TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA 2022/2023

Disusun oleh:

Louis Caesa Kesuma 13521069

TEKNIK INFORMATIKA

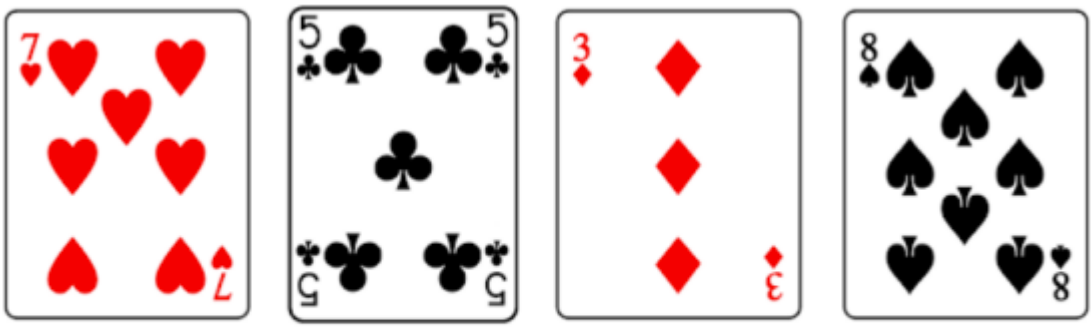
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

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.



MAKE IT 24

BAB II ALGORITMA BRUTE FORCE PERMAINAN 24

Algoritma *brute force* adalah pendekatan untuk menyelesaikan suatu permasalahan dengan metode yang lempang (*straightforward*). Algoritma ini umumnya didasarkan pada pernyataan persoalan (*problem statement*) dan definisi/konsep yang dilibatkan pada permasalahan tersebut. Algoritma tersebut memecahkan persoalan dengan cara yang sangat sederhana, langsung (*to the point*), dan jelas. Contohnya adalah algoritma pencarian beruntun dalam sebuah array, algoritma tersebut bisa dibuat dengan membuat algoritma yang mengakses setiap elemen pada array dan selesai ketika semua elemen pada array telah diakses.

Algoritma *brute force* tentunya dapat digunakan untuk mencari solusi dari permainan 24. Penggunaan algoritma *brute force* pada proses pencarian solusi permainan tersebut adalah dengan membuat setiap kondisi urutan angka (yang mewakilkan kartu-kartu) dan operator-operatornya. Tentunya sebagian besar kondisi tersebut dapat dieliminasi dengan aturan tertentu agar pembuatan program dan proses pencarian solusinya bisa dilakukan dengan lebih cepat. Langkah-langkah algoritma untuk mencari solusi-solusinya adalah sebagai berikut:

1. Membuat setiap kemungkinan operator dan kurung

   Misalkan tiap kartu sebagai variabel (a, b, c, dan d). Kemudian algoritma harus memiliki tiap kemungkinan susunan operator dan kurung yang mungkin, contohnya sebagai berikut:

   a + b + c + d

   a + b + c – d

   …

   …

   (a / (b + c)) * d

2. Meminimalisir kemungkinan-kemungkinan susunan operator dan kurung

   Untuk memudahkan pembuatan algoritmanya, maka susunan operator-operator dan kurung tersebut bisa diminimalisir agar tidak terjadi pengulangan. Operator-operator dan kurung yang repetitif bisa dihilangkan dari algoritma karena sudah diwakilkan oleh susunan yang lain. Contohnya seperti:

   (a + b) + c + d dan (a + (b +c)) + d dan a + b + c + d

   Susunan-susunan tersebut bisa diminimalisir karena solusi yang dihasilkan sesungguhnya sama dengan satu sama lain, dan akan terkesan repetitif.

   Algoritma kemudian bisa disederhanakan lagi dengan cara meminimalisir repetisi operator dan kurung yang jumlahnya sama, contohnya:

--+ dan -+-, atau

(a * b) + c + d dan a + b + (c * d)

Susunan tersebut bisa disederhanakan karena kita melakukan langkah berikutnya.

