Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II Tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh:

Ammar Rasyad Chaeroel 13521136

PROGRAM STUDI
TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO
DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2022

1. LATAR BELAKANG

Permainan kartu merupakan salah satu permainan yang terkenal dan banyak peminatnya. Ada beberapa jenis permainan kartu, seperti yang paling terkenal Poker. Pada makalah ini penulis akan membahas 24 Game, yang merupakan permainan kartu aritmatika dengan tujuan mencari jumlah dari 4 kartu yang menghasilkan jumlah 24. Diambil 4 kartu dari kartu remi yang terdiri dari 52 kartu (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King dengan 4 simbol berbeda).

Dalam permainan ini simbol (wajik, keriting, hati, sekop) tidak diperhitungkan, sehingga yang hanya diperlukan adalah nilai kartu yang didapatkan. As bernilai 1, Jack bernilai 11, Queen bernilai 12, dan yang terakhir King bernilai 13. Pada awal permainan deck akan dikocok dan pemain mengambil 4 kartu. Permainan berakhir ketika salah satu pemain berhasil menebak bagaimana dari 4 kartu tersebut bisa berjumlah 24. Pemain yang tercepat ialah pemenangnya.

Figure 1 Set kartu remi

2. TUJUAN

Tujuan penulis membuat makalah tentang penyelesaian 24 Game dengan Brute Force:

- Mengimplementasikan naïve algorithm (brute force) pada 24 Game
- Menambah wawasan tentang algoritma
- Mengerti permainan kartu 24 dan cara menyelesaikannya.

3. LANDASAN TEORI

3.1 Brute Force

Brute Force merupakan cara pendekatan naif dan lempang dalam memecahkan suatu masalah. Brute Force adalah algoritma yang sederhana dan jelas bagi siapapun yang membaca.

Brute Force disebut sebagai algoritma naif bukan tanpa alasan. Pendekatan algoritma ini cenderung tidak “pintar,”

karena seperti dari namanya algoritma brute force mencoba semua kemungkinan, sehingga membutuhkan langkah yang banyak dan besar dalam penyelesaiannya.

Namun demikian, algoritma brute force digunakan karena pendekatannya yang mudah, dan untuk masalah-masalah skala kecil implementasi algoritma brute force lebih mudah dibandingkan algoritma-algoritma lain. Itu merupakan salah satu kelebihan algoritma brute force. Kekurangannya adalah algoritma brute force cenderung lebih lambat dan cenderung kurang kreatif dibandingkan algoritma-algoritma lain.

3.2 Exhaustive Search

Exhaustive search adalah teknik pencarian solusi secara brute force untuk mencari solusi bagi sebuah masalah. Teknik ini biasanya menggunakan kombinatorika seperti permutasi, kombinasi, dsb. Cara digunakannya teknik ini pada permainan kartu 24:

- Mencari semua permutasi dari input kartu
- Evaluasi setiap kemungkinan solusi satu-per-satu, sesuai dengan implementasi
- Bila pencarian selesai, akan dipilih solusi terbaik

3.3 Permainan Kartu 24

Permainan kartu 24, seperti yang sudah dijelaskan sebelumnya, adalah permainan kartu yang dicampur dengan aritmatika untuk mencari bagaimana cara mendapatkan jumlah 24 dari 4 kartu yang diambil secara random. Aturan dari permainan ini sangat mudah, dengan menggunakan operasi dasar matematika (+, -, *, /, ()), pemain hanya butuh memanipulasi angka-angka dari kartu yang ia punya sehingga kartu-kartu tersebut mencapai nilai 24. Setiap kartu hanya bisa digunakan sekali. Jika tidak ada yang bisa menjawab, maka kartu-kartu akan dikembalikan ke deck dan dikocok ulang. Permainan berakhir ketika deck sudah habis dan pemain dengan kartu terbanyak adalah pemenangnya.

