

PENYELESAIAN PERMAINAN KARTU '24' MENGGUNAKAN ALGORITMA *BRUTE FORCE*

Diajukan sebagai pemenuhan tugas kecil 1.



Oleh:

Antonio Natthan Krishna

13521162

Dosen Pengampu : Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

IF2211 - Strategi Algoritma

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

2022

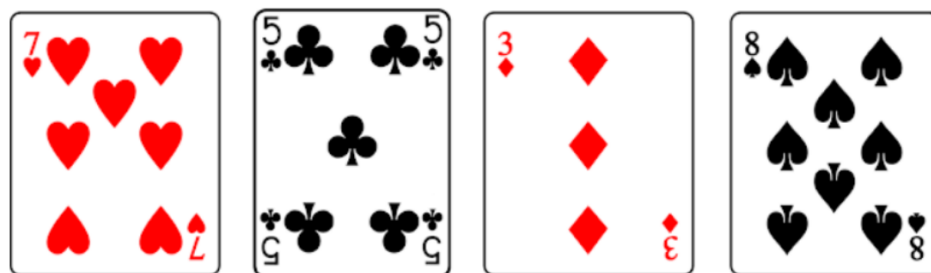
DAFTAR ISI

DAFTAR ISI	2
BAB I	
DESKRIPSI SINGKAT PERSOALAN	3
BAB 2	
LANDASAN TEORI	4
2.1. Permainan Kartu 24	4
2.2. Algoritma Brute Force	5
BAB 3	
IMPLEMENTASI PEMECAHAN PERSOALAN	6
3.1. Metode	6
3.2. Deskripsi dan Struktur Program	7
3.3. Source Program	8
BAB 4	
HASIL PENGUJIAN	24
4.1. Hasil Tangkapan Layar Program	24
4.2. Rangkuman Hasil Pengujian Komprehensif	27
BAB 5	
PENUTUP	29
DAFTAR PUSTAKA	30
LAMPIRAN	31
A. Tautan Repository	31
B. Check List	31

BAB I

DESKRIPSI SINGKAT PERSOALAN

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 (\times), 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/Makalah2016/MakalahStima-2016-038.pdf>).



MAKE IT 24

Gambar 1. Permainan Kartu 24

BAB 2

LANDASAN TEORI

2.1. Permainan Kartu 24

Permainan Kartu 24 adalah salah satu jenis permainan kartu pengaktif otak, yang menantang dan bermanfaat untuk dimainkan. Permainan kartu 24 dapat meningkatkan kecerdasan dan meningkatkan keterampilan menghitung. Sehingga dapat membantu meningkatkan kemampuan matematika anak-anak sekolah dan mengurangi risiko penyakit seperti Alzheimer pada kaum lansia. Permainan ini juga melatih aritmatika mental kita, sehingga bermanfaat untuk semua orang di semua kategori usia. Berikut merupakan aturan permainan kartu 24,

1. Pada awal permainan, pemain atau moderator mengambil 4 kartu secara acak (random) dari dek.
2. Empat kartu tersebut mewakili empat angka antara 1-13. Dengan As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri.
3. Setelah itu, dengan menggunakan operasi dasar matematika, selain ($()$), pengurangan ($-$), perkalian (\times), divisi (\div) dan tanda kurung ($()$) pemain perlu memanipulasi angka-angka tersebut dan mencapai hasil persis bernilai 24.
4. Setiap kartu harus digunakan sekali dan hanya dapat digunakan sekali.
5. Pemain yang dapat menyelesaikan permasalahan tersebut akan mendapatkan 4 buah kartu tersebut.
6. Jika semua pemain menyerah karena tidak dapat menyelesaikan permasalahan dari keempat kartu, maka kartu-kartu tersebut akan dikembalikan ke dek dan dikocok ulang.
7. Permainan berakhir ketika dek telah habis dan pemain dengan kartu di tangan terbanyak yang menang.
8. Jika terjadi draw, maka dilakukan 1 babak lagi untuk menentukan pemenangnya diantara pemain yang mendapatkan jumlah kartu terbanyak yang sama

2.2. Algoritma Brute Force

Brute force adalah sebuah pendekatan yang langsung (straightforward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (obvious way).

