TUGAS KECIL I IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2022/2023

PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA *BRUTE FORCE*

Disusun oleh:

Made Debby Almadea Putri 13521153

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

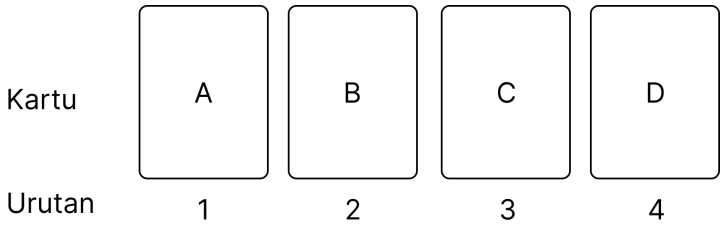
2023

1. Algoritma *Brute Force*

Dalam penyelesaian permainan kartu 24 dengan algoritma *Brute Force* dibagi menjadi 2 algoritma, algoritma dalam menentukan permutasi urutan 4 kartu dan algoritma untuk mencari semua solusi permainan kartu 24.

a. Algoritma *Brute Force* untuk permutasi 4 buah kartu

Algoritma *Brute Force* untuk mencari permutasi dari 4 buah kartu dilakukan secara rekursif. Misalkan kartu pertama yang diambil adalah A, kartu kedua adalah B, kartu ketiga adalah C, dan kartu keempat adalah D. Keempat kartu tersebut dapat bernilai sama dan diurutkan seperti gambar di bawah.



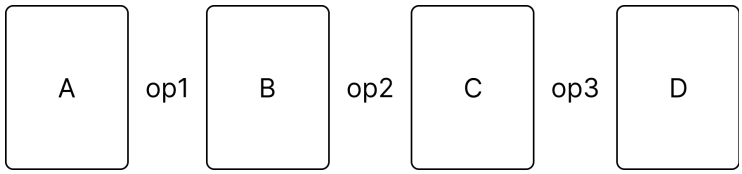
Gambar 1 Kartu dan Urutan Kartu (Sumber: Dokumentasi Pribadi)

1. Tukar kartu pada urutan ke-$i \in \{1, 2, 3, 4\}$ dengan kartu pada urutan pertama. Cek apakah kartu tersebut telah digunakan sebagai kartu pertama sebelumnya. Jika iya, maka tukar kembali kartu tersebut. Jika tidak, maka catat bahwa kartu tersebut telah digunakan sebagai kartu pada urutan pertama. Selanjutnya kita mencari permutasi urutan untuk ketiga kartu lainnya.
2. Tukar kartu pada urutan ke-$j \in \{2, 3, 4\}$ dengan kartu pada urutan kedua. Cek apakah kartu tersebut telah digunakan sebagai kartu kedua sebelumnya. Jika iya, maka tukar kembali kartu tersebut. Jika tidak, maka catat bahwa kartu tersebut telah digunakan sebagai kartu pada urutan kedua. Selanjutnya kita mencari permutasi urutan untuk kedua kartu lainnya.
3. Tukar kartu pada urutan ke-$k \in \{3, 4\}$ dengan kartu pada urutan ketiga. Cek apakah kartu tersebut telah digunakan sebagai kartu ketiga sebelumnya. Jika iya, maka tukar kembali kartu tersebut. Jika tidak, maka catat bahwa kartu tersebut telah digunakan sebagai kartu pada urutan ketiga. Pada tahap ini kita sudah menentukan kartu pada urutan ketiga dan keempat. Urutan kartu ini merupakan salah satu permutasi dari keempat kartu A, B, C, dan D.

b. Algoritma *Brute Force* untuk mencari semua solusi permainan kartu 24

Solusi dari permainan kartu 24 dapat dicari dengan teknik *exhaustive search,* yaitu mencoba semua kombinasi kartu dan operasi lalu menyimpan kombinasi yang memiliki hasil 24. Misalkan keempat buah kartu tersebut adalah A, B, C, dan D. Algoritma dalam mencari semua solusi dari permainan kartu 24 adalah

1. Tentukan permutasi dari keempat kartu tersebut. Pada kasus terburuk, keempat kartu memiliki nilai berbeda, maka terdapat 4! = 24 permutasi.
2. Tentukan op1, op2, dan op3 $\in$ { +, -, × , / } yang akan digunakan sebagai operator yang mengubah keempat nilai pada kartu tersebut menjadi nilai lain. Lihat gambar di bawah sebagai visualisasi



Gambar 2 Kartu dan Operasi pada Kartu (Sumber: Dokumentasi Pribadi)

Sehingga untuk setiap permutasi kartu, terdapat 64 permutasi operator.

