Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Angela Livia Arumsari 13521094

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

BAB I
DESKRIPSI PERMASALAHAN

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.

Tugas kecil ini bertujuan untuk merancang algoritma *brute force* untuk menemukan solusi dari permainan kartu 24 tersebut. Pada tugas kali ini, akan digunakan bahasa C++.

BAB II
ALGORITMA BRUTE FORCE

Algoritma brute force adalah pendekatan yang *straightforward* untuk memecahkan suatu persoalan. Algoritma ini didasarkan pada problem statement dan konsep yang dilibatkan. Algoritma ini akan memecahkan persoalan dengan langsung, jelas, dan sangat sederhana. Pada problem permainan kartu 24, konsep yang digunakan yaitu algoritma ini akan mencari seluruh kemungkinan ekspresi dari empat angka kartu yang ada, empat operasi, dan penempatan tanda kurung. Jika, salah satu ekspresi itu menghasilkan 24, maka dianggap sebagai solusi.

Pada tugas kali ini, algoritma *brute force* untuk menyelesaikan permainan kartu 24 memiliki langkah-langkah sebagai berikut:

1. Mencari seluruh permutasi yang mungkin dari nilai empat buat kartu. Kombinasi empat kartu ini akan disimpan dalam sebuah vektor. Sedangkan, seluruh kombinasi disimpan dalam struktur data set. Penggunaan struktur data set dimaksudkan menghindari duplikat nilai.
2. Mencari seluruh kombinasi tiga operator yang mungkin dari empat operator yang boleh digunakan. Dalam pencarian kombinasi operator, satu operator dapat digunakan berulang kali. Seluruh kombinasi ini disimpan dalam struktur data vektor.
3. Untuk setiap kombinasi empat kartu dan tiga operator, lakukan pengecekan untuk lima buah ekspresi yang berbeda peletakkan tanda kurung. Langkah ini melakukan pengecekkan pada semua kombinasi yang mungkin.
4. Jika didapatkan ekspresi yang menghasilkan 24, maka ekspresi tersebut merupakan salah satu solusi.

BAB III
SOURCE CODE

io.h (Header file io.cpp)

```
#ifndef IO_H
#define IO_H

#include <iostream>
#include <random>
#include <fstream>

using namespace std;

/* Procedure to generate 4 random cards and insert to cards array
*/
void randomCards(int cards[]);

/* Procedure to get cards input from user and input into cards
array */
void readCards(int cards[]);

/* Procedure to print main menu */
void mainMenu();

/* Procedure to print splash screen */
void splashScreen();

/* Procedure to write results into file */
void writeFile(vector<string> result, int cards[], int execTime);

#endif
```

io.cpp (File untuk menangani input output)

```
#include <iostream>
#include <random>
#include <fstream>
#include <vector>

using namespace std;

/* Procedure to print cards */
void printCards(int cards[])
{
    string cardsText[4];

    for (int i = 0; i < 4; i++)
    {
        switch (cards[i])
        {
        case 1:
            cardsText[i] = "A";
            break;
        case 11:
            cardsText[i] = "J";
            break;
        case 12:
            cardsText[i] = "Q";
            break;
        case 13:
            cardsText[i] = "K";
            break;
        default:
            cardsText[i] = to_string(cards[i]);
            break;
        }
    }
    cout << R"(
        ===============  CARDS  ================
        ========================================
        ||                                    ||
                    )" << cardsText[0] << "  " << cardsText[1]
<< "  " << cardsText[2] << "  " << cardsText[3] << R"(
        ||                                    ||
        ========================================
        ========================================
    )";
}

/* Procedure to generate 4 random cards and insert to cards array
*/
void randomCards(int cards[])
{
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> distr(1, 13);

    cout << "Your four cards are as follows. \n";
    for (int i = 0; i < 4; i++)
    {
        cards[i] = distr(gen);
    }
    printCards(cards);
}

/* Procedure to get cards input from user and input into cards
array */
void readCards(int cards[])
{
    string input;
    bool isValid, isPrevOne, isNeedSpace;
    isValid = false;

    do
    {
        cout << "Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J,
Q, K) \n";
        getline(cin, input);
        isPrevOne = isNeedSpace = false;
        int i = 0;
        int ctr = 0;
        while (i <= input.length())
        {
            if (i == input.length())
            {
                if (ctr != 4)
                {
                    cout << "Input is invalid. Please input again!
\n";
                    break;
                }
                else
                {
                    isValid = true;
                }
            }
            else if (input[i] == '1' && !isNeedSpace)
            {
                isPrevOne = true;
            }
            else if (input[i] > '1' && input[i] <= '9' &&
!isNeedSpace && !isPrevOne)
            {
                isPrevOne = false;
                isNeedSpace = true;
                cards[ctr % 4] = input[i] - '0';
                ctr++;
            }
            else if ((input[i] == 'Q' || input[i] == 'q' ||
input[i] == 'J' || input[i] == 'j' || input[i] == 'K' || input[i]
== 'k' || input[i] == 'A' || input[i] == 'a') && !isNeedSpace &&
!isPrevOne)
            {
                isPrevOne = false;
                isNeedSpace = true;
                switch (input[i])
                {
                case 'A':
                case 'a':
                    cards[ctr % 4] = 1;
                    break;
                case 'J':
                case 'j':
                    cards[ctr % 4] = 11;
                    break;
                case 'Q':
                case 'q':
                    cards[ctr % 4] = 12;
                    break;
                case 'K':
                case 'k':
                    cards[ctr % 4] = 13;
                    break;
                default:
                    break;
                }
                ctr++;
            }
            else if (input[i] == '0' && isPrevOne && !isNeedSpace)
            {
                isPrevOne = false;
                isNeedSpace = true;
                cards[ctr % 4] = 10;
                ctr++;
            }
            else if (input[i] == ' ' && !isPrevOne)
            {
                isNeedSpace = false;
            }
            else
            {
                cout << "Input is invalid. Please input again!
\n";
                break;
            }
            i++;
        }
    } while (!isValid);
    printCards(cards);
}

/* Procedure to print main menu */
void mainMenu()
{
    cout << R"(
            ================  MENU  ================
            ========================================
            ||        1. Input your cards         ||
            ||       2. Randomly pick cards       ||
            ||              3. Exit               ||
            ========================================
            ========================================
    )" << endl;
}

/* Procedure to print splash screen */
void splashScreen()
{
    cout << "      _______  _   ___    _______  _______  __   __
_______\n     |       || | |   |  |       ||   _   ||  |_|  ||
|\n     |____   || |_|   |  |    ___||  |_|  ||       ||    ___|\n
____|  ||       |  |   | __ |       ||       ||   |___\n     |
______||___    |  |   ||  ||       ||       ||    ___|\n     |
|_____     |   |  |   |_| ||   _   || ||_|| ||   |___\n
|_______|    |___|  |_______||__| |__||_|   |_||_______|";
    cout << R"(

                 ____    ____    ____    ____
                |2   |  |A   |  |Q   |  |4   |
                |(\/)|  | /\ |  | /\ |  | &  |
                | \/ |  | \/ |  |(__)|  |&|& |
                |   2|  |   A|  | /\Q|  | | 4|
                `----`  `----'  `----'  `----')"
         << "\n\n";
}

/* Procedure to write results into file */
void writeFile(vector<string> result, int cards[], int execTime)
{
    string input, fileName;
    bool isValid = false;

    cout << "Enter file name: ";
    cin >> input;
    fileName = "../test/" + input + ".txt";

    ofstream resultFile(fileName);

    resultFile << "Cards:";
    for (int i = 0; i < 4; i++)
    {
        resultFile << " " << cards[i];
    }
    resultFile << endl
               << result.size() << " solutions found\n";

    for (int i = 0; i < result.size(); i++)
    {
        resultFile << result[i] << endl;
    }
    resultFile << "The execution time is " << execTime << "
microseconds";
    resultFile.close();
}
```

