# LAPORAN
# TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA SEMESTER II TAHUN 2022/2023 PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE


**oleh**

**Fakhri Muhammad Mahendra 13521045**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**
**INSTITUT TEKNOLOGI BANDUNG**
**BANDUNG**
**2023**

# DAFTAR ISI

# BAB 1

# ALGORITMA BRUTE FORCE

## 1.1 Langkah-Langkah Program

1. Program pertama-tama diberi input berupa 4 buah kartu dengan masing-masing nilai berada di antara 1-13. Input bisa dari pengguna maupun secara acak
2. Lalu program akan mencari semua permutasi yang mungkin dari dari 4 buah kartu tersebut dengan fungsi **four_permutation.** Program tetap memperhatikan apabila terdapat pengulangan dari dari angkat sehingga tidak perlu dihitung kembali. Semua hasil permutasi dari 4 buah kartu tersebut akan disimpan pada suatu vektor
3. Setiap permutasi pada vektor tersebut akan diiterasi dalam loop sedalam tiga level, dengan masing-masing loop merepresentasikan keempat operator yang mungkin dipakai pada tiap perhitungannya. Sebagai contoh, jika nilai indeks (i, j, k) dari ketiga loop didapat adalah (1, 1, 1) maka operator yang dipakai adalah (+, +, +), di kasus yang lain jika nilai tupel nya (2, 3, 1), maka operator yang dipakai adalah (- , *, / ) dan seterusnya.
4. Lalu permutasi yang terakhir adalah 5 kemungkinan. 5 kemungkinan ini diambil berdasarkan kombinasi-kombinasi unik dari penempatan tanda kurung pada perhitungan. Sebagai contoh, terdapat kombinasi kurung dengan pola sebagai berikut ( Z # Z ) # ( Z # Z) dan ( Z # ( Z # Z ) ) # Z, dengan Z adalah nilai yang dimiliki kartu dan tanda pagar (#) adalah kemungkinan operator yang mungkin

5. Secara total, pada kemungkinan terburuk, program akan mengevaluasi sebanyak 4! * 4^3 * 5 = 7680 expresi yang mungkin terbuat. Dari expresi-expresi matematika tersebut, dicari yang nilainya sama dengan 24 dan dicatat apabila ada yang memenuhi.