3. Membuat setiap susunan kartu yang mungkin

   Kartu-kartu yang awalnya dimasukkan pada program (misalkan a, b, c, dan d) harus dibuat susunan yang lainnya karena pada metode ini pembuatan setiap kondisi bisa diminimalisir dengan cara membuat susunan operator dan kurung yang tidak repetitif, akan tetapi algoritmanya akan mengetes untuk setiap susunan kartu-kartu yang mungkin dari ke-4 masukan. Susunan yang mungkin adalah:

   a b c d

   a b d c

   …

   …

   d a b c

4. Menyederhanakan susunan kartu yang repetitif

   Karena cara kerja algoritma ini adalah dengan mencoba setiap susunan kartu pada program, maka susunan kartu yang sama dapat dihilangkan untuk menghindari pengulangan solusi yang sama saat mencari solusi.

5. Mencoba setiap susunan kartu

   Susunan kartu-kartu tersebut kemudian dicoba satu persatu kedalam daftar susunan operator dan kurung yang telah dibuat dan disederhanakan pada langkah 1 dan 2.

Berikut adalah contoh source-*code* yang mungkin, sesuai dengan langkah-langkah diatas (dalam Bahasa C++), untuk lebih jelasnya bisa dilihat link repository pada lampiran.

- Algoritma pencari solusi

```
void algo24(int a, int b, int c, int d, fstream * file) {
    // perlu diperhatikan bahwa tidak perlu membuat kasus-kasus dengan operator yang mengulang karena nantinya akan dilakukan permutasi untuk setiap susunan angka
    // contohnya: jika ada ++-, maka tidak perlu membuat +-+ ataupun -++
    if (a + b + c + d == 24) {
        *file << a << " + " << b << " + " << c << " + " << d << "\n";
    } if (a + b + c - d == 24) {
        *file << a << " + " << b << " + " << c << " - " << d << "\n";
    } if (a + b + c * d == 24) {
        *file << a << " + " << b << " + " << c << " * " << d << "\n";
    } if (a*d + b*d + c == 24*d) { // a + b + c / d == 24
        *file << a << " + " << b << " + " << c << " / " << d << "\n";
    } if (a + b - c - d == 24) {
        *file << a << " + " << b << " - " << c << " - " << d << "\n";
    } if (a + b - c * d == 24) {
        *file << a << " + " << b << " - " << c << " * " << d << "\n";
    } if (a*d + b*d - c == 24*d) { // a + b - c / d == 24
        *file << a << " + " << b << " - " << c << " / " << d << "\n";
    } if (a + b * c * d == 24) {
        *file << a << " + " << b << " * " << c << " * " << d << "\n";
    } if (a*d + b * c*d == 24*d) { // a + b * c / d == 24
        *file << a << " + " << b << " * " << c << " / " << d << "\n";
    } if (a*c*d + b == 24*c*d) { // a + b / c / d == 24
        *file << a << " + " << b << " /" << c << " / " << d << "\n";
    } if (a - b - c - d == 24) {
        *file << a << " + " << b << " /" << c << " / " << d << "\n";
    } if (a*d - b*d - c*d == 24*d) {
        *file << a << " - " << b << " - " << c << " / " << d << "\n";
    } if (a - b - c * d == 24) {
        *file << a << " - " << b << " - " << c << " * " << d << "\n";
    } if (a*c*d - b == 24*c*d) {
        *file << a << " - " << b << " / " << c << " / " << d << "\n";
    } if (a*c - b * d == 24*c) {
        *file << a << " - " << b << " / " << c << " * " << d << "\n";
    } if (a - b * c * d == 24) {
        *file << a << " - " << b << " * " << c << " * " << d << "\n";
    } if (a == 24*b*c*d) {
        *file << a << " / " << b << " / " << c << " / " << d << "\n";
    } if (a * d == 24*b*c) {
        *file << a << " / " << b << " / " << c << " * " << d << "\n";
    } if (a * c * d == 24*b) {
        *file << a << " / " << b << " * " << c << " * " << d << "\n";
    } if (a * b * c * d == 24) {
        *file << a << " * " << b << " * " << c << " * " << d << "\n";
    }

    // kasus kurung berisikan 2 angka
    // perhatikan juga karena nanti akan dilakukan permutasi untuk susunan angkanya, maka tidak perlu membuat kurung untuk semua kasus
    // (a op b) + c + d bisa dianggap sama seperti diatas, jadi bisa dihiraukan
    if ((a + b) * c + d == 24) {
        *file << "(" << a << " + " << b << ")" << " * " << c << " + " << d << "\n";
    } if ((a + b) * c * d == 24) {
        *file << "(" << a << " + " << b << ")" << " * " << c << " * " << d << "\n";
    } if ((a + b) * c - d == 24) {