game_solver.h (Header file game_solver.c)

```
#ifndef GAME_SOLVER_H
#define GAME_SOLVER_H

#include <iostream>
#include <vector>
#include <set>

using namespace std;

/* Procedure to permute 4 number of cards */
void permuteCards(int num[], int l, int r, set <vector<int>>&
cards);

/* Procedure to make all the combination of the operator */
void permuteOps(vector <vector<char>>& res);

/* Procedure to transform set to vector */
void setToVector (set <vector<int>> cards, vector <vector<int>>&
res);

/* Function to compute based on the operator */
```

```
double evaluate(double a, double b, char ops);

/* Procedure to get all correct expression that results in 24 */
void getResults(vector<string>& res, vector <vector<int>> cards,
vector <vector<char>> ops);

/* Procedure to solve 24 game with an array of cards*/
void gameSolver(int cards[], vector <string>& result);

#endif
```

game_solver.c (File berisi algoritma brute force untuk mencari solusi)

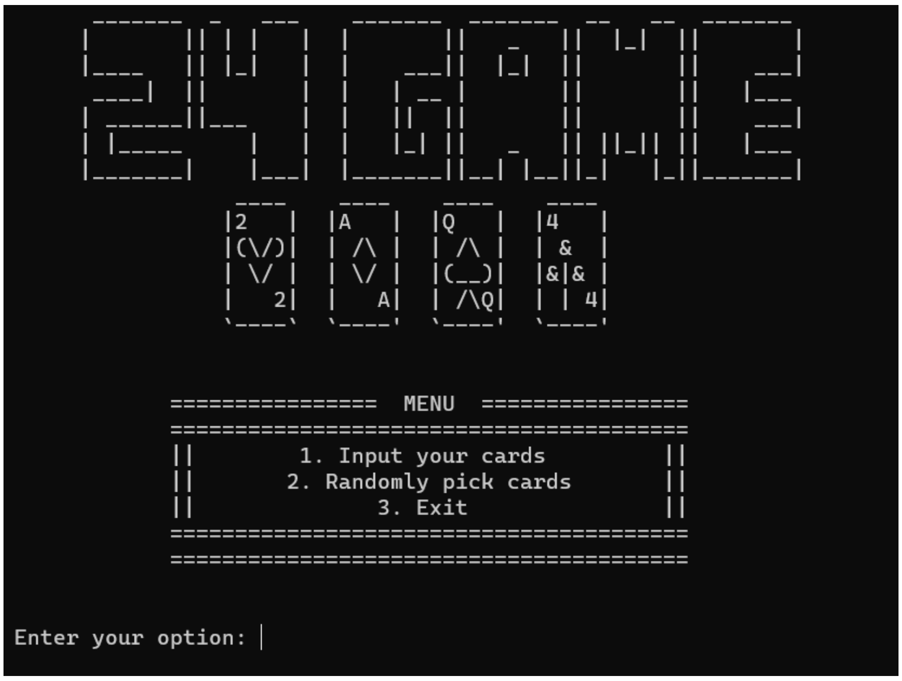
```
#include <iostream>
#include <vector>
#include <set>

using namespace std;

/* Implementation of swap function */
void swapValue(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

/* Function to permute 4 number of cards */
void permuteCards(int num[], int l, int r, set <vector<int>>&
cards)
{
    if (l == r) {
        vector <int> card;
        for (int i=0; i<4;i++) {
            card.push_back(num[i]);
        }
        cards.insert(card);
    }
    else {
        for (int i = l; i <= r; i++) {
            swapValue(&num[l], &num[i]);
            permuteCards(num, l + 1, r, cards);
            // backtrack
            swapValue(&num[l], &num[i]);
        }
    }
}

/* Function to make all the combination of the operator */
void permuteOps(vector <vector<char>>& res) {
    char ops[4] = {'+', '-', '*', '/'};

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            for (int k = 0; k < 4; k++) {
                vector <char> comb;
                comb.push_back(ops[i]);
                comb.push_back(ops[j]);
                comb.push_back(ops[k]);
                res.push_back(comb);
        }
        }
    }

}

/* Function to transform set to vector */
void setToVector (set <vector<int>> cards, vector <vector<int>>&
res) {
    for (auto i = cards.begin(); i != cards.end(); i++) {
        vector <int> card;
        for (int j = 0; j < 4; j++) {
            card.push_back((*i)[j]);
        }
        res.push_back(card);
    }
}

/* Function to compute based on the operator */
double evaluate(double a, double b, char ops) {
    switch (ops)
    {
        case '+':
            return a + b;
        case '-':
            return a - b;
        case '*':
            return a * b;
        default:
            return a / b;
    }
}

/* Function to get all correct expression that results in 24 */
void getResults(vector<string>& res, vector <vector<int>> cards,
vector <vector<char>> ops) {
    for (int i = 0; i < cards.size(); i++) {
        for (int j = 0; j < ops.size(); j++) {
            double card1 = (double) cards[i][0];
            double card2 = (double) cards[i][1];
            double card3 = (double) cards[i][2];
            double card4 = (double) cards[i][3];

            double num12 = evaluate(card1, card2, ops[j][0]);
            double num23 = evaluate(card2, card3, ops[j][1]);
            double num34 = evaluate(card3, card4, ops[j][2]);

            string op1 = " " + string(1, ops[j][0]) + " ";
            string op2 = " " + string(1, ops[j][1]) + " ";
            string op3 = " " + string(1, ops[j][2]) + " ";

            // test every possible precedence
            double prec1 = evaluate(evaluate(num12, card3,
ops[j][1]), card4, ops[j][2]);
            if (prec1 == 24) {
                string sol1 = "(((" + to_string(cards[i][0]) + op1
+ to_string(cards[i][1]) + ")" + op2 + to_string(cards[i][2]) +
")" + op3 + to_string(cards[i][3]) + ")";
                res.push_back(sol1);
            }
            double prec2 = evaluate(num12, num34, ops[j][1]);
            if (prec2 == 24) {
                string sol2 = "((" + to_string(cards[i][0]) + op1
+ to_string(cards[i][1]) + ")" + op2 + "(" +