4. PENERAPAN ALGORITMA BRUTE FORCE

Pada implementasi algoritma ini, solusi dari kombinasi 4 kartu didapatkan dengan mencari semua permutasi dari 4 kartu tersebut terlebih dahulu. Dari 4 kartu akan didapatkan 24 kombinasi sesuai dengan rumus permutasi.

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

Setelah dicari semua kombinasi dari 4 kartu tersebut, lanjut ke bagian pencarian solusi. Penulis menggunakan contoh implementasi seperti:

(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)

(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))

op adalah 3 operasi matematika yang terpilih dari (+, -, *, /).

Evaluasi aritmatika tersebut akan dikumpul dalam set solusi untuk mencegah adanya duplikasi. Setelah semua solusi telah dikumpul, solusi akan ditampilkan di console atau disimpan ke file, sesuai dengan input dari pengguna. Waktu eksekusi akan ditampilkan di akhir program (waktu yang dihitungkan hanya bagian algoritma brute force, tidak termasuk bagian menyimpan ke file atau menampilkan solusi).

5. PENGUJIAN PROGRAM

Manual

```
Input manual atau random? (m/r): m
Masukkan 4 kartu (A, 2-10, J, Q, K): 5 5 5 5
Apakah ingin menyimpan hasil ke file? (y/n): n
(5*5)-(5/5)
(5*5)-5/5
Waktu eksekusi: 0.4009 ms
```

```
Input manual atau random? (m/r): m
Masukkan 4 kartu (A, 2-10, J, Q, K): 6 6 6 6
Apakah ingin menyimpan hasil ke file? (y/n): n
((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)
(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)
Waktu eksekusi: 0.6388 ms
```

```
Input manual atau random? (m/r): m
Masukkan 4 kartu (A, 2-10, J, Q, K): 8 7 3 9
Apakah ingin menyimpan hasil ke file? (y/n): n
((7+9)-8)*3
((7-8)+9)*3
((9+7)-8)*3
((9-8)+7)*3
(7+(9-8))*3
(7+9-8)*3
(7-(8-9))*3
(7-8+9)*3
(9+(7-8))*3
(9+7-8)*3
(9-(8-7))*3
(9-8+7)*3
3*((7+9)-8)
3*((7-8)+9)
3*((9+7)-8)
3*((9-8)+7)
3*(7+(9-8))
3*(7+9)-8
3*(7+9-8)
3*(7-(8-9))
3*(7-8)+9
3*(7-8+9)
3*(9+(7-8))
3*(9+7)-8
3*(9+7-8)
3*(9-(8-7))
3*(9-8)+7
3*(9-8+7)
3*7+(9-8)
3*7-(8-9)
3*9+(7-8)
3*9-(8-7)
Waktu eksekusi: 0.2346 ms
```

Random

```
Input manual atau random? (m/r): r
Nomor-nomor kartu: J 2 10 A
Apakah ingin menyimpan hasil ke file? (y/n): n
((1+10)+11)+2
((1+10)+2)+11
((1+11)+10)+2
((1+11)+2)+10
((1+2)+10)+11
((1+2)+11)+10
((10+1)+11)+2
((10+1)+2)+11
((10+11)+1)+2
((10+11)+2)+1
((10+2)+1)+11
((10+2)+11)+1
((11+1)+10)+2
((11+1)+2)+10
((11+10)+1)+2
((11+10)+2)+1
((11+2)+1)+10
```

Karena ada banyak solusi dan tidak muat, maka gambar hanya sebagian dari solusi.

```
Waktu eksekusi: 0.8962 ms
```

```
Input manual atau random? (m/r): r
Nomor-nomor kartu: 3 4 10 2
Apakah ingin menyimpan hasil ke file? (y/n): n
((10*3)-2)-4
((10*3)-4)-2
((10+2)-4)*3
((10-4)+2)*3
((2+10)-4)*3
((2-4)+10)*3
((3*10)-2)-4
((3*10)-4)-2
((3*4)+10)+2
((3*4)+2)+10
((3+4)*2)+10
((4*3)+10)+2
((4*3)+2)+10
((4+3)*2)+10
(10*3)-(2+4)
(10*3)-(4+2)
(10*3)-2+4
(10*3)-4+2
(10*3-2)-4
(10*3-4)-2
(10+(2-4))*3
(10+(3*4))+2
(10+(4*3))+2
(10+2)+(3*4)
(10+2)+(4*3)
```

