LAPORAN TUGAS KECIL 1 IF2211

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2022/2023

Disusun oleh:

Edia Zaki Naufal Ilman **13521141**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

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 (×), divisi (/) dan tanda kurung (()). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas.

Pada tugas kecil ini, akan digunakan Bahasa pemrograman C untuk membuat program permainan kartu 24 dan dengan mengikuti metode algoritma brute force. Program akan meminta user untuk memasukan empat buah kartu atau memilih untuk dimasukkan secara acak oleh program, kemudian akan dihitung jumlah solusi yang ada serta menampilkan solusi-solusi tersebut. Pada akhir program user dapat memilih untuk menyimpan hasil ke dalam sebuah text file.

.

BAB II
TEORI SINGKAT

2.1. Algoritma *Brute Force*

Algoritma *brute force* adalah suatu pendekatan algoritma yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma ini memecahkan suatu persoalan dengan cara yang sangat sederhana, langsung (*direct*), dan jelas (*obvious*). Pendekatan *brute force* membutuhkan volume komputasi dan waktu penyelesaian yang relative cukup besar. Meskipun begitu, algoritma *brute force* dapat menyelesaikan hamper semua persoalan dan sangat cocok untuk persoalan-persoalan kecil.

2.2. Exhaustive Search

Exhaustive search adalah Teknik pencarian solusi secara *brute force* untuk persoalan-persoalan kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan. *Exhaustive search* dilakukan dengan mengenumerasi setiap kemungkinan solusi dengan cara yang sistematis, mengevaluasi setiap kemungkinan, menyimpan solusi terbaik sejauh ini (*the best solution so far*), dan mengumumkan solusi terbaik (*the winner*). Karena *Exhaustive search* menggunakan algoritma *brute force*, *exhaustive search* juga membutuhkan volume komputasi dan waktu penyelesaian yang relative cukup besar.

BAB III
IMPLEMENTASI PROGRAM

3.1 *Tech Stack*

Untuk pembuatan program digunakan Bahasa pemrograman C dengan beberapa library tambahan, yaitu, stdlib.h, time.h, string.h, windows.h, dan unistd.h. Library stdlib.h digunakan untuk alokasi memori secara dinamis, time.h digunakan untuk menghitung waktu eksekusi program, string.h untuk mengolah berbagai value string, serta windows.h dan unistd.h digunakan untuk menggunakan *function* sleep pada OS windows dan linux agar program tetap terbuka selama waktu tertentu setelah program berakhir.

3.2 Penggunaan ADT

Pada program ini dibuat suatu struct dengan nama "*Combination*" yang membentuk suatu kombinasi urutan angka dan operasi. ADT Combination disusun oleh tiga komponen, yaitu, komponen "*type"* yang menyimpan bentuk dari kombinasi operasi dalam bentuk integer, komponen "*nums"* yang menyimpan empat buah integer untuk menentukan kombinasi dan posisi dari empat buah integer tersebut, dan komponen "*ops*" yang menyimpan empat buah char untuk menentukan kombinasi dan posisi dari empat symbol operasi.

Dibuat juga suatu struct dengan nama "*Buffer"* yang bertujuan menyimpan senarai variable dengan ADT Combination. ADT Buffer disusun oleh dua komponen, yaitu komponen "*buf*" yang menyimpan senarai variable ADT Combination atau kombinasi dari solusi-solusi yang berhasil didapatkan, dan komponen "l*en*" yang menyimpan Panjang efektif dari senarai solusi yang didapatkan.

3.3 Garis Besar Algoritma

a. Masukan Kartu

Pada awal program, user akan diminta untuk memilih cara memasukkan empat buah kartu dari himpunan kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dimana A = 1, J = 11, Q = 12, dan K = 13. Pengguna dapat memasukkan kartu dengan dipilihkan secara acak oleh program atau memilih sendiri. Jika pengguna memilih untuk dimasukkan secara acak, maka program akan mengambil empat buah bilangan antara 1-13 secara acak kemudian ditampilkan ke layar dalam bentuk format kartu. Jika pengguna memilih untuk memasukkan

kartu sendiri, maka pengguna dapat memasukkan kartu yang diinginkan baik dalam format kartu ataupun bilangan numerik.

b. Permutasi Posisi

Dalam penentuan bentuk operasi untuk mendapatkan hasil 24, perlu ditentukan posisi-posisi dari seluruh angka yang dimasukkan dan simbol operasi yang dapat dilakukan. Karena terdapat empat buah bilangan, maka jumlah kemungkinan posisi yang didapatkan adalah $P_4^4 = \frac{4!}{(4-4)!} = 24$. Dari himpunan simbol operasi {+, -, *, /} yang akan diambil tiga untuk setiap kemungkinan operasi dapat dihitung permutasinya sehingga didapatkan $P_3^4 = \frac{4!}{(4-3)!} = 24$. Maka, dengan mengalikan kedua hasil, didapatkan 576 kemungkinan operasi yang dapat dihasilkan.

