Tugas Kecil IF2211 Strategi Algoritma

Laporan Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Oleh:

Ghazi Akmal Fauzan

K01 / 13521058

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

A. Algoritma *Brute Force*

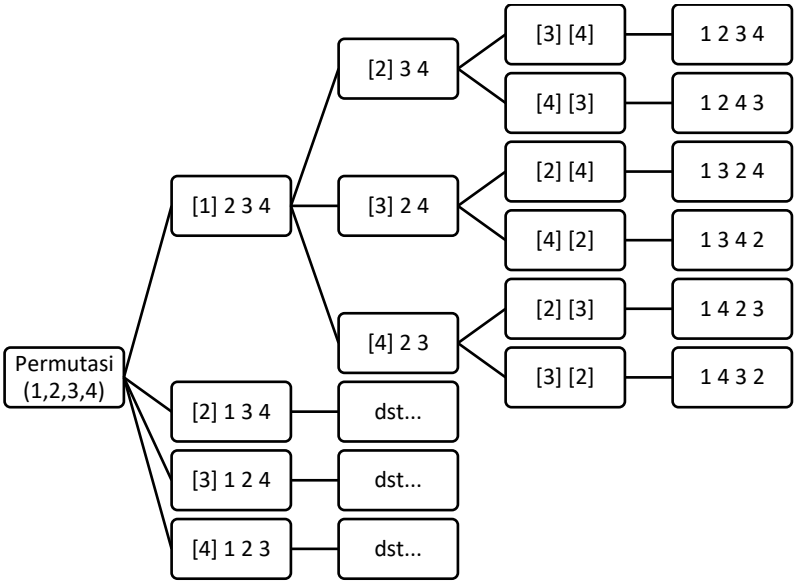
Algoritma *brute force* adalah suatu pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma ini biasanya didasarkan pada pernyataan pada persoalan (*problem statement*) dan definisi konsep yang dilibatkan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, dan dengan cara yang jelas (*obvious way*). Karakteristik algoritma *brute force* umumnya tidak cerdas, karena membutuhkan jumlah langkah yang besar dalam penyelesaiannya. Algoritma *brute force* dapat disebut juga algoritma naif (*naïve algorithm*).

Dalam tugas kecil kali ini algoritma *brute force* diimplementasikan untuk membuat program permainan kartu 24. Program ini merupakan program yang dapat menyelesaikan permainan kartu 24 dengan menggunakan algoritma *brute force*. Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. 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.

Langkah-langkah algoritma *brute force* yang saya implementasikan adalah sebagai berikut.

1. Program membaca input dari pengguna untuk memilih apakah kartu ingin dimasukkan secara manual oleh pengguna atau kartu dipilih secara acak oleh program. Input kartu ini akan divalidasi oleh program.

2. 4 kartu yang terpilih ini akan dilakukan permutasi secara rekursif, sehingga didapatkan urutan angka yang bervariasi. Contoh permutasinya adalah sebagai berikut.



3. Setelah mendapat semua kombinasi angka, angka-angka ini akan dioperasikan atau dihitung dengan menggunakan operator yang ada, kombinasi mana saja yang dapat menghasilkan angka 24. Terdapat 3 posisi operator, 4 angka, dan 5 jenis posisi tanda kurung yang berbeda, sehingga dapat menghasilkan kemungkinan sebagai berikut.
    a. ((N OP N) OP N) OP N
    b. (N OP (N OP N)) OP N
    c. N OP (N OP (N OP N))
    d. N OP ((N OP N) OP N)
    e. (N OP N) OP (N OP N)

    Dengan N adalah angka dan OP adalah operator. Algoritma akan di-*looping* dan semua kemungkinan akan dicoba ke lima kemungkinan di atas. Maka akan didapatkan hasil dari algoritma *brute force*.

4. Untuk menghilangkan hasil yang terduplikasi maka digunakan struktur data set. Contoh duplikasi kartu misal terdapat kartu K 4 3 3, maka terdapat kemungkinan hasil ((13 - 4) * 3') – 3" dengan ((13 - 4) * 3") – 3'.