Algoritma Brute Force memiliki beberapa kelebihan:

1. Algoritma brute force dapat digunakan untuk memecahkan hampir sebagian besar masalah.
2. Sederhana dan mudah dimengerti.
3. Menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, perkalian matriks.
4. Menghasilkan algoritma baku (standar) untuk tugas-tugas komputasi seperti penjumlahan/perkalian N buah bilangan, menentukan elemen minimum atau maksimum di tabel.

Algoritma Brute Force memiliki beberapa kekurangan:

1. Jarang menghasilkan algoritma yang mangkus/efektif.
2. Lambat sehingga tidak dapat diterima.
3. Tidak kreatif teknik pemecahan masalah lainnya

BAB 3

IMPLEMENTASI PEMECAHAN PERSOALAN

3.1. Metode

Pendekatan algoritma yang digunakan untuk memecahkan persoalan permainan kartu '24' adalah algoritma *brute force*. Algoritma brute force merupakan algoritma yang bersifat obvious, jelas, dan straightforward. Dalam permainan kartu '24', algoritma brute force memiliki pendekatan sebagai berikut

1. Satu ekspresi permainan kartu '24' terdiri atas 4 kartu (E, F, G, H) pada deck dan 3 operasi matematika (x, y, z) yang melibatkan keempat kartu tersebut.

E x F y G z H

2. Kartu (E, F, G, H) boleh memiliki rank yang sama, mengingat simbol pada kartu rummy terdiri atas 4: Spades, Hearts, Diamonds, Clubs. Contoh: E, F, G, H dapat diisi dengan angka 5 seluruhnya.
3. Walaupun memiliki rank yang sama, kartu E, F, G, H merupakan kartu yang unik. Dengan kata lain, posisi F tidak dapat digantikan oleh kartu E, meskipun E dan F dapat memiliki rank yang sama.
4. Setiap operator (+, /, *, -) dapat menempati posisi manapun. Contoh: operator '+' dapat menempati x, y, z seluruhnya.
5. Untuk menentukan bagian yang akan dioperasikan terlebih dahulu. Diperlukan kurung. Terdapat lima posisi kurung yang mungkin yang dapat menjadi pedoman perhitungan matematika.

(E x F) y (G z H)

((E x F) y G) z H

(E x (F y G)) z H

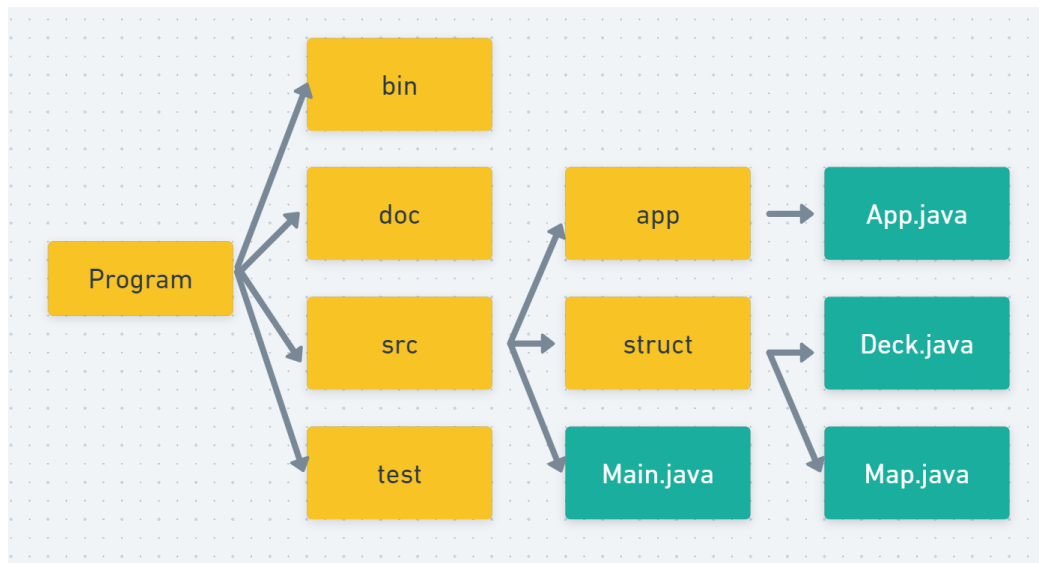
E x ((F y G) z H)