3. Dalam permainan kartu 24 diperbolehkan untuk menggunakan tanda kurung '()' untuk menentukan urutan operasi, sehingga kita tentukan beberapa kombinasi tanda kurung yang mungkin dalam permainan ini. Kombinasi tersebut adalah

$$((A\ op1\ B)\ op2\ C)\ op3\ D$$

$$(A\ op1\ (B\ op2\ C))\ op3\ D$$

$$A\ op1\ ((B\ op2\ C)\ op3\ D)$$

Kemudian untuk setiap permutasi kartu dan operator, kita cari nilai yang dihasilkan untuk setiap kombinasi tanda kurung. Jika nilai yang dihasilkan sama dengan 24, maka kombinasi tersebut merupakan salah satu solusi dari permainan kartu 24.

4. Lakukan langkah 1 - 3 hingga semua permutasi kartu dan operator telah dihitung hasilnya untuk masing-masing kombinasi tanda kurung.

2. *Source* Program dalam Bahasa C++

a. Expression/Expression.hpp

```
#ifndef __EXPRESSION__HPP__
#define __EXPRESSION__HPP__
#include <iostream>

class Expression
{
private:
    double (*func)(double, double);
    char sym;

public:
    Expression(double (*initFunc)(double, double), char initSym);
    double eval(double a, double b);
    char getSymbol();
};

#endif
```

b. Expression/Expression.cpp

```
#include <iostream>
#include "Expression.hpp"

Expression::Expression(double (*initFunc)(double, double), char initSym)
{
    /// @brief  konstruktor class Expression
    /// @param initFunc address dari fungsi untuk mengevaluasi ekspresi yang
menerima 2 buah argumen
    /// @param initSym simbol operator dari ekspresi
    func = initFunc;
    sym = initSym;
}

double Expression::eval(double a, double b)
{
    // @return hasil evaluasi dari a op b
    return (*func)(a, b);
}

char Expression::getSymbol()
{
    // @return simbol dari operator
    return sym;
}
```

c. Solver/Solver.hpp

```
#ifndef __SOLVER__HPP__
#define __SOLVER__HPP__
#include <vector>
#include <iostream>
#include "../Expression/Expression.hpp"

using namespace std;

class Solver
{
private:
    vector<double> num;
    vector<Expression> ops;
    int totalSolution;
    vector<string> solution;
    void getPermutation(vector<double>::iterator vec, int vecLen);
    void swapCard(vector<double>::iterator cardA, vector<double>::iterator
cardB);
    int getUserInput();
    void getRandomizedInput();
    double convertInput(string input);
    void printCards();
    int writeSolution(string path = "", string filename =
"24solver_result.txt");
    void solve();
    void displaySolutions();

public:
    Solver();
    void getInput();
    void solveAll();
};

#endif
```