Penempatan kurung dalam operasi juga mempengaruhi hasil akhir dari operasi, sehingga perlu diperhitungkan juga. Dengan bilangan empat angka, terdapat lima kemungkinan penempatan kurung yang dapat dihasilkan, yaitu:

- (A B) (C D)
- ((A B) C) D
- (A (B C)) D
- A ((B C) D)
- A (B (C D))

Jika dikalikan dengan kemungkinan posisi bilangan dan symbol operasi, maka didapatkan kembali jumlah perhitungan yang perlu dikomputasi adalah 2880 perhitungan.

Menggunakan *nested loop* dan *conditional statements*, seluruh perhitungan akan dicek satu persatu. Perhitungan yang mengasilkan bilangan 24 akan kemudian disimpan dalam sebuah array. Dalam pengecekan terdapat beberapa constraint sehingga tidak terjadi kesalahan, dalam komputasi. Constraint pertama adalah operasi pembagian (/) hanya dapat dilakukan jika menghasilkan bilangan yang bulat dan jika penyebut dari operasi bukan bilangan 0. Constraint yang kedua adalah jika terdapat kartu atau bilangan yang sama dalam masukkan, maka akan dicek terlebih dahulu eksistensi operasi sebelumnya agar tidak terdapat duplikat

operasi yang sama pada hasil akhir.

BAB IV
SOURCE
CODE

4.1 Source Code Header File (CardGame24.h)

```
#ifndef CARDGAME_H
#define CARDGAME_H

typedef struct
{
    int type;
    char ops[3];
    int nums[4];
}Combination;

typedef struct
{
    Combination buf[255];
    int len;
}Buffer;

char* convertToCard(int num);

int convertToNum(char* card);

void displayOps(Buffer buff);

void process(double nums[], int pos[], int posop[], Buffer *buff);

void checkDuplicate(int type, int pos[], int posop[], Buffer *buff);

void saveToText(Buffer buff, double time);

#endif
```

4.2. Source Code Driver File (CardGame24.c)

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <windows.h>

#include <unistd.h>

#include "CardGame24.h"


int main(){

    //Begin time1

    double time1 = 0.0;

    clock_t begin1 = clock();


    //Splash screen

    printf(".------..------.        .------..------..------..------.        .------..------..---
---..------..------.\n");

    printf("|2.--. ||4.--. | .-.    |C.--. ||A.--. ||R.--. ||D.--. | .-.    |G.--. ||A.--. ||M.-
-. ||E.--. ||!.--. |\n");

    printf("| (\\/) || :/\\: |(( ))   | :/\\: || (\\/) || :(): || :/\\: |(( ))   | :/\\: || (\\/)
|| (\\/) || (\\/) || (\\/) |\n");

    printf("| :\\/: || :\\/: | '-.-.  | :\\/: || :\\/: || ()() || (__) | '-.-.  | :\\/: || :\\/:
|| :\\/: || :\\/: || :\\/: |\n");

    printf("| '--'2|| '--'4|  (( )) | '--'C|| '--'A|| '--'R|| '--'D|  (( )) | '--'G|| '--'A|| '-
-'M|| '--'E|| '--'!|\n");

    printf("`------'`------'   '-'  `------'`------'`------'`------'   '-'  `------'`------'`---
---'`------'`------'\n");

    printf("\n");

    printf("|-----------------------------------------------------------------------------
------------------|\n");

    printf("|          Welcome          to          the          24          Card          Game!
|\n");
```

```
    printf("|-------------------------------------------------------------------------------
------------------|\n");
    printf("|       How       do       you       want       to       play       the       game?
|\n");
    printf("|                       1.                       Random                       cards
|\n");
    printf("|                       2.                       Custom                       cards
|\n");
    printf("|-------------------------------------------------------------------------------
------------------|\n");
    printf("| Your input: ");


    //End time1
    clock_t end1 = clock();
    time1 += (double)(end1 - begin1) / CLOCKS_PER_SEC;


    int input;
    scanf("%d", &input);
    while(input < 1 || input > 2){
        printf("|       Masukan       tidak       sesuai.       Silahkan       coba       lagi
|\n");
        printf("| Your input: ");
        scanf("%d", &input);
    }