E x (F y (G z H))

6. Untuk setiap posisi kurung, untuk setiap operator dan untuk setiap deck, dilakukan permutasi. Apabila evaluasi mendapatkan hasil sama dengan 24, maka ekspresi tersebut merupakan salah satu solusi dari permainan kartu '24'.
7. Terdapat deck yang tidak memiliki solusi.

3.2. Deskripsi dan Struktur Program

Program ditulis dengan menggunakan bahasa Java dengan struktur sebagai berikut,



Gambar 2. Struktur Program

Kotak berwarna kuning melambangkan folder dan kotak berwarna hijau melambangkan file program/output.

1. bin, berisikan file hasil compile java, memiliki extension .class.
2. doc, berisi file laporan ini.
3. src, merupakan source program. Terdapat dua folder dan satu main program (Main.java).
 - a. app, yang berisi App.java yang berfungsi mengatur jalannya program.
 - b. struct, merupakan folder struktur data.
 - Map (Dictionary) yang dimuat dalam Map.java, berisi tentang value integer dari sebuah rank (Misal. K -> 13).
 - Deck merupakan tipe data bentukan yang merupakan modifikasi dari array of string. Tipe data ini berfungsi untuk melakukan manipulasi terhadap susunan kartu yang mungkin pada permainan '24'.
4. test, folder tujuan penyimpanan solusi program.

3.3. Source Program

1. App.java

```
package app;

import java.util.Scanner;
import java.util.ArrayList;

import struct.*;

public class App {
    public boolean exe (Scanner inputTray) {
        // Initialization
        String[] inputCards = new String[4];
        Deck Deck = new Deck();

        // Menu Input
        System.out.println("Choose Input");
        System.out.println("1. My Deck");
        System.out.println("2. Random Deck");
        System.out.println("3. Exit");
        System.out.print("My choice (1/2/3): ");
        String choice = inputTray.nextLine();

        while (!choice.equals("1") && !choice.equals("2")
            && !choice.equals("3")) {
            System.out.println("Wrong input! Please type
1/2/3.");
            System.out.print("My choice (1/2/3): ");
            choice = inputTray.nextLine();
        }
        // choice = 1/2/3
    }
}
```



```

        if (choice.equals("1")) {
            for (int i = 0; i < 4; i++) {
                inputCards[i] = inputTray.next();
            }

            Deck.constructDeck(inputCards);

            if (!Deck.isDeckValid(inputCards)) {
                System.out.println("Invalid Card Deck.
Please input the correct one!");

            } else {
                ArrayList<String> result;
                result = Deck.calculateDeck(inputCards);
                Deck.displayResult(result);
                String input = inputTray.nextLine();
                System.out.print("Do you want to save the
results? (Y/N): ");
                input = inputTray.nextLine();

                if (!input.equals("Y") &&
!input.equals("N")) {
                    System.out.print("Wrong input! Please
input (Y/N): ");
                    input = inputTray.nextLine();
                }

                if (input.equals("Y")) {
                    Deck.saveResultsToFile(inputCards,
result);
                }
            }

            return true;
        } else if (choice.equals("2")){

```

```

        System.out.println("Picking Random Card...");
        System.out.print("Your Deck: ");
        inputCards = Deck.randomizeDeck();
        for (int i = 0; i < 4; i++) {
            System.out.print(inputCards[i] + " ");
        }
        System.out.println();
        ArrayList<String> result;
        result = Deck.calculateDeck(inputCards);
        Deck.displayResult(result);
        System.out.print("Do you want to save the
results? (Y/N): ");
        String input = inputTray.nextLine();

        if (!input.equals("Y") && !input.equals("N"))
        {
            System.out.print("Wrong input! Please
input (Y/N): ");
            input = inputTray.nextLine();
        }

        if (input.equals("Y")) {
            Deck.saveResultsToFile(inputCards,
result);
        }
        return true;
    } else { // choice = 3
        System.out.println("Thank you for using our
App!");
        return false;
    }
}
}