d. Solver/Solver.cpp

```
#include <vector>
#include <iostream>
#include <cstdlib>
```

```cpp
#include <time.h>
#include <fstream>
#include <chrono>
#include "Solver.hpp"

double add(double a, double b) { return (a + b); }
double subtract(double a, double b) { return (a - b); }
double multiple(double a, double b) { return (a * b); }
double divide(double a, double b) { return (a / b); }

Solver::Solver()
{
    // @brief konstruktor untuk class Solver

    Expression addExpr(&add, '+');
    Expression subExpr(&subtract, '-');
    Expression mulExpr(&multiple, '*');
    Expression divExpr(&divide, '/');
    ops.push_back(addExpr);
    ops.push_back(subExpr);
    ops.push_back(mulExpr);
    ops.push_back(divExpr);

    totalSolution = 0;
}

double Solver::convertInput(string input)
{
    // @brief mengubah input menjadi angka yang dapat dioperasikan
    /* @return 0 jika input tidak valid, 1 jika input valid
    dan berhasil diubah */
    if (input.length() > 1)
    {
        if (input == "10")
            return 10;
        else
            return 0;
    }
    else
    {
        if (input[0] >= '2' && input[0] <= '9')
            return (int)(input[0]) - 48;

        if (input == "A" || input == "a")
            return 1;

        if (input == "J" || input == "j")
```

```
            return 11;

        if (input == "Q" || input == "q")
            return 12;

        if (input == "K" || input == "k")
            return 13;
    }

    return 0;
}

void Solver::printCards()
{
    // @brief mencetak kartu dengan format A B C D
    for (int i = 0; i < 4; i++)
    {
        if (num[i] == 1)
            cout << "A";
        else if (num[i] == 11)
            cout << "J";
        else if (num[i] == 12)
            cout << "Q";
        else if (num[i] == 13)
            cout << "K";
        else
            cout << num[i];

        cout << " ";
    }
}

int Solver::getUserInput()
{
    // @brief mendapatkan input dari user melalui cli
    // @return 0 jika input tidak valid, 1 jika valid
    int i;
    string inputline;
    string input = "";
    int count = 0;

    cout << ">> Input 4 cards: ";
    getline(cin, inputline);

    for (auto x : inputline)
    {
        if (count > 3)
```

```
        {
            cout << "Invalid input. Too many inputs." << endl;
            return 0;
        }

        if (x == ' ')
        {
            (num).push_back(convertInput(input));
            if (!(num)[count])
            {
                cout << "Invalid input. There's no " << input << " symbol in
cards." << endl;
                return 0;
            }
            input = "";
            count++;
        }
        else
        {
            input = input + x;
        }
    }
    if (input != "")
    {
        (num).push_back(convertInput(input));
        if (!(num)[count])
        {
            cout << "Invalid input. There's no " << input << " symbol in
cards." << endl;
            return 0;
        }
        count++;
    }
    if (count < 4)
    {
        cout << "Invalid input. Too little input." << endl;
        return 0;
    }

    return 1;
}

void Solver::getRandomizedInput()
{
    // @brief mendapatkan input secara random
    int i;
```

```
    srand(time(0));
    for (i = 0; i < 4; i++)
    {
        (num).push_back(rand() % 13 + 1);
    }
}

void Solver::getInput()
{
    // @brief interface untuk memperoleh input kartu
    char choice = '9';
    cout << "Welcome to 24 Solver!" << endl;
    cout << "====================" << endl;
    cout << "Choose input method" << endl
         << "1. Cli Input" << endl
         << "2. Randomized Input" << endl;
    cout << "====================" << endl;
    do
    {
        cout << ">> Your choice: ";
        cin >> choice;
        cout << endl;
        if (int(choice) - 48 == 1)
        {
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            while (!getUserInput())
            {
                num.clear();
                cout << endl;
            }
            cout << endl;
        }
        else if (int(choice) - 48 == 2)
        {
            getRandomizedInput();
            cout << "Your 4 cards are:" << endl;
            printCards();
            cout << endl
                 << endl;
        }
    } while (int(choice) - 48 > 2);
}

int Solver::writeSolution(string path, string filename)
{
    // @brief menulis solusi pada file
    // @param path path dari file
```

```
    // @param filename nama dari file, termasuk extensionnya
    // @return 0 jika gagal menulis file, 1 jika berhasil
    try
    {
        ofstream MyFile(path + filename);
        for (auto sol : solution)
        {
            MyFile << sol << endl;
        }
        MyFile.close();
        return 1;
    }
    catch (const exception &e)
    {
        cerr << e.what() << '\n';
        return 0;
    }
}

void Solver::solve()
{
