Laporan Tugas Kecil 1 IF2211 Straregi Algoritma

Semester II Tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh:

Addin Munawwar Yusuf 13521085

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

BAB I : DOMAIN PERSOALAN

1. Deskripsi Masalah

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. (Paragraf di atas dikutip dari sini: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/Makalah Stima-2016-038.pdf)

2. Spesifikasi Tugas

- Tulislah program sederhana dalam Bahasa C/C++/Java yang mengimplementasikan algoritma Brute Force untuk mencari seluruh solusi permainan kartu 24.
- **Input**: 4 angka/huruf yang terdiri dari:
  (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). Contoh input:

  A 8 9 Q

  Selain itu, input juga dapat dilakukan dengan program men-generate 4 angka/hurufnya sendiri secara random. Pengguna dapat memilih apakah program meminta input dari pengguna atau generate sendiri. Apabila masukan tidak sesuai, maka program menampilkan luaran “Masukan tidak sesuai” dan akan meminta ulang.

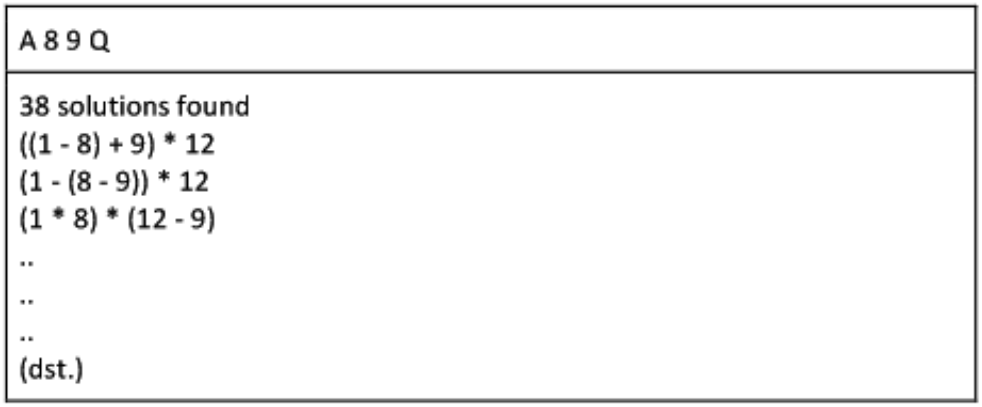
- **Output**:
  1) Banyaknya solusi yang ditemukan.

2) Solusi dari permainan kartu 24 ditampilkan di layar dan terdapat opsi untuk menyimpan solusi dalam file text. Untuk contoh kasus di atas A 8 9 Q, maka salah satu solusinya adalah:

```
(( 9 + A ) - 8 ) * Q atau (( 9 + 1 ) - 8 ) * 12
(Kedua jenis output dibebaskan)
```

Note: Format penulisan output yang dicetak tidak harus persis contoh, yang penting merepresentasikan solusinya sudah cukup. Output apabila tidak ada solusi untuk pasangan kombinasi input, cukup ditampilkan "Tidak ada solusi". Untuk solusi setiap masukan, perlu dipertimbangkan urutan nilai (x1..x4), urutan operator, dan grouping dengan kurung yang mungkin.

Gambaran Apabila Terdapat Solusi:



Gambar 1.2.1 Contoh solusi dari 24 Game Solver

Di akhir, program akan menanyakan "Apakah ingin menyimpan solusi ?". Jika iya, program akan meminta sebuah nama untuk file teks dan semua solusi yang didapat akan disimpan dalam file text tersebut.

3) Waktu eksekusi program (tidak termasuk waktu pembacaan file input).

BAB II : DASAR TEORI

1. Algoritma Brute Force

Algoritma *brute force* merupakan sebuah teknik pemecahan masalah yang lempang (*straightforward*) terhadap sebuah persoalan. Algoritma *brute force* biasanya sangat didasarkan pada pernyataan pada persoalan, serta definisi konsep atau konsep dari persoalan yang dihadapi. Algoritma ini sangatlah sederhana, langsung, dan jelas caranya.

Algoritma ini umumnya tidak “cerdas” dan tidak mangkus, karena *resource* yang dibutuhkan dari segi memori maupun waktu biasanya lebih besar. Itulah alasan kata “force” pada algoritma ini yang mengindikasikan penggunaan tenaga, dibandingkan otak. Maka dari itu, algoritma *brute force* lebih cocok untuk persoalan dengan masukan yang kecil. Meski demikian, algoritma ini masih sering digunakan karena dapat memecahkan hampir sebagian masalah dan mudah dimengerti.