```

2. Deck.java

```
package struct;

import java.util.Random;
import java.io.FileWriter;
import java.util.ArrayList;
import java.io.File;
import java.io.IOException;

public class Deck {
    final String[] allowedChar = {"2", "3", "4", "5",
    "6", "7", "8", "9", "10", "A", "J", "Q", "K"};
    final String[] operators = {"+", "-", "*", "/", "(",
    ")"};
    String[] cardDeck;

    public void constructDeck (String[] inputCards) {
        // Initializing Deck
        this.cardDeck = new String[inputCards.length];
        for (int i = 0; i < inputCards.length; i++) {
            this.cardDeck[i] = inputCards[i];
        }
    }

    public boolean isDeckValid (String[] cardIn) {
        // Checking whether the rank of cards in Deck are
valid
        boolean valid = true;
        int i = 0;
        while (i < cardIn.length && valid) {
```

```

        int j = 0;
        boolean found = false;
        while (j < allowedChar.length && !found) {
            if (allowedChar[j].equals(cardIn[i])) {
                found = true;
            }
            j++;
        }
        valid = found;
        i++;
    }
    return valid;
}

public String[] randomizeDeck () {
    // Get a set of Random Deck
    Random rand = new Random();
    String[] randomDeck = new String[4];

    for (int i = 0; i < 4; i++) {
        randomDeck[i] =
allowedChar[rand.nextInt(13)];
    }
    return randomDeck;
}

static double processCalculation (double card1,
double card2, String operator) {
    if (operator.equals("+")) {
        return card1 + card2;
    } else if (operator.equals("-")) {
        return card1 - card2;
    } else if (operator.equals("*")) {
        return card1 * card2;
    }
}

```



```
processCalculation(processCalculation(dict.getValue(inputDeck[i]), dict.getValue(inputDeck[j]), operators[m]), processCalculation(dict.getValue(inputDeck[k]), dict.getValue(inputDeck[l]), operators[o]), operators[n]));

if (temp == 24.0) {

    if (!res.contains("(" + inputDeck[i] + " " + operators[m] + " " + inputDeck[j] + ") " + operators[n] + " (" + inputDeck[k] + " " + operators[o] + " " + inputDeck[l] + ")")) {

        res.add("(" + inputDeck[i] + " " + operators[m] + " " + inputDeck[j] + ") " + operators[n] + " (" + inputDeck[k] + " " + operators[o] + " " + inputDeck[l] + ")");

    }

}

}
```

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        if (i != j) {
            for (int k = 0; k < 4; k++) {
                if (i != k && j != k) {
```

```

        for (int l = 0; l < 4; l++) {
            if (i != l && j != l && k
!= l) {
                for (int m = 0; m <
4; m++) {
                    for (int n = 0; n
< 4; n++) {
                        for (int o =
0; o < 4; o++) {
                            double
temp =
processCalculation(processCalculation(processCalculation(
dict.getValue(inputDeck[i]), dict.getValue(inputDeck[j]),
operators[m]), dict.getValue(inputDeck[k]),
operators[n]), dict.getValue(inputDeck[l]),
operators[o]));
                            if (temp
== 24.0) {
                                if
(!res.contains("(" + inputDeck[i] + " " + operators[m] +
" " + inputDeck[j] + ") " + operators[n] + " " +
inputDeck[k] + ") " + operators[o] + " " + inputDeck[l]))
{
res.add("(" + inputDeck[i] + " " + operators[m] + " " +
inputDeck[j] + ") " + operators[n] + " " + inputDeck[k] +
") " + operators[o] + " " + inputDeck[l]);
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
}

for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        if (i != j) {
            for (int k = 0; k < 4; k++) {
                if (i != k && j != k) {
                    for (int l = 0; l < 4; l++) {
                        if (i != l && j != l && k
!= l) {
                            for (int m = 0; m <
4; m++) {
                                for (int n = 0; n
< 4; n++) {
                                    for (int o =
0; o < 4; o++) {
                                        double
temp =
processCalculation(processCalculation(dict.getValue(input
Deck[i]), processCalculation(dict.getValue(inputDeck[j]),
dict.getValue(inputDeck[k]), operators[n]),
operators[m]), dict.getValue(inputDeck[l]),
operators[o]));