        *file << "(" << a << " + " << b << ")" << " * " << c << " - " << d << "\n";
    } if ((a + b) * c == 24*d) {
        *file << "(" << a << " + " << b << ")" << " * " << c << " / " << d << "\n";
    } if ((a + b) + d*c == 24*c) {
        *file << "(" << a << " + " << b << ")" << " / " << c << " + " << d << "\n";
    } if ((a + b) == 24*c*d) {
        *file << "(" << a << " + " << b << ")" << " / " << c << " / " << d << "\n";
    } if ((a + b) - d*c == 24*c) {
        *file << "(" << a << " + " << b << ")" << " / " << c << " - " << d << "\n";
    } if ((a - b) * c + d == 24) {
        *file << "(" << a << " - " << b << ")" << " * " << c << " + " << d << "\n";
    } if ((a - b) * c * d == 24) {
        *file << "(" << a << " - " << b << ")" << " * " << c << " * " << d << "\n";
    } if ((a - b) * c - d == 24) {
        *file << "(" << a << " - " << b << ")" << " * " << c << " - " << d << "\n";
    } if ((a - b) * c == 24*d) {
        *file << "(" << a << " - " << b << ")" << " * " << c << " / " << d << "\n";
    } if ((a - b) + d*c == 24*c) {
        *file << "(" << a << " - " << b << ")" << " / " << c << " + " << d << "\n";
    } if ((a - b) == 24*c*d) {
        *file << "(" << a << " - " << b << ")" << " / " << c << " / " << d << "\n";
    } if ((a - b) - d*c == 24*c) {
        *file << "(" << a << " - " << b << ")" << " / " << c << " - " << d << "\n";
    }

    // kasus kurung berisi 2 angka sebanyak 2 buah
    if ((a + b) * (c + d) == 24) {
        *file << "(" << a << " + " << b << ")" << " * " << "(" << c << " + " << d << ")" << "\n";
    } if ((a + b) == 24*(c + d)) {
        *file << "(" << a << " + " << b << ")" << " / " << "(" << c << " + " << d << ")" << "\n";
    } if ((a - b) * (c + d) == 24) {
        *file << "(" << a << " - " << b << ")" << " * " << "(" << c << " + " << d << ")" << "\n";
    } if ((a - b) == 24*(c + d)) {
        *file << "(" << a << " - " << b << ")" << " / " << "(" << c << " + " << d << ")" << "\n";
    } if ((a - b) * (c - d) == 24) {
        *file << "(" << a << " - " << b << ")" << " * " << "(" << c << " - " << d << ")" << "\n";
    } if ((a - b) == 24*(c - d)) {
        if ((a - b) & (c - d) != 0) { // khusus jika c-d == 0
            *file << "(" << a << " - " << b << ")" << " / " << "(" << c << " - " << d << ")" << "\n";
        }
    }

    // kasus kurung berisi tiga angka
    if ((a + b + c) * d == 24) {
```

```
    // kasus kurung berisi tiga angka
    if ((a + b + c) * d == 24) {
        *file << "(" << a << " + " << b << " + " << c << ")" << " * " << d << "\n";
    } if ((a + b + c) == 24*d) {
        *file << "(" << a << " + " << b << " + " << c << ")" << " / " << d << "\n";
    } if ((a + b - c) * d == 24) {
        *file << "(" << a << " + " << b << " - " << c << ")" << " * " << d << "\n";
    } if ((a + b - c) == 24*d) {
        *file << "(" << a << " + " << b << " - " << c << ")" << " / " << d << "\n";
    } if ((a - b - c) * d == 24) {
        *file << "(" << a << " - " << b << " - " << c << ")" << " * " << d << "\n";
    } if ((a - b - c) == 24*d) {
        *file << "(" << a << " - " << b << " - " << c << ")" << " / " << d << "\n";
    } if ((a + b * c) * d == 24) {
        *file << "(" << a << " + " << b << " * " << c << ")" << " * " << d << "\n";
    } if ((a + b * c) == 24*d) {
        *file << "(" << a << " + " << b << " * " << c << ")" << " / " << d << "\n";
    } if ((a - b * c) * d == 24) {
        *file << "(" << a << " - " << b << " * " << c << ")" << " * " << d << "\n";
    } if ((a - b * c) == 24*d) {
        *file << "(" << a << " - " << b << " * " << c << ")" << " / " << d << "\n";
    } if ((a*c + b) * d == 24*c) {
        *file << "(" << a << " + " << b << " / " << c << ")" << " * " << d << "\n";
    } if ((a*c + b) == 24*c*d) {
        *file << "(" << a << " + " << b << " / " << c << ")" << " / " << d << "\n";
    } if ((a*c - b) == 24*c*d) {
        *file << "(" << a << " - " << b << " / " << c << ")" << " / " << d << "\n";
    } if ((a*c - b) * d == 24*c) {
        *file << "(" << a << " - " << b << " / " << c << ")" << " * " << d << "\n";
    }
}
```

