Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*

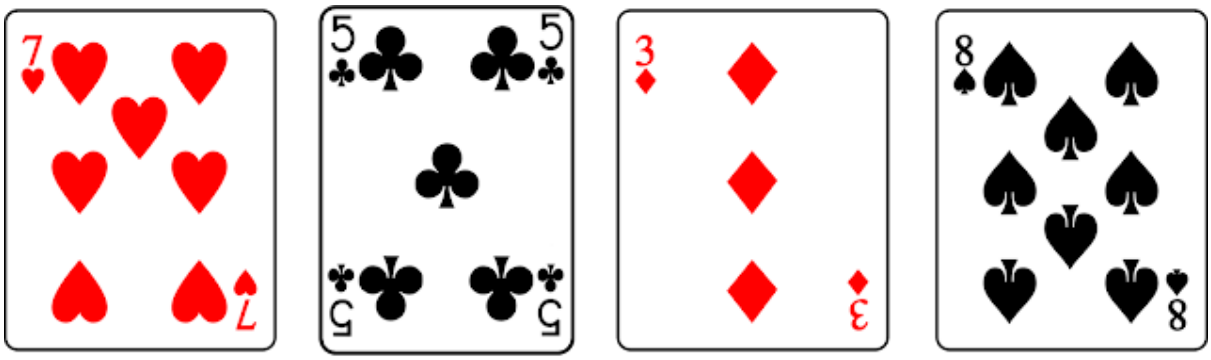
Ditulis oleh:

Muhammad Rifko Favian 13521075

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022/2023

BAGIAN 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.



Gambar 1. Permainan Kartu 24

BAGIAN II
LANGKAH ALGORITMA *BRUTE FORCE*

Berikut merupakan langkah-langkah mencari solusi dari penyelesaian permainan kartu 24 dengan algoritma *brute force* :

1. Memilih 4 kartu bebas yang akan digunakan mencari solusinya.
2. Mencari semua kemungkinan urutan dari 4 kartu tersebut dengan permutasi. Dalam *worst-case scenario* di mana tidak ada kartu yang memiliki angka/huruf yang sama, maka jumlah kemungkinannya adalah 4!.
3. Mencari semua kemungkinan posisi atau penempatan kurung. Di sini, ditemukan bahwa kemungkinan penempatan tersebut ada 5 banyak, yaitu :
    - (num op num) op (num op num)
    - ((num op num) op num) op num
    - (num op (num op num)) op num
    - num op ((num op num) op num)
    - num op (num op (num op num))

    di mana num adalah angka dan op adalah operasi matematika yang dipakai (+, -, *, /).
4. Mencari semua kemungkinan penggunaan operasi dalam ekspresi matematika tersebut. Karena terdapat 4 operasi yang bisa digunakan yang dipakai di 3 tempat berbeda, maka banyak kemungkinannya ada 4 * 4 * 4 atau 4^3.
5. Dari gabungan 3 poin percobaan sebelumnya, cari ekspresi mana yang menghasilkan solusi bilangan 24 dan catat semua kemungkinan ekspresi tersebut.