    //Begin time2
    double time2 = 0.0;
    double time2_5 = 0.0;
    clock_t begin2 = clock();


    int num1, num2, num3, num4;
```

```
    char *card1 = malloc(sizeof(char) * 2);
    char *card2 = malloc(sizeof(char) * 2);
    char *card3 = malloc(sizeof(char) * 2);
    char *card4 = malloc(sizeof(char) * 2);
    if(input == 1){
        srand(time(NULL));

        num1 = (rand() % 12) + 1;
        num2 = (rand() % 12) + 1;
        num3 = (rand() % 12) + 1;
        num4 = (rand() % 12) + 1;

        // printf("THESE: %d %d %d %d\n", num1,num2,num3,num4);

        card1 = convertToCard(num1);
        card2 = convertToCard(num2);
        card3 = convertToCard(num3);
        card4 = convertToCard(num4);

        //End time2
        clock_t end2 = clock();
        time2 += (double)(end2 - begin2) / CLOCKS_PER_SEC;
        //printf("%s %s %s %s\n", card1, card2, card3, card4);

    }

    else{
        printf("|-------------------------------------------------------------------------------
----------------------|\n");
        printf("| Please input 4 cards: ");
```

I

```
    //End time2
    clock_t end2 = clock();
    time2 += (double)(end2 - begin2) / CLOCKS_PER_SEC;

    scanf("%s %s %s %s", card1, card2, card3, card4);

    //Begin time2_5
    clock_t begin2_5 = clock();
    num1 = convertToNum(card1);
    num2 = convertToNum(card2);
    num3 = convertToNum(card3);
    num4 = convertToNum(card4);

    while(num1 == 0 || num2 == 0 || num3 == 0 || num4 == 0){
        printf("| Invalid card(s)! Please try again\n");
        printf("| Please input 4 cards: ");
        scanf("%s %s %s %s", card1, card2, card3, card4);
        num1 = convertToNum(card1);
        num2 = convertToNum(card2);
        num3 = convertToNum(card3);
        num4 = convertToNum(card4);
    }
    //End time2_5
    clock_t end2_5 = clock();
    time2_5 += (double)(end2_5 - begin2_5) / CLOCKS_PER_SEC;

}
```

I

```
    //Begin time3
    double time3 = 0.0;
    clock_t begin3 = clock();

    printf("|--------------------------------------------------------------------------------
------------------|\n");
    printf("|--------------------------------------------------------------------------------
------------------|\n");
    printf("| Your cards: |                                          %s       %s       %s       %s
|\n", card1, card2, card3, card4);
    int count = 0;
    double nums[4];
    nums[0] = num1;
    nums[1] = num2;
    nums[2] = num3;
    nums[3] = num4;
    int pos[4];
    int posOP[4];
    Buffer buff;
    buff.len = 0;
    int current = 0;
    process(nums, pos, posOP, &buff);

    printf("|--------------------------------------------------------------------------------
------------------|\n");

    if(buff.len == 0){
        printf("|                                                  There are no solutions!
|\n");
    }
    else{
```

I

```
        printf("|                                                                    %d solutions found
||\n", buff.len);
        printf("|-------------------------------------------------------------------------------
----------------------|\n");
        displayOps(buff);
    }
    printf("|-------------------------------------------------------------------------------
------------------|\n");
    printf("|   Do   you   want   to   save   the   results   to   a   text   file?   (y/n)
|\n");
    printf("|-------------------------------------------------------------------------------
------------------|\n");
    char yon;
    printf("| Your input: ");

    //End time3
    clock_t end3 = clock();
    time3 += (double)(end3 - begin3) / CLOCKS_PER_SEC;

    scanf(" %c", &yon);
    //scanf("%c", &yon);
    while(yon != 'Y' && yon != 'y' && yon != 'N' && yon != 'n'){
        printf("| Invalid input. Please try again\n");
        printf("| Your input: ");
        scanf("%c", &yon);
    }

    double timeAmount = time1 + time2 + time2_5 + time3;

    if((yon == 'Y' || yon == 'y') && buff.len == 0){
        printf("|-------------------------------------------------------------------------
```

I

```
----------------------|\n");

        printf("| There are no solutions available, do you still want to save the results? (y/n)
|\n");

        printf("|-------------------------------------------------------------------------------
----------------------|\n");

        printf("| Your input: ");

        scanf(" %c", &yon);

        while(yon != 'Y' && yon != 'y' && yon != 'N' && yon != 'n'){

            printf("| Invalid input. Please try again\n");

            printf("| Your input: ");

            scanf("%c", &yon);

        }

    }

    if(yon == 'Y' || yon == 'y'){

        saveToText(buff, timeAmount);