...

```
Waktu eksekusi: 0.4376 ms
```

```
Input manual atau random? (m/r): r
Nomor-nomor kartu: A K 7 4
Apakah ingin menyimpan hasil ke file? (y/n): n
((1*13)+4)+7
((1*13)+7)+4
((1*13)-7)*4
((1*4)+13)+7
((1*4)+7)+13
((1*7)+13)+4
((1*7)+4)+13
((13*1)+4)+7
((13*1)+7)+4
((13*1)-7)*4
((13+4)*1)+7
((13+4)+7)*1
((13+4)+7)/1
((13+4)/1)+7
((13+7)*1)+4
((13+7)+4)*1
((13+7)+4)/1
((13+7)/1)+4
((13-7)*1)*4
((13-7)*4)*1
((13-7)*4)/1
((13-7)/1)*4
((13/1)+4)+7
```

…

```
Waktu eksekusi: 0.8854 ms
```

Jika tidak ada solusi:

```
Input manual atau random? (m/r): m
Masukkan 4 kartu (A, 2-10, J, Q, K): 6 6 K 4
Apakah ingin menyimpan hasil ke file? (y/n): y
Tidak ada solusi.
Solusi kosong maka tidak akan di simpan ke file.
Waktu eksekusi: 0.3437 ms
```

Untuk output file:

Untuk input: 1 13 7 4

Ada 424 solusi:
((1*13)+4)+7
((1*13)+7)+4
((1*13)-7)*4
((1*4)+13)+7
((1*4)+7)+13

((1*7)+13)+4
((1*7)+4)+13
((13*1)+4)+7
((13*1)+7)+4
((13*1)-7)*4
((13+4)*1)+7
((13+4)+7)*1
((13+4)+7)/1
((13+4)/1)+7
((13+7)*1)+4
((13+7)+4)*1
((13+7)+4)/1
((13+7)/1)+4
((13-7)*1)*4
((13-7)*4)*1
((13-7)*4)/1
((13-7)/1)*4
((13/1)+4)+7
((13/1)+7)+4
((13/1)-7)*4
((4*1)+13)+7
((4*1)+7)+13
((4+13)*1)+7
((4+13)+7)*1
((4+13)+7)/1
((4+13)/1)+7
((4+7)*1)+13
((4+7)+13)*1
((4+7)+13)/1
((4+7)/1)+13
((4/1)+13)+7
((4/1)+7)+13
((7*1)+13)+4
((7*1)+4)+13
((7+13)*1)+4
((7+13)+4)*1
((7+13)+4)/1
((7+13)/1)+4
((7+4)*1)+13
((7+4)+13)*1
((7+4)+13)/1
((7+4)/1)+13
((7/1)+13)+4
((7/1)+4)+13
(1*(13+4))+7

(1*(13+7))+4
(1*(13-7))*4
(1*(4+13))+7
(1*(4+7))+13
(1*(7+13))+4
(1*(7+4))+13
(1*13)+(4+7)
(1*13)+(7+4)
(1*13)+4+7
(1*13)+7+4
(1*13+4)+7
(1*13+7)+4
(1*13-7)*4
(1*4)*(13-7)
(1*4)*13-7
(1*4)+(13+7)
(1*4)+(7+13)
(1*4)+13+7
(1*4)+7+13
(1*4+13)+7
(1*4+7)+13
(1*7)+(13+4)
(1*7)+(4+13)
(1*7)+13+4
(1*7)+4+13
(1*7+13)+4
(1*7+4)+13
(13*1)+(4+7)
(13*1)+(7+4)
(13*1)+4+7
(13*1)+7+4
(13*1+4)+7
(13*1+7)+4
(13*1-7)*4
(13+(1*4))+7
(13+(1*7))+4
(13+(4*1))+7
(13+(4+7))*1
(13+(4+7))/1
(13+(4/1))+7
(13+(7*1))+4
(13+(7+4))*1
(13+(7+4))/1
(13+(7/1))+4