2. Implementasi Algoritma Brute Force dalam 24 Game Solver

Dalam persoalan 24 game solver, teknik yang digunakan adalah dengan *exhaustive search*, yaitu salah satu jenis dari *brute force* dimana kita menjelajahi segala kemungkinan kombinatorik yang ada untuk menemukan solusi yang diinginkan. Untuk dapat menganalisis persoalan kombinatorik pada 24 game solver ini, penulis membagi persoalan ke dalam tiga komponen kombinatorik yang penting yaitu **operator, kurung,** dan **kartu**.

```
a op b op c op d
```

Gambar 2.2.1 Pengaturan kartu dalam permainan 24 game solver

Operator adalah simbol yang melambangkan operasi antara 2 ekspresi. Pada permainan 24 game solver ini, operator yang digunakan adalah penjumlahan (+), pengurangan (-), perkalian (×), divisi (/). Berdasarkan *rule* dari 24 game solver, kita memerlukan 3 operator untuk digunakan dalam satu kombinasi operasi. Dengan menggunakan aturan perkalian, maka dapat kita ketahui bahwa ada $4 \times 4 \times 4 = 64$ permutasi dari operator yang mungkin. Kombinasi operator yang didapat adalah sebagai berikut:

+++	++-	++*	++/	+-+	+--	+-*	+-/
+*+	+*-	+**	+*/	+/+	+/-	+/*	+//
-++	-+-	-+*	-+/	--+	---	--*	--/
-*+	-*-	-**	-*/	-/+	-/-	-/*	-//
*++	*+-	*+*	*+/	*-+	*--	*-*	*-/
**+	**-	***	**/	*/+	*/-	*/*	*//
/++	/+-	/+*	/+/	/-+	/--	/-*	/-/
/*+	/*-	/**	/*/	//+	//-	//*	///

Gambar 2.2.2 64 kombinasi operator yang mungkin

Selain operator, kombinasi **tanda kurung** juga perlu menjadi perhatian. Untuk persoalan 24 game solver, terdapat 5 kombinasi tanda kurung yang dapat terjadi:

```
(a op b) op (c op d)
```

```
((a op b) op c) op d
(a op (b op c)) op d
a op ((b op c) op d)
a op (b op (c op d))
```

Gambar 2.2.3 Kombinasi tanda kurung pada 24 game solver

Adapun beberapa kombinasi tanda kurung tidak disebutkan, seperti (a op b) op c op d, dan lain-lain, disebabkan kombinasi tersebut sudah terwakili oleh 5 kombinasi di atas, sehingga penambahannya hanya akan menjadi sia-sia.

Terakhir yang menjadi perhatian adalah kombinasi **kartu**. Terdapat 4 kartu yang dapat dipermutasikan. Dengan aturan permutasi, maka kita bisa mendapatkan 24 kemungkinan permutasi yang mungkin untuk 24 kartu.

Setelah 3 komponen di atas sudah ditemukan segala kombinasi kemungkinannya, maka kita dapat melakukan *exhaustive search* untuk mencari kombinasi operasi yang menghasilkan solusi yang diinginkan, yaitu menghasilkan 24. Pencarian dilakukan dengan mencocokkan segala operasi dari gabungan semua kombinasi operator, tanda kurung, dan kartu yang ada. Jika hasil evaluasinya menghasilkan 24, maka operasi tersebut merupakan solusi, dan jika sebaliknya, maka operasi tersebut bukan merupakan solusi.

Maka, secara garis besar, tahapan penyelesaian algoritma 24 game solver dengan *brute force* adalah:

- Cari semua kombinasi operator yang ada.
- Dari kombinasi semua operator tersebut, lakukan untuk semua kemungkinan tanda kurung yang ada.
- Dari kombinasi operator dan kurung, enumerasi untuk semua kemungkinan kartu yang ada.
- Lakukan *exhaustive search* untuk mendapatkan solusi yang diinginkan.