    }

    printf("|-------------------------------------------------------------------------------
-------------------|\n");

    printf("|-------------------------------------------------------------------------------
-------------------|\n");

    printf("|                                          Thank you for playing the 24 Card Game!
|\n");

    printf("|-------------------------------------------------------------------------------
-------------------|\n");

    printf("| Execution time: %f                                                            |\n",
timeAmount);

    printf("|-------------------------------------------------------------------------------
-------------------|\n");

    sleep(3);

    return 0;

}
```

I

```
char* convertToCard(int num){
    if(num == 1){
        return "A";
    }
    if(num == 2){
        return "2";
    }
    if(num == 3){
        return "3";
    }
    if(num == 4){
        return "4";
    }
    if(num == 5){
        return "5";
    }
    if(num == 6){
        return "6";
    }
    if(num == 7){
        return "7";
    }
    if(num == 8){
        return "8";
    }
    if(num == 9){
        return "9";
    }
    if(num == 10){
```

I

```
        return "10";
    }
    if(num == 11){
        return "J";
    }
    if(num == 12){
        return "Q";
    }
    if(num == 13){
        return "K";
    }
}

int convertToNum(char* card){
    if(strcmp(card, "A") == 0 || strcmp(card, "1") == 0){
        return 1;
    }
    if(strcmp(card, "2") == 0){
        return 2;
    }
    if(strcmp(card, "3") == 0){
        return 3;
    }
    if(strcmp(card, "4") == 0){
        return 4;
    }
    if(strcmp(card, "5") == 0){
        return 5;
    }
```

I

```
    if(strcmp(card, "6") == 0){
        return 6;
    }
    if(strcmp(card, "7") == 0){
        return 7;
    }
    if(strcmp(card, "8") == 0){
        return 8;
    }
    if(strcmp(card, "9") == 0){
        return 9;
    }
    if(strcmp(card, "10") == 0){
        return 10;
    }
    if(strcmp(card, "J") == 0 || strcmp(card, "11") == 0){
        return 11;
    }
    if(strcmp(card, "Q") == 0 || strcmp(card, "12") == 0){
        return 12;
    }
    if(strcmp(card, "K") == 0 || strcmp(card, "13") == 0){
        return 13;
    }

    return 0;
}

void displayOps(Buffer buff){
```

I

```
    int res1, res2, res;
    char ops[4];
    ops[0] = '+'; ops[1] = '-'; ops[2] = '*'; ops[3] = '/';

    for(int i = 0; i < buff.len; i++){
        if(buff.buf[i].type == 1){
            printf("| (%d %c %d) %c (%d %c %d)\n", buff.buf[i].nums[0], buff.buf[i].ops[0],
buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],      buff.buf[i].ops[2],
buff.buf[i].nums[3]);
        }

        if(buff.buf[i].type == 2){
            printf("| ((%d %c %d) %c %d) %c %d\n", buff.buf[i].nums[0], buff.buf[i].ops[0],
buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],      buff.buf[i].ops[2],
buff.buf[i].nums[3]);
        }

        if(buff.buf[i].type == 3){
            printf("| (%d %c (%d %c %d)) %c %d\n", buff.buf[i].nums[0], buff.buf[i].ops[0],
buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],      buff.buf[i].ops[2],
buff.buf[i].nums[3]);
        }

        if(buff.buf[i].type == 4){
            printf("| %d %c ((%d %c %d) %c %d)\n", buff.buf[i].nums[0], buff.buf[i].ops[0],
buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],      buff.buf[i].ops[2],
buff.buf[i].nums[3]);
        }

        if(buff.buf[i].type == 5){
            printf("| %d %c (%d %c (%d %c %d))\n", buff.buf[i].nums[0], buff.buf[i].ops[0],
buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],      buff.buf[i].ops[2],
```

I

```
buff.buf[i].nums[3]);
        }
    }
}


void process(double nums[], int pos[], int posop[], Buffer *buff){
    char ops[4];
    ops[0] = '+'; ops[1] = '-'; ops[2] = '*'; ops[3] = '/';
    for(int i = 0; i < 4; i++){
        pos[0] = nums[i];
        for(int j = 0; j < 4; j++){
            if(j!= i){
                pos[1] = nums[j];
                for(int k = 0; k < 4; k++){
                    if(k != i && k != j){
                        pos[2] = nums[k];
                        for(int l = 0; l < 4; l++){
                            if(l != i && l != j && l !=k){
                                pos[3] = nums[l];


                                for(int z = 0; z < 4; z++){
                                    posop[0] = z;
                                    for(int x = 0; x < 4; x++){
                                        posop[1] = x;
                                        for(int c = 0; c < 4; c++){
                                            posop[2] = c;
                                            int res1, res2, res;
                                            char ops[4];
                                            ops[0] = '+'; ops[1] = '-'; ops[2] = '*'; ops[3] =
'/';
```

I