(13+4)+(1*7)
(13+4)+(7*1)
(13+4)+(7/1)
(13+4)+1*7
(13+4)+7*1
(13+4)+7/1
(13+4*1)+7
(13+4+7)*1
(13+4+7)/1
(13+4/1)+7
(13+7)+(1*4)
(13+7)+(4*1)
(13+7)+(4/1)
(13+7)+1*4
(13+7)+4*1
(13+7)+4/1
(13+7*1)+4
(13+7+4)*1
(13+7+4)/1
(13+7/1)+4
(13-(1*7))*4
(13-(7*1))*4
(13-(7/1))*4
(13-7)*(1*4)
(13-7)*(4*1)
(13-7)*(4/1)
(13-7)*1*4
(13-7)*4*1
(13-7)*4/1
(13-7)/(1/4)
(13-7)/1/4
(13-7*1)*4
(13-7*4)*1
(13-7*4)/1
(13-7/1)*4
(13/1)+(4+7)
(13/1)+(7+4)
(13/1)+4+7
(13/1)+7+4
(13/1+4)+7
(13/1+7)+4
(13/1-7)*4
(4*(13-7))*1
(4*(13-7))/1
(4*1)*(13-7)

(4*1)*13-7
(4*1)+(13+7)
(4*1)+(7+13)
(4*1)+13+7
(4*1)+7+13
(4*1+13)+7
(4*1+7)+13
(4+(1*13))+7
(4+(1*7))+13
(4+(13*1))+7
(4+(13+7))*1
(4+(13+7))/1
(4+(13/1))+7
(4+(7*1))+13
(4+(7+13))*1
(4+(7+13))/1
(4+(7/1))+13
(4+13)+(1*7)
(4+13)+(7*1)
(4+13)+(7/1)
(4+13)+1*7
(4+13)+7*1
(4+13)+7/1
(4+13*1)+7
(4+13+7)*1
(4+13+7)/1
(4+13/1)+7
(4+7)+(1*13)
(4+7)+(13*1)
(4+7)+(13/1)
(4+7)+1*13
(4+7)+13*1
(4+7)+13/1
(4+7*1)+13
(4+7+13)*1
(4+7+13)/1
(4+7/1)+13
(4/1)*(13-7)
(4/1)*13-7
(4/1)+(13+7)
(4/1)+(7+13)
(4/1)+13+7
(4/1)+7+13
(4/1+13)+7

(4/1+7)+13
(7*1)+(13+4)
(7*1)+(4+13)
(7*1)+13+4
(7*1)+4+13
(7*1+13)+4
(7*1+4)+13
(7+(1*13))+4
(7+(1*4))+13
(7+(13*1))+4
(7+(13+4))*1
(7+(13+4))/1
(7+(13/1))+4
(7+(4*1))+13
(7+(4+13))*1
(7+(4+13))/1
(7+(4/1))+13
(7+13)+(1*4)
(7+13)+(4*1)
(7+13)+(4/1)
(7+13)+1*4
(7+13)+4*1
(7+13)+4/1
(7+13*1)+4
(7+13+4)*1
(7+13+4)/1
(7+13/1)+4
(7+4)+(1*13)
(7+4)+(13*1)
(7+4)+(13/1)
(7+4)+1*13
(7+4)+13*1
(7+4)+13/1
(7+4*1)+13
(7+4+13)*1
(7+4+13)/1
(7+4/1)+13
(7/1)+(13+4)
(7/1)+(4+13)
(7/1)+13+4
(7/1)+4+13
(7/1+13)+4
(7/1+4)+13
1*((13+4)+7)
1*((13+7)+4)