BAB III : IMPLEMENTASI

Program ditulis dengan bahasa pemrograman C++. Sistem direktori file yang digunakan adalah sebagai berikut:

```
.
├── bin
│   └── 24solver.exe
├── docs
│   └── Tucil1_K1_13521085_Addin Munawwar Yusuf.pdf
├── src
│   ├── brutecoder.cpp
│   ├── brutecoder.h
│   ├── cards.cpp
```

```
│   ├── cards.h
│   ├── evaluate.txt
│   ├── main.cpp
│   ├── main.h
│   ├── vectoralgo.cpp
│   └── vectoralgo.h
├── test
│   ├── test1.txt
│   ├── test2.txt
│   ├── test3.txt
│   ├── test4.txt
│   ├── test5.txt
│   ├── test6.txt
│   ├── test7.txt
│   └── test8.txt
├── README.md
└── .gitignore
```

1. brutecoder.cpp

Untuk permasalahan kombinatorik operator dan tanda kurung, penulis melakukan pendekatan dengan melakukan *hard code* untuk segala kemungkinan yang ada. Penulis menggunakan pendekatan *hard code* agar program utama 24 game solver tidak perlu mencari kombinasi atau permutasi dari operator dan tanda kurung setiap saat program dijalankan. Akan tetapi, dalam melakukan *hard coding* tersebut sendiri, penulis tidak secara manual melakukan *hard code*, melainkan menggunakan "script" brutecoder.cpp yang melakukan *hard code*.

brutecoder.cpp

```cpp
#include "./brutecoder.h"
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

vector<char> ops = {'+', '-', '*', '/'}; // operator
ofstream ofile; // Output file : evaluate.txt

int main() {
/* I.S : Sembarang
   F.S : Kode lengkap yang dapat mengevaluasi ekspresi dengan 4 variabel
dan semua permutasi dari 3 operator
         tertulis di evaluate.txt */

    ofile.open("evaluate.txt");
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
```

```
        {
            for (int k = 0; k < 4; k++)
            {
                // Tulis kode untuk setiap permutasi operator
                writeCode(i, j, k);
            }
        }
    }

    ofile.close();
    return 0;
}

void writeCode(int i, int j, int k) {
/* I.S : Sembarang
   F.S : Kode yang dapat mengevaluasi ekspresi dengan 4 variabel dan
sebuah permutasi dari 3 operator
         tertulis di evaluate.txt */

    char codeline[75]; // Container sementara untuk menampung kode

    // Evaluasi untuk setiap kasus tanda kurung

    // --- A. 1 group of 2 ---
    // Dapat diabaikan karena hasil dari kasus ini tertampung di kasus
lain

    // --- B. 2 group of 2 ---
    // Satu-satunya kasus (a op b) op (c op d)
    if (ops[i] == '/'){ // Handle division by zero
        sprintf(codeline, "if (b != 0) {");
        ofile << codeline << endl;
    }
    if (ops[k] == '/'){
        sprintf(codeline, "if (d != 0) {");
        ofile << codeline << endl;
    }

    if (ops[j] == '/'){
        sprintf(codeline, "if (c %c d != 0) {", ops[k]);
        ofile << codeline << endl;
    }

    // Tulis kode ke evaluate.txt
    sprintf(codeline, "if ((float)((a %c b) %c (c %c d)) ==
(float)RESULT) {", ops[i], ops[j], ops[k]);
    ofile << codeline << endl;
```

```
    sprintf(codeline, "    sprintf(sol_char, \"(%%d %c %%d) %c (%%d %c
%%d)\", (int)a, (int)b, (int)c, (int)d);", ops[i], ops[j], ops[k]);
    ofile << codeline << endl;
    sprintf(codeline, "    sol_str.assign(sol_char);");
    ofile << codeline << endl;
    sprintf(codeline, "    if (!isMember(solutions, sol_str)){");
    ofile << codeline << endl;
    sprintf(codeline, "    solutions.push_back(sol_str);", ops[i],
ops[j], ops[k]);
    ofile << codeline << endl;
    ofile << "}" << endl;
    ofile << "}" << endl;

    if (ops[i] == '/'){
        ofile << "}" << endl;
    }
    if (ops[j] == '/'){
        ofile << "}" << endl;
    }

    if (ops[k] == '/'){
        ofile << "}" << endl;
    }

    // ... dan seterusnya untuk kasus-kasus kurung lainnya
}
```