```
//TYPE 1
//RES1
if(posop[0] == 0){
    res1 = pos[0] + pos[1];
}
if(posop[0] == 1){
    res1 = pos[0] - pos[1];
}
if(posop[0] == 2){
    res1 = pos[0] * pos[1];
}
if(posop[0] == 3){
    if(pos[1] != 0 && pos[0] % pos[1] == 0){
        res1 = pos[0] / pos[1];
    }
    else{
        res1 = 1000;
    }
}

//RES2
if(posop[2] == 0){
    res2 = pos[2] + pos[3];
}
if(posop[2] == 1){
    res2 = pos[2] - pos[3];
}
if(posop[2] == 2){
```

I

```
        res2 = pos[2] * pos[3];
    }
    if(posop[2] == 3){
        if(pos[3] != 0 && pos[2] % pos[3] == 0){
            res2 = pos[2] / pos[3];
        }
        else{
            res2 = 1000;
        }
    }

    //RESULT
    if(posop[1] == 0){
        res = res1 + res2;
    }
    if(posop[1] == 1){
        res = res1 - res2;
    }
    if(posop[1] == 2){
        res = res1 * res2;
    }
    if(posop[1] == 3){
        if(res2 != 0 && res1 % res2 == 0){
            res = res1 / res2;
        }
        else{
            res = 1000;
        }
    }
```

I

```
//check
if(res == 24){
    checkDuplicate(1, pos, posop, &(*buff));
}

//TYPE 2
//printf("TYPE 2 TEST\n");
//RES1
if(posop[0] == 0){
    res1 = pos[0] + pos[1];
}
if(posop[0] == 1){
    res1 = pos[0] - pos[1];
}
if(posop[0] == 2){
    res1 = pos[0] * pos[1];
}
if(posop[0] == 3){
    if(pos[1] != 0 && pos[0] % pos[1] == 0){
        res1 = pos[0] / pos[1];
    }
    else{
        res1 = 1000;
    }
}

//RES2
if(posop[1] == 0){
```

I

```
        res2 = res1 + pos[2];
    }
    if(posop[1] == 1){
        res2 = res1 - pos[2];
    }
    if(posop[1] == 2){
        res2 = res1 * pos[2];
    }
    if(posop[1] == 3){
        if(pos[2] != 0 && res1 % pos[2] == 0){
            res2 = res1 / pos[2];
        }
        else{
            res2 = 1000;
        }
    }

    //RESULT
    if(posop[2] == 0){
        res = res2 + pos[3];
    }
    if(posop[2] == 1){
        res = res2 - pos[3];
    }
    if(posop[2] == 2){
        res = res2 * pos[3];
    }
    if(posop[2] == 3){
        if(pos[3] != 0 && res2 % pos[3] == 0){
```

I

```
            res = res2 / pos[3];
        }
        else{
            res = 1000;
        }
    }

    //CHECK
    if(res == 24){
        checkDuplicate(2, pos, posop, &(*buff));
    }

    //TYPE 3
    //printf("TYPE 3 TEST\n");
    //RES1
    if(posop[1] == 0){
        res1 = pos[1] + pos[2];
    }
    if(posop[1] == 1){
        res1 = pos[1] - pos[2];
    }
    if(posop[1] == 2){
        res1 = pos[1] * pos[2];
    }
    if(posop[1] == 3){
        if(pos[2] != 0 && pos[1] % pos[2] == 0){
            res1 = pos[1] / pos[2];
        }
        else{
```

I

```
            res1 = 1000;
        }
    }

    //RES2
    if(posop[0] == 0){
        res2 = pos[0] + res1;
    }
    if(posop[0] == 1){
        res2 = pos[0] - res1;
    }
    if(posop[0] == 2){
        res2 = pos[0] * res1;
    }
    if(posop[0] == 3){
        if(res1 != 0 && pos[0] % res1 == 0){
            res2 = pos[0] / res1;
        }
        else{
            res2 = 1000;
        }
    }

    //RESULT
    if(posop[2] == 0){
        res = res2 + pos[3];
    }
    if(posop[2] == 1){
        res = res2 - pos[3];
```

I

```
}
if(posop[2] == 2){
    res = res2 * pos[3];
}
if(posop[2] == 3){
    if(pos[3] != 0 && res2 % pos[3] == 0){
        res = res2 / pos[3];
    }
    else{
        res = 1000;
    }
}

//CHECK
if(res == 24){
    checkDuplicate(3, pos, posop, &(*buff));
}

//TYPE 4
//printf("TYPE 4 TEST\n");
//RES1
if(posop[1] == 0){
    res1 = pos[1] + pos[2];
}
if(posop[1] == 1){
    res1 = pos[1] - pos[2];
}
if(posop[1] == 2){
    res1 = pos[1] * pos[2];
```

I

