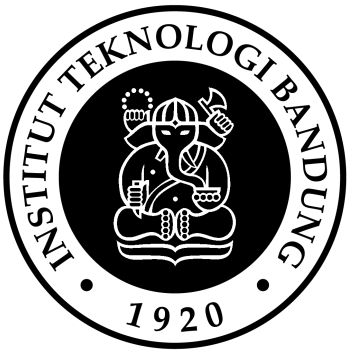
IF2211 Strategi Algoritma

IMPLEMENTASI ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN PERMAINAN KARTU 24

Laporan Tugas Kecil I

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma
pada Semester II (dua) Tahun Akademik 2022/2023.



Oleh

Alisha Listya Wardhani 13521171

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Permainan kartu 24 merupakan permainan kartu aritmatika dengan tujuan untuk mencari kombinasi yang tepat sehingga 4 buah angka pada kartu jika dioperasikan akan menghasilkan angka 24. Permainan ini dimainkan dengan kartu remi, yang terdiri dari kartu AS, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, dan K. Pada sebuah dek kartu remi, terdapat 4 simbol yang berbeda (hati, sekop, keriting, dan wajik) sehingga memungkinkan 4 angka yang sama untuk muncul. Nilai pada setiap kartu bergantung pada angka yang disimbolkan pada kartu. Aturan tambahan meliputi: AS bernilai 1, J bernilai 11, Q bernilai 12, dan K bernilai 13. Permainan dimulai ketika 4 kartu sudah dipilih. Nilai pada setiap kartu tersebut dimanipulasi sedemikian sehingga nilai akhirnya adalah 24. Pengubahan nilai dapat dilakukan dengan operator penjumlahan (+), pengurangan (-), pembagian (/), ataupun perkalian (*).

Pada makalah ini, dirancang sebuah algoritma berbasis brute force untuk menemukan solusi dari 4 kartu yang diberikan pada permainan kartu 24. Program yang diciptakan meliputi masukkan nilai (kartu), solusi yang ditemukan, dan waktu eksekusi program. Program yang diciptakan juga diharapkan dapat menyimpan solusi dalam file teks.

BAB II
TEORI SINGKAT

Algoritma *brute force* merupakan algoritma dengan pendekatan yang lempang dalam menyelesaikan suatu permasalahan. Metode yang digunakan pada algoritma ini adalah *exhaustion*, yaitu iterasi terhadap semua pilihan jawaban hingga ditemukan jawaban yang memenuhi solusi permasalahan. Kompleksitas waktu algoritma brute force sering kali proporsional dengan jumlah masukkan. Oleh karena itu, algoritma ini sering diklasifikasikan sebagai algoritma yang tidak mangkus, dengan O(n) lebih buruk dibandingkan polinomial.

Ciri khas algoritma dengan pendekatan *brute force* yaitu memiliki konsep penyelesaian yang sederhana, langsung, jelas, dan intuitif. Algoritma ini merupakan algoritma dasar yang dapat menyelesaikan hampir semua permasalahan. Jika sebuah solusi ada di dalam koleksi, maka algoritma ini dapat menanganinya walaupun biaya yang dibutuhkan sebanding dengan jumlah kandidat solusi. Pada dunia nyata, karena banyak permasalahan yang mempunyai jumlah kemungkinan yang terlalu tinggi, metode brute-force diimplementasikan secara heuristik khusus sampai kepada jumlah yang dapat ditangani.

BAB III
RANCANGAN DAN IMPLEMENTASI PROGRAM

3.1. Rancangan Algoritma

Dalam menyelesaikan permainan kartu 24 secara algoritmik, digunakan pendekatan brute force. Adapun langkah-langkah penyelesaian permasalahan dalam algoritma dijelaskan sebagai berikut:

1. Siapkan 4 buah kartu yang masing-masih merepresentasikan sebuah nilai.

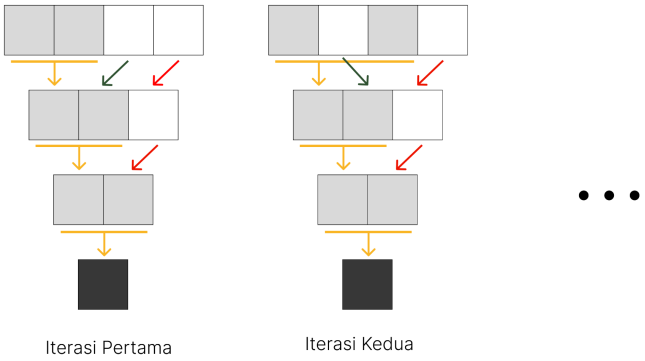
   Kartu dapat berasal dari masukan manual pengguna ataupun dibangkitkan secara acak oleh program. Jika masukkan kartu tidak sesuai dengan ketentuan, maka akan diminta sampai semua ketentuan terpenuhi.
2. Tampung input pada struktur data larik agar dapat diakses oleh program.

   Struktur data yang digunakan pada program ini merupakan himpunan dan larik (senarai) dengan jumlah masing-masing 4, 3, 2, dan 1. Himpunan digunakan untuk menyimpan solusi unik yang memenuhi nilai 24. Terdapat dua larik yang digunakan pada algoritma ini, larik berisi pengolahan nilai dan larik berisikan pengolahan ekspresi matematika.
3. Untuk menemukan solusi, dilakukan permutasi dari setiap larik.

   A) Iterasi kemungkinan operator sebanyak 4*4*4 = 64 kemungkinan.

