Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh:

Brian Kheng (13521049)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

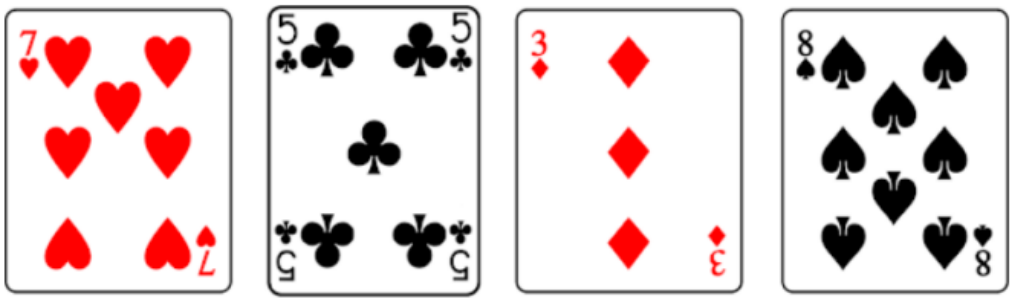
2023

DAFTAR ISI

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. (Paragraf di atas dikutip dari sini: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/MakalahStima-2016-038.pdf).



Gambar 1. Permainan Kartu 24 (Sumber:

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf)

SPESIFIKASI TUGAS

- Tulislah program sederhana dalam Bahasa C/C++/Java yang mengimplementasikan algoritma *Brute Force* untuk mencari seluruh solusi permainan kartu 24.
- **Input**: 4 angka/huruf yang terdiri dari:

  (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). Contoh input:

  A 8 9 Q

  Selain itu, input juga dapat dilakukan dengan program men-generate 4 angka/hurufnya sendiri secara random. Pengguna dapat memilih apakah program meminta input dari pengguna atau generate sendiri. Apabila masukan tidak sesuai, maka program menampilkan luaran "Masukan tidak sesuai" dan akan meminta ulang.

- **Output**:

  1. Banyaknya solusi yang ditemukan.

2. Solusi dari permainan kartu 24 ditampilkan di layar dan **terdapat opsi untuk menyimpan solusi** dalam file text. Untuk contoh kasus di atas A 8 9 Q, maka salah satu solusinya adalah:

   ((9 + A) - 8) * Q atau ((9 + 1) - 8) * 12 (Kedua jenis output dibebaskan)

   Note: Format penulisan output yang dicetak tidak harus persis contoh, yang penting merepresentasikan solusinya sudah cukup. Output apabila tidak ada solusi untuk pasangan kombinasi input, cukup ditampilkan "Tidak ada solusi". Untuk solusi setiap masukan, perlu dipertimbangkan urutan nilai (x1..x4), urutan operator, dan grouping dengan kurung yang mungkin.

   Gambaran apabila terdapat solusi:

A 8 9 Q
38 solutions found ((1 - 8) + 9) * 12 (1 - (8 - 9)) * 12 (1 * 8) * (12 - 9) .. .. .. (dst.)

   Di akhir, program akan menanyakan "Apakah ingin menyimpan solusi ?". Jika iya, program akan meminta sebuah nama untuk file teks dan semua solusi yang didapat akan disimpan dalam file text tersebut.

3. Waktu eksekusi program (tidak termasuk waktu pembacaan file input).

BAB II

TEORI SINGKAT

Algoritma *brute force* adalah metode yang digunakan untuk mencari solusi suatu masalah dengan cara mencoba semua kemungkinan yang ada. Algoritma ini sangat sederhana dan mudah diimplementasikan, tetapi juga sangat lambat dan tidak efisien saat digunakan untuk masalah yang memiliki jumlah kemungkinan yang besar. Dalam konteks komputasi, algoritma *brute force* digunakan untuk melakukan pencarian atau proses pemecahan kode, seperti *cracking password* atau pencarian kunci enkripsi.

Proses algoritma *brute force* dilakukan dengan mencoba semua kombinasi karakter yang mungkin sampai ditemukan kombinasi yang benar. Namun, metode ini sangat lambat dan memerlukan banyak sumber daya komputasi saat digunakan untuk masalah yang memiliki jumlah kombinasi yang besar. Oleh karena itu, algoritma *brute force* lebih cocok digunakan untuk masalah yang memiliki jumlah kombinasi yang terbatas.