to_string(cards[i][2]) + op3 + to_string(cards[i][3]) + "))";
                res.push_back(sol2);
            }
            double prec3 = evaluate(evaluate(card1, num23,
ops[j][0]), card4, ops[j][2]);
            if (prec3 == 24) {
                string sol3 = "((" + to_string(cards[i][0]) + op1
+ "(" + to_string(cards[i][1]) + op2 + to_string(cards[i][2]) +
"))" + op3 + to_string(cards[i][3]) + ")";
                res.push_back(sol3);
            }
            double prec4 = evaluate(card1, evaluate(num23, card4,
ops[j][2]), ops[j][0]);
            if (prec4 == 24) {
                string sol4 = "(" + to_string(cards[i][0]) + op1 +
"((" + to_string(cards[i][1]) + op2 + to_string(cards[i][2]) + ")"
+ op3 + to_string(cards[i][3]) + "))";
                res.push_back(sol4);
            }
            double prec5 = evaluate(card1, evaluate(card2, num34,
ops[j][1]), ops[j][0]);
            if (prec5 == 24) {
                string sol5 = "(" + to_string(cards[i][0]) + op1 +
"(" + to_string(cards[i][1]) + op2 + "(" + to_string(cards[i][2])
+ op3 + to_string(cards[i][3]) + ")))";
                res.push_back(sol5);
            }
        }
    }
}

void gameSolver(int cards[], vector <string>& result) {
    set <vector<int>> cardComb;
    vector <vector<char>> opComb;
    vector <vector<int>> cardCombVec;

    permuteCards(cards, 0, 3, cardComb);
    setToVector(cardComb, cardCombVec);
    permuteOps(opComb);
    getResults(result, cardCombVec, opComb);
}
```

main.c (Program utama)

```
#include <iostream>
#include "io.h"
#include "game_solver.h"
#include <chrono>
#include <limits>

using namespace std::chrono;
using namespace std;

int main()
{
    int opt;
    splashScreen();
    mainMenu();
    cout << "\nEnter your option: ";
    cin >> opt;
    cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    while (opt != 3)
    {
        char save;
        int cards[4];
        vector<string> result;
        if (opt == 1)
        {
            readCards(cards);
        }
        else if (opt == 2)
        {
            randomCards(cards);
        }
        else {
            cout << "The option is not available.\n";
        }

        if (opt == 1 || opt == 2) {
            auto start = high_resolution_clock::now();
            gameSolver(cards, result);
            auto stop = high_resolution_clock::now();
            auto execTime = duration_cast<microseconds>(stop -
start);

            if (result.size() > 0)
            {
                cout << endl
                    << result.size() << " solutions found\n";
                for (int i = 0; i < result.size(); i++)
                {
                    cout << i + 1 << ". " << result[i] << endl;
                }
            }
            else
            {
                cout << "There is no solution\n";
            }
            cout << "The execution time is " << execTime.count()
<< " microseconds.\n";
            cout << "Do you want to save the solution to a file?
(Y/N)\n";
            cin >> save;
            bool isSaveValid = false;
            while (!isSaveValid) {
                if (save == 'y' || save == 'Y')
                {
                    writeFile(result, cards, execTime.count());
                    cout << "Solution is succesfully saved!\n";
                }
                if (save == 'y' || save == 'Y' || save == 'n' ||
save == 'N') {
                    isSaveValid = true;
                }
                else {
                    cout << "Your input is invalid.\n";
                    cout << "Do you want to save the solution to a
file? (Y/N)\n";
                    cin >> save;
                }
            }

        }

        mainMenu();
        cout << "\nEnter your option: ";
        cin >> opt;
        cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
    }
    cout << "Thank you for using this program!\n";
}
```

BAB IV
UJI COBA PROGRAM

Program untuk menyelesaikan permainan kartu 24 ini dikompilasi dengan menjalankan run.bat pada direktori src, jika digunakan sistem operasi Windows. Setelah dikompilasi kemudian dijalankan, program akan mengeluarkan tampilan awal sebagai berikut.