# BAB II

# SOURCE CODE PROGRAM DALAM BAHASA C++

## 2.1 module.cpp

```cpp
#include <iostream>
#include <bits/stdc++.h>
#include <vector>
#include <string>
#include <cstdlib>
#include <time.h>

using namespace std;

/**
 * @brief Fungsi menerima masukan bilangan bulat, dan mengeluarkan
 * string kartu yang merepresentasikan nilai tersebut
 *
 *
 * @param i masukan nilai bilangan bulat
 * @return string kartu yang direpresentasikan
 */
string cardToString(int i) {
    switch(i) {
        case 1:
            return "A";
        case 2:
            return "2";
        case 3:
            return "3";
        case 4:
            return "4";
        case 5:
            return "5";
        case 6:
            return "6";
        case 7:
            return "7";
        case 8:
            return "8";
        case 9:
            return "9";
        case 10:
            return "10";
        case 11:
            return "J";
        case 12:
            return "Q";
        case 13:
            return "K";
        default:
            return "0";
```

```
    }
}
/**
 * @brief Fungsi merubah representasi kartu dalam char
 * dengan nilai bilangan bulatnya
 *
 * @param c string masukan
 * @return int bilangan bulat yang merepresentasikan kartu
 * tersebut
 */
int stringToCard(char c) {
    switch(c) {
        case 'A':
            return 1;
        case '2':
            return 2;
        case '3':
            return 3;
        case '4':
            return 4;
        case '5':
            return 5;
        case '6':
            return 6;
        case '7':
            return 7;
        case '8':
            return 8;
        case '9':
            return 9;
        case 'J':
            return 11;
        case 'Q':
            return 12;
        case 'K':
            return 13;
        default:
            return 0;
    }
}
/**
 * @brief Fungsi yang mengembalikan suatu vektor beranggotakan 4
 * yang merepresentasikan nilai dari 4 kartu yang dipilih acak
 *
 * @return vector<int> Keluaran vector berisi 4 kartu
 */
vector<int> random_input () {
    // KAMUS LOKAL
    int i, holder;
    vector<int> card;
    // ALGORITMA
    srand(time(0));
    printf("Masukan Random:\n");
    for (i = 0; i <= 3; i++) {
```

```
        holder = (rand() % 12) + 1;
        card.push_back(holder);
        cout << cardToString(holder) + " ";
    }
    cout << "\n";
    return card;
}
/**
 * @brief Fungsi menghandle validasi sekaligus mengambil input
 * pengguna dari 4 kartu yang bisa dipilih
 *
 * @return vector<int> 4 kartu pilihan pengguna
 */
vector<int> user_input () {
    // KAMUS LOKAL
    bool valid;
    int i;
    int string_length;
    string input_string;
    vector<int> card;
    // ALGORITMA
    valid = false;
    while (!valid) {
        printf("Masukkan 4 simbol kartu\n");
        printf("Daftar simbol valid: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\n");
        printf("Input: ");
        cin.clear(); cin.sync();
        getline (cin, input_string);
        string_length = input_string.size();
        card.clear();
        i = 0;
        if ((string_length < 7) || (input_string[0] == ' ')) {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        if ((input_string[i] == '1') && (input_string[i+1] == '0')) {
            card.push_back(10);
            i = i + 2;
        } else if (stringToCard(input_string[i]) != 0) {
            card.push_back(stringToCard((input_string[i])));
            i = i + 1;
        } else {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        if (input_string[i] != ' ') {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        i = i+1;
        if ((input_string[i] == '1') && (input_string[i+1] == '0')) {
            card.push_back(10);
            i = i + 2;
        } else if (stringToCard(input_string[i]) != 0) {
```

```
            card.push_back(stringToCard((input_string[i])));
            i = i + 1;
        } else {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        if (input_string[i] != ' ') {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        i = i+1;
        if ((i + 1 < string_length) && (input_string[i] == '1') && (input_string[i+1] == '0')) {
            card.push_back(10);
            i = i + 2;
        } else if ((i < string_length) && (stringToCard(input_string[i]) != 0)) {
            card.push_back(stringToCard((input_string[i])));
            i = i + 1;
        } else {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        if (input_string[i] != ' ') {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        i = i+1;
        if ((i + 1 < string_length) && (input_string[i] == '1') && (input_string[i+1] == '0')) {
            card.push_back(10);
            i = i + 2;
        } else if ((i < string_length) && (stringToCard(input_string[i]) != 0)) {
            card.push_back(stringToCard((input_string[i])));
            i = i + 1;
        } else {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        if (i != string_length) {
            printf("Masukkan kurang tepat, coba diulangi lagi\n");
            continue;
        }
        valid = true;
    }
    return card;

}
/**
 * @brief Fungsi melakukan sorting dengan skema selection sort dari
 * 4 kartu yang sudah terpilih
 *
 * @param card vector yang berisi 4 kartu
 * @return vector<int> vector yang berisi 4 kartu terurut menaik
 */
vector<int> four_sort (vector<int> card) {
    // KAMUS LOKAL
```

```
    int i, pass, imin, temp;
    // ALGORITMA
    for (pass = 0; pass <= 2; pass++) {
        imin = pass;
        for (i = pass+1; i <= 3; i++) {
            if (card.at(i) < card.at(imin)) {
                imin = i;
            }
        }
        temp = card.at(pass);
        card.at(pass) = card.at(imin);
        card.at(imin) = temp;
    }
    return card;
}
/**
 * @brief Fungsi melakukan permutasi pada vektor berisi 4 kartu
 * dan mengembalikan vector yang berisi seluruh kemungkinan
 * permutasi tersebut
 *
 * @param card 4 kartu yang ingin di permutasi