                                        if (temp
== 24.0) {
                                            if
(!res.contains("(" + inputDeck[i] + " " + operators[m] +
" (" + inputDeck[j] + " " + operators[n] + " " +
inputDeck[k] + ")) " + operators[o] + " " +
inputDeck[l])) {

```



```

res.add("(" + inputDeck[i] + " " + operators[m] + " (" +
inputDeck[j] + " " + operators[n] + " " + inputDeck[k] +
")) " + operators[o] + " " + inputDeck[l]);

    }
}
}
}
}
}
}
}
}
}

for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        if (i != j) {
            for (int k = 0; k < 4; k++) {
                if (i != k && j != k) {
                    for (int l = 0; l < 4; l++) {
                        if (i != l && j != l && k
!= l) {

                            for (int m = 0; m <
4; m++) {

                                for (int n = 0; n
< 4; n++) {

                                    for (int o =
0; o < 4; o++) {

                                        double
temp = processCalculation(dict.getValue(inputDeck[i]),
processCalculation(processCalculation(dict.getValue(input

```

```
Deck[j]), dict.getValue(inputDeck[k]), operators[n]),
dict.getValue(inputDeck[l]), operators[o]),
operators[m]);

                                if (temp
== 24.0) {

                                    if
(!res.contains(inputDeck[i] + " " + operators[m] + " (" +
+ inputDeck[j] + " " + operators[n] + " " + inputDeck[k]
+ ") " + operators[o] + " " + inputDeck[l] + "))") {

res.add(inputDeck[i] + " " + operators[m] + " (" +
inputDeck[j] + " " + operators[n] + " " + inputDeck[k] +
") " + operators[o] + " " + inputDeck[l] + "));

                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }

for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        if (i != j) {
            for (int k = 0; k < 4; k++) {
                if (i != k && j != k) {
                    for (int l = 0; l < 4; l++) {
                        if (i != l && j != l && k
!= l) {
```

```

        for (int m = 0; m <
4; m++) {
                                for (int n = 0; n
< 4; n++) {
                                    for (int o =
0; o < 4; o++) {
                                        double
temp = processCalculation(dict.getValue(inputDeck[i]),
processCalculation(dict.getValue(inputDeck[j]),
processCalculation(dict.getValue(inputDeck[k]),
dict.getValue(inputDeck[l]), operators[o]),
operators[n]), operators[m]);
                                                if (temp
== 24.0) {
                                                    if
(!res.contains(inputDeck[i] + " " + operators[m] + " (" +
inputDeck[j] + " " + operators[n] + " (" + inputDeck[k] +
" " + operators[o] + " " + inputDeck[l] + "))")) {
res.add(inputDeck[i] + " " + operators[m] + " (" +
inputDeck[j] + " " + operators[n] + " (" + inputDeck[k] +
" " + operators[o] + " " + inputDeck[l] + "))");
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        long end = System.nanoTime();
        long time = end - start;
        res.add(String.valueOf(time));
        return res;
    }

    public void displayResult (ArrayList<String> result)
    {
        // Display Result to Terminal
        if (result.size() == 1) {
            System.out.println("No solution found");
            System.out.println("Time taken: " +
Integer.parseInt(result.get(0))/1000000000.0);

            } else if (result.size() == 2) {

System.out.println(String.valueOf(result.size()-1) + "
solution found");
            for (int i = 0; i < result.size()-1; i++) {
                System.out.println(result.get(i));
                System.out.println("Time taken: " +
Integer.parseInt(result.get(1))/1000000000.0);
            }

            } else {

System.out.println(String.valueOf(result.size()-1) + "
solutions found");
            for (int i = 0; i < result.size()-1; i++) {
                System.out.println(result.get(i));
            }
            System.out.println("Time taken: " +
Integer.parseInt(result.get(result.size()-1))/1000000000.
0);

```