output brutecoder.cpp (di-*copy* manual ke program utama)

```
if ((float)((a + b) + (c + d)) == (float)RESULT) {
    sprintf(sol_char, "(%d + %d) + (%d + %d)", (int)a, (int)b, (int)c, (int)d);
    sol_str.assign(sol_char);
    if (!isMember(solutions, sol_str)){
    solutions.push_back(sol_str);
}
}
if ((float)(((a + b) + c) + d) == (float)RESULT) {
    sprintf(sol_char, "((%d + %d) + %d) + %d", (int)a, (int)b, (int)c, (int)d);
    sol_str.assign(sol_char);
    if (!isMember(solutions, sol_str)){
    solutions.push_back(sol_str);
}
}
if ((float)((a + (b + c)) + d) == (float)RESULT) {
    sprintf(sol_char, "(%d + (%d + %d)) + %d", (int)a, (int)b, (int)c, (int)d);
    sol_str.assign(sol_char);
    if (!isMember(solutions, sol_str)){
    solutions.push_back(sol_str);
}
}
if ((float)(a + ((b + c) + d)) == (float)RESULT) {
    sprintf(sol_char, "%d + ((%d + %d) + %d)", (int)a, (int)b, (int)c, (int)d);
    sol_str.assign(sol_char);
    if (!isMember(solutions, sol_str)){
    solutions.push_back(sol_str);
```

2. vectoralgo.cpp

Melakukan operasi-operasi vector dalam 24 game solver.

```
#include "./vectoralgo.h"

bool isMember(vector<string> v, string s) {
/*Mengecek apakah sebuah string t (yang berupa ekspresi) adalah
anggota dari vector v*/
    for (int i = 0; i < v.size(); i++)
    {
        if (v[i] == s)
        {
            return true;
        }
    }
    return false;
}

vector<vector<int>> permutateVector4(vector<int> v){
/*Menghasilkan semua permutasi dari vector v yang berisi 4 buah
bilangan*/
    vector<vector<int>> result;
    for (int i = 0; i < v.size(); i++)
    {
        for (int j = 0; j < v.size(); j++)
        {
            for (int k = 0; k < v.size(); k++)
            {
                for (int l = 0; l < v.size(); l++)
                {
                    if (i != j && i != k && i != l && j != k && j !=
l && k != l)
                    {
                        vector<int> temp;
                        temp.push_back(v[i]);
                        temp.push_back(v[j]);
                        temp.push_back(v[k]);
                        temp.push_back(v[l]);
                        result.push_back(temp);
                    }
                }
            }
        }
    }
    return result;
}
```

3. cards.cpp

Modul untuk menjalankan operasi kartu yang diperlukan dalam 24 game solver

```
bool isValid(char c) {
//Mengecek apakah sebuah karakter merupakan kartu yang valid
    return (c == 'A' || c == '1' || c == '2' || c == '3' || c == '4'
|| c == '5' || c == '6' || c == '7' || c == '8'
    || c == '9' || c == 'J' || c == 'Q' || c == 'K');
}

int charToInt(char c) {
//Mengubah sebuah karakter kartu menjadi bilangan
    if (c == 'A') {
        return 1;
    } else if (c == 'J') {
        return 10;
    } else if (c == 'Q') {
        return 11;
    } else if (c == 'K') {
        return 12;
    } else {
        return c - '0';
    }
}

char intToChar(int i) {
//Mengubah sebuah bilangan kartu menjadi karakter
    if (i == 1) {
        return 'A';
    } else if (i == 10) {
        return 'J';
    } else if (i == 11) {
        return 'Q';
    } else if (i == 12) {
        return 'K';
    } else {
        return i + '0';
    }
}
```