 * @return vector<vector<int>> daftar semua permutasi yang mungkin
 */
vector<vector<int>> four_permutator (vector<int> card) {
    // KAMUS LOKAL
    int i, j, k;
    int temp;
    vector<int> sorted, card1, card2, card3;
    vector<vector<int>> result;
    // ALGORITMA
    sorted = four_sort(card);
    // Memilih acak dari 4 kartu terdepan dan menaruhnya ke paling belakang
    for (i = 0; i <= 3; i++) {
        card = sorted;
        if ((i >= 1) && (card.at(i-1) == card.at(i))) {
            continue;
        } else {
            temp = card.at(i);
            card.erase(card.begin() + i);
            card.push_back(temp);
            card1 = card;
            // Memilih acak dari 3 kartu terdepan dan menaruhnya ke paling belakang
            for (j = 0; j <= 2; j++) {
                card = card1;
                if ((j >= 1) && (card.at(j-1) == card.at(j))) {
                    continue;
                } else {
                    temp = card.at(j);
                    card.erase(card.begin() + j);
                    card.push_back(temp);
                    card2 = card;
                    // Memilih acak dari 2 kartu terdepan dan menaruhnya ke paling belakang
                    for (k = 0; k <= 1; k++) {
                        card = card2;
```

```
                        if ((k >= 1) && (card.at(k-1) == card.at(k))) {
                            continue;
                        } else {
                            temp = card.at(k);
                            card.erase(card.begin() + k);
                            card.push_back(temp);
                            // Menyimpan hasilnya pada vektor result
                            result.push_back(card);
                        }
                    }
                }
            }
        }
    }
    return result;
}
/**
 * @brief Fungsi menghitung nilai sebuah expresi matematika dari dua
 * bilangan bergantung kepada jenis operator yang menjadi parameter
 *
 * @param num1
 * @param num2
 * @param opType jenis operator yang ingin dibuat
 * @return double hasil perhitungannya
 */
double op(double num1, double num2, int opType) {
    switch(opType) {
        case 1:
            return num1 + num2;
        case 2:
            return num1 - num2;
        case 3:
            return num1 * num2;
        case 4:
            return num1 / num2;
        default:
            return NAN;
    }
}
/**
 * @brief Dari suatu 4 kartu, dilakukan perhitungan sesuai skema kurung yang menjadi
 * bagian dari parameter, dan operator operator yang dipilih
 *
 * @param card kumpulan nilai dari 4 kartu
 * @param bracketType skema kurung yang dipilih (1-5)
 * @param op1
 * @param op2
 * @param op3
 * @return double  hasilnya
 */
double evaluator(vector<int> card, int bracketType, int op1, int op2, int op3) {
    switch(bracketType) {
        case 1:
            return op(
```

```
                        op(
                            op(double(card.at(0)),
                                double(card.at(1)), op1),
                                    double(card.at(2)), op2),
                                        double(card.at(3)), op3);
        case 2:
            return op(double(card.at(0)),
                        op(double(card.at(1)),
                            op(double(card.at(2)),double(card.at(3)), op3), op2), op1);
        case 3:
            return op(
                        op(double(card.at(0)),
                            op(double(card.at(1)),
                                double(card.at(2)), op2), op1),
                                    double(card.at(3)),  op3);
        case 4:
            return op(double(card.at(0)),
                        op(
                            op(double(card.at(1)),
                                double(card.at(2)), op2),
                                    double(card.at(3)), op3), op1);
        case 5:
            return op(
                        op(double(card.at(0)), double(card.at(1)), op1),
                            op(double(card.at(2)), double(card.at(3)), op3), op2);
        default:
            return NAN;
    }
}
/**
 * @brief Fungsi mengembalikan string yang sesuai dari jenis operator
 * pada parameter
 *
 * @param opType
 * @return string
 */
string opToStr(int opType) {
    switch(opType) {
        case 1:
            return "+";
        case 2:
            return "-";
        case 3:
            return "*";
        case 4:
            return "/";
        default:
            return "";
    }
}
/**
 * @brief Mengmbalikan representasi string dari parameter-parameter berupa
 * tipe kurung, susuanan 4 kartu, dan operator-operator yang tersedia
 *
```

```
 * @param card vector berisi 4 kartu
 * @param bracketType skema kurung yang dipilih (1-5)
 * @param op1
 * @param op2
 * @param op3
 * @return string
 */
string stringConverter(vector<int> card, int bracketType, int op1, int op2, int op3) {
    switch(bracketType) {
        case 1:
            return "((" + to_string(card.at(0)) + opToStr(op1) + to_string(card.at(1)) + ")"
                + opToStr(op2) + to_string(card.at(2)) + ")" + opToStr(op3) + to_string(card.at(3));
        case 2:
            return to_string(card.at(0)) + opToStr(op1) + "(" + to_string(card.at(1))
                 + opToStr(op2) + "(" + to_string(card.at(2)) + opToStr(op3) + to_string(card.at(3))
+ "))";
        case 3:
            return "(" + to_string(card.at(0)) + opToStr(op1) + "(" + to_string(card.at(1))
                              +  opToStr(op2)  +  to_string(card.at(2))  +  "))"  +  opToStr(op3)  +
to_string(card.at(3));
        case 4:
            return to_string(card.at(0)) + opToStr(op1) + "((" + to_string(card.at(1))
                 + opToStr(op2) + to_string(card.at(2)) + ")" + opToStr(op3) + to_string(card.at(3))
+ ")";
        case 5:
            return "(" + to_string(card.at(0)) + opToStr(op1) + to_string(card.at(1)) + ")"
                + opToStr(op2) + "(" + to_string(card.at(2))  + opToStr(op3) + to_string(card.at(3))
+ ")";
        default:
            return "";
    }
}
```