1*((13-7)*4)
1*((4+13)+7)
1*((4+7)+13)
1*((7+13)+4)
1*((7+4)+13)
1*(13+(4+7))
1*(13+(7+4))
1*(13+4)+7
1*(13+4+7)
1*(13+7)+4
1*(13+7+4)
1*(13-7)*4
1*(13-7*4)
1*(4*(13-7))
1*(4+(13+7))
1*(4+(7+13))
1*(4+13)+7
1*(4+13+7)
1*(4+7)+13
1*(4+7+13)
1*(7+(13+4))
1*(7+(4+13))
1*(7+13)+4
1*(7+13+4)
1*(7+4)+13
1*(7+4+13)
1*13+(4+7)
1*13+(7+4)
1*4*(13-7)
1*4+(13+7)
1*4+(7+13)
1*7+(13+4)
1*7+(4+13)
13+((1*4)+7)
13+((1*7)+4)
13+((4*1)+7)
13+((4+7)*1)
13+((4+7)/1)
13+((4/1)+7)
13+((7*1)+4)
13+((7+4)*1)
13+((7+4)/1)
13+((7/1)+4)
13+(1*(4+7))

13+(1*(7+4))
13+(1*4)+7
13+(1*4+7)
13+(1*7)+4
13+(1*7+4)
13+(4*1)+7
13+(4*1+7)
13+(4+(1*7))
13+(4+(7*1))
13+(4+(7/1))
13+(4+7)*1
13+(4+7)/1
13+(4+7*1)
13+(4+7/1)
13+(4/1)+7
13+(4/1+7)
13+(7*1)+4
13+(7*1+4)
13+(7+(1*4))
13+(7+(4*1))
13+(7+(4/1))
13+(7+4)*1
13+(7+4)/1
13+(7+4*1)
13+(7+4/1)
13+(7/1)+4
13+(7/1+4)
13+1*(4+7)
13+1*(7+4)
13+4+(1*7)
13+4+(7*1)
13+4+(7/1)
13+7+(1*4)
13+7+(4*1)
13+7+(4/1)
4*((1*13)-7)
4*((13*1)-7)
4*((13-7)*1)
4*((13-7)/1)
4*((13/1)-7)
4*(1*(13-7))
4*(1*13)-7
4*(1*13-7)
4*(13*1)-7
4*(13*1-7)

4*(13-(1*7))
4*(13-(7*1))
4*(13-(7/1))
4*(13-7)*1
4*(13-7)/1
4*(13-7*1)
4*(13-7/1)
4*(13/1)-7
4*(13/1-7)
4*1*(13-7)
4*13-(1*7)
4*13-(7*1)
4*13-(7/1)
4+((1*13)+7)
4+((1*7)+13)
4+((13*1)+7)
4+((13+7)*1)
4+((13+7)/1)
4+((13/1)+7)
4+((7*1)+13)
4+((7+13)*1)
4+((7+13)/1)
4+((7/1)+13)
4+(1*(13+7))
4+(1*(7+13))
4+(1*13)+7
4+(1*13+7)
4+(1*7)+13
4+(1*7+13)
4+(13*1)+7
4+(13*1+7)
4+(13+(1*7))
4+(13+(7*1))
4+(13+(7/1))
4+(13+7)*1
4+(13+7)/1
4+(13+7*1)
4+(13+7/1)
4+(13/1)+7
4+(13/1+7)
4+(7*1)+13
4+(7*1+13)
4+(7+(1*13))
4+(7+(13*1))

4+(7+(13/1))
4+(7+13)*1
4+(7+13)/1
4+(7+13*1)
4+(7+13/1)
4+(7/1)+13
4+(7/1+13)
4+1*(13+7)
4+1*(7+13)
4+13+(1*7)
4+13+(7*1)
4+13+(7/1)
4+7+(1*13)
4+7+(13*1)
4+7+(13/1)
4/(1/(13-7))
4/1/(13-7)
7+((1*13)+4)
7+((1*4)+13)
7+((13*1)+4)
7+((13+4)*1)
7+((13+4)/1)
7+((13/1)+4)
7+((4*1)+13)
7+((4+13)*1)
7+((4+13)/1)
7+((4/1)+13)
7+(1*(13+4))
7+(1*(4+13))
7+(1*13)+4
7+(1*13+4)
7+(1*4)+13
7+(1*4+13)
7+(13*1)+4
7+(13*1+4)
7+(13+(1*4))
7+(13+(4*1))
7+(13+(4/1))
7+(13+4)*1
7+(13+4)/1
7+(13+4*1)
7+(13+4/1)
7+(13/1)+4
7+(13/1+4)
7+(4*1)+13