4. main.cpp

Melakukan operasi-operasi vector dalam 24 game solver.

```
#include "./main.h"
#include "./vectoralgo.h"
#include "./cards.h"
#include <iostream>
```

```cpp
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <map>

using namespace std;
using namespace std::chrono;

int RESULT = 24;
vector<string> solutions;

double execTime;

int main(){
/* Program utama yang menjalankan 24 game solver */
    vector<int> nums = handleInputCards(); // Input kartu

    auto start_s = high_resolution_clock::now(); // Mulai timer
    vector<vector<int>> permutations = permutateVector4(nums); //
Permutasi kartu
    for (int i = 0; i < permutations.size(); i++){ // Evaluasi
permutasi kartu
        evaluate(permutations[i][0], permutations[i][1],
permutations[i][2], permutations[i][3]);
    }

    // Output
    if (solutions.size() == 0) { // Jika tidak ada solusi
        cout << "Tidak ada solusi." << endl;
    } else { // Jika ada solusi
        cout << "Solusi ditemukan: " << solutions.size() << endl;
        for (int i = 0; i < solutions.size(); i++) {
            cout << solutions[i] << endl;
        }
    }
    auto stop_s = high_resolution_clock::now(); // Stop timer

    // Hitung waktu eksekusi
    execTime = duration_cast<milliseconds>(stop_s - start_s).count();
    cout << endl << "Execution time: " << execTime << " ms" << endl;

    // Simpan ke file (jika diminta)
    handleFileSave(nums, solutions);

    return 0;
}
```

```
vector<int> handleInputCards(){
/* Mengembalikan input kartu, baik melalui input maupun random
generation */
    cout << endl << "Apakah anda ingin memasukkan input sendiri atau
generate secara random?" << endl << "1. Masukkan sendiri" << endl <<
"2. Generate random" << endl << endl;
    int input;
    cout << "Masukkan pilihan: ";
    cin >> input;
    if (input == 1){
        char a, b, c, d;
        cout << endl << "Masukkan 4 kartu yang ingin dihitung dengan
spasi sebagai pemisah: " << endl << "Contoh: A 3 4 Q" << endl <<
endl;
        cin >> a >> b >> c >> d; // Input kartu
        while (!isValid(a) || !isValid(b) || !isValid(c) ||
!isValid(d)){
            cout << "Input tidak valid! Silahkan ulangi!" << endl;
            cin >> a >> b >> c >> d; // Input kartu
        }
        vector<int> cards = {charToInt(a), charToInt(b),
charToInt(c), charToInt(d)};
        return cards;
    } else if (input == 2){
        srand(time(0));
        vector<int> cards = {rand() % 12 + 1, rand() % 12 + 1, rand()
% 12 + 1, rand() % 12 + 1}; // Generate kartu secara random
        cout << endl << "Kartu yang dihasilkan: " <<
intToChar(cards[0]) << " " <<
        intToChar(cards[1]) << " " << intToChar(cards[2]) << " " <<
intToChar(cards[3]) << endl << endl;
        return cards;
    } else {
        cout << "Input tidak valid! Silahkan ulangi!" << endl;
        return handleInputCards();
    }
}

void handleFileSave(vector<int> cards, vector<string> solutions){
/*I.S : Solusi dari 24 game solver sudah ditemukan
  F.S : Solusi disimpan (ataupun tidak) ke dalam file di folder
test*/
    cout << endl << "Apakah anda ingin menyimpan hasil ke file?" <<
endl << "1. Ya" << endl << "2. Tidak" << endl << endl;
    int input;
    cout << "Masukkan pilihan: ";
    cin >> input;
    if (input == 1){
```

```
        cout << "Masukkan nama file: "; // Input nama file
        string filename;
        cin >> filename;
        cout << endl;
        ofstream file;

        string filepath = "../test/" + filename + ".txt";

        file.open(filepath);

        // Tulis ke file
        file << "Kartu : " << intToChar(cards[0]) << " " <<
intToChar(cards[1]) << " " << intToChar(cards[2]) << " " <<
intToChar(cards[3]) << endl << endl;
        file << "Solusi ditemukan : " << solutions.size() << endl;
        for (int i = 0; i < solutions.size(); i++) {
            file << solutions[i] << endl;
        }
        file << endl << "Execution time : " << execTime << " ms" <<
endl;
        file.close();
        cout << "File berhasil disimpan di folder test" << endl;
    } else if (input == 2){
        cout << "File tidak disimpan" << endl;
    } else {
        cout << "Input tidak valid! Silahkan ulangi!" << endl;
        handleFileSave(cards, solutions);
    }
}

void evaluate(float a, float b, float c, float d) {
/*I.S : Input kartu sudah ada
  F.S : Evaluasi kombinasi kartu dengan segala kemungkinan permutasi
operator dan tanda kurung yang ada
        dan menyimpan solusi yang menghasilkan 24

  Kode pada prosedur ini digenerate oleh brutecoder.cpp*/

    char sol_char[50];
    string sol_str;

    // Hasil dari brutecoder.cpp
}
```

BAB IV : HASIL

1. Input sendiri dan benar

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 1

Masukkan 4 kartu yang ingin dihitung dengan spasi sebagai pemisah:
Contoh: A 3 4 Q