```

    }

}

    public void saveResultsToFile (String[] inputDeck,
ArrayList<String> result) {
    // Save Results to File
    try {
        File buffer = new File("../test\\Solution "
+ inputDeck[0] + " " + inputDeck[1] + " " + inputDeck[2]
+ " " + inputDeck[3] + ".txt");
        if (buffer.createNewFile()) {
            try {
                try (FileWriter writer = new
FileWriter(buffer)) {
                    if (result.size() == 1) {
                        writer.write("No solution
found");

                    } else if (result.size() == 2) {
                        writer.write("1 solution
found\n");

                        writer.write(result.get(0));
                    } else {
                        writer.write(result.size()-1
+ " solutions found\n");

                        for (int i = 0; i <
result.size()-1; i++) {

writer.write(result.get(i) + "\n");

                        }
                    }

                    writer.write("Time taken: " +
Integer.parseInt(result.get(result.size()-1))/1000000000.
0);

                }
            }
        }
    }
}

```

```

        } catch (IOException e) {
            System.out.println("Something went
wrong. Please try again.");
        }
    } else {
        System.out.println("File already
exists.");
    }
} catch (IOException e) {
    System.out.println("Something went wrong.
Please try again.");
}
}
}

```

3. Map.java

```

package struct;
import java.util.*;

public class Map {
    Dictionary<String, Integer> dict = new
Hashtable<String, Integer>();

    public Map() {
        // Dictionary Initialization
        dict.put("A", 1);
        dict.put("2", 2);
        dict.put("3", 3);
        dict.put("4", 4);
        dict.put("5", 5);
        dict.put("6", 6);
        dict.put("7", 7);
        dict.put("8", 8);
    }
}

```

```
        dict.put("9", 9);
        dict.put("10", 10);
        dict.put("J", 11);
        dict.put("Q", 12);
        dict.put("K", 13);
    }

    public int getValue (String inputSuit) {
        // Getting the value of the suit
        return dict.get(inputSuit);
    }
}
```

4. Main.java

```
import app.*;
import java.util.Scanner;

public class Main {
    public static void main (String[] args) {
        App executable = new App();
        boolean choicevalid = true;
        Scanner inputTray = new Scanner(System.in);

        System.out.println("\'24\' Card Game Solver");
        while (choicevalid) {
            choicevalid = executable.exe(inputTray);
            inputTray = new Scanner(System.in);
        }
        inputTray.close();
    }
}
```

BAB 4

HASIL PENGUJIAN

4.1. Hasil Tangkapan Layar Program

a. Testing 1

Input dari pengguna dan penyimpanan pada file (file belum ada)

Tampilan Program