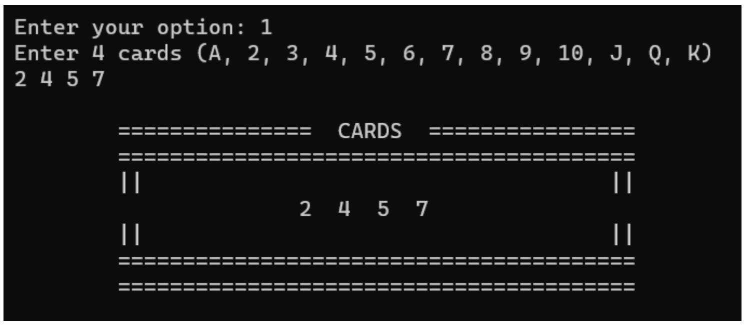


```
                 ____    ____    ____    ____
                |2   |  |A   |  |Q   |  |4   |
                |(\/)|  | /\ |  | /\ |  | &  |
                | \/ |  | \/ |  |(__)|  |&|& |
                |   2|  |   A|  | /\Q|  | | 4|
                `----`  `----'  `----'  `----'


            ===============  MENU  ===============
            ======================================
            ||        1. Input your cards       ||
            ||       2. Randomly pick cards     ||
            ||              3. Exit             ||
            ======================================
            ======================================


Enter your option:
```

Gambar 5 Tampilan Awal Program

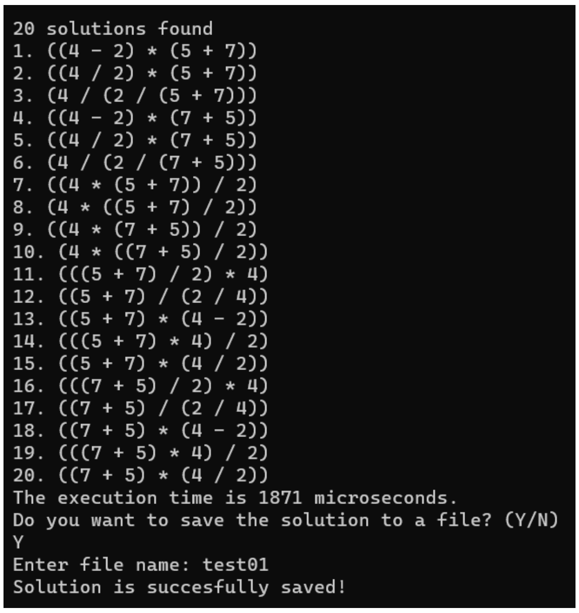
Berikutnya pengguna dapat memilih sesuai pilihan yang tersedia pada menu. Berikut beberapa data uji sebagai contoh input output program.

1. Data Uji 1



```
Enter your option: 1
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
2 4 5 7

        ===============  CARDS  ===============
        =======================================
        ||                                   ||
                      2  4  5  7
        ||                                   ||
        =======================================
        =======================================
```

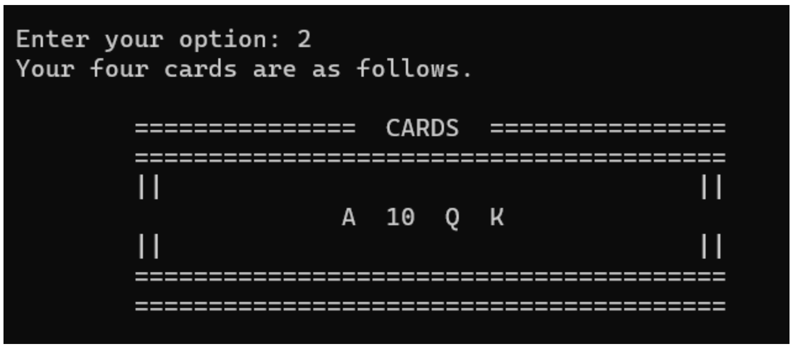
Gambar 5.1 Input Data Uji 1



```
20 solutions found
1. ((4 - 2) * (5 + 7))
2. ((4 / 2) * (5 + 7))
3. (4 / (2 / (5 + 7)))
4. ((4 - 2) * (7 + 5))
5. ((4 / 2) * (7 + 5))
6. (4 / (2 / (7 + 5)))
7. ((4 * (5 + 7)) / 2)
8. (4 * ((5 + 7) / 2))
9. ((4 * (7 + 5)) / 2)
10. (4 * ((7 + 5) / 2))
11. (((5 + 7) / 2) * 4)
12. ((5 + 7) / (2 / 4))
13. ((5 + 7) * (4 - 2))
14. (((5 + 7) * 4) / 2)
15. ((5 + 7) * (4 / 2))
16. (((7 + 5) / 2) * 4)
17. ((7 + 5) / (2 / 4))
18. ((7 + 5) * (4 - 2))
19. (((7 + 5) * 4) / 2)
20. ((7 + 5) * (4 / 2))
The execution time is 1871 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test01
Solution is succesfully saved!
```