7+(4*1+13)
7+(4+(1*13))
7+(4+(13*1))
7+(4+(13/1))
7+(4+13)*1
7+(4+13)/1
7+(4+13*1)
7+(4+13/1)
7+(4/1)+13
7+(4/1+13)
7+1*(13+4)
7+1*(4+13)
7+13+(1*4)
7+13+(4*1)
7+13+(4/1)
7+4+(1*13)
7+4+(13*1)
7+4+(13/1)

Waktu eksekusi: 0.8907 ms

Source Code

main.cpp

```
#include <iostream>
#include <chrono>
#include "inputhandler.h"
#include "solver.h"

using namespace std;

int main() {
    // Step 1: Input
    vector<int> out;
    fill_array(out);

    // Step 2: Ask if the user wants to save to file
    bool to_file;
    char dump;
    cout << "Apakah ingin menyimpan hasil ke file? (y/n): ";
    cin >> dump;
    if (dump == 'y') {
        to_file = true;
    } else {
        if (dump != 'n') {
            cout << "Input tidak valid. Program akan tidak akan
save file." << endl;
        }
        to_file = false;
    }

    // Step 3: Solve
```

```
    auto start = chrono::high_resolution_clock::now();
    solve(out);
    auto duration = chrono::duration<double,
milli>(chrono::high_resolution_clock::now() - start);

    if (!to_file) {
        print_solutions();
    }
    save(out, duration);

    cout << "Waktu eksekusi: " << duration.count() << " ms" <<
endl;
    return 0;
}
```

inputhandler.h

```
#ifndef STIMATUCIL1_INPUTHANDLER_H
#define STIMATUCIL1_INPUTHANDLER_H

#include <sstream>
#include <vector>

using namespace std;

#define ACE 1
#define JACK 11
#define QUEEN 12
#define KING 13

/*
 * String to integer array. Converts a string to an array of
integers.
 * It is ensured that the output always contains 4 integers.
 * Could just use a single cin and then split the string, but
this is more fun.
 * @param out The array to store the integers in.
 */
int stoia(vector<int>& out) {
    string input;
    getline(cin >> ws, input);
    string const temp = input;
    for (auto c : temp) {
        switch (toupper(c)) {
            case 'A':
                input.replace(input.find(c), 1, "1");
                break;
            case 'J':
                input.replace(input.find(c), 1, "11");
                break;
            case 'Q':
                input.replace(input.find(c), 1, "12");
                break;
            case 'K':
                input.replace(input.find(c), 1, "13");
                break;
            default:
                continue;
        }
    }
```

```
    istringstream iss(input);
    for (int i = 0; i < 4; i++) {
        int t;
        iss >> t;
        if (t < 1 || t > 13) {
            cerr << "Masukan tidak sesuai. Hanya menerima (A, 2-
10, J, Q, K)." << endl;
            return 1;
        }
        out.push_back(t);
    }
    return 0;
}

ostream& operator<<(ostream& out, const vector<int>& cards) {
    for (auto& i : cards) {
        switch (i) {
            case ACE:
                out << "A ";
                break;
            case JACK:
                out << "J ";
                break;
            case QUEEN:
                out << "Q ";
                break;
            case KING:
                out << "K ";
                break;
            default:
                out << i << " ";
                break;
        }
    }
    return out;
}

/*
 * Checks whether the user would like to input cards manually or
randomly.
 */
void fill_array(vector<int>& out) {
    bool valid = false;
    do {
        cout << "Input manual atau random? (m/r): ";
        string input;
        getline(cin >> ws, input);
        transform(input.begin(), input.end(),
input.begin(), ::tolower);
        if (input == "m") {
            int result;
            cout << "Masukkan 4 kartu (A, 2-10, J, Q, K): ";
            do {
                result = stoia(out);
            } while (result != 0);
            valid = true;
        } else if (input == "r") {
            unsigned int card;
```

```
            __builtin_ia32_rdrand32_step(&card);
            for (int i = 0; i < 4; i++) {
                out.push_back((card % 13) + 1);
                __builtin_ia32_rdrand32_step(&card);
            }
            valid = true;
            cout << "Nomor-nomor kartu: " << out << endl;
        } else {
            cout << "Masukan tidak sesuai" << endl;
        }
    } while (!valid);
}

#endif
```