- Algoritma pencari susunan kartu

```
void setpermutasi(int a, int b, int c, int d, Matrix * M) {
    ELMT(*M, 0, 0) = a;
    ELMT(*M, 0, 1) = b;
    ELMT(*M, 0, 2) = c;
    ELMT(*M, 0, 3) = d;

    ELMT(*M, 1, 0) = a;
    ELMT(*M, 1, 1) = b;
    ELMT(*M, 1, 2) = d;
    ELMT(*M, 1, 3) = c;

    ELMT(*M, 2, 0) = a;
    ELMT(*M, 2, 1) = c;
    ELMT(*M, 2, 2) = b;
    ELMT(*M, 2, 3) = d;

    ELMT(*M, 3, 0) = a;
    ELMT(*M, 3, 1) = c;
    ELMT(*M, 3, 2) = d;
    ELMT(*M, 3, 3) = b;

    ELMT(*M, 4, 0) = a;
    ELMT(*M, 4, 1) = d;
    ELMT(*M, 4, 2) = b;
    ELMT(*M, 4, 3) = c;

    ELMT(*M, 5, 0) = a;
    ELMT(*M, 5, 1) = d;
    ELMT(*M, 5, 2) = c;
    ELMT(*M, 5, 3) = b;

    ELMT(*M, 6, 0) = b;
    ELMT(*M, 6, 1) = a;
    ELMT(*M, 6, 2) = c;
    ELMT(*M, 6, 3) = d;

    ELMT(*M, 7, 0) = b;
    ELMT(*M, 7, 1) = a;
    ELMT(*M, 7, 2) = d;
    ELMT(*M, 7, 3) = c;

    ELMT(*M, 8, 0) = b;
    ELMT(*M, 8, 1) = d;
    ELMT(*M, 8, 2) = a;
    ELMT(*M, 8, 3) = c;

    ELMT(*M, 9, 0) = b;
    ELMT(*M, 9, 1) = d;
    ELMT(*M, 9, 2) = c;
```

```
    ELMT(*M, 9, 2) = c;
    ELMT(*M, 9, 3) = a;

    ELMT(*M, 10, 0) = b;
    ELMT(*M, 10, 1) = c;
    ELMT(*M, 10, 2) = a;
    ELMT(*M, 10, 3) = d;

    ELMT(*M, 11, 0) = b;
    ELMT(*M, 11, 1) = c;
    ELMT(*M, 11, 2) = d;
    ELMT(*M, 11, 3) = a;

    ELMT(*M, 12, 0) = c;
    ELMT(*M, 12, 1) = a;
    ELMT(*M, 12, 2) = b;
    ELMT(*M, 12, 3) = d;

    ELMT(*M, 13, 0) = c;
    ELMT(*M, 13, 1) = a;
    ELMT(*M, 13, 2) = d;
    ELMT(*M, 13, 3) = b;

    ELMT(*M, 14, 0) = c;
    ELMT(*M, 14, 1) = b;
    ELMT(*M, 14, 2) = d;
    ELMT(*M, 14, 3) = a;

    ELMT(*M, 15, 0) = c;
    ELMT(*M, 15, 1) = b;
    ELMT(*M, 15, 2) = a;
    ELMT(*M, 15, 3) = d;

    ELMT(*M, 16, 0) = c;
    ELMT(*M, 16, 1) = d;
    ELMT(*M, 16, 2) = a;
    ELMT(*M, 16, 3) = b;

    ELMT(*M, 17, 0) = c;
    ELMT(*M, 17, 1) = d;
    ELMT(*M, 17, 2) = b;
    ELMT(*M, 17, 3) = a;

    ELMT(*M, 18, 0) = d;
    ELMT(*M, 18, 1) = a;
    ELMT(*M, 18, 2) = b;
    ELMT(*M, 18, 3) = c;
```

```
    ELMT(*M, 19, 0) = d;
    ELMT(*M, 19, 1) = a;
    ELMT(*M, 19, 2) = c;
    ELMT(*M, 19, 3) = b;

    ELMT(*M, 20, 0) = d;
    ELMT(*M, 20, 1) = b;
    ELMT(*M, 20, 2) = a;
    ELMT(*M, 20, 3) = c;

    ELMT(*M, 21, 0) = d;
    ELMT(*M, 21, 1) = b;
    ELMT(*M, 21, 2) = c;
    ELMT(*M, 21, 3) = a;

    ELMT(*M, 22, 0) = d;
    ELMT(*M, 22, 1) = c;
    ELMT(*M, 22, 2) = a;
    ELMT(*M, 22, 3) = b;

    ELMT(*M, 23, 0) = d;
    ELMT(*M, 23, 1) = c;
    ELMT(*M, 23, 2) = b;
    ELMT(*M, 23, 3) = a;

    EFF(*M) = 24;
}
```