## 2.2 main.cpp

```
#include <iostream>
#include <bits/stdc++.h>
#include <chrono>
#include <vector>
#include "module.cpp"
#include <cstdio>

using namespace std;
using namespace std::chrono;

int main () {
    // KAMUS
    bool valid;
    int i, j, k, m, holder;
    int op1, op2, op3, bracketType;
    int permutation_count, solution_count, string_length;
    string input_string, file_name;
    vector<int> card;
    vector<vector<int>> permutation_result;
```

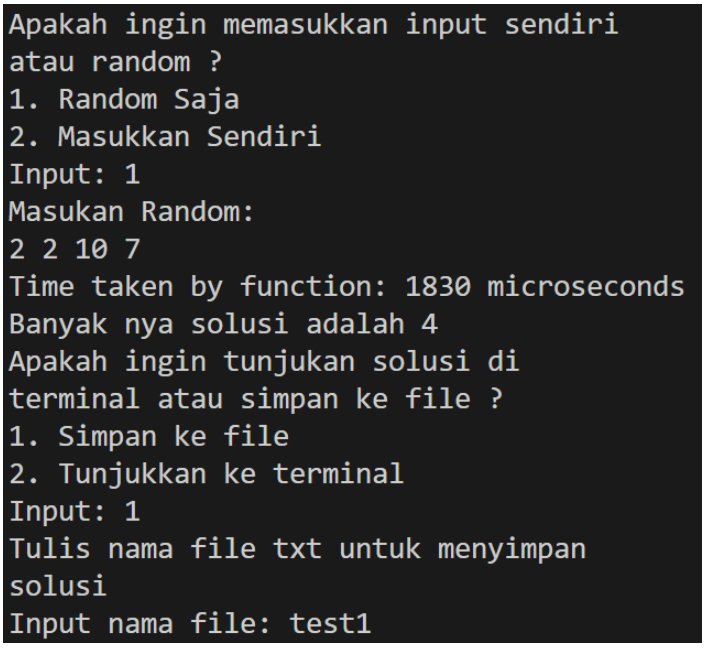
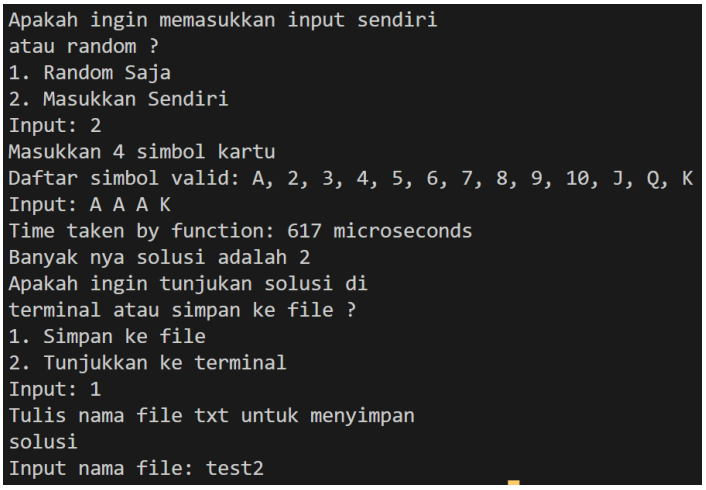
```
    vector<string> solution;

    // ALGORITMA UTAMA
    // Inisialisasi
    valid = false;
    // Validasi input pilihan 4 kartu
    while (!valid) {
        printf("Apakah ingin memasukkan input sendiri\natau random ?\n");
        printf("1. Random Saja\n");
        printf("2. Masukkan Sendiri\n");
        printf("Input: ");
        cin >> input_string;
        if (input_string == "1") {
            card = random_input();
            valid = true;
        } else if (input_string  == "2") {
            card = user_input();
            valid = true;
        } else {
            printf("Masukkan input yang sesuai (1 atau 2)\n");
        }
    }
    // Dimulainya perhitungan waktu
    auto start = high_resolution_clock::now();
    // Didapatkan daftar dari semua permutas 4 kartu yang mungkin
    permutation_result = four_permutator(card);
    permutation_count = permutation_result.size();

    // Dilakukan iterasi terhadap seluruh kemungkinan operator yang mungkin dan seluruh kemungkinan
    // skema kurung yang mungkin
    for (i = 0; i <= permutation_count - 1; i++) {
        for (op1 = 1; op1 <= 4; op1++) {
            for (op2 = 1; op2 <= 4; op2++) {
                for (op3 = 1; op3 <= 4; op3++) {
                    for (bracketType = 1; bracketType <= 5; bracketType++) {
                        // Dilakukan evaluasi dari nilai-nilai operator dan skema yang didapat
                        if (24 == evaluator(permutation_result.at(i), bracketType, op1, op2, op3)) {
                            // Jika nilai nya sudah pas 24, maka representasi string nya akan
dimasukan ke vector solution
                            solution.push_back(stringConverter(permutation_result.at(i),
bracketType, op1, op2, op3));
                        }
                    }
                }
            }
        }
    }
    // Perhentian menghitung waktu dan handling output waktu
    auto stop = high_resolution_clock::now();
    solution_count = solution.size();
    auto duration = duration_cast<microseconds>(stop - start);
    cout << "Time taken by function: "
         << duration.count() <<  " microseconds" << endl;
```

```
    // Handling kasus tidak ada solusi
    if (solution_count == 0) {
        printf("Tidak ada solusi yang ditemukan");
    } else {
        cout << "Banyak nya solusi adalah " << solution_count;
        cout << "\n";

        valid = false;
        // Handling valisasi input, dimana pengguna bisa memilih tempat
        // Untuk menaruh jawaban
        while (!valid) {
            printf("Apakah ingin tunjukan solusi di\nterminal atau simpan ke file ?\n");
            printf("1. Simpan ke file\n");
            printf("2. Tunjukkan ke terminal\n");
            printf("Input: ");
            scanf("%d", &holder);
            if (holder == 1) {
                printf("Tulis nama file txt untuk menyimpan\nsolusi\n");
                printf("Input nama file: ");
                cin >> file_name;
                file_name = "test/" + file_name + ".txt";
                freopen(file_name.c_str(),"w+",stdout);
                for (i = 0; i <= solution_count - 1; i++) {
                    cout << solution.at(i) + "\n";
                }
                valid = true;
            } else if (holder == 2) {
                for (i = 0; i <= solution_count - 1; i++) {
                    cout << solution.at(i) + "\n";
                }
                valid = true;
            } else {
                printf("Masukkan input yang sesuai (1 atau 2)\n");
            }

        }
    }



    return 0;
}
```