```
Choose Input
1. My Deck
2. Random Deck
3. Exit
My choice (1/2/3): 1
5 5 7 6
16 solutions found
(5 * 5) - (7 - 6)
(5 * 5) + (6 - 7)
(5 * 7) - (5 + 6)
(5 * 7) - (6 + 5)
(7 * 5) - (5 + 6)
(7 * 5) - (6 + 5)
(6 - 7) + (5 * 5)
((5 * 5) - 7) + 6
((5 * 5) + 6) - 7
((5 * 7) - 5) - 6
((5 * 7) - 6) - 5
((7 * 5) - 5) - 6
((7 * 5) - 6) - 5
(6 + (5 * 5)) - 7
6 + ((5 * 5) - 7)
6 - (7 - (5 * 5))
Time taken: 0.0034469
Do you want to save the results? (Y/N): Y
```


Tampilan file	<pre> test > ≡ Solution 5 5 7 6.txt 1 16 solutions found 2 (5 * 5) - (7 - 6) 3 (5 * 5) + (6 - 7) 4 (5 * 7) - (5 + 6) 5 (5 * 7) - (6 + 5) 6 (7 * 5) - (5 + 6) 7 (7 * 5) - (6 + 5) 8 (6 - 7) + (5 * 5) 9 ((5 * 5) - 7) + 6 10 ((5 * 5) + 6) - 7 11 ((5 * 7) - 5) - 6 12 ((5 * 7) - 6) - 5 13 ((7 * 5) - 5) - 6 14 ((7 * 5) - 6) - 5 15 (6 + (5 * 5)) - 7 16 6 + ((5 * 5) - 7) 17 6 - (7 - (5 * 5)) 18 Time taken: 0.0034469 </pre>
Keterangan	-

b. Testing 2

Input dari pengguna dan penyimpanan pada file (file telah tersedia)	
Tampilan Program	<pre> Choose Input 1. My Deck 2. Random Deck 3. Exit My choice (1/2/3): 1 5 5 7 6 16 solutions found (5 * 5) - (7 - 6) (5 * 5) + (6 - 7) (5 * 7) - (5 + 6) (5 * 7) - (6 + 5) (7 * 5) - (5 + 6) (7 * 5) - (6 + 5) (6 - 7) + (5 * 5) ((5 * 5) - 7) + 6 ((5 * 5) + 6) - 7 ((5 * 7) - 5) - 6 ((5 * 7) - 6) - 5 ((7 * 5) - 5) - 6 ((7 * 5) - 6) - 5 (6 + (5 * 5)) - 7 6 + ((5 * 5) - 7) 6 - (7 - (5 * 5)) Time taken: 0.0037658 Do you want to save the results? (Y/N): Y File already exists. </pre>

Hasil	<pre> test > ≡ Solution 5 5 7 6.txt 1 16 solutions found 2 (5 * 5) - (7 - 6) 3 (5 * 5) + (6 - 7) 4 (5 * 7) - (5 + 6) 5 (5 * 7) - (6 + 5) 6 (7 * 5) - (5 + 6) 7 (7 * 5) - (6 + 5) 8 (6 - 7) + (5 * 5) 9 ((5 * 5) - 7) + 6 10 ((5 * 5) + 6) - 7 11 ((5 * 7) - 5) - 6 12 ((5 * 7) - 6) - 5 13 ((7 * 5) - 5) - 6 14 ((7 * 5) - 6) - 5 15 (6 + (5 * 5)) - 7 16 6 + ((5 * 5) - 7) 17 6 - (7 - (5 * 5)) 18 Time taken: 0.0034469 </pre>
Analisis	File tidak berubah dikarenakan memiliki isi informasi yang sama sehingga tidak dilakukan kembali penulisan ke file txt kembali karena operasi write merupakan operasi yang “mahal”

c. Testing 3

Input tidak valid	
Tampilan Program	<pre> Choose Input 1. My Deck 2. Random Deck 3. Exit My choice (1/2/3): 1 1 2 3 4 Invalid Card Deck. Please input the correct one! </pre>
Keterangan	Meskipun 1 merupakan angka yang valid pada perhitungan, angka satu tidak terdapat pada paket kartu rummy. Untuk memilih nilai 1, pengguna dapat memberi masukan ‘A’

d. Testing 4

No Solution Deck

Tampilan Program	<pre> Choose Input 1. My Deck 2. Random Deck 3. Exit My choice (1/2/3): 1 J J J J No solution found Time taken: 0.003127 Do you want to save the results? (Y/N): N </pre>
Keterangan	-

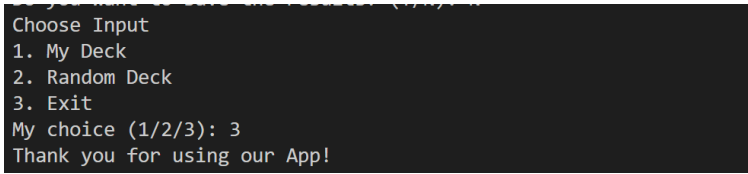
e. Testing 5

Random Deck	
Tampilan Program	<pre> Choose Input 1. My Deck 2. Random Deck 3. Exit My choice (1/2/3): Wrong input! Please type 1/2/3. My choice (1/2/3): 2 Picking Random Card... Your Deck: 2 7 A 7 1 solution found ((7 * 7) - A) / 2 Time taken: 0.0016023 </pre>
Keterangan	Random deck tidak dapat menghasilkan deck yang tidak valid

f. Testing 6

Random Deck (variasi No solution)	