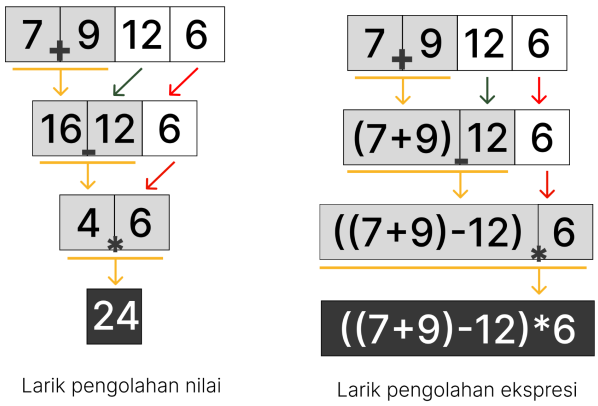
   B) Iterasi kemungkinan penempatan angka sebanyak 4*3*2*1 = 24 kemungkinan

   Setelah dilakukan iterasi semua kemungkinan operator dan penempatan angka, dilakukan langkah-langkah sebagai berikut.



(Gambar 3.1 Iterasi Penyelesaian)

Dari setiap larik dengan panjang n, diambil 2 elemen sembarang dan operasikan elemen tersebut. Hasil dari operasi tersebut dan elemen yang tidak dioperasikan akan dimasukkan ke larik dengan panjang (n-1). Lakukanlah langkah ini sampai larik memiliki panjang 1.



(Gambar 3.1 Larik Pengolahan Penyelesaian)

4. Jika larik sudah memiliki panjang 1, identifikasi apakah hasil akhirnya sama dengan 24. Pada program ini, karena dipakai real division, maka penulis memilih untuk menyediakan range nilai akhir yaitu $23.9 < N < 24.1$, maka jika nilainya 24.001 akan termasuk jawaban yang benar. Ekspresi dari solusi tersebut kemudian dimasukkan ke dalam struktur data himpunan. Jika hasilnya tidak sama dengan 24, maka melanjutkan iterasi selanjutnya.
5. Iterasi dilakukan sampai semua kemungkinan telah dicek. Jika sudah, keluarkan solusi tersebut ke terminal.

3.2. Implementasi Program dalam Bahasa C++

Dalam pembuatan program ini, penulis menggunakan bahasa pemrograman C++. Struktur program ini terdiri dari 4 file: cards.cpp, decorators.cpp, operators.cpp, dan main.cpp.

3.2.1. Cards.cpp

File ini berisi fungsi-fungsi yang berkaitan dengan pemilihan dan validasi kartu. Berikut merupakan deskripsi dari setiap fungsi yang terdapat pada file ini.

Nama Fungsi/Prosedur	Deskripsi
bool splitCards (string str, char sep, string cardArr[4])	Prosedur ini merupakan fungsi untuk menghasilkan array berisi string nama kartu, misalnya A, 2, 3, J, Q, ataupun K.

bool validateCards (string cardArr[4], int numArr[4])	Fungsi ini mengembalikan value berupa true/false tergantung kartu yang dimasukkan sesuatu dengan ketentuan.
string numToCards (int numCard)	Prosedur ini mengembalikan simbol yang berpadanan dengan nilai pada kartu. Misalnya jika kartu bernilai 1 maka akan menghasilkan “A”.
void randCards (int ArrNum[4])	Prosedur ini membangkitkan kartu acak dengan ketentuan sesuai dengan kartu permainan (remi). I.S. ArrNum merupakan larik terdefinisi sembarang. F.S. ArrNum berisikan 4 elemen dengan nilai sembarang dari 1 sampai 13.
void inputCards (int ArrNum[4])	Prosedur ini menerima masukkan kartu dari pengguna berupa string. Pada prosedur ini juga dicek ketentuan sesuai dengan kartu menggunakan fungsi validateCards. I.S. ArrNum merupakan larik terdefinisi sembarang. F.S. ArrNum berisikan 4 elemen dengan nilai sembarang dari 1 sampai 13.

3.2.2. Decorators.cpp

File ini berisi fungsi-fungsi yang berkaitan dengan dekorasi dan splash screen. Berikut merupakan deskripsi dari setiap fungsi yang terdapat pada file ini.

Nama Fungsi/Prosedur	Deskripsi
void printAscii()	Prosedur ini merupakan prosedur untuk menampilkan splash screen.
void printCards (string card1, string card2, string card3, string card4)	Prosedur ini merupakan prosedur yang menampilkan gambar dari kartu-kartu yang telah dipilih.
void printCardsClosed()	Prosedur ini merupakan prosedur untuk menampilkan ASCII Art berupa gambar kartu dengan kartu tertutup.

3.2.3. Operators.cpp

File ini berisi fungsi-fungsi yang berkaitan . Berikut merupakan deskripsi dari setiap fungsi yang terdapat pada file ini.

Nama Fungsi/Prosedur	Deskripsi
void writeSolutions(set <string> StrRes, string filename)	Prosedur ini merupakan prosedur yang menulis semua solusi yang memenuhi permainan kartu 24 ke dalam sebuah file bernama filename.

double solvOps(char ops, double a, double b)	Fungsi ini merupakan fungsi yang menyelesaikan sebuah operasi aritmetika jika diberikan operator ops, nilai pertama a, dan nilai kedua b.
int solvCards(int ArrCards[4];	Prosedur ini merupakan prosedur algoritma brute force yang mencari solusi permainan 24 Card-Solver.