- Prosedur untuk menyederhanakan susunan kartu

```
void sederhanakanpermutasi(Matrix M, Matrix * M2) {
    // untuk menyederhanakan permutasi yang telah dibuat

    for (int k = 0; k <= 3; k++) {
        ELMT(*M2, 0, k) =  ELMT(M, 0, k);
    }
    EFF(*M2) = 1;

    for (int i = 1; i <= 23; i++) {
        bool ada = false;
        int j = 0;

        while (j < EFF(*M2) & !ada) {
            int ctr = 0;
            for (int k = 0; k <= 3; k++) {
                if (ELMT(*M2, j, k) ==  ELMT(M, i, k)) {
                    ctr++;
                }
            }

            if (ctr == 4) {
                ada = true;
            }
            j++;
        }

        if (!ada) {
            for (int k = 0; k <= 3; k++) {
                ELMT(*M2, EFF(*M2), k) =  ELMT(M, i, k);
            }
            EFF(*M2)++;
        }
    }
}
```

- Prosedur untuk menyimpan solusi

```
void savesolusi(string from, string to) {
    std::ifstream  src(from, std::ios::binary);
    std::ofstream  dst(to, std::ios::binary);

    dst << src.rdbuf();
}
```

- Prosedur untuk menampilkan solusi

```
void displaysolusi(fstream* file) {
    string baris;

    while (getline(*file, baris)) {
        cout << baris << "\n";
    }
}
```

- Prosedur untuk menampilkan jumlah solusi

```
void displayjumlahsolusi(fstream* file) {
    int jumlahsolusi = 0;
    string baris;

    while (getline(*file, baris)) {
        jumlahsolusi++;
    }

    if (jumlahsolusi != 0) {
        cout << "Terdapat " << jumlahsolusi << " buah solusi\n";
    } else {
        cout << "Tidak ada solusi\n";
    }
}
```