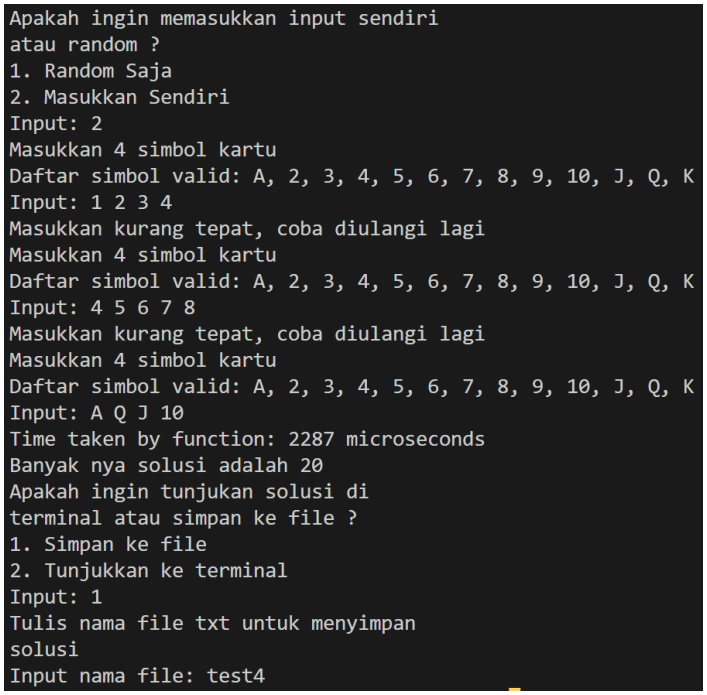
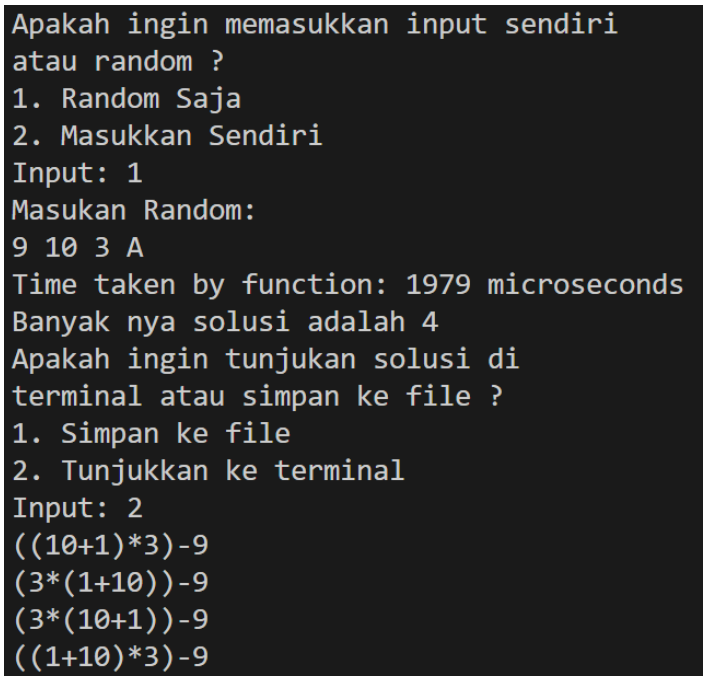
# BAB III