Walaupun algoritma *brute force* memiliki kelemahan dalam efisiensi, namun algoritma ini masih digunakan dalam beberapa kasus karena kehandalannya dalam menemukan solusi yang benar. Contohnya dalam kasus *cracking password* yang memiliki panjang dan kompleksitas yang rendah. Namun algoritma ini tidak efektif digunakan dalam kasus *cracking password* yang memiliki panjang dan kompleksitas yang tinggi.

BAB III

ALGORITMA BRUTE FORCE

1. Cari semua permutasi dari 4 kartu. Dapat dilakukan dengan melakukan *backtracking* yakni, mula-mula fungsi rekursif kita menerima 2 argumen (indeks ke-l & indeks ke-r). Kita kemudian melakukan iterasi i dari [l…r] lalu menukar kartu di indeks ke-l dengan kartu di indeks ke-i dan memanggil fungsi rekursif kita kembali dengan mengubah argumen pertama menjadi indeks ke-l + 1 dan mengulang kembali langkah sebelumnya. Fungsi rekursif akan berhenti ketika indeks ke-l telah mencapai indeks ke-r sehingga telah ditemukan satu susunan kartu-kartu yang valid. Lalu, tidak lupa kita mengembalikan posisi kartu menjadi semula dengan menukarkan kembali kartu di indeks ke-l dengan kartu di indeks ke-i. Hal ini dilakukan agar kita dapat kembali menukar kartu di indeks ke-l dengan indeks ke-i selanjutnya (i + 1) dan *backtracking* dapat berhasil.
2. Ketika satu kombinasi susunan kartu-kartu telah ditemukan, maka kita perlu untuk mencari semua kombinasi susunan operasi di antara kartu-kartu tersebut guna membentuk suatu persamaan yang valid. Caranya ialah dengan kembali melakukan *backtracking*. Fungsi rekursif kita menerima 2 argumen (susunan persamaan sampai sekarang, indeks kartu berikutnya). Mula-mula kita memanggil fungsi rekursif kita dengan argumen (kartu pertama, 1), lalu untuk setiap operasi (+, -, x, /) kita tambahkan dari awal (operasi ke-1 terlebih dahulu, kemudian diikuti operasi berikutnya setelah operasi sebelumnya selesai dilakukan) ke persamaan dan juga kita selingi dengan menambahkan kartu dengan indeks di argumen ke-2 ke dalam persamaan, kemudian memanggil kembali fungsi rekursif kita namun menambahkan 1 pada argumen ke-2.

Kemudian, ulangi kembali langkah sebelumnya. Fungsi rekursif akan berhenti ketika argumen ke-2 kita mencapai 4 (indeks yang tidak terdefinisi) yang menyatakan bahwa susunan kartu beserta persamaannya telah lengkap. Tidak lupa, setelah fungsi rekursif dikembalikan, kita menghapus kartu terakhir dan operasi terakhir dari persamaan untuk diisikan dengan operasi selanjutnya.

3. Sesudah suatu persamaan valid telah berhasil dibentuk, kita akan melakukan uji dengan 4 buah kasus yang valid yakni, ((1 op 2) op 3) op 4, (1 op 2) op (3 op 4), (1 op (2 op 3)) op 4, dan 1 op ((2 op 3) op 4) dengan angka mewakili kartu dan “op” mewakili operasi. Operasi yang dilakukan ialah operasi bilangan riil, sehingga untuk mengetahui hasilnya ialah 24 akan dilakukan perbandingan selisih dengan epsilon. Jika selisihnya kurang dari epsilon, maka persamaan tersebut merupakan salah satu solusi valid.