3.3. Source Code Program

decorators.cpp

```
#include <iostream>
using namespace std;

void printAscii(){
   cout << "\n ================================================================================ \n\n";
   cout << "     ____  _  _        ____                _   ____        _                \n"  ;
   cout << "    |___ \\| || |     / ___|__ _ _ __ __| | / ___|  ___ | |_  _____ _ __ \n";
   cout << "      __) | || |_   | |   / _` | '__/ _` | \\___\\ / _\\|\\\\ / / _\\ '__| \n";
   cout << "    / __/|__   _|   | |__| (_| | | | (_| |  ___) | (_) | |\\ V /  __/ |   \n";
   cout << "    |_____|  |_|     \\____\\__,_|_|  \\__,_| |____/ \\___/|_| \\_/ \\___|_| \n";
   cout << "\n ================================================================================ \n\n";
}
void printCards(string card1, string card2, string card3, string card4){
   printf( "┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐    \n"
           "|%s       |  |%s       |  |%s       |  |%s       |   \n"
           "|         |  |         |  |         |  |         |    \n"
           "|         |  |         |  |         |  |         |     \n"
           "|    ♥    |  |    ♠    |  |    ♦    |  |    ♣    |    \n"
           "|         |  |         |  |         |  |         |     \n"
           "|         |  |         |  |         |  |         |     \n"
           "|       %s|  |       %s|  |       %s|  |       %s|     \n"
           "└─────────┘  └─────────┘  └─────────┘  └─────────┘     \n", card1.c_str(), card2.c_str(),
card3.c_str(), card4.c_str(), card1.c_str(), card2.c_str(),card3.c_str(),card4.c_str()
           );
}
void printCardsClosed(){
   printf( "┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐    \n"
           "|▒▒▒▒▒▒▒▒▒|  |    ^    |  |    _    |  |▒▒▒▒▒▒▒▒▒|    \n"
           "|▒▒▒▒▒▒▒▒▒|  |   / \\   |  |   / \\   |  |▒▒▒▒▒▒▒▒▒|     \n"
           "|▒▒▒▒▒▒▒▒▒|  |  /   \\  |  |  _\\ /_  |  |▒▒▒▒▒▒▒▒▒|      \n"
           "|▒▒▒▒▒▒▒▒▒|  | (     ) |  | /     \\ |  |▒▒▒▒▒▒▒▒▒|     \n"
           "|▒▒▒▒▒▒▒▒▒|  |  \\   /  |  | \\_/ \\_/ |  |▒▒▒▒▒▒▒▒▒|      \n"
           "|▒▒▒▒▒▒▒▒▒|  |   \\ /   |  |   |_|   |  |▒▒▒▒▒▒▒▒▒|      \n"
           "|▒▒▒▒▒▒▒▒▒|  |    v    |  |         |  |▒▒▒▒▒▒▒▒▒|      \n"
           "└─────────┘  └─────────┘  └─────────┘  └─────────┘     \n"

           );
}
```

cards.cpp

```cpp
#include <iostream>
#include <cstring>
#include <set>
using namespace std;

bool splitCards(string str, char sep, string cardArr[4]){
   // Initialize
   int strLen = str.length();
   int j = 0;
   int count = 0;

   // Iterasi untuk split
   for(int i = 0; i < strLen+1; i++){
       if (count > 3){
           return false;
       }
       if (str[i] == sep || i == strLen){
           string substr = "";
           substr.append(str, j, i-j);
           cardArr[count] = substr;
           count ++;
           j = i + 1 ;
       }
   }
   return (count == 4);
}

bool validateCards(string cardArr[4], int numArr[4]){
   // Initialize
   int strLen;

   for(int i=0; i<4; i++){
       // cout << cardArr[i][0] << "\n";
       // Validasi jika bukan Char
       strLen = cardArr[i].length();
       if (strLen > 1 && cardArr[i] != "10"){
           return false;
       }
       // Jika char
       if (cardArr[i] == "A"){
           numArr[i] = 1;
       } else if (cardArr[i] == "J"){
           numArr[i] = 11;
       } else if (cardArr[i] == "Q"){
           numArr[i] = 12;
       } else if (cardArr[i] == "K"){
           numArr[i] = 13;
       }
```

```
        // Jika number
        else if (cardArr[i] == "10"){
            numArr[i] = 10;
        } else if (cardArr[i][0] > 49 && cardArr[i][0] < 59){
            numArr[i] = cardArr[i][0] - 48;
        }
        // lainnya
        else {return false;}
    }
    return true;
}

string numToCards(int numCard){
    if (numCard == 1){
        return "A";
    } else if(numCard == 11){
        return "J";
    } else if(numCard == 12){
        return "Q";
    } else if(numCard == 13){
        return "K";
    } else { return(to_string(numCard));}
}


// CARDS
void randCards(int ArrNum[4]){
    // Use current time as seed for random generator
    srand(time(0));
    int ub = 13;
    int lb = 1;
    for (int i = 0; i < 4; i++){
        ArrNum[i] = (rand() % (ub - lb + 1)) + lb;
    }
}

void inputCards(int ArrNum[4]){
    // Initialize
    string cardString;
    string ArrStr[4];
    bool cardValid;

    // Input Kartu
    do {
        // Masukkan inout
        cout << "\nMasukkan 4 kartu: ";
        getline(cin, cardString);

        // Validasi Input
        cardValid = splitCards(cardString, ' ', ArrStr);
        if (cardValid){
            cardValid = validateCards(ArrStr, ArrNum);
        }
```

```
        if (!cardValid) { cout << "Masukan salah! Coba lagi. \n";}
    } while (!cardValid);
}
```

operators.cpp

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <time.h>
#include <chrono>
#include <set>
using namespace std;