8 6 A A
Solusi ditemukan: 8
(8 * 6) / (1 + 1)
8 * (6 / (1 + 1))
(8 / (1 + 1)) * 6
8 / ((1 + 1) / 6)
(6 * 8) / (1 + 1)
6 * (8 / (1 + 1))
(6 / (1 + 1)) * 8
6 / ((1 + 1) / 8)

Execution time: 2 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 1
Masukkan nama file: test2

File berhasil disimpan di folder test
```

2. Input random dan benar

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 2

Kartu yang dihasilkan: 4 4 3 7

Solusi ditemukan: 20
((4 / 4) + 7) * 3
4 * ((3 - 4) + 7)
4 * (3 - (4 - 7))
4 * ((3 + 7) - 4)
4 * (3 + (7 - 4))
4 * ((7 - 4) + 3)
4 * (7 - (4 - 3))
4 * ((7 + 3) - 4)
4 * (7 + (3 - 4))
3 * ((4 / 4) + 7)
((3 - 4) + 7) * 4
```

```
(3 - (4 - 7)) * 4
((3 + 7) - 4) * 4
(3 + (7 - 4)) * 4
3 * (7 + (4 / 4))
(7 + (4 / 4)) * 3
((7 - 4) + 3) * 4
(7 - (4 - 3)) * 4
((7 + 3) - 4) * 4
(7 + (3 - 4)) * 4

Execution time: 4 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 1
Masukkan nama file: test1.txt

File berhasil disimpan di folder test
```

3. Input sendiri dan salah

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 1

Masukkan 4 kartu yang ingin dihitung dengan spasi sebagai pemisah:
Contoh: A 3 4 Q

R T 5 6
Input tidak valid! Silahkan ulangi!
5 6 7 8
Solusi ditemukan: 28
((5 + 7) - 8) * 6
(5 + (7 - 8)) * 6
(5 + 7) * (8 - 6)
((5 - 8) + 7) * 6
(5 - (8 - 7)) * 6
6 * ((5 + 7) - 8)
6 * (5 + (7 - 8))
6 * ((5 - 8) + 7)
6 * (5 - (8 - 7))
6 * ((7 + 5) - 8)
6 * (7 + (5 - 8))
(6 / (7 - 5)) * 8
6 / ((7 - 5) / 8)
6 * ((7 - 8) + 5)
6 * (7 - (8 - 5))
(6 * 8) / (7 - 5)
```

```
6 * (8 / (7 - 5))
((7 + 5) - 8) * 6
(7 + (5 - 8)) * 6
(7 + 5) * (8 - 6)
((7 - 8) + 5) * 6
(7 - (8 - 5)) * 6
(8 - 6) * (5 + 7)
(8 - 6) * (7 + 5)
(8 * 6) / (7 - 5)
8 * (6 / (7 - 5))
(8 / (7 - 5)) * 6
8 / ((7 - 5) / 6)

Execution time: 5 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 1
Masukkan nama file: test3.txt

File berhasil disimpan di folder test
```

4. Tidak ada solusi

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 1

Masukkan 4 kartu yang ingin dihitung dengan spasi sebagai pemisah:
Contoh: A 3 4 Q

A 2 A 3
Tidak ada solusi.

Execution time: 0 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 1
Masukkan nama file: test5

File berhasil disimpan di folder test
```

5. Tidak menyimpan ke file

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 1

Masukkan 4 kartu yang ingin dihitung dengan spasi sebagai pemisah:
Contoh: A 3 4 Q

3 2 A 2
Tidak ada solusi.