SOURCE PROGRAM DALAM BAHASA C++

```
#include <bits/stdc++.h>

using namespace std;

mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());

const double EPS = 1e-7;
bool valid;
set<string> solutions;
vector<string> cards;
string ops[4] = {"+", "-", "*", "/"}, validCards[13] = {"A", "2", "3",
"4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"}, choice, inp;

void output()
{
    if (solutions.empty())
    {
        cout << "Tidak ada solusi!"
```

```
            << "\n";
    }
    else
    {
        cout << solutions.size() << ' ' << "solusi ditemukan!"
             << "\n"
             << "\n";
        for (auto it : solutions)
            cout << it << "\n";
    }
}

double operation(double c1, double c2, string op)
{
    if (op == "+")
        return c1 + c2;
    else if (op == "-")
        return c1 - c2;
    else if (op == "*")
        return c1 * c2;
    else if (op == "/")
        return c1 / c2;
}

void solveEquation(vector<string> eq)
{
    // Case 1: ((1 op 2) op 3) op 4
    if (abs(operation(operation(operation(stod(eq[0]), stod(eq[2]),
eq[1]), stod(eq[4]), eq[3]), stod(eq[6]), eq[5]) - 24) <= EPS)
    {
        solutions.insert("((" + eq[0] + " " + eq[1] + " " + eq[2] + ") "
+ eq[3] + " " + eq[4] + ") " + eq[5] + " " + eq[6]);
    }
    // Case 2: (1 op 2) op (3 op 4)
    if (abs(operation(operation(stod(eq[0]), stod(eq[2]), eq[1]),
operation(stod(eq[4]), stod(eq[6]), eq[5]), eq[3]) - 24) <= EPS)
    {
        solutions.insert("(" + eq[0] + " " + eq[1] + " " + eq[2] + ") "
+ eq[3] + " (" + eq[4] + " " + eq[5] + " " + eq[6] + ")");
    }
```

```
    // Case 3: (1 op (2 op 3)) op 4
    if (abs(operation(operation(stod(eq[0]), operation(stod(eq[2]),
stod(eq[4]), eq[3]), eq[1]), stod(eq[6]), eq[5]) - 24) <= EPS)
    {
        solutions.insert("(" + eq[0] + " " + eq[1] + " (" + eq[2] + " "
+ eq[3] + " " + eq[4] + ")) " + eq[5] + " " + eq[6]);
    }
    // Case 4: 1 op ((2 op 3) op 4)
    if (abs(operation(stod(eq[0]), operation(operation(stod(eq[2]),
stod(eq[4]), eq[3]), stod(eq[6]), eq[5]), eq[1]) - 24) <= EPS)
    {
        solutions.insert(eq[0] + " " + eq[1] + " ((" + eq[2] + " " +
eq[3] + " " + eq[4] + ") " + eq[5] + " " + eq[6] + ")");
    }
}

void findEquation(vector<string> eq, int idx)
{
    if (idx == 4)
    {
        solveEquation(eq);
        return;
    }
    for (auto op : ops)
    {
        eq.push_back(op);
        eq.push_back(cards[idx]);
        findEquation(eq, idx + 1);
        eq.pop_back();
        eq.pop_back();
    }
}

void tukar(string &s1, string &s2)
{
    string temp = s1;
    s1 = s2;
    s2 = temp;
}
```

```
void permutation(int l, int r)
{
    if (l == r)
    {
        findEquation({cards[0]}, 1);
    }
    else
    {
        for (int i = l; i <= r; i++)
        {
            tukar(*(cards.begin() + l), *(cards.begin() + i));
            permutation(l + 1, r);
            tukar(*(cards.begin() + l), *(cards.begin() + i));
        }
    }
}

vector<string> tokenize(string s)
{
    vector<string> tokens;
    string cur = "";
    for (auto it : s)
    {
        if (it == ' ')
        {
            tokens.push_back(cur);
            cur = "";
        }
        else
            cur += it;
    }
    tokens.push_back(cur);
    return tokens;
}

int main()
{
    // INPUT
    cout << "~~~ Selamat Datang Di Permainan Kartu 24! ~~~\n\n";
```

```
    cout << "1. Input dari pengguna\n2. Input auto-generate\n\nMasukan
pilihan: ";
    getline(cin, choice);

    while (choice != "1" && choice != "2")
    {
        cout << "\nMasukan tidak sesuai!\nMasukan pilihan: ";
        getline(cin, choice);
    }

    // TAKE CARD
    if (choice == "1")
    {
        do
        {
            valid = true;
            cout << "\nPilihan kartu:\n";
            getline(cin, inp);
            cards = tokenize(inp);
            if (cards.size() == 4)
            {
                for (int i = 0; i < 4; i++)
                {
                    if (cards[i] == "2" || cards[i] == "3" || cards[i]
== "4" || cards[i] == "5" || cards[i] == "6" ||
                        cards[i] == "7" || cards[i] == "8" || cards[i]
== "9" || cards[i] == "10")
                        continue;
                    else if (cards[i] == "A")
                        cards[i] = "1";
                    else if (cards[i] == "J")
                        cards[i] = "11";
                    else if (cards[i] == "Q")
                        cards[i] = "12";
                    else if (cards[i] == "K")
                        cards[i] = "13";
                    else
                        valid = false;
                }
            }
```

```
            else
                valid = false;
            if (!valid)
            {
                cout << "\nMasukan tidak sesuai!"
                     << "\n";
            }
            if (valid)
                cout << "\n";
        } while (!valid);
    }
    else
    {
        cout << "\nPilihan kartu:\n";
        for (int i = 0; i < 4; i++)
        {
            cards.push_back(validCards[rnd() % 13]);
            cout << cards[i] << ' ';
            if (cards[i] == "A")
                cards[i] = "1";
            else if (cards[i] == "J")
                cards[i] = "11";
            else if (cards[i] == "Q")
                cards[i] = "12";
            else if (cards[i] == "K")
                cards[i] = "13";
        }
        cout << "\n\n";
    }

    // FIND SOLUTION
    auto start = chrono::high_resolution_clock::now();
    permutation(0, 3);
    auto stop = chrono::high_resolution_clock::now();

    // OUTPUT
    output();

    cout << "\nApakah ingin menyimpan solusi? (Y/N)\nMasukan pilihan: ";
    getline(cin, choice);
```

```
    while (choice != "Y" && choice != "N" && choice != "y" && choice !=
"n")
    {
        cout << "\nMasukan tidak sesuai!\nMasukan pilihan: ";
        getline(cin, choice);
    }

    if (choice == "Y" || choice == "y")
    {
        string fileName;
        cout << "\nMasukan nama file keluaran (tanpa .txt): ";
        getline(cin, fileName);
        fileName = "../test/" + fileName + ".txt";
        freopen(fileName.c_str(), "w", stdout);
        output();
        freopen("CON", "w", stdout);
    }

    auto duration = chrono::duration_cast<chrono::milliseconds>(stop -
start);
    cout << "\nWaktu eksekusi program: "
         << duration.count() << " ms\n";

    cout << "\n\n~~~ Terima kasih telah menggunakan program ini!
~~~\n\n";