solver.cpp

```
//
// Created by ammar on 19/01/2023.
//

#include <fstream>
#include <set>
#include <chrono>
#include "solver.h"

const vector<char> operators = {'+', '-', '*', '/'};
set<string> solutions;

/*
 * Evaluates an expression.
 * @param a The first operand.
 * @param op The operator.
 * @param b The second operand.
 */
float operate(const float& a, const char& op, const float& b) {
    switch (op) {
        case '+':
            return a + b;
        case '-':
            return a - b;
        case '*':
            return a * b;
        case '/':
            return a / b;
        default:
            return 0;
    }
}

/*
 * Solves the 24 game.
 * @param cards The input cards.
 * @param to_file Whether to save the output to a file.
 */
void solve(vector<int>& cards) {
    vector<vector<int>> permutations =
get_all_permutations(cards);
    for (auto& p : operators) {
        for (auto& q : operators) {
```

```
            for (auto& r : operators) {
                for (auto& perm : permutations) {
                    int &a = perm[0];
                    int &b = perm[1];
                    int &c = perm[2];
                    int &d = perm[3];

                    // (a b) c d
                    float result = operate(operate(a, p, b), q,
operate(c, r, d));
                    if (result == TARGET) {
                        solutions.insert("(" + to_string(a) + p
+ to_string(b) + ")" + q + to_string(c) + r + to_string(d));
                    }
                    // a (b c) d
                    result = operate(a, p, operate(operate(b, q,
c), r, d));
                    if (result == TARGET) {
                        solutions.insert(to_string(a) + p + "("
+ to_string(b) + q + to_string(c) + ")" + r + to_string(d));
                    }
                    // a b (c d)
                    result = operate(a, p, operate(b, q,
operate(c, r, d)));
                    if (result == TARGET) {
                        solutions.insert(to_string(a) + p +
to_string(b) + q + "(" + to_string(c) + r + to_string(d) + ")");
                    }
                    // (a b c) d
                    result = operate(operate(operate(a, p, b),
q, c), r, d);
                    if (result == TARGET) {
                        solutions.insert("(" + to_string(a) + p
+ to_string(b) + q + to_string(c) + ")" + r + to_string(d));
                    }
                    // a (b c d)
                    result = operate(a, p, operate(operate(b, q,
c), r, d));
                    if (result == TARGET) {
                        solutions.insert(to_string(a) + p + "("
+ to_string(b) + q + to_string(c) + r + to_string(d) + ")");
                    }
                    // (a b) (c d)
                    result = operate(operate(a, p, b), q,
operate(c, r, d));
                    if (result == TARGET) {
                        solutions.insert("(" + to_string(a) + p
+ to_string(b) + ")" + q + "(" + to_string(c) + r + to_string(d)
+ ")");
                    }
                    // ((a b) c) d
                    result = operate(operate(operate(a, p, b),
q, c), r, d);
                    if (result == TARGET) {
                        solutions.insert("((" + to_string(a) + p
+ to_string(b) + ")" + q + to_string(c) + ")" + r +
to_string(d));
                    }
```

```
                    // (a (b c)) d
                    result = operate(operate(a, p, operate(b, q,
c)), r, d);
                    if (result == TARGET) {
                        solutions.insert("(" + to_string(a) + p
+ "(" + to_string(b) + q + to_string(c) + "))" + r +
                                         to_string(d));
                    }
                    // a ((b c) d)
                    result = operate(a, p, operate(operate(b, q,
c), r, d));
                    if (result == TARGET) {
                        solutions.insert(to_string(a) + p + "(("
+ to_string(b) + q + to_string(c) + ")" + r + to_string(d) +