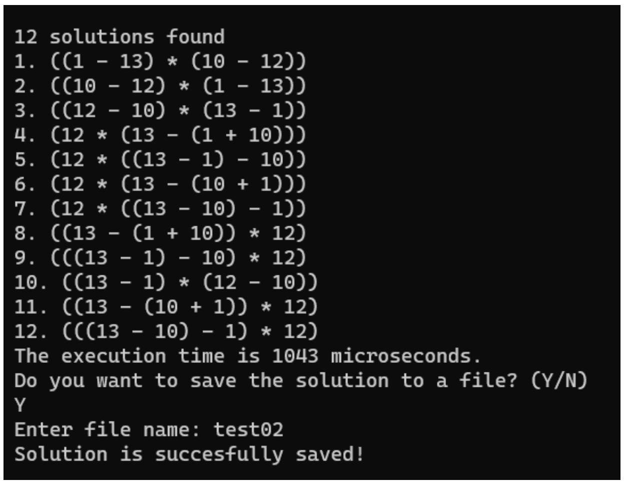
Gambar 5.1 Output Data Uji 1

2. Data Uji 2



```
Enter your option: 2
Your four cards are as follows.

          ===============  CARDS  ================
          ========================================
          ||                                    ||
                       A  10  Q  K
          ||                                    ||
          ========================================
          ========================================
```

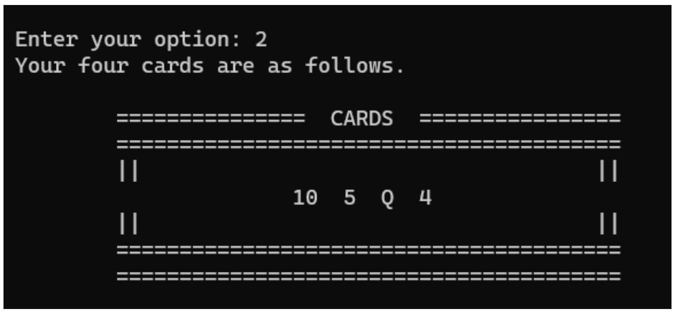
Gambar 5.2 Input Data Uji 2



```
12 solutions found
1. ((1 - 13) * (10 - 12))
2. ((10 - 12) * (1 - 13))
3. ((12 - 10) * (13 - 1))
4. (12 * (13 - (1 + 10)))
5. (12 * ((13 - 1) - 10))
6. (12 * (13 - (10 + 1)))
7. (12 * ((13 - 10) - 1))
8. ((13 - (1 + 10)) * 12)
9. (((13 - 1) - 10) * 12)
10. ((13 - 1) * (12 - 10))
11. ((13 - (10 + 1)) * 12)
12. (((13 - 10) - 1) * 12)
The execution time is 1043 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test02
Solution is succesfully saved!
```

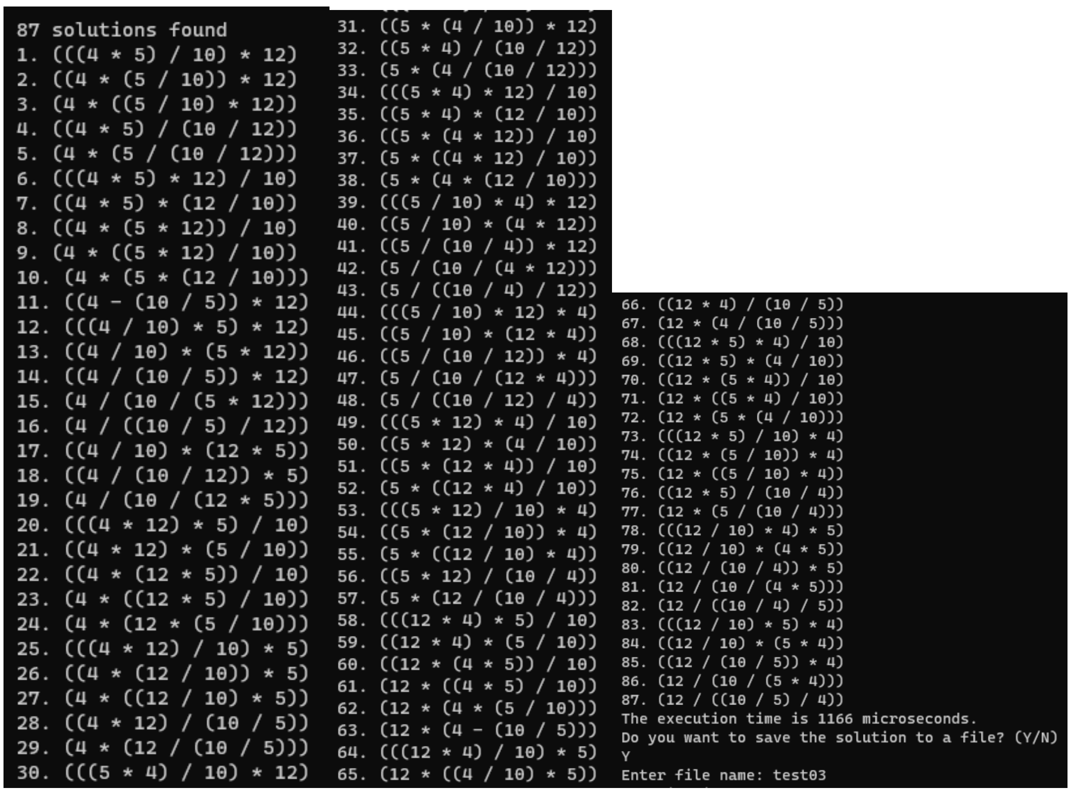
Gambar 5.2 Output Data Uji 2

3. Data Uji 3



```
Enter your option: 2
Your four cards are as follows.

        ===============  CARDS  ================
        ========================================
        ||                                    ||
                      10  5  Q  4
        ||                                    ||
        ========================================
        ========================================
```