Tampilan Program	<pre> Choose Input 1. My Deck 2. Random Deck 3. Exit My choice (1/2/3): 2 Picking Random Card... Your Deck: 2 9 7 9 No solution found Time taken: 0.0018778 </pre>
Keterangan	Meskipun tidak dapat menghasilkan Deck yang tidak valid, Random Deck dapat menghasilkan Deck yang tidak memiliki solusi

g. Testing 7

Exit	
Tampilan Program	
Keterangan	Program akan terus berjalan hingga pengguna memutuskan untuk keluar dari program

4.2. Rangkuman Hasil Pengujian Komprehensif

No	Deck	Banyak Solusi	Waktu Pengujian
1	2 2 5 8	4	0.0033111
2	10 Q K J	90	0.0079924
3	3 7 K 6	22	0.0062769
4	J 5 Q 3	8	0.0031436
5	7 K 8 3	16	0.0069512
6	J 4 4 5	64	0.0050998
7	6 6 6 6	7	0.0051934
8	A 2 3 4	242	0.0036355
9	A Q A 8	0	0.0016103
10	10 5 7 J	2	0.0008574
11	3 5 2 Q	1	0.0016045
12	9 K 4 K	0	0.0026992

13	10 4 7 Q	26	0.0064206
14	K 2 A 5	28	0.0060499
15	4 3 2 8	32	0.0035380
16	2 7 4 K	0	0.0025151
17	J 2 J 10	8	0.0060234
18	Q 10 8 4	60	0.0027687
19	7 2 7 10	4	0.0018227
20	4 9 7 Q	114	0.0024532
Rata Rata			0.003998345

BAB 5

PENUTUP

Demikianlah laporan tugas pemrograman ini saya tulis. Tugas ini sangat membantu saya memahami abstraksi terkait algoritma *brute force* yang memiliki penyelesaian yang bersifat *obvious*. Selain itu, saya juga turut senang dapat menuliskan pemecahan persoalan ini menggunakan bahasa pemrograman Java yang mana merupakan bahasa pemrograman yang jarang saya gunakan.

Banyak hal yang dapat dikembangkan dari program ini sendiri, salah satunya dengan pembuatan *Graphical User Interface* (GUI) yang dapat memudahkan pengguna menggunakan aplikasi dengan lebih intuitif. Meskipun demikian, program yang berbasis *Command Line Interface* (CLI) ini sudah dapat digunakan dengan efisien.

Akhir kata, dengan waktu yang cukup singkat, saya mengucapkan terima kasih kepada diri saya sendiri karena sudah dapat memenuhi seluruh spesifikasi program yang diberikan

dengan sangat baik. Tak lupa saya mengucapkan terima kasih kepada Bu Ulfa selaku dosen pengampu saya yang sudah sangat jelas dalam menjelaskan mekanisme algoritma *brute force* sebagai bahan mata kuliah IF2211 Strategi Algoritma. Saya berharap dapat belajar lebih banyak kedepannya terkait pemecahan masalah dengan metode lainnya.

DAFTAR PUSTAKA

Bayu Widia Santoso, Firdiansyah Sundawa, Muhammad Azhari. 2016. Implementasi Algoritma Brute Force Sebagai Mesin Pencari (Search Engine) Berbasis Web Pada Database. Diakses pada 22 Januari 2023 melalui <https://core.ac.uk/download/pdf/288088999.pdf>

Evita Chandra. 2015. Penerapan Algoritma Brute Force pada Permainan Kartu 24 (24 game). Diakses pada 22 Januari 2023 melalui <https://docplayer.info/37153843-Penerapan-algoritma-brute-force-pada-permainan-kartu-24-24-game.html>

LAMPIRAN

A. Tautan Repository

Link repository GitHub: https://github.com/natthankrish/Tucil1_13521162

B. Check List

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	V	
Program berhasil <i>running</i>	V	
Program dapat membaca input/generate sendiri	V	

dan memberikan luaran		
Solusi yang diberikan program memenuhi (berhasil mencapai 24)	V	
Program dapat menyimpan solusi dalam file teks	V	