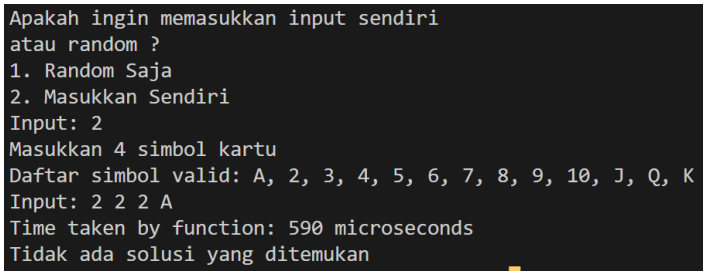
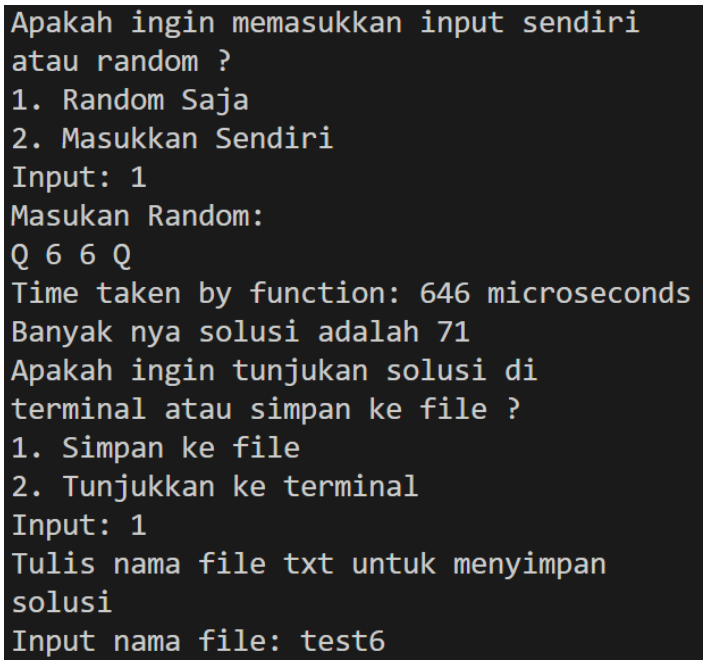
# EXPERIMEN