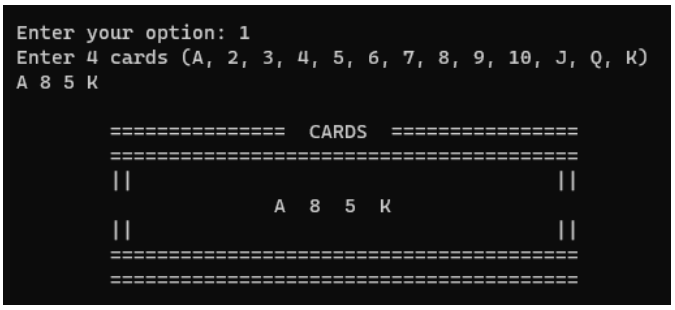
Gambar 5.3 Input Data Uji 3



```
87 solutions found
1. (((4 * 5) / 10) * 12)
2. ((4 * (5 / 10)) * 12)
3. (4 * ((5 / 10) * 12))
4. ((4 * 5) / (10 / 12))
5. (4 * (5 / (10 / 12)))
6. (((4 * 5) * 12) / 10)
7. ((4 * 5) * (12 / 10))
8. ((4 * (5 * 12)) / 10)
9. (4 * ((5 * 12) / 10))
10. (4 * (5 * (12 / 10)))
11. ((4 - (10 / 5)) * 12)
12. (((4 / 10) * 5) * 12)
13. ((4 / 10) * (5 * 12))
14. ((4 / (10 / 5)) * 12)
15. (4 / (10 / (5 * 12)))
16. (4 / ((10 / 5) / 12))
17. ((4 / 10) * (12 * 5))
18. ((4 / (10 / 12)) * 5)
19. (4 / (10 / (12 * 5)))
20. (((4 * 12) * 5) / 10)
21. ((4 * 12) * (5 / 10))
22. ((4 * (12 * 5)) / 10)
23. (4 * ((12 * 5) / 10))
24. (4 * (12 * (5 / 10)))
25. (((4 * 12) / 10) * 5)
26. ((4 * (12 / 10)) * 5)
27. (4 * ((12 / 10) * 5))
28. ((4 * 12) / (10 / 5))
29. (4 * (12 / (10 / 5)))
30. (((5 * 4) / 10) * 12)
31. ((5 * (4 / 10)) * 12)
32. ((5 * 4) / (10 / 12))
33. (5 * (4 / (10 / 12)))
34. (((5 * 4) * 12) / 10)
35. ((5 * 4) * (12 / 10))
36. ((5 * (4 * 12)) / 10)
37. (5 * ((4 * 12) / 10))
38. (5 * (4 * (12 / 10)))
39. (((5 / 10) * 4) * 12)
40. ((5 / 10) * (4 * 12))
41. ((5 / (10 / 4)) * 12)
42. (5 / (10 / (4 * 12)))
43. (5 / ((10 / 4) / 12))
44. (((5 / 10) * 12) * 4)
45. ((5 / 10) * (12 * 4))
46. ((5 / (10 / 12)) * 4)
47. (5 / (10 / (12 * 4)))
48. (5 / ((10 / 12) / 4))
49. (((5 * 12) * 4) / 10)
50. ((5 * 12) * (4 / 10))
51. ((5 * (12 * 4)) / 10)
52. (5 * ((12 * 4) / 10))
53. (((5 * 12) / 10) * 4)
54. ((5 * (12 / 10)) * 4)
55. (5 * ((12 / 10) * 4))
56. ((5 * 12) / (10 / 4))
57. (5 * (12 / (10 / 4)))
58. (((12 * 4) * 5) / 10)
59. ((12 * 4) * (5 / 10))
60. ((12 * (4 * 5)) / 10)
61. (12 * ((4 * 5) / 10))
62. (12 * (4 * (5 / 10)))
63. (12 * (4 - (10 / 5)))
64. (((12 * 4) / 10) * 5)
65. (12 * ((4 / 10) * 5))
66. ((12 * 4) / (10 / 5))
67. (12 * (4 / (10 / 5)))
68. (((12 * 5) * 4) / 10)
69. ((12 * 5) * (4 / 10))
70. ((12 * (5 * 4)) / 10)
71. (12 * ((5 * 4) / 10))
72. (12 * (5 * (4 / 10)))
73. (((12 * 5) / 10) * 4)
74. ((12 * (5 / 10)) * 4)
75. (12 * ((5 / 10) * 4))
76. ((12 * 5) / (10 / 4))
77. (12 * (5 / (10 / 4)))
78. (((12 / 10) * 4) * 5)
79. ((12 / 10) * (4 * 5))
80. ((12 / (10 / 4)) * 5)
81. (12 / (10 / (4 * 5)))
82. (12 / ((10 / 4) / 5))
83. (((12 / 10) * 5) * 4)
84. ((12 / 10) * (5 * 4))
85. ((12 / (10 / 5)) * 4)
86. (12 / (10 / (5 * 4)))
87. (12 / ((10 / 5) / 4))
The execution time is 1166 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test03
```

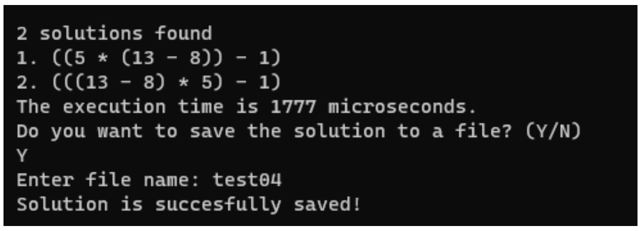
Gambar 5.3 Output Data Uji 3

4. Data Uji 4



```
Enter your option: 1
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
A 8 5 K

        ===============  CARDS  ================
        ========================================
        ||                                    ||
                      A  8  5  K
        ||                                    ||
        ========================================
        ========================================
```

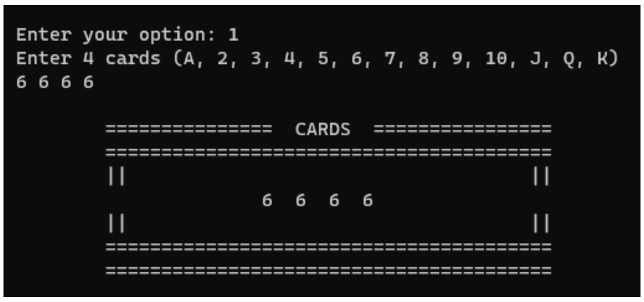
Gambar 5.4 Input Data Uji 4



```
2 solutions found
1. ((5 * (13 - 8)) - 1)
2. (((13 - 8) * 5) - 1)
The execution time is 1777 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test04
Solution is succesfully saved!
```