"))");
                    }
                    // a (b (c d))
                    result = operate(a, p, operate(b, q,
operate(c, r, d)));
                    if (result == TARGET) {
                        solutions.insert( to_string(a) + p + "("
+ to_string(b) + q + "(" + to_string(c) + r + to_string(d) +
"))");
                    }
                }
            }
        }
    }
    if (solutions.empty()) {
        cout << "Tidak ada solusi.";
        cout << endl;
        return;
    }
//    if (to_file) {
//        try {
//            ofstream file("solutions.txt");
//            if (file.is_open()) {
//                file << "Untuk input: " << cards[0] << " " <<
cards[1] << " " << cards[2] << " " << cards[3] << endl << endl;
//                file << "Ada " << solutions.size() << "
solusi:" << endl;
//                for (auto& s : solutions) {
//                    file << s << endl;
//                }
//                file.close();
//            } else {
//                cout << "File handle tidak bisa dibuka." <<
endl;
//            }
//        } catch (const exception& e) {
//            cerr << "Error: " << e.what() << endl;
//        }
//    }
}

void print_solutions() {
    for (auto& s : solutions) {
        cout << s << endl;
    }
```

```
}

void save(const vector<int>& cards, const
chrono::duration<double, milli>& duration) {
    if (!solutions.empty()) {
        try {
            ofstream file("solutions.txt");
            if (file.is_open()) {
                file << "Untuk input: " << cards[0] << " " <<
cards[1] << " " << cards[2] << " " << cards[3] << endl
                     << endl;
                file << "Ada " << solutions.size() << " solusi:"
<< endl;
                for (auto &s: solutions) {
                    file << s << endl;
                }
                file << endl;
                file << "Waktu eksekusi: " << duration.count()
<< " ms" << endl;
                file.close();
            } else {
                cout << "File handle tidak bisa dibuka." <<
endl;
            }
        } catch (const exception &e) {
            cerr << "Error: " << e.what() << endl;
        }
    } else {
        cout << "Solusi kosong maka tidak akan di simpan ke
file." << endl;
    }
}

/*
 * Swaps the values of two integers.
 * @param a The first integer
 * @param b The second integer
 */
void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

/*
 * Helper function to do permutations from a given array of
cards.
 * @param result The multidimensional vector to store the
permutations
 * @param cards The array of cards
 * @param start The starting index of the array
 * @param end The ending index of the array
 */
void permutations(vector<vector<int>>& result, vector<int>&
cards, int start, int end) {
    if (start == end) {
        result.push_back(cards);
    } else {
```

```
        for (int i = start; i <= end; i++) {
            swap(cards[start], cards[i]);
            permutations(result, cards, start + 1, end);
            swap(cards[start], cards[i]);
        }
    }
}

/*
 * Get all permutations from a given array of cards.
 * @param cards The set of cards
 */
vector<vector<int>> get_all_permutations(vector<int>& cards) {
    vector<vector<int>> result;
    permutations(result, cards, 0, cards.size() - 1);
    return result;
}
```

solver.h

```
#ifndef TUCIL1_13521136_SOLVER_H
#define TUCIL1_13521136_SOLVER_H

#include <iostream>
#include <vector>

using namespace std;

#define TARGET 24

void permutations(vector<vector<int>>& result, vector<int>&
cards, int start, int end);
vector<vector<int>> get_all_permutations(vector<int>& cards);
void solve(vector<int>& permutation);
void save(const vector<int>& cards, const
chrono::duration<double, milli>& duration);
void print_solutions();

#endif
```

Link Repo GitHub
https://github.com/ammarasyad/Tucil1_13521136

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	✓	