    return 0;
}
```

BAB IV

TESTING PROGRAM

1. Test Input: **A 8 9 Q**

```
~~~ Selamat Datang Di Permainan Kartu 24! ~~~

1. Input dari pengguna
2. Input auto-generate

Masukan pilihan: 1

Pilihan kartu:
A 8 9 Q

39 solusi ditemukan!

((1 * 12) - 9) * 8
((1 + 9) - 8) * 12
((1 - 8) + 9) * 12
((12 * 1) - 9) * 8
((12 - 9) * 1) * 8
((12 - 9) * 8) * 1
((12 - 9) * 8) / 1
((12 - 9) / 1) * 8
((12 / 1) - 9) * 8
((9 + 1) - 8) * 12
((9 - 8) + 1) * 12
(1 * (12 - 9)) * 8
(1 * 8) * (12 - 9)
(1 + (9 - 8)) * 12
(1 - (8 - 9)) * 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)
(12 - 9) / (1 / 8)
(8 * (12 - 9)) * 1
(8 * (12 - 9)) / 1

Apakah ingin menyimpan solusi? (Y/N)
Masukan pilihan: Y

Masukan nama file keluaran (tanpa .txt): test-1

Waktu eksekusi program: 14 ms


~~~ Terima kasih telah menggunakan program ini! ~~~
```

2. Test Input: **2 2 9 7**

```
~~~ Selamat Datang Di Permainan Kartu 24! ~~~

1. Input dari pengguna
2. Input auto-generate

Masukan pilihan: 1

Pilihan kartu:
2 2 9 7

Tidak ada solusi!

Apakah ingin menyimpan solusi? (Y/N)
Masukan pilihan: Y

Masukan nama file keluaran (tanpa .txt): test-2

Waktu eksekusi program: 14 ms


~~~ Terima kasih telah menggunakan program ini! ~~~
```

3. Test Input: **9 Q 9 9**