- Main code program

```
int main() {
    // kamus untuk kartu
    int kartu[4];

    // kamus general
    int i;
    int option;
    int simpan;
    int lanjut;
    Matrix temp, M;
    string path = "../test/";

    bool end = false;

    fstream tempfile;

    // looping untuk program
    while (!end) {
        std::chrono::steady_clock::time_point awal, akhir;
        tempfile.open("../test/temp.txt", std::ios_base::app);

        cout << "Pilihan: \n";
        cout << "1. Input kartu dari user \n";
        cout << "2. Input kartu dari program (random) \n";
        cout << "Nomor selain opsi di atas untuk keluar \n\n";
        cout << "Pilihanmu: \n";
        cin >> option;
        cin.ignore();

        if (option == 1) {
            bool inputbenar = false;

            // inisiasi kartu
            for (int j = 0; j <= 3; j++) {
                kartu[j] = 0;
            }

            // memasukkan input pengguna kedalam array
            while (!inputbenar) {
                int ctr = 0;
                int ctrinput = 0;
                string tempinput;
                bool inputsalah = false;
                cout << "Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi) \n";
                cout << "Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13\n";

                // INPUT KARTU DARI PENGGUNA
                getline (cin, tempinput);
                if (tempinput.length() > 11) {
                    inputsalah = true;
                } else {
                    while (ctrinput <= tempinput.length()-1 & !inputsalah) { // simpan input kedalam variabel
                        if (kartu[ctr] == 0) {
                            if (tempinput[ctrinput] == '1') {
                                kartu[ctr] = 1;
                            } else if (tempinput[ctrinput] == 'A') {
                                kartu[ctr] = 1;
                            } else if (tempinput[ctrinput] == '2') {
                                kartu[ctr] = 2;
                            } else if (tempinput[ctrinput] == '3') {
                                kartu[ctr] = 3;
                            } else if (tempinput[ctrinput] == '4') {
                                kartu[ctr] = 4;
                            } else if (tempinput[ctrinput] == '5') {
                                kartu[ctr] = 5;
                            } else if (tempinput[ctrinput] == '6') {
                                kartu[ctr] = 6;
                            } else if (tempinput[ctrinput] == '7') {
                                kartu[ctr] = 7;
                            } else if (tempinput[ctrinput] == '8') {
                                kartu[ctr] = 8;
                            } else if (tempinput[ctrinput] == '9') {
                                kartu[ctr] = 9;
                            } else if (tempinput[ctrinput] == 'J') {
                                kartu[ctr] = 11;
                            } else if (tempinput[ctrinput] == 'Q') {
                                kartu[ctr] = 12;
                            } else if (tempinput[ctrinput] == 'K') {
                                kartu[ctr] = 13;
                            } else {
                                inputsalah = true;
                            }
                        } else {
                            if (tempinput[ctrinput] == '0') {
                                if (kartu[ctr] == 1) {
                                    kartu[ctr] = 10;
                                } else {
                                    inputsalah = true;
                                }
                            } else if (tempinput[ctrinput] == ' ') {
                                ctr++;
                            } else {
                                inputsalah = true;
                            }
                        }
                        ctrinput++;
                    }
```

```
            }
        }

        if (ctr != 3) {
            inputsalah = true;
        } else {
            // cek apakah ada yg belum terisi
            for (int j = 0; j <= 3; j++) {
                if (kartu[j] == 0) {
                    inputsalah = true;
                }
            }
        }

        if (inputsalah) {
            system("cls");
            cout << "Input salah, silahkan coba masukkan kartu-kartu yang benar\n";
        } else {
            inputbenar = true;
        }
    }

    // MENCARI SOLUSI
    awal = std::chrono::steady_clock::now();
    setpermutasi(kartu[0], kartu[1], kartu[2], kartu[3], &temp);
    sederhanakanpermutasi(temp, &M);