// WRITE TO FILE
void writeSolutions(set <string> StrRes, string filename){
    // Buka file
    ofstream fw("./test/"+filename +".txt");
    if (fw.is_open()){
        auto it = StrRes.begin();
        while (it != StrRes.end()){
            fw << (*it) << "\n";
            it++;
        }
        fw.close();
    }
    else { cout << "Ada kesalahan pembukaan file!";}
}

// Solve Operators
double solvOps(char ops, double a, double b){
    if (ops == '/' && b == 0){
        return 0;
    } else if (ops == '/'){
        return a / b;
    } else if (ops == '+'){
        return a + b;
    } else if (ops == '-'){
        return a - b;
    } else if (ops == '*'){
        return a * b;
    } else {
        return 0;
    }
}

int solvCards(int ArrCards[4]){
    // Initialize solutions
    int solutions = 0;
    char optionSave;
    // Initialize number arrays
    double Num1[4];
```

```
double Num2[3];
double Num3[2];
double res[1];
// Initialize string arrays
string Str1[4];
string Str2[3];
string Str3[2];
string Str4[1];
// Initialize set results
set <string> StrRes;
// Initialize operators
char ops[4] = {'/', '*', '-', '+'};

// Initialize number
for (int i=0; i<4; i++){
    Num1[i] = ArrCards[i];
}

// Initializing Strings
Str1[0] = to_string(ArrCards[0]);
Str1[1] = to_string(ArrCards[1]);
Str1[2] = to_string(ArrCards[2]);
Str1[3] = to_string(ArrCards[3]);
chrono::steady_clock::time_point begin = chrono::steady_clock::now();

    // Looping operators
for(int a=0; a<4; a++){
    for(int b=0; b<4; b++){
        for(int c=0; c<4; c++){

                // Solve iterasi pertama
                    for (int i=0; i<4; i++){
                        for(int j=0; j<4; j++){
                            for(int k=0; k<4; k++){
                                for(int l=0; l<4; l++){
                                    if((i != j) && (i != k) && (i != l) && (j != k) && (j != l) && (k != l)){
                                        // Generalized Solution
                                        // Masukkan number ke array
                                        double res1 = solvOps(ops[a], Num1[i], Num1[j]);
                                        Num2[0] = res1;
                                        Num2[1] = Num1[k];
                                        Num2[2] = Num1[l];
                                        // Masukkan char ke array string
                                        Str2[0] = "(" + Str1[i] + " " + ops[a] + " " + Str1[j] + ")";
                                        Str2[1] = Str1[k];
                                        Str2[2] = Str1[l];

                                        // Solve iterasi kedua
                                        for(int h=0; h<3; h++){
                                            for(int g=0; g<3; g++){
                                                for(int f=0; f<3; f++){
                                                    if((g != h) && (g != f) && (h != f)){

                                                        // Menghitung
                                                        double res2 = solvOps(ops[b], Num2[h], Num2[g]);
                                                        Num3[0] = res2;
                                                        Num3[1] = Num2[f];
                                                        // Masukkan char ke array string
                                                        Str3[0] = "(" + Str2[h] + " " + ops[b] + " " + Str2[g] + ")";
                                                        Str3[1] = Str2[f];

                                                        // Solve iterasi ketiga
```

```
                                                    for (int y=0; y<2; y++){
                                                        for(int x=0; x<2; x++){
                                                            if((y != x)){
                                                                // Solve results
                                                                res[0] = solvOps(ops[c], Num3[y], Num3[x]);
                                                                // Masukkan char ke array string
                                                                Str4[0] = Str3[y] + " " + ops[c] + " " + Str3[x];
                                                                // cout << Str4[0] << "\n";

                                                                if (res[0] <= 24.1 && res[0] >= 23.9){
                                                                    StrRes.insert(Str4[0]);
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    chrono::steady_clock::time_point end = chrono::steady_clock::now();
    cout << "Terdapat: " << StrRes.size() << " solusi\n";
    auto it = StrRes.begin();
    while (it != StrRes.end()){
        cout << (*it) << "\n";
        // cout << (solutions) << "\n";
        it++;
    }

    double time_elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - begin).count();
    cout << "Time elapsed: " << time_elapsed / 1000000000 << " s" << std::endl;

    // Save
            cout << "\nSimpan [y/n]? ";
            cin >> optionSave;
            if (optionSave == 'y'){
                string filename;
                cin.ignore();
                cout << "Masukkan nama file: ";
                getline(cin, filename);
                writeSolutions(StrRes, filename);}
            else {cout << "Ok! Tidak menyimpan\n";}

    return 0;
}
```

main.cpp

```cpp
#include <iostream>
#include "./utils/decorators.cpp"
#include "./utils/operators.cpp"
#include "./utils/cards.cpp"
using namespace std;

int main(){
   // Initialize Value
   int menuOption = 0;
   int arrCardsNum[4];
   string arrCardsStr[4];
   bool gameValid = true;
   char optionValid;

   // Splash Screen
   printAscii();
   printCardsClosed();