    // @brief mendapatkan solusi untuk 1 buah permutasi kartu
    string strA, strB, strC, strD;
    double eps = 1e-9;
    strA = to_string((int)num[0]);
    strB = to_string((int)num[1]);
    strC = to_string((int)num[2]);
    strD = to_string((int)num[3]);

    for (auto op1 : ops)
    {
        for (auto op2 : ops)
        {
            for (auto op3 : ops)
            {
                // ((a op1 b) op2 c) op3 d
                if (abs(op3.eval(op2.eval(op1.eval(num[0], num[1]), num[2]),
num[3]) - 24) < eps)
                {
                    solution.push_back("((" + strA + " " + op1.getSymbol() +
" " + strB + ")" + " " + op2.getSymbol() + " " + strC + ")" + " " +
op3.getSymbol() + " " + strD + ")");
                    totalSolution++;
                }

                // (a op1 (b op2 c)) op3 d
                if (abs(op3.eval(op1.eval(num[0], op2.eval(num[1], num[2])),
```

```
num[3]) - 24) < eps)
                {
                    solution.push_back("(" + strA + " " + op1.getSymbol() +
" " + "(" + strB + " " + op2.getSymbol() + " " + strC + "))" + " " +
op3.getSymbol() + " " + strD);
                    totalSolution++;
                }

                // a op1 ((b op2 c) op3 d)
                if (abs(op1.eval(num[0], op3.eval(op2.eval(num[1], num[2]),
num[3])) - 24) < eps)
                {
                    solution.push_back(strA + " " + op1.getSymbol() + " " +
"((" + strB + " " + op2.getSymbol() + " " + strC + ")" + " " +
op3.getSymbol() + " " + strD + ")");
                    totalSolution++;
                }

                // a op1 (b op2 (c op3 d))
                if (abs(op1.eval(num[0], op2.eval(num[1], op3.eval(num[2],
num[3])))) - 24) < eps)
                {
                    solution.push_back(strA + " " + op1.getSymbol() + " " +
"(" + strB + " " + op2.getSymbol() + " " + "(" + strC + " " +
op3.getSymbol() + " " + strD + "))");
                    totalSolution++;
                }

                // (a op1 b) op2 (c op3 d)
                if (abs(op2.eval(op1.eval(num[0], num[1]), op3.eval(num[2],
num[3])) - 24) < eps)
                {
                    solution.push_back("(" + strA + " " + op1.getSymbol() +
" " + strB + ")" + " " + op2.getSymbol() + " " + "(" + strC + " " +
op3.getSymbol() + " " + strD + ")");
                    totalSolution++;
                }
            }
        }
    }
}

void Solver::displaySolutions()
{
    // @brief mencetak semua solusi
    if (totalSolution == 0)
    {
```

```
        cout << "No Solution." << endl;
    }
    else
    {
        cout << "Total solution: " << totalSolution << endl
             << "List of solution(s):" << endl;
        for (auto x : solution)
        {
            cout << x << endl;
        }
    }
}

void Solver::swapCard(vector<double>::iterator cardA,
vector<double>::iterator cardB)
{
    // @brief menukar urutan 2 buah kartu cardA dan cardB
    double tempA = *(cardA);
    *(cardA) = *(cardB);
    *(cardB) = tempA;
}

void Solver::getPermutation(vector<double>::iterator vec, int vecLen)
{
    /* @brief mencari semua permutasi dari kartu sekaligus
    memanggil prosedur solve untuk mendapatkan solusi
    dari permutasi tersebut */
    bool visited[13];
    for (int i = 0; i < 13; i++)
    {
        visited[i] = false;
    }

    if (vecLen == 1)
    {
        solve();
    }
    else
    {
        for (int i = 0; i < vecLen; i++)
        {
            if (!visited[(int)*(vec + i) - 1])
            {
                visited[(int)*(vec + i) - 1] = true;
                swapCard(vec, (vec + i));
                getPermutation(vec + 1, vecLen - 1);
                swapCard(vec, (vec + i));
```

```
            }
        }
    }
}

void Solver::solveAll()
{
    /* @brief mencari semua solusi dari semua permutasi dari
     permainan kartu 24 */
    char choice = '9';
    auto start = chrono::high_resolution_clock::now();
    getPermutation(num.begin(), 4);
    auto end = chrono::high_resolution_clock::now();
    displaySolutions();
    cout << endl
         << "Elapsed time: " <<
(double)(chrono::duration_cast<chrono::microseconds>(end - start).count()) /
1000 << "ms" << endl;

    cout << endl
         << "====================" << endl
         << "Save this solution?" << endl
         << "1. Yes" << endl
         << "2. No" << endl
         << "====================" << endl;
    do
    {
        cout << ">> Your choice: ";
        cin >> choice;
        if (int(choice) - 48 == 1)
        {
            string filename;

            cout << ">> Input filename (incl .txt): ";
            cin >> filename;

            while (!writeSolution("test/", filename))
            {
                cout << ">> Input filename (incl .txt): ";
                cin >> filename;
            }

            cout << "File saved in folder test!" << endl;
        }

    } while (int(choice) - 48 > 2);
}
```