5. Kemudian ditampilkan banyaknya solusi yang ditemukan dan dicetak semua kombinasinya serta lama waktu eksekusi.

6. Terakhir program akan meminta input apakah solusi ini ingin disimpan dalam file atau tidak. User akan memasukkan nama file txt apabila ingin disimpan dan file akan disimpan pada folder 'test'.

B. Source Program

1. Library and Setup

```cpp
/* I/O Stream */
#include <iostream>

/* STL */
#include <sstream>
#include <vector>
#include <set>

/* Execution Time */
#include <chrono>

/* File Writing */
#include <fstream>

/* Sleep for Splash Screen */
#include <thread>

/* UI (Color) and Setup */
using namespace std;
#define RESET  "\033[0m"
#define RED    "\033[1m\033[31m"
#define GREEN  "\033[1m\033[32m"
#define YELLOW "\033[1m\033[33m"
#define WHITE  "\033[1m\033[37m"
```

2. splash

```
void splash() {
    /* Program splash screen */

    cout << endl << GREEN << "    /$$      /$$/$$$$$$$$/$$       /$$$$$$  /$$$$$$ /$$      /$$/$$$$$$$$/$$" << endl;
    cout <<                  "   | $$  /$ | $| $$_____| $$      /$$__  $$/$$__  $| $$$    /$$| $$_____| $$" << endl;
    cout <<                  "   | $$ /$$$| $| $$     | $$     | $$  \\__| $$  \\ $| $$$$  /$$$| $$     | $$" << endl;
    cout <<                  "   | $$/$$ $$ $| $$$$$  | $$     | $$     | $$  | $| $$ $$/$$ $| $$$$$  | $$" << endl;
    cout <<                  "   | $$$$_  $$$| $$__/  | $$     | $$     | $$  | $| $$  $$$| $| $$__/  |__/" << endl;
    cout <<                  "   | $$$/ \\  $$| $$     | $$     | $$    $| $$  | $| $$\\  $ | $| $$         " << endl;
    cout <<                  "   | $$/   \\  $| $$$$$$$| $$$$$$$|  $$$$$$|  $$$$$$| $$ \\/  | $| $$$$$$$$/$$" << endl;
    cout <<                  "   |__/     \\__|________|_______/ \\______/ \\______/|__/     |__|________|__/" << RESET << endl;

    cout << endl << WHITE << "Welcome to 24 Cards Solver" << RESET << endl;
    cout << YELLOW << "Made By Ghazi Akmal Fauzan (13521058)" << RESET << endl;
    cout << GREEN << "Loading" << RESET;
    this_thread::sleep_for(chrono::seconds(1));
    cout << GREEN << "." << RESET;
    this_thread::sleep_for(chrono::seconds(1));
    cout << GREEN << "." << RESET;
    this_thread::sleep_for(chrono::seconds(1));
    cout << GREEN << "." << RESET << RESET << endl;
    cout << WHITE << "Solver Loaded!" << RESET << endl << endl;
}
```

3. command1 (Command Start/Exit)

```
void command1() {
    /* Command Start/Exit */

    cout << WHITE << "=============================================" << RESET << endl;
    cout << RED <<   "|               START/EXIT                 |" << RESET << endl;
    cout << WHITE << "=============================================" << RESET << endl;
    cout << RED << "1." << WHITE << " START" << RESET << endl;
    cout << RED << "2." << WHITE << " EXIT" << RESET << endl;
}
```

4. command2 (Command List Input)

```
void command2() {
    /* Command List Input */

    cout << WHITE << "=============================================" << RESET << endl;
    cout << RED <<   "|              List Input                   |" << RESET << endl;
    cout << WHITE << "=============================================" << RESET << endl;
    cout << RED << "1." << WHITE << " Input 4 Cards [A, 2, ... , 10, J, Q, K]" << RESET << endl;
    cout << RED << "2." << WHITE << " Generate 4 Random Cards" << RESET << endl;
}
```

