

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

PENYELESAIAN PERMAINAN KARTU 24 DENGAN
BRUTE FORCE ALGORITHM



DISUSUN OLEH :
JEFFREY CHOW
13521046 – K02

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

DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	2
BAB II.....	3
BAB III	4
BAB IV	13
BAB V	17
BAB IV	18

BAB I

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 (\times), divisi (/) dan tanda kurung (). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas. (Source : <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf>)

Pada laporan ini akan dibahas sebuah program yang menggunakan Algoritma *Brute Force* untuk menghasilkan solusi dari permainan kartu 24. Program akan ditulis dalam bahasa Java.

BAB II

ALGORITMA BRUTE FORCE PROGRAM

Pada awal program, program akan meminta input 4 buah kartu atau kartu dapat di acak oleh program. Untuk menyelesaikan permasalahan dengan menggunakan algoritma *brute force*, program harus mencari dan mengecek seluruh kemungkinan yang dapat terjadi.

Pertama, dicari semua kemungkinan permutasi dari 4 angka kartu yang digunakan. Algoritma permutasi yang digunakan diimplementasikan dalam `public static void permute (int[] arr, int index)` pada `file solution.java`. Seluruh hasil permutasi disimpan ke dalam sebuah array of array of integer.

Kedua, setelah seluruh permutasi dari 4 angka didapat, program akan mengecek seluruh kemungkinan kombinasi antara operator dan angka atau ekspresi. Program akan menggunakan konsep rekursif untuk mengimplementasikan operator kurung. Dalam proses rekursifnya, angka-angka yang disimpan dalam bentuk array akan dipartisi menjadi 2 buah array. Kemudian, proses rekursif berlangsung dengan memanggil fungsi itu sendiri dengan array baru hasil partisi sebagai parameter. Basis dari rekursi adalah ketika array hanya tersisa satu elemen. Ketika mencapai basis, fungsi akan mengembalikan elemen tersebut dan akan dioperasikan dengan ekspresi dari hasil rekursif array yang lain. Operasi matematika tambah, kurang, kali, dan bagi akan dioperasikan diantara dua hasil rekursif tersebut dan akan dicoba seluruh kombinasi operator antar kedua ekspresi. Proses ini diimplementasikan dalam `public static Set<StrDoub> rec(Double[] arr)` pada `file solution.java`.

Pada akhir proses rekursi, jika hasil dari operasi ekspresi yang dicoba adalah 24, maka ekspresi akan ditambahkan ke set `solutions`. Penggunaan tipe data set ditujukan agar tidak ada dua ekspresi yang sama yang ditampilkan sebagai solusi. Program juga memiliki data type `StrDoub` yaitu data type yang menyimpan nilai String dan juga Double. String digunakan untuk menyimpan ekspresi dan Double digunakan untuk menyimpan hasil. Double digunakan karena terdapat operasi pembagian yang akan menghasilkan bilangan riil.

Algoritma ini akan menghasilkan ekspresi matematis yang tidak mempertimbangkan ekspresi yang mirip. Contohnya $(3*(3*3))-3$ dan $((3*3)*3)-3$ akan dihitung sebagai dua solusi yang berbeda.

BAB III

KODE PROGRAM DALAM BAHASA JAVA

File : **main.java**

```
import java.util.*;
import java.security.SecureRandom;
import java.io.*;

public class main extends shortcut {
    // global variables

    public static Scanner scan = new Scanner(System.in);
    public static long startTime = 0;
    public static long endTime = 0;

    public static void main(String[] args) {
        // splash screen
        println("\nWelcome to 24 Games Solver");
        println("=====");
        println();

        // menu options
        println("Options: ");
        println("[0] Exit");
        println("[1] Input Cards from Console");
        println("[2] Auto Generate cards");
        println();

        int option;

        do { // input validation
            print("Enter option [0-2]: ");
            option = scan.nextInt();
            scan.nextLine();
        } while (option < 0 || option > 2);

        println();

        switch(option){
            case 1:
                userInput();
                break;
            case 2:
                autoInput();
                break;
            default:
                println("See ya! Thank you mate!");
                System.exit(0);
        }
    }
}
```

```

        break;
    }

    println("\nExecution time : " + (endTime-startTime) + " milliseconds");
}

public static void userInput(){
    println("User Input");
    println("=====");
    println();

    String[] inputs;
    String input;
    boolean reinput;

    do{ // inputs validation
        reinput = false;

        // user input
        print("Input : ");
        input = scan.nextLine();

        // trim and split the input
        inputs = Arrays.copyOfRange(input.trim().split("[ ]+"), 0, 4);

        // validation of each input
        int i = 0;
        while (!reinput && i < inputs.length){
            int ascii = (int)(inputs[i].toLowerCase().charAt(0));
            if (inputs[i].length() > 1){
                reinput = !(inputs[i].equals("10"));
            } else if ((ascii == 97) || (ascii == 106) || (ascii == 107) || (ascii == 113) || (ascii >= 50 &&
ascii <= 57)) {
                reinput = false;
            } else {
                reinput = true;
            }
            i++;
        }

    } while (reinput);

    parseInput(inputs);

}

public static void autoInput(){
    println("Auto Generate Cards");
}