e. main.cpp

```cpp
#include "Solver/Solver.hpp"

int main()
{
    // main program
    Solver solver;
    solver.getInput();
    solver.solveAll();

    return 0;
}
```

3. Screenshot input dan output

Kartu	Input	Output

7 4 9 2	Welcome to 24 Solver! ========== Choose input method 1. Cli Input 2. Randomized Input ========== >> Your choice: 2 Your 4 cards are: 7 4 9 2	Total solution: 40 List of solution(s): ((7 - 4) + 9) * 2) (7 - (4 - 9)) * 2 (7 + (4 * 2)) + 9 7 + ((4 * 2) + 9) 7 + (9 + (4 * 2)) (7 + 9) + (4 * 2) ((7 + 9) - 4) * 2) (7 + (9 - 4)) * 2 7 + (9 + (2 * 4)) (7 + 9) + (2 * 4) (7 + (2 * 4)) + 9 7 + ((2 * 4) + 9) ((4 * 2) + 9) + 7) (4 * 2) + (9 + 7) ((4 * 2) + 7) + 9) (4 * 2) + (7 + 9) ((9 - 4) + 7) * 2) (9 - (4 - 7)) * 2 (9 + (4 * 2)) + 7 9 + ((4 * 2) + 7) 9 + (7 + (4 * 2)) (9 + 7) + (4 * 2) ((9 + 7) - 4) * 2) (9 + (7 - 4)) * 2 9 + (7 + (2 * 4)) (9 + 7) + (2 * 4) (9 + (2 * 4)) + 7 9 + ((2 * 4) + 7) ((2 * 4) + 9) + 7) (2 * 4) + (9 + 7) ((2 * 4) + 7) + 9) (2 * 4) + (7 + 9) 2 * ((9 - 4) + 7) 2 * (9 - (4 - 7)) 2 * ((9 + 7) - 4) 2 * (9 + (7 - 4)) 2 * ((7 + 9) - 4) 2 * (7 + (9 - 4)) 2 * ((7 - 4) + 9) 2 * (7 - (4 - 9)) Elapsed time: 0.298ms

6 6 6 Q	Welcome to 24 Solver! ========== Choose input method 1. Cli Input 2. Randomized Input ========== >> Your choice: 2 Your 4 cards are: 6 6 6 Q	Total solution: 10 List of solution(s): ((6 + 6) / 6) * 12) (6 + 6) / (6 / 12) ((6 + 6) * 12) / 6) (6 + 6) * (12 / 6) 6 * (6 - (12 / 6)) (6 - (12 / 6)) * 6 (12 * (6 + 6)) / 6 12 * ((6 + 6) / 6) (12 / 6) * (6 + 6) 12 / (6 / (6 + 6)) Elapsed time: 0.119ms

3 4 8 2	Welcome to 24 Solver! ========== Choose input method 1. Cli Input 2. Randomized Input ========== >> Your choice: 2 Your 4 cards are: 3 4 8 2	Total solution: 32 List of solution(s): 3 * (4 + (8 / 2)) (3 * (8 - 4)) * 2 3 * ((8 - 4) * 2) 3 * ((8 / 2) + 4) 3 * (2 * (8 - 4)) (3 * 2) * (8 - 4) ((4 - 3) + 2) * 8) (4 - (3 - 2)) * 8 (4 + (8 / 2)) * 3 ((4 + 2) - 3) * 8) (4 + (2 - 3)) * 8 ((8 - 4) * 3) * 2) (8 - 4) * (3 * 2) 8 * ((4 - 3) + 2) 8 * (4 - (3 - 2)) ((8 - 4) * 2) * 3) (8 - 4) * (2 * 3) 8 * ((4 + 2) - 3) 8 * (4 + (2 - 3)) 8 * ((2 - 3) + 4) 8 * (2 - (3 - 4)) 8 * ((2 + 4) - 3) 8 * (2 + (4 - 3)) ((8 / 2) + 4) * 3) ((2 + 4) - 3) * 8) (2 + (4 - 3)) * 8 (2 * (8 - 4)) * 3 2 * ((8 - 4) * 3) 2 * (3 * (8 - 4)) (2 * 3) * (8 - 4) ((2 - 3) + 4) * 8) (2 - (3 - 4)) * 8 Elapsed time: 0.597ms