   // Menu
   while (gameValid){
       cout << "Selamat datang di 24 Card Solver!\n\n";
       cout << "======================\n"
               "         MENU         \n"
               "======================\n";
       cout << "1. Pilih Kartu Sendiri \n"
               "2. Random aja!\n"
               "3. Keluar\n";

       do {
           cout << "\nPilih menu: ";
           cin >> menuOption;
           if (!cin) { // cin is not in a good state
               cerr << "Inputnya bukan angka! :(\n";
               menuOption = 3;}

       } while (menuOption < 1 || menuOption > 3);

       // Pemilihan menu
       switch (menuOption){
       case 1:
           // Input cards
           cin.ignore();
           inputCards(arrCardsNum);
           // Show cards
           for (int i = 0; i< 4; i++){
               arrCardsStr[i] = numToCards(arrCardsNum[i]);
           }
           cout << "Berikut kartu yang kamu pilih!\n";
           printCards(arrCardsStr[0],arrCardsStr[1],arrCardsStr[2],arrCardsStr[3]);
```

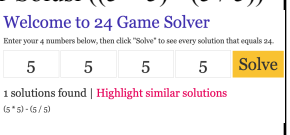
```
            cout << "\n";
            // Solve
            solvCards(arrCardsNum);
            // Round 2?
            cout << "\nIngin bermain lagi [y/n]? ";
            cin >> optionValid;
            if (optionValid != 'y'){gameValid = false;}
            else {gameValid = true;};
            break;