```

```

println("=====");
println();

String[] inputs = new String[4];
String[] cards = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

// randomizer
SecureRandom rand = new SecureRandom();

print("Your cards are : ");

// random 4 cards
for (int i = 0; i < 4; i++){
    inputs[i] = cards[rand.nextInt(13)];
    print(inputs[i] + ' ');
}

println();

parseInput(inputs);
}

public static void parseInput(String[] inputs){
    int[] numbers = new int[4];

    // create dictionary
    Hashtable<String, Integer> dict = new Hashtable<String, Integer>();
    dict.put("A", 1);
    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("j", 11);
    dict.put("Q", 12);
    dict.put("q", 12);
    dict.put("K", 13);
    dict.put("k", 13);

    // parse
    for (int i = 0; i < inputs.length; i++){
        numbers[i] = dict.get(inputs[i]);
    }
}

```

```

    }

    // start timer
    startTime = System.currentTimeMillis();

    // call for solution
    solution.solution(numbers);

}

public static void output(Set<String> solutions){
    // end timer
    endTime = System.currentTimeMillis();

    println("\nOutput options : ");
    println("[1] Console");
    println("[2] Text File");
    println();

    int option;

    do { // input validation
        print("Enter option [1-2]: ");
        option = scan.nextInt();
        scan.nextLine();
    } while (option < 1 || option > 2);

    switch(option){
        case 1:
            outputConsole(solutions);
            break;
        case 2:
            try {
                outputFile(solutions);
            } catch (IOException ex) {
                println("Failed to create file. Please try again...");
                output(solutions);
            }
            break;
    }
}

public static void outputConsole(Set<String> solutions){
    println((solutions.size() == 0) ? "There is no solution" : ("There are " + solutions.size() + "
solutions\n"));
    int i = 1;

    // print all results

```



```

        for (String solution : solutions){
            print(i + ". ");
            println(solution);
            i++;
        }
    }

    public static void outputFile(Set<String> solutions) throws IOException{
        // user input filename
        print("Output file name [____.txt] : ");
        String filename = scan.nextLine();

        // initialize buffer
        String output = "";

        // header
        output += (solutions.size() == 0) ? "There is no solution" : ("There are " + solutions.size() + "
solutions\n");

        // write contents
        int i = 1;
        for (String solution : solutions){
            output += (i + ". " + solution + '\n');
            i++;
        }

        // write to file
        FileWriter writer = new FileWriter("../test/" + filename);
        writer.write(output);
        writer.close();

        // success message
        print("Successfully added "+ filename +" to test folder.");
    }
}

class shortcut {

    public static void print(char item){
        System.out.print(item);
    }

    public static void print(int item){
        System.out.print(item);
    }

    public static void print(String item){

```

```

        System.out.print(item);
    }

    public static void println(){
        System.out.println();
    }

    public static void println(char item){
        System.out.println(item);
    }

    public static void println(int item){
        System.out.println(item);
    }

    public static void println(String item){
        System.out.println(item);
    }

    public static void printArr(int[] arr){
        Arrays.stream(arr).forEach(System.out::print);
        println();
    }
}

```

File : solution.java

```

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import datatype.*;

public class solution extends shortcut {
    // global variable
    public static int indexPermute = 0;
    public static int[][] permutations = new int[24][4];

    public static void solution(int[] nums) {
        // initialize set for solutions
        Set<String> solutions = new HashSet<>();

        // nums permutation
        permute(nums, 0);

        // loop through all possible combination
        for (int i = 0; i < permutations.length; i++){
            Double[] numTemp = new Double[4];

```

```

        int[] toProcess = permutations[i];

        // cast type to double
        for (int j=0; j < toProcess.length; j++){
            double temp = toProcess[j];
            numTemp[j] = temp;
        }

        // solve
        Set<StrDoub> ans = rec(numTemp);

        // append to solutions set
        for (StrDoub item : ans){
            solutions.add(item.expr);
        }
    }

    main.output(solutions);
}

// permute through numbers
public static void permute(int[] arr, int index) {
    if (index == arr.length) {
        // initialize temp array
        int[] tempArr = new int[arr.length];

        // copy to temp array
        System.arraycopy(arr, 0, tempArr, 0, arr.length);

        // assign array to permutations global variable
        permutations[indexPermute] = tempArr;

        // increment the index
        indexPermute++;

    } else {
        for (int i = index; i < arr.length; i++) {
            swap(arr, index, i);
            permute(arr, index + 1);
            swap(arr, index, i);
        }
    }
}

// swap items in array
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];