5. command3 (Command Save)

```
void command3() {
    /* Command Save */

    cout << endl << WHITE << "=============================================" << RESET << endl;
    cout << RED <<           "|   Do you want to save the solutions?    |" << RESET << endl;
    cout << WHITE <<         "=============================================" << RESET << endl;
    cout << RED << "1." << WHITE <<" Yes" << RESET << endl;
    cout << RED << "2." << WHITE <<" No" << RESET << endl;
}
```

6. countWords

```
int countWords(string str) {
    /* Count how many words used in a single line of string */

    bool space = true;
    int count = 0;

    for (auto c:str) {
        if (isspace(c)) {
            space = true;
        }
        else {
            if (space) {
                ++count;
                space = false;
            }
        }
    }

    return count;
}
```

7. isCardValid

```
bool isCardValid(string str) {
    /* Check if the input from user is [A, 2, ... , Q, K] */

    /* 1 char */
    if (str.length() == 1) {
        if (str[0] == '2' || str[0] == '3' || str[0] == '4' || str[0] == '5' || str[0]
== '6' || str[0] == '7' || str[0] == '8' || str[0] == '9' || str[0] == 'A' || str[0] ==
'J' || str[0] == 'Q' || str[0] == 'K') {
            return true;
        }
        else {
            return false;
        }
    }
    /* 2 chars */
    else if (str.length() == 2) {
        if (str[0] == '1' || str[1] == '0') {
            return true;
        }
        else {
            return false;
        }
    }
    /* > 3 chars */
    else {
        return false;
    }
}
```

8. isAll4Valid

```
bool isAll4Valid(string str) {
    /* Check if the input from user is consist of 4 valid digits or words */

    if (countWords(str) != 4) {
        return false;
    }

    vector <string> input_splitted;
    string input2;
    stringstream str2(str);

    while (getline(str2, input2, ' ')) {
        input_splitted.push_back(input2);
    }

    if (isCardValid(input_splitted[0]) && isCardValid(input_splitted[1]) &&
isCardValid(input_splitted[2]) && isCardValid(input_splitted[3])) {
        return true;
    }
    else {
        return false;
    }
}
```

9. strToNum

```
int strToNum(string str) {
    /* Convert string to number (integer) */

    if (str[0] == 'A') {
        return 1;
    }
    else if (str[0] == 'J') {
        return 11;
    }
    else if (str[0] == 'Q') {
        return 12;
    }
    else if (str[0] == 'K') {
        return 13;
    }
    else if (str.length() == 2 && str[0] == '1' && str[1] == '0') {
            return 10;
    }
    else {
        return str[0] - '0';
    }
}
```

10. numToString

```
string numToString(int num) {
    /* Convert number (integer) to string */

    switch(num) {
        case 1:
            return "A";
        case 11:
            return "J";
        case 12:
            return "Q";
        case 13:
            return "K";
        default:
            return to_string(num);
    }
}
```

11. opToNum

```
int opToNum(char op) {
    /* Convert operator to number (integer) */

    switch(op) {
        case '+':
            return 0;
        case '-':
            return 1;
        case '*':
            return 2;
        default:
            return 3;
    }
}
```

12. absolute

```
double absolute(double x) {
    /* Convert negative number to positive number */

    if (x < 0) {
        return -x;
    }
    else {
        return x;
    }
}
```

13. calc

```
double calc(double a, char op, double b) {
    /* Calculate 2 numbers using 4 different operators */

    switch(op) {
        case '+':
            return a + b;
        case '-':
            return a - b;
        case '*':
            return a * b;
        default:
            if (b != 0) {
                return a / b;
            }
            else {
                return -9999;
            }
    }
}
```