    for (i = 0; i<EFF(M); i++) {
        algo24(ELMT(M, i, 0), ELMT(M, i, 1), ELMT(M, i, 2), ELMT(M, i, 3), &tempfile);
    }
    akhir = std::chrono::steady_clock::now();

    tempfile.close();

    // MENAMPILKAN SOLUSI
    tempfile.open("../test/temp.txt", ios::in);
    displayjumlahsolusi(&tempfile);
    tempfile.close();
    tempfile.open("../test/temp.txt", ios::in);
    displaysolusi(&tempfile);
    tempfile.close();

    cout << "Apakah anda mau menyimpan solusinya?\n";
    cout << "1. Iya\n";
    cout << "Nomor selain opsi diatas untuk tidak\n";
    cin >> simpan;
    cin.ignore();

    // OPSI UNTUK MENYIMPAN FILE
    if (simpan == 1) {
        // menyimpan file
        cout << "Masukkan namafile, contoh: solusi.txt\n";
        cout << "Note: jika namafile sudah ada di directory, maka program akan meng-overwrite file tersebut\n";
        string namafile;
        fstream savefile;

        getline (cin, namafile);
        remove(namafile.c_str());
        savesolusi("../test/temp.txt", path+namafile);

        // closing
        tempfile.close();
        remove("../test/temp.txt");

        cout << "File telah disimpan\n";
    } else {
        // closing
        tempfile.close();
        remove("../test/temp.txt");
    }

    auto durasi = std::chrono::duration_cast<std::chrono::nanoseconds>(akhir-awal).count();
    std::cout << "Waktu eksekusi program : " << durasi << " nanodetik\n";

    // OPSI UNTUK MELANJUTKAN PROGRAM
    cout << "Apakah anda ingin menggunakan programnya lagi?\n";
    cout << "1. Iya\n";
    cout << "Nomor selain opsi diatas untuk tidak\n";
    cin >> lanjut;
    cin.ignore();
    if (lanjut == 1) {
        system("cls");
    } else {
        end = true;
    }
} else if (option == 2) {
    int random;
    int jumlahrandom;

    cout << "Kartu yang dirandom oleh program:\n";
    for (jumlahrandom = 0; jumlahrandom <= 3; jumlahrandom++) {
        random = (rand()) % 14;
        if (random == 13) {
            cout << "K ";
        } else if (random == 12) {
            cout << "Q ";
        } else if (random == 11) {
            cout << "J ";
```

```
                cout << "J ";
            } else if (random == 0) {
                random = 1;
                cout << random << " ";
            } else {
                cout << random << " ";
            }

            kartu[jumlahrandom] = random;
        }
        cout << "\n";

        // MENCARI SOLUSI
        auto awal = std::chrono::steady_clock::now();
        setpermutasi(kartu[0], kartu[1], kartu[2], kartu[3], &temp);
        sederhanakanpermutasi(temp, &M);

        for (i = 0; i<EFF(M); i++) {
            algo24(ELMT(M, i, 0), ELMT(M, i, 1), ELMT(M, i, 2), ELMT(M, i, 3), &tempfile);
        }
        auto akhir = std::chrono::steady_clock::now();

        tempfile.close();

        // MENAMPILKAN SOLUSI
        tempfile.open("../test/temp.txt", ios::in);
        displayjumlahsolusi(&tempfile);
        tempfile.close();
        tempfile.open("../test/temp.txt", ios::in);
        displaysolusi(&tempfile);
        tempfile.close();

        cout << "Apakah anda mau menyimpan solusinya?\n";
        cout << "1. Iya\n";
        cout << "Nomor selain opsi diatas untuk tidak\n";
        cin >> simpan;
        cin.ignore();

        // OPSI UNTUK MENYIMPAN FILE
        if (simpan == 1) {
            // menyimpan file
            cout << "Masukkan namafile, contoh: solusi.txt\n";
            cout << "Note: jika namafile sudah ada di directory, maka program akan meng-overwrite file tersebut\n";
            string namafile;
            fstream savefile;

            getline (cin, namafile);
            remove(namafile.c_str());
            savesolusi("../test/temp.txt", path+namafile);

            // closing
            tempfile.close();
            remove("../test/temp.txt");

            cout << "File telah disimpan\n";
        } else {
            // closing
            tempfile.close();
            remove("../test/temp.txt");
        }

        auto durasi = std::chrono::duration_cast<std::chrono::nanoseconds>(akhir-awal).count();
        std::cout << "Waktu eksekusi program : " << durasi << " nanodetik\n";

        // OPSI UNTUK MELANJUTKAN PROGRAM
        cout << "Apakah anda ingin menggunakan programnya lagi?\n";
        cout << "1. Iya\n";
        cout << "Nomor selain opsi diatas untuk tidak\n";
        cin >> lanjut;
        cin.ignore();
        if (lanjut == 1) {
            system("cls");
        } else {
            end = true;
        }

    } else {
        end = true;
    }
  }

  return 0;
}
```