```
~~~ Selamat Datang Di Permainan Kartu 24! ~~~

1. Input dari pengguna
2. Input auto-generate

Masukan pilihan: 2

Pilihan kartu:
9 Q 9 9

7 solusi ditemukan!

((9 + 9) * 12) / 9
((9 + 9) / 9) * 12
(12 * (9 + 9)) / 9
(12 / 9) * (9 + 9)
(9 + 9) * (12 / 9)
(9 + 9) / (9 / 12)
12 * ((9 + 9) / 9)

Apakah ingin menyimpan solusi? (Y/N)
Masukan pilihan: Y

Masukan nama file keluaran (tanpa .txt): test-3

Waktu eksekusi program: 14 ms


~~~ Terima kasih telah menggunakan program ini! ~~~
```

4. Test Input: **A J Q K**

```
~~~ Selamat Datang Di Permainan Kartu 24! ~~~

1. Input dari pengguna
2. Input auto-generate

Masukan pilihan: 1

Pilihan kartu:
A J Q K

26 solusi ditemukan!

((1 * 13) - 11) * 12
((13 * 1) - 11) * 12
((13 - 11) * 1) * 12
((13 - 11) * 12) * 1
((13 - 11) * 12) / 1
((13 - 11) / 1) * 12
((13 / 1) - 11) * 12
(1 * (13 - 11)) * 12
(1 * 12) * (13 - 11)
(12 * (13 - 11)) * 1
(12 * (13 - 11)) / 1
(12 * 1) * (13 - 11)
(12 / 1) * (13 - 11)
(13 - (1 * 11)) * 12
(13 - (11 * 1)) * 12
(13 - (11 / 1)) * 12
(13 - 11) * (1 * 12)
(13 - 11) * (12 * 1)
(13 - 11) * (12 / 1)
(13 - 11) / (1 / 12)
1 * ((13 - 11) * 12)
12 * ((1 * 13) - 11)
12 * ((13 * 1) - 11)
12 * ((13 - 11) * 1)
12 * ((13 - 11) / 1)
12 * ((13 / 1) - 11)

Apakah ingin menyimpan solusi? (Y/N)
Masukan pilihan: Y

Masukan nama file keluaran (tanpa .txt): test-4

Waktu eksekusi program: 15 ms


~~~ Terima kasih telah menggunakan program ini! ~~~
```

5. Test Input: **Q 8 Q 8**

```
~~~ Selamat Datang Di Permainan Kartu 24! ~~~

1. Input dari pengguna
2. Input auto-generate

Masukan pilihan: 2

Pilihan kartu:
Q 8 Q 8

61 solusi ditemukan!

((12 * 8) / 8) + 12
((12 + 12) * 8) / 8
((12 + 12) + 8) - 8
(12 + (12 + 8)) - 8
(12 + (12 - 8)) + 8
(12 + (8 + 12)) - 8
(12 + (8 - 8)) + 12
(12 + 12) * (8 / 8)
(12 + 12) + (8 - 8)
(12 + 12) - (8 - 8)
(12 + 12) / (8 / 8)
(12 + 8) + (12 - 8)
(12 + 8) - (8 - 12)
(12 - (8 - 12)) + 8
(12 - (8 - 8)) + 12
(12 - 8) + (12 + 8)
(12 - 8) + (8 + 12)
(12 / (12 - 8)) * 8
(12 / (8 / 8)) + 12
(8 * (12 + 12)) / 8
(8 * (12 / 8)) + 12
(8 * 12) / (12 - 8)
(8 + (12 + 12)) - 8
(8 + (12 - 8)) + 12
(8 + 12) + (12 - 8)
(8 + 12) - (8 - 12)
(8 - (8 - 12)) + 12
(8 - 8) + (12 + 12)
(8 / (12 - 8)) * 12
(8 / (8 / 12)) + 12
(8 / 8) * (12 + 12)
12 + ((12 * 8) / 8)
12 + ((12 + 8) - 8)
12 + ((12 - 8) + 8)
12 + ((12 / 8) * 8)
12 + ((8 * 12) / 8)
12 + ((8 + 12) - 8)
12 + ((8 - 8) + 12)
12 + ((8 / 8) * 12)
12 - ((8 - 12) - 8)
12 - ((8 - 8) - 12)
12 / ((12 - 8) / 8)
8 * ((12 + 12) / 8)
8 + ((12 + 12) - 8)
8 + ((12 - 8) + 12)
8 - ((8 - 12) - 12)
8 / ((12 - 8) / 12)

Apakah ingin menyimpan solusi? (Y/N)
Masukan pilihan: Y

Masukan nama file keluaran (tanpa .txt): test-5

Waktu eksekusi program: 16 ms