K 4 Q 4	Welcome to 24 Solver! ========== Choose input method 1. Cli Input 2. Randomized Input ========== >> Your choice: 1 >> Input 4 cards: k 4 q 4	Total solution: 10 List of solution(s): ((13 - 4) * 4) - 12) (13 - (4 / 4)) + 12 13 - ((4 / 4) - 12) 13 + (12 - (4 / 4)) (13 + 12) - (4 / 4) (4 * (13 - 4)) - 12 (12 - (4 / 4)) + 13 12 - ((4 / 4) - 13) 12 + (13 - (4 / 4)) (12 + 13) - (4 / 4) Elapsed time: 0.357ms

Q Q Q Q	Welcome to 24 Solver! ========== Choose input method 1. Cli Input 2. Randomized Input ========== >> Your choice: 1 >> Input 4 cards: Q Q Q Q	Total solution: 31 List of solution(s): ((12 + 12) + 12) - 12) (12 + (12 + 12)) - 12 12 + ((12 + 12) - 12) 12 + (12 + (12 - 12)) (12 + 12) + (12 - 12) ((12 + 12) - 12) + 12) (12 + (12 - 12)) + 12 12 + ((12 - 12) + 12) 12 + (12 - (12 - 12)) (12 + 12) - (12 - 12) ((12 + 12) * 12) / 12) 12 + ((12 * 12) / 12) 12 + (12 * (12 / 12)) (12 + 12) * (12 / 12) ((12 + 12) / 12) * 12) 12 + ((12 / 12) * 12) 12 + (12 / (12 / 12)) (12 + 12) / (12 / 12) ((12 - 12) + 12) + 12) (12 - 12) + (12 + 12) (12 - (12 - 12)) + 12 12 - (12 - (12 + 12)) 12 - ((12 - 12) - 12) (12 * (12 + 12)) / 12 12 * ((12 + 12) / 12) ((12 * 12) / 12) + 12) (12 * (12 / 12)) + 12 ((12 / 12) * 12) + 12) (12 / 12) * (12 + 12) (12 / (12 / 12)) + 12 12 / (12 / (12 + 12)) Elapsed time: 0.101ms
9 5 4 8	Welcome to 24 Solver! ========== Choose input method 1. Cli Input 2. Randomized Input ========== >> Your choice: 2 Your 4 cards are: 9 5 4 8	

		Total solution: 22 List of solution(s): ((9 - 5) * 4) + 8 ((9 + 5) - 8) * 4) (9 + (5 - 8)) * 4 ((9 - 8) + 5) * 4) (9 - (8 - 5)) * 4 ((5 + 9) - 8) * 4) (5 + (9 - 8)) * 4 ((5 - 8) + 9) * 4) (5 - (8 - 9)) * 4 4 * ((5 + 9) - 8) 4 * (5 + (9 - 8)) 4 * ((5 - 8) + 9) 4 * (5 - (8 - 9)) 4 * ((9 + 5) - 8) 4 * (9 + (5 - 8)) (4 * (9 - 5)) + 8 4 * ((9 - 8) + 5) 4 * (9 - (8 - 5)) 8 - ((5 - 9) * 4) 8 - (4 * (5 - 9)) 8 + (4 * (9 - 5)) 8 + ((9 - 5) * 4) Elapsed time: 0.425ms

4. Link to Repository

https://github.com/debbyalmadea/Tucil1_13521153

5. Checklist

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