- Struktur data untuk menyimpan susunan kartu (disimpan dalam ADT matrix yang berisi matrix dan integer effective yang berupa banyak elemen pada matrix tersebut)

```c
#ifndef MATRIX_H
#define MATRIX_H

// struktur matrix untuk permainan "Make it 24"
typedef struct
{
    int mem[24][4];
    int effective;
} Matrix;

// Selektor
#define ELMT(M, i, j) (M).mem[(i)][(j)]
#define EFF(M) (M).effective

#endif
```

BAB III PERCOBAAN

1. Percobaan untuk kartu 6 6 6 6

```
Pilihan:
1. Input kartu dari user
2. Input kartu dari program (random)
Nomor selain opsi di atas untuk keluar

Pilihanmu:
1
Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi)
Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13
6 6 6 6
Terdapat 1 buah solusi
6 + 6 + 6 + 6
Apakah anda mau menyimpan solusinya?
1. Iya
Nomor selain opsi diatas untuk tidak
0
Waktu eksekusi program : 1014000 nanodetik
Apakah anda ingin menggunakan programnya lagi?
1. Iya
Nomor selain opsi diatas untuk tidak
0
```

2. Percobaan untuk kartu A 8 9 Q

```
Pilihan:
1. Input kartu dari user
2. Input kartu dari program (random)
Nomor selain opsi di atas untuk keluar

Pilihanmu:
1
Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi)
Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13
A 8 9 Q
Terdapat 8 buah solusi
(1 + 9 - 8) * 12
(9 + 1 - 8) * 12
(12 - 1 * 9) * 8
(12 - 9) * 1 * 8
(12 - 9 * 1) * 8
(12 - 9 / 1) * 8
(12 - 9) * 8 * 1
(12 - 9) * 8 / 1
Apakah anda mau menyimpan solusinya?
1. Iya
Nomor selain opsi diatas untuk tidak
0
Waktu eksekusi program : 0 nanodetik
Apakah anda ingin menggunakan programnya lagi?
1. Iya
Nomor selain opsi diatas untuk tidak
0
```

3. Percobaan untuk kartu K Q 6 1

```
Pilihanmu:
1
Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi)
Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13
K Q 6 1
Terdapat 1 buah solusi
(13 - 1) * 12 / 6
Apakah anda mau menyimpan solusinya?
1. Iya
Nomor selain opsi diatas untuk tidak
0
Waktu eksekusi program : 0 nanodetik
Apakah anda ingin menggunakan programnya lagi?
1. Iya
Nomor selain opsi diatas untuk tidak
0
```

4. Percobaan untuk kartu Q 3 2 1

```
Pilihan:
1. Input kartu dari user
2. Input kartu dari program (random)
Nomor selain opsi di atas untuk keluar

Pilihanmu:
1
Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi)
Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13
Q 3 2 1
Terdapat 4 buah solusi
(3 + 1) * 12 / 2
(3 + 1 - 2) * 12
(1 + 3) * 12 / 2
(1 + 3 - 2) * 12
Apakah anda mau menyimpan solusinya?
1. Iya
Nomor selain opsi diatas untuk tidak
0
Waktu eksekusi program : 0 nanodetik
Apakah anda ingin menggunakan programnya lagi?
1. Iya
Nomor selain opsi diatas untuk tidak
0
```

5. Percobaan untuk kartu Q Q Q Q

```
Pilihan:
1. Input kartu dari user
2. Input kartu dari program (random)
Nomor selain opsi di atas untuk keluar

Pilihanmu:
1
Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi)
Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13
Q Q Q Q
Terdapat 2 buah solusi
12 + 12 + 12 - 12
(12 + 12) * 12 / 12
Apakah anda mau menyimpan solusinya?
1. Iya
Nomor selain opsi diatas untuk tidak
0
Waktu eksekusi program : 0 nanodetik
Apakah anda ingin menggunakan programnya lagi?
1. Iya
Nomor selain opsi diatas untuk tidak
0
```

6. Percobaan untuk kartu J J 2 A

```
Pilihan:
1. Input kartu dari user
2. Input kartu dari program (random)
Nomor selain opsi di atas untuk keluar

Pilihanmu:
1
Masukkan kartu-kartu dengan pemisah spasi, contoh: A 2 8 A atau 1 2 8 1 (kapital dan jumlah spasi berpengaruh, jadi untuk akhir dari input tidak boleh ada spasi)
Untuk J, Q dan K tidak boleh diisi dengan 11, 12, atau 13
J J 2 A
Terdapat 21 buah solusi
11 + 11 + 2 * 1
11 + 11 + 2 / 1
(11 + 11 + 2) * 1
(11 + 11 + 2) / 1
11 + 11 + 1 * 2
(11 + 11) * 1 + 2
(11 + 11) / 1 + 2
11 + 2 + 11 * 1
11 + 2 + 11 / 1
(11 + 2 + 11) * 1
(11 + 2 + 11) / 1
11 + 2 + 1 * 11
(11 + 2) * 1 + 11
(11 + 2) / 1 + 11
2 + 11 + 11 * 1
2 + 11 + 11 / 1
(2 + 11 + 11) * 1
(2 + 11 + 11) / 1
2 + 11 + 1 * 11
(2 + 11) * 1 + 11
(2 + 11) / 1 + 11
Apakah anda mau menyimpan solusinya?
1. Iya
Nomor selain opsi diatas untuk tidak
0
Waktu eksekusi program : 0 nanodetik
Apakah anda ingin menggunakan programnya lagi?
1. Iya
Nomor selain opsi diatas untuk tidak
0
```

LAMPIRAN

Link github: https://github.com/Ainzw0rth/Tucil1_13521069.git

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 ini memenuhi (berhasil mencapai 24)	✓	
5. Program dapat menyimpan solusi dalam file teks	✓	

REFERENSI

Chandra, Evita. 2015. Penerapan Algoritma Brute Force pada Permainan Kartu 24 (24 *game*). Bandung. Departemen Teknik Informatika ITB.