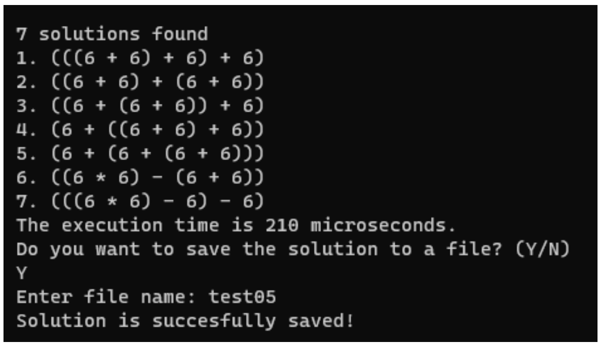
Gambar 5.4 Output Data Uji 4

5. Data Uji 5



```
Enter your option: 1
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
6 6 6 6

        ===============  CARDS  ================
        ========================================
        ||                                    ||
                      6  6  6  6
        ||                                    ||
        ========================================
        ========================================
```

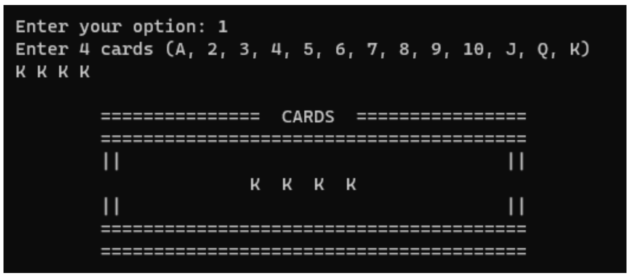
Gambar 5.5 Input Data Uji 5



```
7 solutions found
1. (((6 + 6) + 6) + 6)
2. ((6 + 6) + (6 + 6))
3. ((6 + (6 + 6)) + 6)
4. (6 + ((6 + 6) + 6))
5. (6 + (6 + (6 + 6)))
6. ((6 * 6) - (6 + 6))
7. (((6 * 6) - 6) - 6)
The execution time is 210 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test05
Solution is succesfully saved!
```

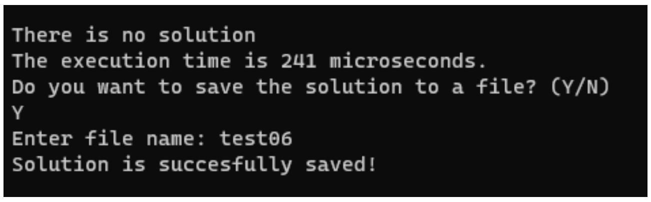
Gambar 5.5 Output Data Uji 5

6. Data Uji 6



```
Enter your option: 1
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
K K K K

        ===============  CARDS  ================
        ========================================
        ||                                    ||
                      K  K  K  K
        ||                                    ||
        ========================================
        ========================================
```

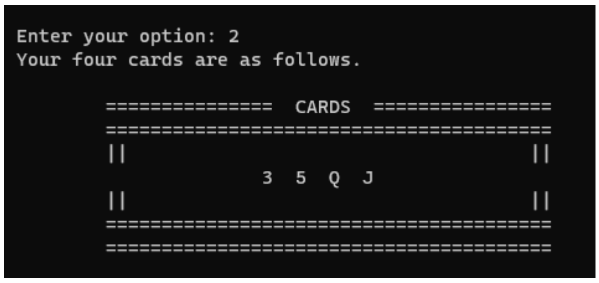
Gambar 5.6 Input Data Uji 6



```
There is no solution
The execution time is 241 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test06
Solution is succesfully saved!
```

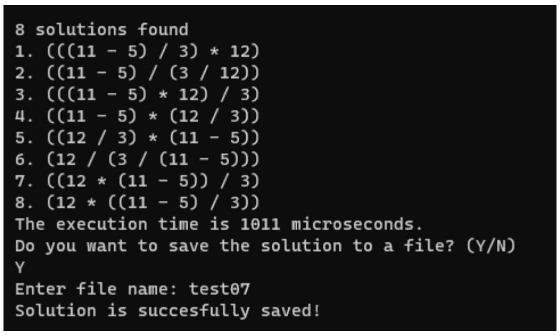
Gambar 5.6 Output Data Uji 6

7. Data Uji 7



```
Enter your option: 2
Your four cards are as follows.

        ===============  CARDS  ================
        ========================================
        ||                                    ||
                      3  5  Q  J
        ||                                    ||
        ========================================
        ========================================
```

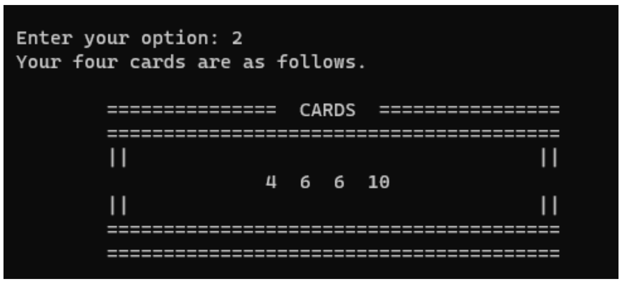
Gambar 5.7 Input Data Uji 7



```
8 solutions found
1. (((11 - 5) / 3) * 12)
2. ((11 - 5) / (3 / 12))
3. (((11 - 5) * 12) / 3)
4. ((11 - 5) * (12 / 3))
5. ((12 / 3) * (11 - 5))
6. (12 / (3 / (11 - 5)))
7. ((12 * (11 - 5)) / 3)
8. (12 * ((11 - 5) / 3))
The execution time is 1011 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test07
Solution is succesfully saved!
```