        case 2:
            // Cards
            randCards(arrCardsNum);
            // Show cards
            for (int i = 0; i< 4; i++){
                arrCardsStr[i] = numToCards(arrCardsNum[i]);
            }
            cout << "Berikut kartu yang kamu pilih!\n";
            printCards(arrCardsStr[0],arrCardsStr[1],arrCardsStr[2],arrCardsStr[3]);
            cout << "\n";
            // Solve
            solvCards(arrCardsNum);
            // Round 2?
            cout << "\nIngin bermain lagi [y/n]? ";
            cin >> optionValid;
            if (optionValid != 'y'){gameValid = false;}
            else {gameValid = true;};
            break;
        case 3:
            gameValid = false;
            break;
        }
    }
    cout << "\nTerimakasih telah bermain!\n";
    return 0;
}
```

BAB 4
EKSPERIMEN

4.1. Eksperimen Program

Pada bab ini, akan dijabarkan penyelesaian serta perbandingan dengan referensi yang terdapat pada spesifikasi Tugas Kecil 1 ini. Berikut merupakan tautan referensi yang digunakan pada eksperimen ini : http://24solver.us-west-2.elasticbeanstalk.com.

(Tabel 4.1. Eksperimen Algoritma Brute Force dalam Penyelesaian Kartu 24)

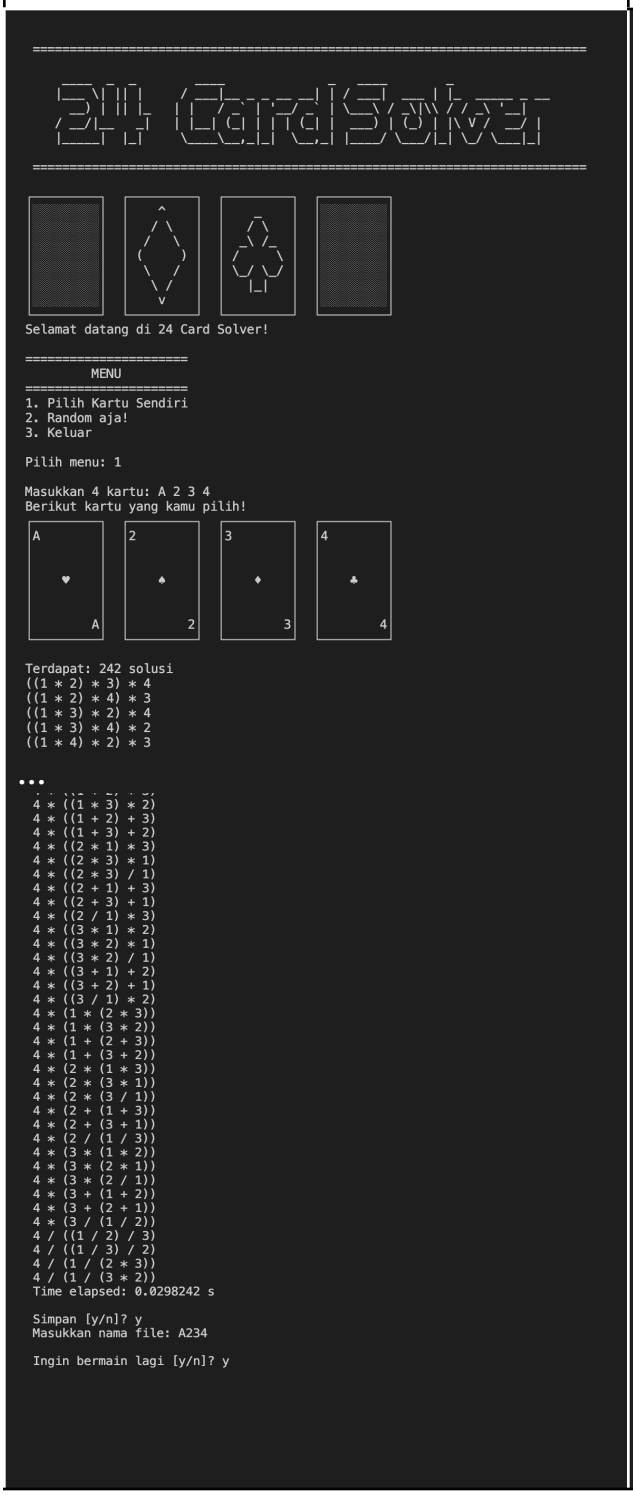
Soal		Penyelesaian	Referensi	Analisis
a)	A 2 3 4 (Masukkan)	242 Solusi Time : 0.0298242 s	194 Solusi 	Analisis: Terdapat ketidaklengkapan pada referensi misalnya 1*(4*(2*3))) tidak ada pada referensi sementara ((1*4)*2)*3 ada. Hal ini disebabkan banyak operasi bersifat asosiatif yang dihitung sama pada referensi. Semua solusi pada referensi terdapat pada penyelesaian.
		Selisih : 48 1*(4*(2*3)), 4*(2*(1*3)), 2*(3*(4*1)), 4/(1/(3*2)), 2*(3*(1*4)), 3*(4*(2/1)), 3*(1*(4*2)), 4*(2*(3/1)), 4*(2*(3*1)), 3*(4*(2*1)), 3*(4*(1*2)), 4*(3*(2*1)), 4/(1/(2*3)), 4*(1*(3*2)), 2*(4/(1/3)), 3/(1/(4*2)), 4*(3*(2/1)), 4*(3/(1/2)), 3*(2*(4/1)), 3/(1/(2*4)), 2/(1/(4*3)), 3*(4/(1/2)), 4*(2/(1/3)), 3*(2*(1*4)), 4*(3*(1*2)), 1*(2*(3*4)), 2*(3/(1/4)), 3*(2/(1/4)), 4*(1*(2*3)), 2*(4*(3*1)), 1*(3*(2*4)), 1*(3*(4*2)), 2/(1/(3*4)), 2*(4*(3/1)), 2*(4*(1*3)), 1*(4*(3*2)), 1*(2*(4*3)), 3*(1*(2*4)), 4*(3+(1+2)), 4*(3+(2+1)), 4*(2+(3+1)), 2*(1*(3*4)), 3*(2*(4*1)), 4*(1+(3+2)), 2*(1*(4*3)), 2*(3*(4/1)), 4*(2+(1+3)), 4*(1+(2+3))	**Selisih : 0**	
b)	5 5 5 5 (Masukkan)	1 Solusi ((5 * 5) - (5 / 5)) Time elapsed: 0.0239677 s	1 Solusi ((5 * 5) - (5 / 5)) 	Analisis: Solusi pada program dan referensi memuat hasil yang sama.
		Selisih : 0	**Selisih : 0**	
c)	A K A K (Masukkan)	15 Solusi Time elapsed: 0.0286801 s	12 Solusi 	Analisis: Solusi pada program dan referensi terdapat perbedaan dimana terletak pada penggunaan kurung atau sifat asosiatif. Semua solusi pada referensi terdapat
		Selisih: 3 13+(13-(1+1)), 13-(1+(1-13)), 13-(1-(13-1))	**Selisih: 0**	

				pada penyelesaian.
d)	9 A J 3 (Random)	154 Solusi Time elapsed: 0.0326203 s	130 Solusi 	Analisis: Terdapat ketidaklengkapan pada referensi misalnya 1*(4*(2*3))) tidak ada pada referensi sementara (9+11)+(1+3) ada. Hal ini disebabkan banyak operasi bersifat asosiatif yang dihitung sama pada referensi. Semua solusi pada referensi terdapat pada penyelesaian.
		Selisih: 24 9+(11+(1+3)), 1+(3+(11+9)), 3+(11+(1+9)) 1+(9+(3+11)), 11+(3+(1+9)), 3+(1+(9+11)) 11+(3+(9+1)), 11+(1+(9+3)), 1+(3+(9+11)) 9+(1+(11+3)), 9+(1+(3+11)), 11+(1+(3+9)) 11+(9+(3+1)), 11+(9+(1+3)), 1+(11+(3+9)) 3+(9+(11+1)), 3+(1+(11+9)), 9+(11+(3+1)) 3+(9+(1+11)), 1+(9+(11+3)), 9+(3+(1+11)) 1+(11+(9+3)), 9+(3+(11+1)), 3+(11+(9+1))	**Selisih: 0**	
e)	J 7 A 6 (Random)	214 Solusi Time elapsed: 0.0303135 s	178 Solusi 	Analisis: Terdapat ketidaklengkapan pada referensi misalnya 6+(11+(1*7) tidak ada pada referensi sementara (6+11)+(1*7) ada. Hal ini disebabkan banyak operasi bersifat asosiatif yang dihitung sama pada referensi. Semua solusi pada referensi terdapat pada penyelesaian.
		Selisih: 36 6+(11+(1*7)), 7+(11+(6/1)), 6+(1*(11+7)) 11+(6+(7/1)), 6+(7+(11*1)), 11+(7+(1*6)) 11+(6+(7*1)), 7+(11+(6*1)), 1*(6+(11+7)) 11+(6+(1*7)), 7+(6+(11*1)), 11+(7+(6*1)), 1*(7+(11+6)), 1*(7+(6+11)), 1*(6*(11-7)), 6*(1*(11-7)), 6+(11+(7/1)), 6+(7+(11/1)) 7+(1*(6+11)), 6+(1*(7+11)), 1*(11+(7+6)) 6*(11-(7/1)), 11+(1*(6+7)), 6/(1/(11-7)) 6+(7+(1*11)), 6+(11+(7*1)), 11+(1*(7+6)) 7+(6+(1*11)), 1*(6+(7+11)), 11+(7+(6/1)) 6*(11-(7*1)), 6*(11-(1*7)), 7+(11+(1*6)) 7+(1*(11+6)), 1*(11+(6+7)), 7+(6+(11/1))	**Selisih: 0**	
f)	7 J 8 9 (Random)	0 Solusi Time elapsed: 0.0263611 s	0 Solusi 	Analisis: Solusi pada program dan referensi memuat hasil yang sama.
		Selisih: 0	**Selisih: 0**	

4.2. Masukkan / Keluaran Program

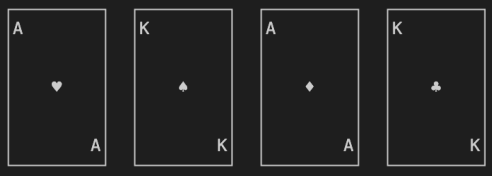
Berikut merupakan keluaran program penyelesaian kartu 24.

(Tabel 4.2. Keluaran Program)

Soal a	Soal b
	

Soal c

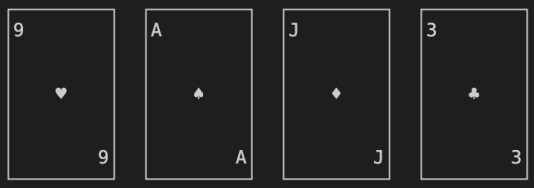
```
====================
        MENU
====================
1. Pilih Kartu Sendiri
2. Random aja!
3. Keluar