```
}
if(posop[1] == 3){
    if(pos[2] != 0 && pos[1] % pos[2] == 0){
        res1 = pos[1] / pos[2];
    }
    else{
        res1 = 1000;
    }
}

//RES2
if(posop[2] == 0){
    res2 = res1 + pos[3];
}
if(posop[2] == 1){
    res2 = res1 - pos[3];
}
if(posop[2] == 2){
    res2 = res1 * pos[3];
}
if(posop[2] == 3){
    if(pos[3] != 0 && res1 % pos[3] == 0){
        res2 = res1 / pos[3];
    }
    else{
        res2 = 1000;
    }
}
```

I

```
//RESULT
if(posop[0] == 0){
    res = pos[0] + res2;
}
if(posop[0] == 1){
    res = pos[0] - res2;
}
if(posop[0] == 2){
    res = pos[0] * res2;
}
if(posop[0] == 3){
    if(res2 != 0 && pos[0] % res2 == 0){
        res = pos[0] / res2;
    }
    else{
        res = 1000;
    }
}

//CHECK
if(res == 24){
    checkDuplicate(4, pos, posop, &(*buff));
}

//TYPE 5
//printf("TYPE 5 TEST\n");
//RES1
if(posop[2] == 0){
    res1 = pos[2] + pos[3];
```

I

```
}
if(posop[2] == 1){
    res1 = pos[2] - pos[3];
}
if(posop[2] == 2){
    res1 = pos[2] * pos[3];
}
if(posop[2] == 3){
    if(pos[3] != 0 && pos[2] % pos[3] == 0){
        res1 = pos[2] / pos[3];
    }
    else{
        res1 = 1000;
    }
}

//RES2
if(posop[1] == 0){
    res2 = pos[1] + res1;
}
if(posop[1] == 1){
    res2 = pos[1] - res1;
}
if(posop[1] == 2){
    res2 = pos[1] * res1;
}
if(posop[1] == 3){
    if(res1 != 0 && pos[1] % res1 == 0){
        res2 = pos[1] / res1;
```

I

```
    }
    else{
        res2 = 1000;
    }
}

//RESULT
if(posop[0] == 0){
    res = pos[0] + res2;
}
if(posop[0] == 1){
    res = pos[0] - res2;
}
if(posop[0] == 2){
    res = pos[0] * res2;
}
if(posop[0] == 3){
    if(res2 != 0 && pos[0] % res2 == 0){
        res = pos[0] / res2;
    }
    else{
        res = 1000;
    }
}

//CHECK
if(res == 24){
    checkDuplicate(5, pos, posop, &(*buff));
}
```

I

```
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}


void checkDuplicate(int type, int pos[], int posop[], Buffer *buff){
    //printf("TEST HERE\n%d\n", (*buff).len);
    char ops[4];
    ops[0] = '+'; ops[1] = '-'; ops[2] = '*'; ops[3] = '/';


    if((*buff).len == 0){
        for(int i = 0; i < 4; i++){
            (*buff).buf[0].nums[i] = pos[i];
        }
        for(int i = 0; i < 3; i++){
            (*buff).buf[0].ops[i] = ops[posop[i]];
        }
        (*buff).buf[0].type = type;
        (*buff).len++;
    }


    else{
```

I

```
        int flag1;
        int flag2;
        for(int i = 0; i < (*buff).len; i++){
            flag1 = 1;
            flag2 = 1;
            //Check if it's the same type
            if((*buff).buf[i].type == type){
                for(int j = 0; j < 4; j++){
                    if((*buff).buf[i].nums[j] != pos[j]){
                        flag1 = 0;
                        break;
                    }
                }
                for(int j = 0; j < 3; j++){
                    if((*buff).buf[i].ops[j] != ops[posop[j]]){
                        flag2 = 0;
                        break;
                    }
                }
            }else{
                flag1 = 0;
                flag2 = 0;
            }

            if(flag1 == 1 && flag2 == 1){
                //printf("fail\n");
                break;
            }
        }
```

I

```
        //If there's no duplicates, then add
        if(flag1 == 0 || flag2 == 0){
            //printf("Succeed\n");
            for(int i = 0; i < 4; i++){
                (*buff).buf[(*buff).len].nums[i] = pos[i];
            }
            for(int i = 0; i < 3; i++){
                (*buff).buf[(*buff).len].ops[i] = ops[posop[i]];
            }
            (*buff).buf[(*buff).len].type = type;
            (*buff).len++;
        }
    }
}

void saveToText(Buffer buff, double time){
    printf("|-------------------------------------------------------------------------------------
-------------------|\n");
    char name[100];
    printf("| Type a name for the file: ");
    scanf(" %[^\n]%*c", name);
    char text[5] = ".txt";
    char filename[100];
    int i = 0; int j = 0;

    while (name[i] != '\0') {
        filename[j] = name[i];
        i++;
        j++;
```

I