## 3.1 Eksperimen Program

| Input/Output | Deskripsi |
|---|---|
|  | 1. Input diambil secara acak dan output disimpan di test/test1.txt. |
|  | 2. Dilakukan pengujian terhadap pembacaan karakter non-integer.<br><br>Hasil disimpan pada test/test2.txt |

| | |
|---|---|
| Apakah ingin memasukkan input sendiri<br>atau random ?<br>1. Random Saja<br>2. Masukkan Sendiri<br>Input: 3<br>Masukkan input yang sesuai (1 atau 2)<br>Apakah ingin memasukkan input sendiri<br>atau random ?<br>1. Random Saja<br>2. Masukkan Sendiri<br>Input: 1<br>Masukan Random:<br>A 2 J J<br>Time taken by function: 1183 microseconds<br>Banyak nya solusi adalah 90<br>Apakah ingin tunjukan solusi di<br>terminal atau simpan ke file ?<br>1. Simpan ke file<br>2. Tunjukkan ke terminal<br>Input: 10<br>Masukkan input yang sesuai (1 atau 2)<br>Apakah ingin tunjukan solusi di<br>terminal atau simpan ke file ?<br>1. Simpan ke file<br>2. Tunjukkan ke terminal<br>Input: 1<br>Tulis nama file txt untuk menyimpan<br>solusi<br>Input nama file: test3 | 3. Dilakukan pengujian terhadap validasi input.<br><br>Hasil disimpan pada test/test3.txt |

| | |
|---|---|
|  | 4. Menguji validitas dari input kartu<br><br>Hasil disimpan pada test/test4.txt |
|  | 5. Menguji mengeluarkan jawaban pada terminal<br><br>Hasil disimpan pada test/test5.txt |

| | |
|---|---|
|  | 6. Menguji kombinasi kartu yang tidak memiliki solusi<br><br>Hasil tidak disimpan |
|  | 7. Menguji pada kombinasi yang memiliki banyak solusi<br><br>Hasil disimpan pada test/test6.txt |

# BAB IV

# KESIMPULAN, SARAN, DAN REFLEKSI

## 4.1 Kesimpulan

Brute Force merupakan salah satu pendekatan yang menarik dalam memilih algoritma. Walaupun pendekatan *brute force* sering dibilang yang paling *straightforward* namun masih diperlukan strategi yang matang, untuk memastikan seluruh kemungkinan sudah ter-*cover*.

## 4.2 Saran

Untuk kedepannya sebaiknya penulis melakukan *research* terhadap pendekatan algoritma yang lainnya, karena sebelumnya penulis mengerjakan tugas ini dengan pendekatan yang lebih mendekati pendekatan *backtracking* daripada *brute force* sehingga penulis harus menulis ulang kode yang telah dibuat.

# DAFTAR PUSTAKA

Munir, Rinaldi. (2022). *Tucil1-Stima-2023.pdf*. Institut Teknologi Bandung. Diakses pada 25 Januari 2023, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf

# LAMPIRAN

## TAUTAN REPOSITORY GITHUB

https://github.com/Fakhrimm/Tucil1_13521045

## TABEL ASISTEN

| 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 text | ✓ | |