Pilih menu: 1

Masukkan 4 kartu: 1 K 1 K
Masukan salah! Coba lagi.

Masukkan 4 kartu: A K A K
Berikut kartu yang kamu pilih!
```




```
Terdapat: 15 solusi
((13 + 13) - 1) - 1
((13 - 1) + 13) - 1
((13 - 1) - 1) + 13
(13 + (13 - 1)) - 1
(13 + 13) - (1 + 1)
(13 - (1 + 1)) + 13
(13 - (1 - 13)) - 1
(13 - 1) + (13 - 1)
(13 - 1) - (1 - 13)
13 + ((13 - 1) - 1)
13 + (13 - (1 + 1))
13 - ((1 + 1) - 13)
13 - ((1 - 13) + 1)
13 - (1 + (1 - 13))
13 - (1 - (13 - 1))
Time elapsed: 0.0286801 s


Simpan [y/n]? y
Masukkan nama file: AKAK

Ingin bermain lagi [y/n]? y
```

Soal d

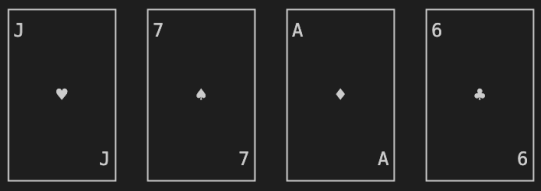
```
======================
        MENU
======================
1. Pilih Kartu Sendiri
2. Random aja!
3. Keluar

Pilih menu: 2
Berikut kartu yang kamu pilih!
```




```
Terdapat: 154 solusi
((1 * 11) * 3) - 9
((1 * 3) * 11) - 9
((1 + 11) + 3) + 9
((1 + 11) + 9) + 3
((1 + 3) + 11) + 9
((1 + 3) + 9) + 11
((1 + 9) + 11) + 3
((1 + 9) + 3) + 11
((11 * 1) * 3) - 9
((11 * 3) * 1) - 9
((11 * 3) - 9) * 1
((11 * 3) - 9) / 1
((11 * 3) / 1) - 9
((11 + 1) + 3) + 9
((11 + 1) + 9) + 3
((11 + 3) + 1) + 9
((11 + 3) + 9) + 1
((11 + 9) + 1) + 3
((11 + 9) + 3) + 1
((11 / 1) * 3) - 9
((11 / 3) - 1) * 9
((3 * 1) * 11) - 9
((3 * 11) * 1) - 9
((3 * 11) - 9) * 1
((3 * 11) - 9) / 1
((3 * 11) / 1) - 9
((3 + 1) + 11) + 9
((3 + 1) + 9) + 11
...
   3 + ((9 + 1) + 11)
   3 + ((9 + 11) + 1)
   3 + (1 + (11 + 9))
   3 + (1 + (9 + 11))
   3 + (11 + (1 + 9))
   3 + (11 + (9 + 1))
   3 + (9 + (1 + 11))
   3 + (9 + (11 + 1))
   9 * ((11 / 3) - 1)
   9 + ((1 + 11) + 3)
   9 + ((1 + 3) + 11)
   9 + ((11 + 1) + 3)
   9 + ((11 + 3) + 1)
   9 + ((3 + 1) + 11)
   9 + ((3 + 11) + 1)
   9 + (1 + (11 + 3))
   9 + (1 + (3 + 11))
   9 + (11 + (1 + 3))
   9 + (11 + (3 + 1))
   9 + (3 + (1 + 11))
   9 + (3 + (11 + 1))
   Time elapsed: 0.0326203 s