14. permutation

```
vector <vector<int>> permutation(vector<int> &nums) {
    /* Calculate vector of vector containing permutations */

    if (nums.size() <= 1) {
        return {nums};
    }

    vector <vector<int>> result;
    for (int i = 0; i < nums.size(); i ++) {
        vector<int> v(nums.begin(), nums.end());
        v.erase(v.begin() + i);
        auto res = permutation(v);

        for (int j = 0; j < res.size(); j ++) {
            vector<int> _v = res[j];
            _v.insert(_v.begin(), nums[i]);
            result.push_back(_v);
        }
    }

    return result;
}
```

15. userInput

```
vector <int> userInput() {
    /* User input */

    string input, input2;
    vector <int> numbers;

    cout << endl << WHITE << "Input format (1 2 3 4)" << RESET << endl;
    cout << WHITE << "Enter 4 cards: " << RESET;
    getline(cin, input);

    while (!isAll4Valid(input)) {
        cout << endl << RED << "Please enter 4 valid value [A, 2,  ... , 10, J, Q, K]" <<
RESET << endl;
        cout << WHITE << "Input format (1 2 3 4)" << RESET << endl;
        cout << WHITE <<  "Enter 4 cards: " << RESET;
        getline(cin, input);
    }

    stringstream ss(input);

    while (getline(ss, input2, ' ')) {
        numbers.push_back(strToNum(input2));
    }

    return numbers;
}
```

16. randomInput

```
vector <int> randomInput() {
    /* Random generated card */

    srand(time(0));
    vector <int> random;

    for (int i = 0; i < 4; i ++) {
        random.push_back((rand() % 13) + 1);
    }

    return random;
}
```

17. commandInput

```cpp
int commandInput() {
    /* Input for command */
    /* Validate input, must be 1 or 2 */

    int input;

    while (true) {
        cout << WHITE << ">> " << RESET;
        cin >> input;
        if (input == 1 || input == 2) {
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            break;
        }
        else {
            cout << endl << RED << "Please enter a valid input! (1/2)" << RESET << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }
    }

    return input;
}
```