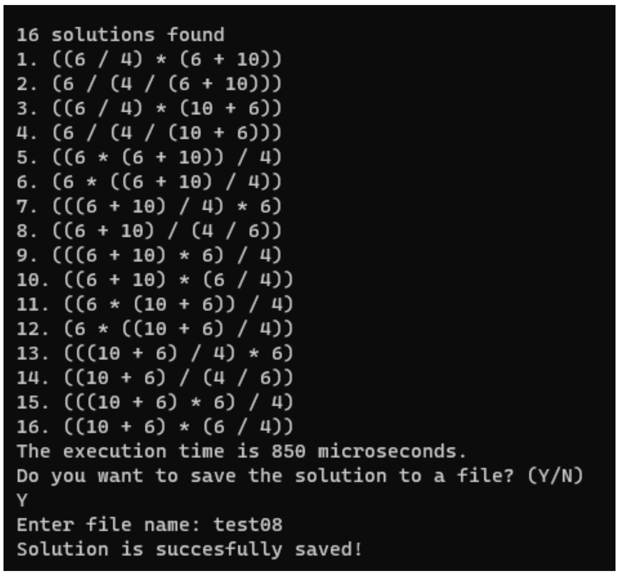
Gambar 5.7 Output Data Uji 7

8. Data Uji 8



```
Enter your option: 2
Your four cards are as follows.

        ===============  CARDS  ================
        ========================================
        ||                                    ||
                      4  6  6  10
        ||                                    ||
        ========================================
        ========================================
```

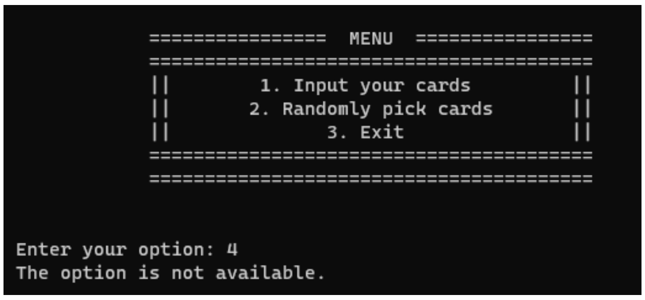
Gambar 5.8 Input Data Uji 8



```
16 solutions found
1. ((6 / 4) * (6 + 10))
2. (6 / (4 / (6 + 10)))
3. ((6 / 4) * (10 + 6))
4. (6 / (4 / (10 + 6)))
5. ((6 * (6 + 10)) / 4)
6. (6 * ((6 + 10) / 4))
7. (((6 + 10) / 4) * 6)
8. ((6 + 10) / (4 / 6))
9. (((6 + 10) * 6) / 4)
10. ((6 + 10) * (6 / 4))
11. ((6 * (10 + 6)) / 4)
12. (6 * ((10 + 6) / 4))
13. (((10 + 6) / 4) * 6)
14. ((10 + 6) / (4 / 6))
15. (((10 + 6) * 6) / 4)
16. ((10 + 6) * (6 / 4))
The execution time is 850 microseconds.
Do you want to save the solution to a file? (Y/N)
Y
Enter file name: test08
Solution is succesfully saved!
```

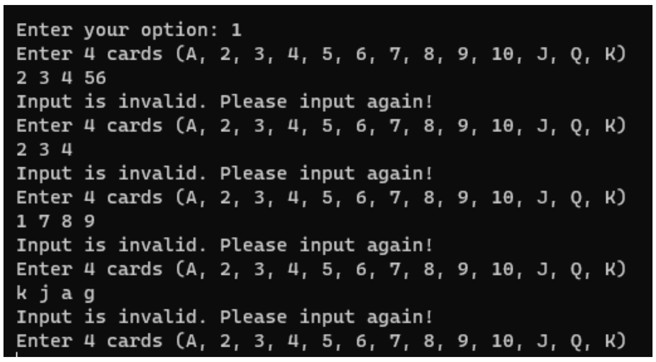
Gambar 5.8 Output Data Uji 8

Program juga melakukan validasi terhadap input user. Berikut beberapa tampilan validasi yang dilakukan program.



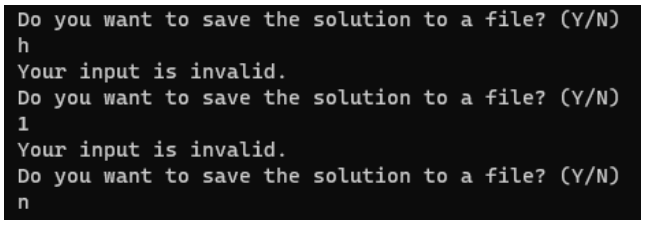
```
            ================  MENU  ================
            ========================================
            ||        1. Input your cards         ||
            ||       2. Randomly pick cards       ||
            ||              3. Exit               ||
            ========================================
            ========================================


Enter your option: 4
The option is not available.
```

Gambar 5.9 Validasi Opsi Menu



```
Enter your option: 1
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
2 3 4 56
Input is invalid. Please input again!
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
2 3 4
Input is invalid. Please input again!
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
1 7 8 9
Input is invalid. Please input again!
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
k j a g
Input is invalid. Please input again!
Enter 4 cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)
```

Gambar 5.10 Validasi Input Kartu



```
Do you want to save the solution to a file? (Y/N)
h
Your input is invalid.
Do you want to save the solution to a file? (Y/N)
1
Your input is invalid.
Do you want to save the solution to a file? (Y/N)
n
```

Gambar 5.11 Validasi Opsi Save

DAFTAR REFERENSI

Algoritma Brute Force. (n.d.). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

Chandra, Evita. 2015. *Penerapan Algoritma Brute Force pada Permainan Kartu 24 (24 game)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf

LAMPIRAN

Repository Github
https://github.com/liviaarumsari/Tucil1_13521094