   Simpan [y/n]? y
   Masukkan nama file: R_93JJ
```

Soal e

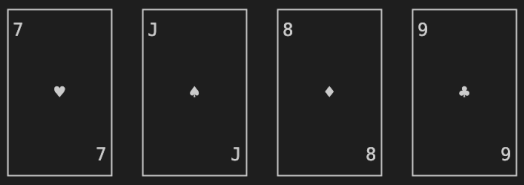
```
======================
        MENU
======================
1. Pilih Kartu Sendiri
2. Random aja!
3. Keluar

Pilih menu: 2
Berikut kartu yang kamu pilih!
```




```
Terdapat: 214 solusi
((1 * 11) + 6) + 7
((1 * 11) + 7) + 6
((1 * 11) - 7) * 6
((1 * 6) + 11) + 7
((1 * 6) + 7) + 11
((1 * 7) + 11) + 6
((1 * 7) + 6) + 11
((11 * 1) + 6) + 7
((11 * 1) + 7) + 6
((11 * 1) - 7) * 6
((11 + 6) * 1) + 7
((11 + 6) + 7) * 1
((11 + 6) + 7) / 1
((11 + 6) / 1) + 7
((11 + 7) * 1) + 6
((11 + 7) + 6) * 1
((11 + 7) + 6) / 1
((11 + 7) / 1) + 6
((11 - 7) * 1) * 6
((11 - 7) * 6) * 1
((11 - 7) * 6) / 1
((11 - 7) / 1) * 6
((11 / 1) + 6) + 7
((11 / 1) + 7) + 6
((11 / 1) - 7) * 6
((6 * 1) + 11) + 7
((6 * 1) + 7) + 11
...
 7 + ((11 + 6) / 1)
 7 + ((11 / 1) + 6)
 7 + ((6 * 1) + 11)
 7 + ((6 + 11) * 1)
 7 + ((6 + 11) / 1)
 7 + ((6 / 1) + 11)
 7 + (1 * (11 + 6))
 7 + (1 * (6 + 11))
 7 + (11 + (1 * 6))
 7 + (11 + (6 * 1))
 7 + (11 + (6 / 1))
 7 + (6 + (1 * 11))
 7 + (6 + (11 * 1))
 7 + (6 + (11 / 1))
 Time elapsed: 0.0303135 s

 Simpan [y/n]? y
 Masukkan nama file: R_76JA

 Ingin bermain lagi [y/n]? y
```

Soal f

```
Selamat datang di 24 Card Solver!

======================
        MENU
======================
1. Pilih Kartu Sendiri
2. Random aja!
3. Keluar

Pilih menu: 2
Berikut kartu yang kamu pilih!
```




```
Terdapat: 0 solusi
Time elapsed: 0.0263611 s

Simpan [y/n]? y
Masukkan nama file: R_7J89

Ingin bermain lagi [y/n]? n

Terimakasih telah bermain!
➜  IF2211-24-Card-Solver git:(main) ✗
```

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Pada tugas kecil I IF2211 Strategi Algoritma ini telah diimplementasikan algoritma berbasis brute force beserta fungsi-fungsi pendukung dalam tujuan untuk menyelesaikan permainan kartu 24. Fungsi-fungsi tersebut mencakup fungsi yang menangani masukkan kartu, fungsi penyelesaian, dan fungsi yang bersifat dekoratif. Implementasi tersebut kemudian berhasil direalisasikan dalam sebuah program C++.

Algoritma brute force pada tugas ini sangatlah exhaustive dan mengecek semua kemungkinan yang dapat terjadi. Pendekatan ini membuat algoritma berjalan lama dan tidak praktis. Adapun terdapat perbedaan yang dihasilkan antara penyelesaian dan referensi solusi yang telah diberikan. Setelah ditelaah, perbedaan terjadi akibat sifat asosiatif pada operasi. Dari tabel 4.1, terlihat bahwa referensi tidak menyediakan kemungkinan jika telah terpenuhi sifat asosiatifnya. Misalnya penyelesaian (2+3)+4-2 dan 2+(3+4)-2 dianggap satu solusi yang sama.

Dengan demikian, penulis menyimpulkan bahwa melalui Tugas kecil 1 IF2211 Strategi Algoritma ini, dapat dibuat sebuah algoritma berbasis brute force diimplementasikan dalam suatu program yang mengkomputasi kemungkinan penyelesaian permainan kartu 24.

5.2. Saran

Tugas Kecil IF2211 Strategi Algoritma Semester II Tahun 2022/2023 menjadi salah satu tugas yang memberikan pelajaran baru bagi penulis. Berdasarkan pengalaman penulis mengerjakan tugas ini, berikut merupakan saran untuk pembaca yang ingin melakukan atau mengerjakan hal yang serupa.

1. Mengingat bahwa tugas ini merupakan tugas kecil individual, manajemen waktu yang baik sangat diperlukan untuk menyelesaikan tugas ini secara efektif dan efisien. Jika pada sebelumnya tugas besar dilakukan berkelompok, pada tugas ini semua bergantung kepada penulis secara individu.
2. Perancangan algoritma dan struktur program perlu diperhatikan dalam mengerjakan tugas ini. Hal ini disebabkan spesifikasi program yang *loose* dan memungkinkan penulis untuk melakukan eksplorasi lebih. Penggunaan prosedur dan fungsi yang baik sangat disarankan agar membuat struktur program menjadi lebih rapi dan dapat digunakan berulang kali oleh program.

LAMPIRAN

Repository Github

https://github.com/alishalistyaa/IF2211-24-Card-Solver.git

Checklist Fitur

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