18. app

```cpp
void app() {
    /* APP */

    int input;
    vector <int> numbers;


    /* Cards Input (User or random) */
    cout << endl;
    command2();

    input = commandInput();
    if (input == 1) {
        numbers = userInput();
    }
    else {
        numbers = randomInput();
        cout << endl << GREEN << "Selected cards: " << RESET;
        for (auto x:numbers) {
            cout << WHITE << numToString(x) << " " << RESET;
        }
        cout << endl;
    }
    /* Start Execution Time */
    auto start = chrono::high_resolution_clock::now();


    /* Main Algorithm (Brute Force) */
    vector <vector<int>> permutateNumbers = permutation(numbers);
    vector <vector<int>> hasil;

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

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        for (int k = 0; k < 4; k++) {
            for (int x = 0; x < 24; x++) {
                int a = permutateNumbers[x][0];
                int b = permutateNumbers[x][1];
                int c = permutateNumbers[x][2];
                int d = permutateNumbers[x][3];

                // ((N OP N) OP N) OP N
                if (absolute(calc(calc(calc(a, operators[i], b), operators[j], c), operators[k], d) - 24) < __FLT_EPSILON__) {
                    hasil.push_back({1, a, opToNum(operators[i]), b, opToNum(operators[j]), c, opToNum(operators[k]), d});
                }

                // (N OP (N OP N)) OP N
                if (absolute(calc(calc(a, operators[i], calc(b, operators[j], c)), operators[k], d) - 24) < __FLT_EPSILON__) {
                    hasil.push_back({2, a, opToNum(operators[i]), b, opToNum(operators[j]), c, opToNum(operators[k]), d});
                }

                // (N OP N) OP (N OP N)
                if (absolute(calc(calc(a, operators[i], b), operators[j], calc(c, operators[k], d)) - 24) < __FLT_EPSILON__) {
                    hasil.push_back({0, a, opToNum(operators[i]), b, opToNum(operators[j]), c, opToNum(operators[k]), d});
                }

                // N OP ((N OP N) OP N)
                if (absolute(calc(a, operators[i], calc(calc(b, operators[j], c), operators[k], d)) - 24) < __FLT_EPSILON__) {
                    hasil.push_back({4, a, opToNum(operators[i]), b, opToNum(operators[j]), c, opToNum(operators[k]), d});
                }

                // N OP (N OP (N OP N))
                if (absolute(calc(a, operators[i], calc(b, operators[j], calc(c, operators[k], d))) - 24) < __FLT_EPSILON__) {
                    hasil.push_back({3, a, opToNum(operators[i]), b, opToNum(operators[j]), c, opToNum(operators[k]), d});
                }
            }
        }
    }
}
set <vector<int>> s( hasil.begin(), hasil.end() );
hasil.assign( s.begin(), s.end() );


/* Stop Execution Time */
auto stop = chrono::high_resolution_clock::now();
/* Print Solutions */
if (s.size() != 0) {
    cout << endl << GREEN << s.size() << " Solutions Found" << RESET << endl;
    std::set< std::vector<int> >::iterator it;
    for (it = s.begin(); it != s.end(); it++) {
        const std::vector<int>& x = (*it);
        switch (x[0]) {
            case 0:
                cout << WHITE << "(" << x[1] << " " << operators[x[2]] << " " << x[3] << ") " << operators[x[4]] << " (" << x[5] <<
                " " << operators[x[6]] << " " << x[7] << ")" << RESET << endl;
                break;
            case 1:
                cout << WHITE << "(("<< x[1] << " " << operators[x[2]] << " " << x[3] << ") " << operators[x[4]] << " " << x[5] <<
                ") " << operators[x[6]] << " " << x[7] << RESET << endl;
                break;
            case 2:
                cout << WHITE << "(" << x[1] << " " << operators[x[2]] << " (" << x[3] << " " << operators[x[4]] << " " << x[5] <<
                ")) " << operators[x[6]] << " " << x[7] << RESET << endl;
                break;
            case 3:
                cout << WHITE << x[1] << " " << operators[x[2]] << " " << "(" << x[3] << " " << operators[x[4]] << " (" << x[5] << "
                " << operators[x[6]] << " " << x[7] << "))" << RESET << endl;
                break;
            case 4:
                cout << WHITE << x[1] << " " << operators[x[2]] << " ((" << x[3] << " " << operators[x[4]] << " " << x[5] << ") " <<
                operators[x[6]] << " " << x[7] << ")" << RESET << endl;
                break;
        }
    }
}
else {
    cout << endl << GREEN << "No Solutions Found" << RESET << endl;
}
```

