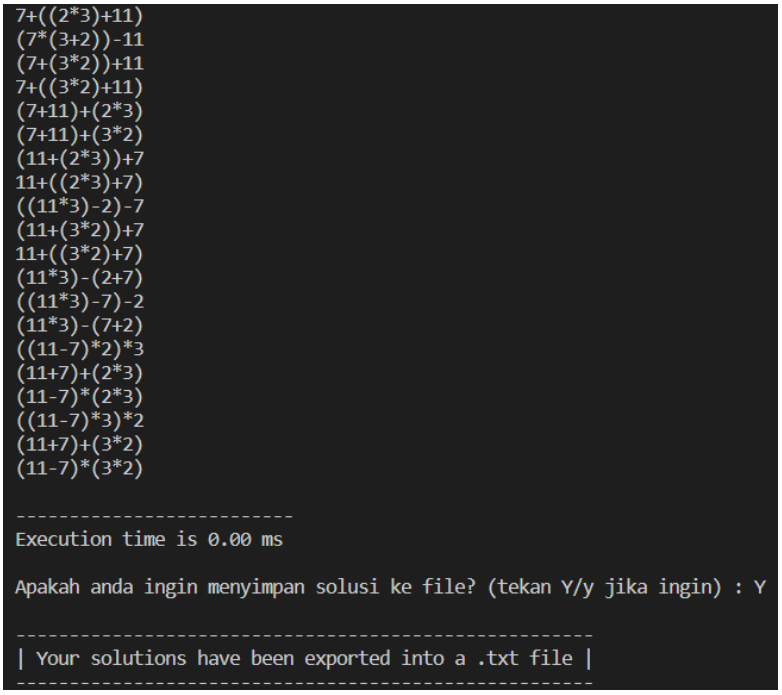
```
 Permainan kartu 24 menggunakan algoritma brute force
------------------------------------------------------------
           Name : Muhammad Naufal Nalendra
           NIM  : 13521152
------------------------------------------------------------

Silahkan pilih cara memberikan input :
------------------------------------------------------------
1. Input secara manual
2. Input secara acak
3. Keluar
------------------------------------------------------------
Input :
2
 _______  _   ___   _______ _______ __   __ _______
|       || | |   | |       |       ||  |_|  ||       |
|____   || |_|   | |    ___|  _____||       ||    ___|
 ____|  ||       | |   |___ | |_____ |       ||   |___
| ______||___    | |    ___||_____  ||       ||    ___|
| |_____     |   | |   |___  _____| || ||_|| ||   |___
|_______|    |___| |_______||_______||_|   |_||_______|
------------------------------------------------------------
 Permainan kartu 24 menggunakan algoritma brute force
------------------------------------------------------------
           Name : Muhammad Naufal Nalendra
           NIM  : 13521152
------------------------------------------------------------

Random Input :
6 8 K 7

Output :
No Solutions found!

--------------------------
Execution time is 0.00 ms
```

```
 _______  _   ___   _______ _______ __   __ _______
|       || | |   | |       |       ||  |_|  ||       |
|____   || |_|   | |    ___|  _____||       ||    ___|
 ____|  ||       | |   |___ | |_____ |       ||   |___
| ______||___    | |    ___||_____  ||       ||    ___|
| |_____     |   | |   |___  _____| || ||_|| ||   |___
|_______|    |___| |_______||_______||_|   |_||_______|
------------------------------------------------------------
 Permainan kartu 24 menggunakan algoritma brute force
------------------------------------------------------------
           Name : Muhammad Naufal Nalendra
           NIM  : 13521152
------------------------------------------------------------

Random Input :
K 4 8 6

Output :

There are 2 solutions!

((13-6)-4)*8
(13-(6+4))*8

--------------------------
Execution time is 0.00 ms
```


Permainan kartu 24 menggunakan algoritma brute force
Name : Muhammad Naufal Nalendra
NIM : 13521152
Random Input :
8 10 2 3
Output :
There are 11 solutions!
(8+10)+(3*2)
(10+(2*3))+8
10+((2*3)+8)
((10*3)+2)-8
(10+(3*2))+8
10+((3*2)+8)
(10*3)+(2-8)
((10*3)-8)+2
(10*3)-(8-2)
(10+8)+(2*3)
(10+8)+(3*2)
Execution time is 0.00 ms

4.3 Test Save


Permainan kartu 24 menggunakan algoritma brute force
Name : Muhammad Naufal Nalendra
NIM : 13521152
Masukkan input 4 kartu yang dipisahkan dengan spasi :
Petunjuk --> (A = 1, J = 11, Q = 12, K = 13)
Input :
2 7 J 3
Output :
There are 36 solutions!
(2*(11-7))*3
2*((11-7)*3)
((3+2)*7)-11
((3*2)+7)+11
(3*2)+(7+11)
((3*2)+11)+7
(3*2)+(11+7)
(3*2)*(11-7)
((3*11)-2)-7
(3*11)-(2+7)
((3*11)-7)-2
(3*(11-7))*2
3*((11-7)*2)
(3*11)-(7+2)
(7*(2+3))-11
(7+(2*3))+11



Link Repo : https://github.com/Naufal-Nalendra/Tucil1_13521152

5. Tabel Penilaian

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	✓	