~~~ Terima kasih telah menggunakan program ini! ~~~
```

6. Test Input: **A K 3 9**

```
~~~ Selamat Datang Di Permainan Kartu 24! ~~~

1. Input dari pengguna
2. Input auto-generate

Masukan pilihan: 2

Pilihan kartu:
A K 3 9

72 solusi ditemukan!

((9 + 3) + 13) - 1
((9 + 3) - 1) + 13
((9 - 1) + 13) + 3
((9 - 1) + 3) + 13
(13 + (3 + 9)) - 1
(13 + (3 - 1)) + 9
(13 + (9 + 3)) - 1
(13 + (9 - 1)) + 3
(13 + 3) + (9 - 1)
(13 + 3) - (1 - 9)
(13 + 9) + (3 - 1)
(13 + 9) - (1 - 3)
(13 - (1 - 3)) + 9
(13 - (1 - 9)) + 3
(13 - 1) + (3 + 9)
(13 - 1) + (9 + 3)
(3 + (13 + 9)) - 1
(3 + (13 - 1)) + 9
(3 + (9 + 13)) - 1
(3 + (9 - 1)) + 13
(3 + 13) + (9 - 1)
(3 + 13) - (1 - 9)
(3 + 9) + (13 - 1)
(3 + 9) - (1 - 13)
(3 - (1 - 13)) + 9
(3 - (1 - 9)) + 13
(3 - 1) + (13 + 9)
(3 - 1) + (9 + 13)
(9 + (13 + 3)) - 1
(9 + (13 - 1)) + 3
(9 + (3 + 13)) - 1
(9 + (3 - 1)) + 13
(9 + 13) + (3 - 1)
(9 + 13) - (1 - 3)
(9 + 3) + (13 - 1)
(9 + 3) - (1 - 13)
(9 - (1 - 13)) + 3
(9 - (1 - 3)) + 13
(9 - 1) + (13 + 3)
(9 - 1) + (3 + 13)
13 + ((3 + 9) - 1)
13 + ((3 - 1) + 9)
13 + ((9 + 3) - 1)
13 + ((9 - 1) + 3)
13 - ((1 - 3) - 9)
13 - ((1 - 9) - 3)
3 + ((13 + 9) - 1)
3 + ((13 - 1) + 9)
3 + ((9 + 13) - 1)
3 + ((9 - 1) + 13)
3 - ((1 - 13) - 9)
3 - ((1 - 9) - 13)
9 + ((13 + 3) - 1)
9 + ((13 - 1) + 3)
9 + ((3 + 13) - 1)
9 + ((3 - 1) + 13)
9 - ((1 - 13) - 3)
9 - ((1 - 3) - 13)

Apakah ingin menyimpan solusi? (Y/N)
Masukan pilihan: Y

Masukan nama file keluaran (tanpa .txt): test-6

Waktu eksekusi program: 15 ms


~~~ Terima kasih telah menggunakan program ini! ~~~
```

BAB V

KESIMPULAN

Algoritma *brute force* dapat digunakan untuk mencari semua permutasi dari 4 kartu yang diberikan, serta mencari semua kombinasi penyusunan simbol operasi untuk membentuk suatu persamaan yang valid dan menghasilkan nilai 24. Program dapat dijalankan dengan lumayan cepat, hal ini dikarenakan kompleksitas waktunya adalah O(N! . M!), dengan N = 4 dan M = 4, maka estimasi banyaknya operasi yang dilakukan hanyalah 576 operasi.

SARAN

Program dapat dijadikan sebagai referensi untuk menyelesaikan permainan lain yang serupa.

DAFTAR PUSTAKA

SUMBER PUSTAKA

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/Makalah Stima-2016-038.pdf
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf
3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

LAMPIRAN

1. Github : https://github.com/briankheng/Tucil1_13521049
2.

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	✔	