```

```

        arr[j] = temp;
    }

    public static StrDoub operate(char operator, StrDoub a, StrDoub b, Boolean bracket){
        // variable declaration
        StrDoub result = new StrDoub(null, 0.0);

        // assign expr
        result.expr = (bracket ? '(' : "") + a.expr + operator + b.expr + (bracket ? ')' : "");

        // assign res
        switch(operator){
            case '-':
                result.res = a.res-b.res;
                break;
            case '*':
                result.res = a.res*b.res;
                break;
            case '/':
                result.res = a.res/b.res;
                break;
            case '+':
                result.res = a.res+b.res;
                break;
        }

        return result;
    }

    public static Set<StrDoub> rec(Double[] arr){
        // initialize variables
        Double[] head;
        Double[] tail;

        // initialize results set
        Set<StrDoub> results = new HashSet<>();

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

        if (arr.length > 1){
            for (int i = 1; i < arr.length; i++){
                // split array
                head = Arrays.copyOfRange(arr, 0, i);
                tail = Arrays.copyOfRange(arr, i, arr.length);

                // recursion
            }
        }
    }

```

```

        Set<StrDoub> headRes = rec(head);
        Set<StrDoub> tailRes = rec(tail);

        // Loop through operators, headRes, and tailRes
        for(char operator : operators){
            for (StrDoub x : headRes){
                for (StrDoub y : tailRes){
                    if (operator != '/' || (operator == '/' && y.res != 0)){
                        StrDoub ans = operate(operator, x, y, arr.length != 4);
                        if ((arr.length == 4 && ans.res.equals(24.0)) || arr.length != 4){
                            results.add(ans);
                        }
                    }
                }
            }
        }
    }
}

else { // if arr.length == 1
    StrDoub res = new StrDoub(Integer.toString(arr[0].intValue()), arr[0]);
    results.add(res);
}

return results;
}
}

```

File : StrDoub.java

```

package datatype;

public class StrDoub {
    public String expr;
    public Double res;

    public StrDoub (String expr, Double res){
        this.expr = expr;
        this.res = res;
    }
}

```

BAB IV INPUT/OUTPUT PROGRAM






Kondisi 1		
Input from console Output from console		<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 1 User Input ===== Input : 3 3 3 3 Output options : [1] Console [2] Text File Enter option [1-2]: 1 There are 2 solutions 1. (3*(3*3))-3 2. ((3*3)*3)-3 Execution time : 39 milliseconds </pre>
Kondisi 2		
Input random Output from console		<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 2 Auto Generate Cards ===== Your cards are : 8 K 4 3 Output options : [1] Console [2] Text File Enter option [1-2]: 1 There are 8 solutions 1. 8/(3/(13-4)) 2. (8*(13-4))/3 3. (13-4)/(3/8) 4. (13-4)*(8/3) 5. (8/3)*(13-4) 6. ((13-4)*8)/3 7. 8*((13-4)/3) 8. ((13-4)/3)*8 Execution time : 26 milliseconds </pre>

Kondisi 3		
<p>Input from console Output from file</p>		<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 1 User Input ===== Input : A K 7 6 Output options : [1] Console [2] Text File Enter option [1-2]: 2 Output file name [____.txt] : ans.txt Successfully added ans.txt to test folder. Execution time : 41 miliseconds test > ≡ ans.txt 1 There is no solution </pre>
Kondisi 4		
<p>Input random Output from file</p>		<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 2 Auto Generate Cards ===== Your cards are : J 6 8 6 Output options : [1] Console [2] Text File Enter option [1-2]: 2 Output file name [____.txt] : ans.txt Successfully added ans.txt to test folder. Execution time : 24 miliseconds test > ≡ ans.txt 1 There are 6 solutions 2 1. 6-((8-11)*6) 3 2. 6+(6*(11-8)) 4 3. (6*(11-8))+6 5 4. 6-(6*(8-11)) 6 5. ((11-8)*6)+6 7 6. 6+((11-8)*6) </pre>

Kondisi 5		
Validation input		<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 1 User Input ===== Input : 1 2 3 4 Input : 2 3 4 5 Output options : [1] Console [2] Text File Enter option [1-2]: 3 Enter option [1-2]: 6 Enter option [1-2]: 1 There are 40 solutions </pre>
	<p>Apabila input tidak sesuai, program akan terus meminta input sampai user melakukan input dengan benar.</p>	<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 1 User Input ===== Input : j K q A Output options : [1] Console [2] Text File Enter option [1-2]: 1 There are 32 solutions 1. (1*(13-11))*12 2. ((13-11)*12)*1 3. ((1*13)-11)*12 </pre>

	<pre> Welcome to 24 Games Solver ===== Options: [0] Exit [1] Input Cards from Console [2] Auto Generate cards Enter option [0-2]: 1 User Input ===== Input : 3 4 5 6 7 8 9 10 Output options : [1] Console [2] Text File Enter option [1-2]: 1 There are 16 solutions 1. ((3-4)+5)*6 2. 6*(5-(4-3)) </pre> <p>Program juga menerima apabila terdapat space diantara kartu-kartu input user. Jika angka yang diinput user lebih dari 4, maka akan diambil 4 angka pertama.</p>
--	--

BAB V
TABEL PENILAIAN

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		

BAB IV

REPOSITORY GITHUB

https://github.com/JeffreyChow19/Tucil1_13521046.git