Execution time: 0 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 2
File tidak disimpan
```

6. Hasil banyak

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 1

Masukkan 4 kartu yang ingin dihitung dengan spasi sebagai pemisah:
Contoh: A 3 4 Q

K K 4 4
Solusi ditemukan: 77
(12 + 12) + (4 - 4)
((12 + 12) + 4) - 4
(12 + (12 + 4)) - 4
12 + ((12 + 4) - 4)
12 + (12 + (4 - 4))
((12 + 12) - 4) + 4
(12 + (12 - 4)) + 4
12 + ((12 - 4) + 4)
(12 + 12) - (4 - 4)
12 + (12 - (4 - 4))
(12 + 12) * (4 / 4)
((12 + 12) * 4) / 4
12 + ((12 * 4) / 4)
12 + (12 * (4 / 4))
((12 + 12) / 4) * 4
12 + ((12 / 4) * 4)
(12 + 12) / (4 / 4)
12 + (12 / (4 / 4))
(12 * (12 - 4)) / 4
12 * ((12 - 4) / 4)
```

```
(12 + 4) + (12 - 4)
((12 + 4) + 12) - 4
(12 + (4 + 12)) - 4
12 + ((4 + 12) - 4)
12 + (4 + (12 - 4))
12 + ((4 * 12) / 4)
12 + (4 * (12 / 4))
(12 - 4) + (12 + 4)
((12 - 4) + 12) + 4
(12 - (4 - 12)) + 4
12 - (4 - (12 + 4))
12 - ((4 - 12) - 4)
(12 - 4) * (12 / 4)
((12 - 4) * 12) / 4
(12 / 4) * (12 - 4)
12 / (4 / (12 - 4))
((12 + 4) - 4) + 12
(12 + (4 - 4)) + 12
12 + ((4 - 4) + 12)
(12 + 4) - (4 - 12)
12 + (4 - (4 - 12))
12 + ((4 / 4) * 12)
12 + (4 / (4 / 12))
(12 - 4) + (4 + 12)
((12 - 4) + 4) + 12
(12 - (4 - 4)) + 12
12 - (4 - (4 + 12))
12 - ((4 - 4) - 12)
((12 - 4) / 4) * 12
(12 - 4) / (4 / 12)
((12 * 4) / 4) + 12
(12 * (4 / 4)) + 12
((12 / 4) * 4) + 12
(12 / (4 / 4)) + 12
(4 + 12) + (12 - 4)
((4 + 12) + 12) - 4
(4 + (12 + 12)) - 4
4 + ((12 + 12) - 4)
4 + (12 + (12 - 4))
(4 * (12 + 12)) / 4
4 * ((12 + 12) / 4)
((4 + 12) - 4) + 12
(4 + (12 - 4)) + 12
4 + ((12 - 4) + 12)
(4 + 12) - (4 - 12)
4 + (12 - (4 - 12))
((4 * 12) / 4) + 12
(4 * (12 / 4)) + 12
(4 - 4) + (12 + 12)
((4 - 4) + 12) + 12
(4 - (4 - 12)) + 12
4 - (4 - (12 + 12))
```

```
4 - ((4 - 12) - 12)
(4 / 4) * (12 + 12)
((4 / 4) * 12) + 12
(4 / (4 / 12)) + 12
4 / (4 / (12 + 12))

Execution time: 19 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 1
Masukkan nama file: test4

File berhasil disimpan di folder test
```

7. Kartu semuanya sama (6 6 6 6)

```
Apakah anda ingin memasukkan input sendiri atau generate secara random?
1. Masukkan sendiri
2. Generate random

Masukkan pilihan: 1

Masukkan 4 kartu yang ingin dihitung dengan spasi sebagai pemisah:
Contoh: A 3 4 Q

6 6 6 6
Solusi ditemukan: 7
(6 + 6) + (6 + 6)
((6 + 6) + 6) + 6
(6 + (6 + 6)) + 6
6 + ((6 + 6) + 6)
6 + (6 + (6 + 6))
(6 * 6) - (6 + 6)
((6 * 6) - 6) - 6

Execution time: 2 ms

Apakah anda ingin menyimpan hasil ke file?
1. Ya
2. Tidak

Masukkan pilihan: 1
Masukkan nama file: test8

File berhasil disimpan di folder test
```

BAB V : DAFTAR PUSTAKA

1. Algoritma Brute Force (Bagian 1) [Online]. Rinaldi Munir. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf. Diakses pada 20 Januari 2023.
2. Tugas Kecil 1 IF2211 Strategi Algoritma 2023 [Online]. Rinaldi Munir. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf. Diakses pada 20 Januari 2023.

BAB IV : LAMPIRAN

1. Tautan Repository Github

Repository untuk source code program yang penulis buat dapat diakses pada: https://github.com/moonawar/Tucil1_13521085

2. Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan		
2. Program berhasil *running*		
3. Program dapat membaca input / generate sendiri dapat memberikan luaran		
4. Solusi yang diberikan program memenuhi (berhasil mencapai 24)		
5. Progran dapat menyimpan solusi dalam file teks		