```
    }


    i = 0;
    while (text[i] != '\0') {
        filename[j] = text[i];
        i++;
        j++;
    }
    filename[j] = '\0';



    FILE *file = fopen(filename, "w");


    if(buff.len != 0){
        fprintf(file, "%d solutions found\n", buff.len);
        for(int i = 0; i < buff.len; i++){
            if(buff.buf[i].type == 1){
                fprintf(file,   "(%d   %c   %d)   %c   (%d   %c   %d)\n",   buff.buf[i].nums[0],
buff.buf[i].ops[0],      buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],
buff.buf[i].ops[2], buff.buf[i].nums[3]);
            }else if(buff.buf[i].type == 2){
                fprintf(file,   "((%d   %c   %d)   %c   %d)   %c   %d\n",   buff.buf[i].nums[0],
buff.buf[i].ops[0],      buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],
buff.buf[i].ops[2], buff.buf[i].nums[3]);
            }else if(buff.buf[i].type == 3){
                fprintf(file,   "(%d   %c   (%d   %c   %d))   %c   %d\n",   buff.buf[i].nums[0],
buff.buf[i].ops[0],      buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],
buff.buf[i].ops[2], buff.buf[i].nums[3]);
            }else if(buff.buf[i].type == 4){
                fprintf(file,   "%d   %c   ((%d   %c   %d)   %c   %d)\n",   buff.buf[i].nums[0],
buff.buf[i].ops[0],      buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],
buff.buf[i].ops[2], buff.buf[i].nums[3]);
```

I