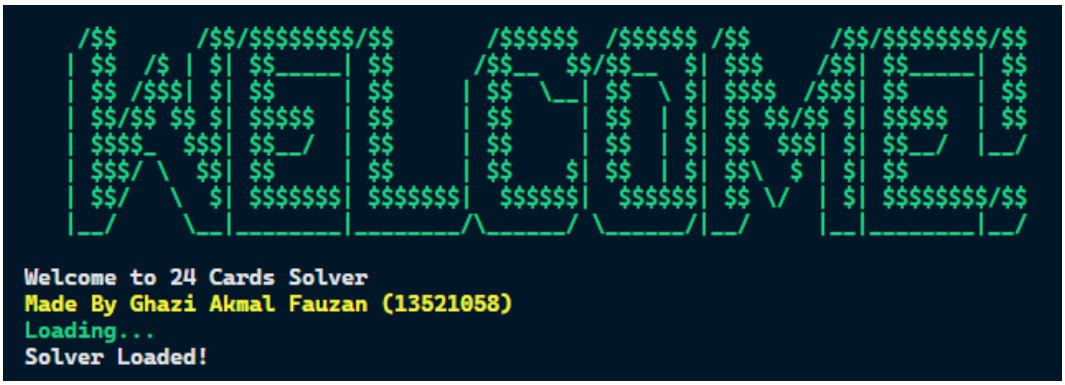
```
    /* Execution Time Calculation and Print */
    auto duration = chrono::duration_cast<std::chrono::microseconds>(stop - start);
    cout << GREEN << "Execution Time: " << RESET << WHITE << duration.count() << " microseconds" << RESET << endl;


    /* Save Algorithm (Writing to File) */
    command3();
    int inputSave;

    inputSave = commandInput();

    if (inputSave == 1) {
        string filename;

        cout << endl << WHITE << "Input Filename: " << RESET;
        cin >> filename;

        ofstream MyFile;

        // Open file
        MyFile.open("test/" + filename + ".txt");

        // Write to file
        for (auto x:numbers) {
            MyFile << numToString(x) << " ";
        }
        MyFile << endl;

        if (s.size() == 0) {
            MyFile << "No Solutions Found" << endl;
        }
        else {
            MyFile << s.size() << " Solutions Found" << endl;

            std::set< std::vector<int> >::iterator itr;
            for (itr = s.begin(); itr != s.end(); itr++) {
                const std::vector<int>& x = (*itr);
                switch (x[0]) {
                    case 0:
                        MyFile << "(" << x[1] << " " << operators[x[2]] << " " << x[3] << ") " << operators[x[4]] << " (" << x[5] << " "
                        << operators[x[6]] << " " << x[7] << ")" << endl;
                        break;
                    case 1:
                        MyFile << "((" << x[1] << " " << operators[x[2]] << " " << x[3] << ") " << operators[x[4]] << " " << x[5] << ") "
                        << operators[x[6]] << " " << x[7] << endl;
                        break;
                    case 2:
                        MyFile << "(" << x[1] << " " << operators[x[2]] << " (" << x[3] << " " << operators[x[4]] << " " << x[5] << "))
                        " << operators[x[6]] << " " << x[7] << endl;
                        break;
                    case 3:
                        MyFile << x[1] << " " << operators[x[2]] << " " << "(" << x[3] << " " << operators[x[4]] << " (" << x[5] << " "
                        << operators[x[6]] << " " << x[7] << "))" << endl;
                        break;
                    case 4:
                        MyFile << x[1] << " " << operators[x[2]] << " ((" << x[3] << " " << operators[x[4]] << " " << x[5] << ") " <<
                        operators[x[6]] << " " << x[7] << ")" << endl;
                        break;
                }
            }
        }

        // Close file
        MyFile.close();

        cout << endl << GREEN << "File saved succesfully" << RESET << endl;
    }
}
```

19. main

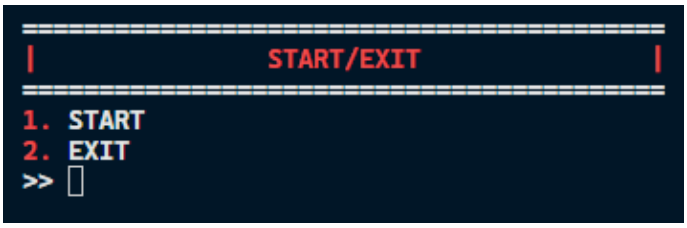
```
int main () {
    /* Main Program */

    int startInput;

    splash();
    command1();

    startInput = commandInput();

    while (startInput == 1) {
        app();
        cout << endl << GREEN << "Do you want to try again?" << RESET << endl << endl;
        command1();
        startInput = commandInput();
    }

    cout << endl << GREEN << "Thank You For Using!" << RESET << endl << endl;
    exit(0);

    return 0;
}
```