```
            }else{
                fprintf(file,   "%d   %c   (%d   %c   (%d   %c   %d))\n",   buff.buf[i].nums[0],
buff.buf[i].ops[0],      buff.buf[i].nums[1],      buff.buf[i].ops[1],      buff.buf[i].nums[2],
buff.buf[i].ops[2], buff.buf[i].nums[3]);
            }
        }
    }else{
        fprintf(file, "There are no solutions!\n");
    }
    fprintf(file, "Execution time: %f\n", time);
    fclose(file);
    printf("|-----------------------------------------------------------------------------------
------------------|\n");
    printf("|                                        The solutions have been successfully saved!
|\n");
}
```

I

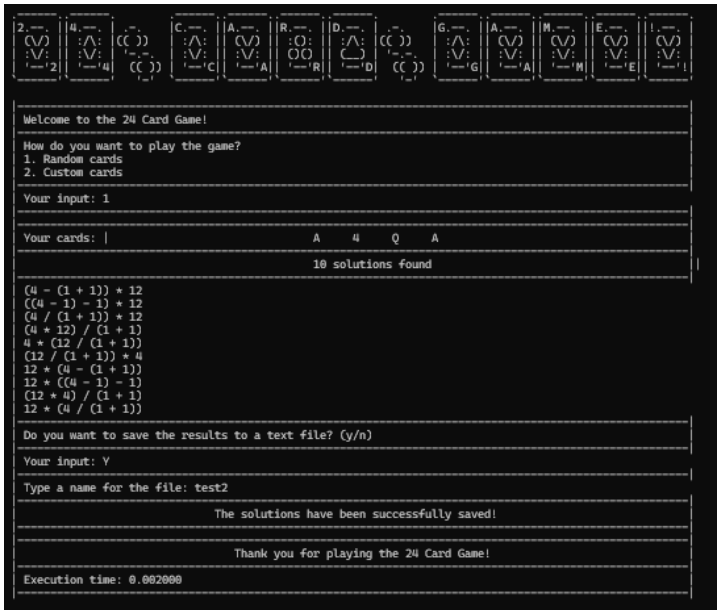
BAB V

EKSPERIMEN

6.1. Hasil Input dan Output Program

1. Testcase 1

2. Testcase 2

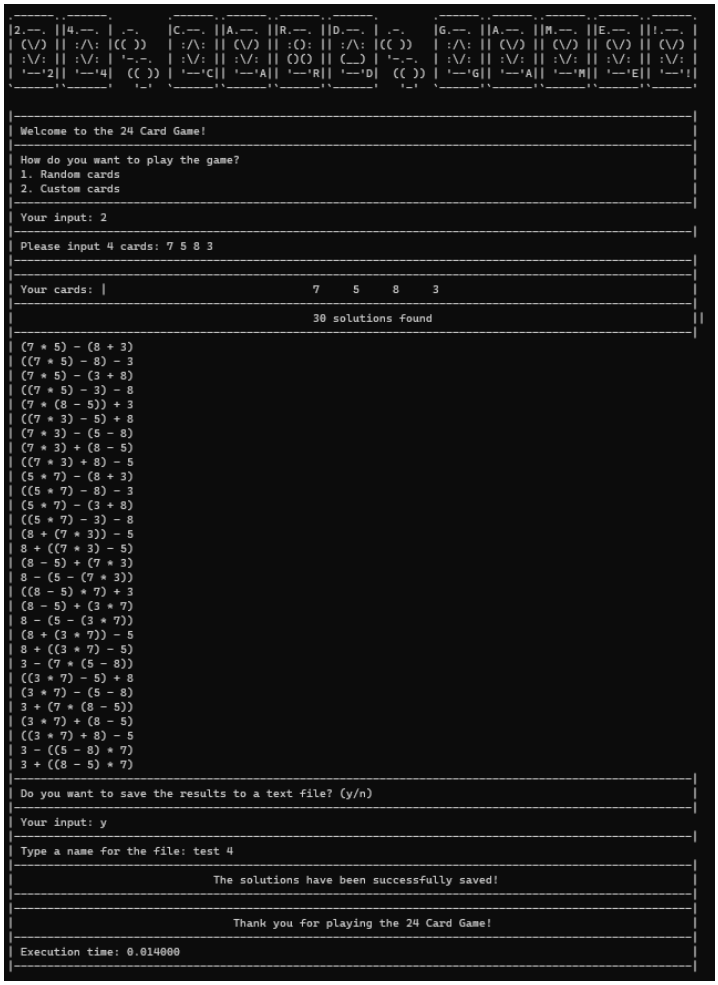


3. Testcase 3

4. Testcase 4

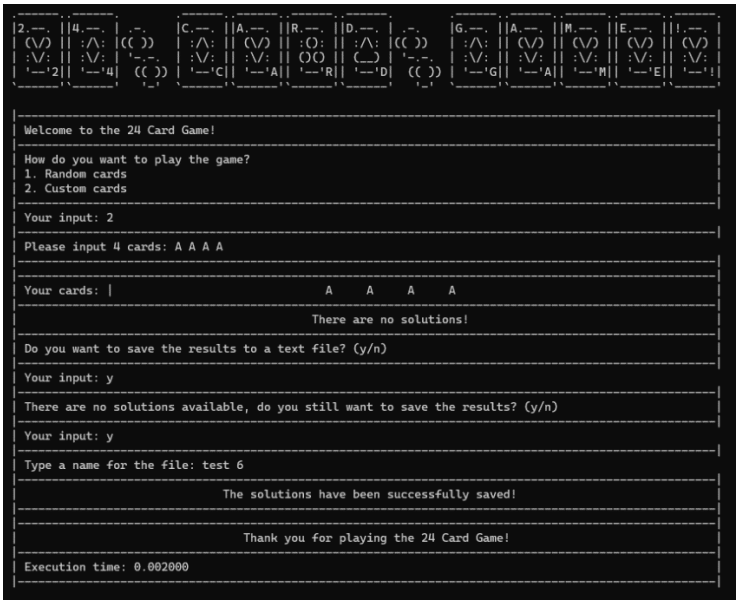
I

5. Testcase 5



6. Testcase 6

7. Testcase 7



8. Testcase 8

I

BAB VI
KESIMPULAN, SARAN, DAN REFLEKSI

6.2. Kesimpulan

Permainan kartu 24 dapat diselesaikan dengan berbagai cara, salah satunya adalah dengan menggunakan algoritma *brute force*. Algoritma *brute force* merupakan algoritma yang tidak efektif namun dapat menyelesaikan seluruh persoalan termasuk permainan kartu 24 ini. Dengan cara mengkomputasi seluruh kemungkinan bentuk operasi yang berjumlah 2880 kemungkinan, satu persatu, algoritma *brute force* membutuhkan volume dan waktu komputasi yang realtif besar dari algoritma lain. Hal tersebut dapat dilihat dari hasil program permainan kartu 24 dengan C yang telah berhasil dibuat.

Dengan menentukan seluruh kemungkinan posisi keempat bilangan dan simbol operasi, program dapat mengecek dan mendapatkan seluruh kemungkinan yang dapat menghasilkan bilangan 24. Beberapa constraint juga diberlakukan dalam pengecekan sehingga tidak terjadi duplikat dan kesalahan komputasi.

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

LAMPIRAN

Link github:
https://github.com/Ezaaan/Tucil1_13521141

Tabel pengerjaan:

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	✓	