C. Input dan Output

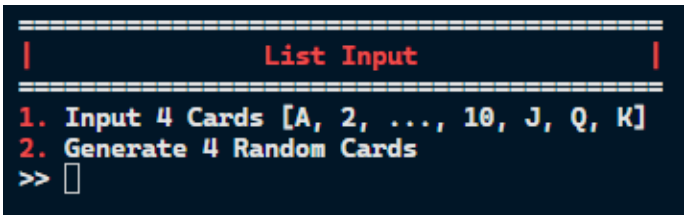
1. Splash Screen



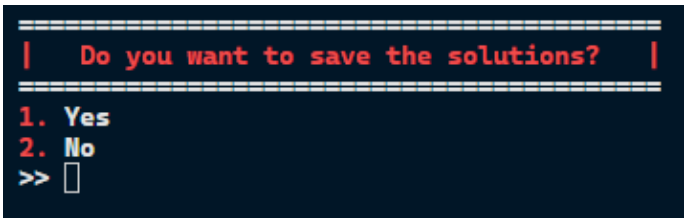
2. Start/Exit Command



3. List Input Command

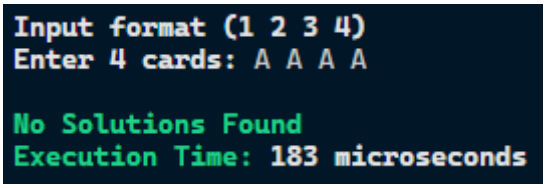


4. Save Command



5. Test Case 1

   Kartu: A A A A (User)



6. Test Case 2
   Kartu: 5 6 7 8 (User)

```
Input format (1 2 3 4)
Enter 4 cards: 5 6 7 8

28 Solutions Found
(5 + 7) * (8 - 6)
(6 * 8) / (7 - 5)
(7 + 5) * (8 - 6)
(8 - 6) * (5 + 7)
(8 - 6) * (7 + 5)
(8 * 6) / (7 - 5)
((5 + 7) - 8) * 6
((5 - 8) + 7) * 6
((7 + 5) - 8) * 6
((7 - 8) + 5) * 6
(5 + (7 - 8)) * 6
(5 - (8 - 7)) * 6
(6 / (7 - 5)) * 8
(7 + (5 - 8)) * 6
(7 - (8 - 5)) * 6
(8 / (7 - 5)) * 6
6 * (5 + (7 - 8))
6 * (5 - (8 - 7))
6 * (7 + (5 - 8))
6 * (7 - (8 - 5))
6 * (8 / (7 - 5))
8 * (6 / (7 - 5))
6 * ((5 + 7) - 8)
6 * ((5 - 8) + 7)
6 * ((7 + 5) - 8)
6 * ((7 - 8) + 5)
6 / ((7 - 5) / 8)
8 / ((7 - 5) / 6)
Execution Time: 251 microseconds
```

7. Test Case 3
   Kartu: K K K K (User)

```
Input format (1 2 3 4)
Enter 4 cards: K K K K

No Solutions Found
Execution Time: 182 microseconds
```

8. Test Case 4
   Kartu: K 4 3 3 (Random)

```
Selected cards: K 4 3 3

2 Solutions Found
((13 - 4) * 3) - 3
(3 * (13 - 4)) - 3
Execution Time: 228 microseconds
```

9. Test Case 5
   Kartu: A 5 6 J (Random)

```
Selected cards: A 5 6 J

6 Solutions Found
((1 + 6) * 5) - 11
((6 + 1) * 5) - 11
((11 - 6) * 5) - 1
(5 * (1 + 6)) - 11
(5 * (6 + 1)) - 11
(5 * (11 - 6)) - 1
Execution Time: 213 microseconds
```

10. Test Case 6
    Kartu: 5 2 8 8 (Random)

```
Selected cards: 5 2 8 8

8 Solutions Found
(5 * 8) - (2 * 8)
(5 * 8) - (8 * 2)
(8 * 5) - (2 * 8)
(8 * 5) - (8 * 2)
((5 * 8) + 8) / 2
((8 * 5) + 8) / 2
(8 + (5 * 8)) / 2
(8 + (8 * 5)) / 2
Execution Time: 210 microseconds
```

11. Txt File
    No Solutions

```
test > TestCase1.txt
1   A A A A
2   No Solutions Found
3
```

    Solutions Found

```
test > TestCase4.txt
1   K 4 3 3
2   2 Solutions Found
3   ((13 - 4) * 3) - 3
4   (3 * (13 - 4)) - 3
5
```

D. *Link to Repository*

https://github.com/ghaziakmalf/Tucil1_13521058

E. *Check List Table*

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	✓	