BAGIAN III
PROGRAM SOURCE

```
#include <iostream>
#include <set>
#include <string>
#include <chrono>
#include <fstream>

using namespace std;
set<string> solution;

bool isInputValid(string inp)
{
    if (inp.length() == 1)
    {
        if (!isdigit(inp[0]))
        {
            if (inp[0] != 'A' && inp[0] != 'J' && inp[0] != 'Q' && inp[0]
!= 'K')
            {
                return false;
            }
        }
        else
        {
            if (inp[0] == '0' or inp[0] == '1')
            {
                return false;
            }
        }
    }

    else if (inp.length() == 2)
    {
        if (inp[0] != '1' || inp[1] != '0')
        {
            return false;
        }
    }

    else
    {
        return false;
    }

    return true;
}

string numToString(int c)
{
    switch (c)
    {
    case 1:
        return "A";
```

```
    case 11:
        return "J";
    case 12:
        return "Q";
    case 13:
        return "K";
    default:
        return to_string(c);
    }
}

int stringToNum(string c)
{
    if (c[0] == 'A')
    {
        return 1;
    }
    else if (c[0] == 'J')
    {
        return 11;
    }
    else if (c[0] == 'Q')
    {
        return 12;
    }
    else if (c[0] == 'K')
    {
        return 13;
    }
    else
    {
        if (c.length() == 2 && c[0] == '1' && c[1] == '0')
        {
            return 10;
        }
        else
        {
            return c[0] - '0';
        }
    }
}

double count(double num1, double num2, char op)
{
    switch (op)
    {
    case '+':
        return num1 + num2;
    case '-':
        return num1 - num2;
    case '*':
        return num1 * num2;
    default:
        if (num2 != 0)
        {
            return num1 / num2;
        }
```

```
        else
        {
            return -9999;
        }
    }
}

void inp_way(int inp[4])
{
    while (true)
    {
        int choseCard;
        cout << "Pilih cara untuk memasukkan 4 kartu:\n";
        cout << "1. Dipilih dengan input manual\n";
        cout << "2. Dipilih secara random\n\n";
        cout << "Ketik 1 atau 2: ";

        cin >> choseCard;

        if (choseCard < 1 || choseCard > 2)
        {
            cout << "Masukan tidak valid! Silahkan coba lagi.\n\n";
            continue;
        }

        if (choseCard == 1)
        {
            string inp_card1, inp_card2, inp_card3, inp_card4;
            while (true)
            {
                cout << "Kartu ke-1 : ";
                cin >> inp_card1;
                cout << "Kartu ke-2 : ";
                cin >> inp_card2;
                cout << "Kartu ke-3 : ";
                cin >> inp_card3;
                cout << "Kartu ke-4 : ";
                cin >> inp_card4;

                if (!isInputValid(inp_card1) || !isInputValid(inp_card2) ||
!isInputValid(inp_card3) || !isInputValid(inp_card4))
                {
                    cout << "Masukan kartu tidak valid! Silahkan coba
lagi.\n\n";
                    continue;
                }
                else
                {
                    break;
                }
            }
            inp[0] = stringToNum(inp_card1);
            inp[1] = stringToNum(inp_card2);
            inp[2] = stringToNum(inp_card3);
            inp[3] = stringToNum(inp_card4);
            break;
        }
```

```
        else
        {
            for (int i = 0; i < 4; i++)
            {
                int random = rand() % 13 + 1;
                inp[i] = random;
            }
        }
        break;
    }
}

void kurung1(int a, int b, int c, int d, char op[4])
{
    double solve1, solve2, solve3;
    int i;
    string num1, num2, num3, num4;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                solve1 = count(a, b, op[i]);
                solve2 = count(c, d, op[k]);
                solve3 = count(solve1, solve2, op[j]);

                if (24 - solve3 <= 0.00001 && 24 - solve3 >= -0.00001)
                {
                    num1 = to_string(a);
                    num2 = to_string(b);
                    num3 = to_string(c);
                    num4 = to_string(d);
                    solution.insert("(" + num1 + " " + op[i] + " " + num2 +
") " + op[j] + " (" + num3 + " " + op[k] + " " + num4 + ")");
                }
            }
        }
    }
}

void kurung2(int a, int b, int c, int d, char op[4])
{
    double solve1, solve2, solve3;
    int i;
    string num1, num2, num3, num4;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                solve1 = count(a, b, op[i]);
                solve2 = count(solve1, c, op[j]);
                solve3 = count(solve2, d, op[k]);
                if (24 - solve3 <= 0.00001 && 24 - solve3 >= -0.00001)
```

```
                {
                    num1 = to_string(a);
                    num2 = to_string(b);
                    num3 = to_string(c);
                    num4 = to_string(d);
                    solution.insert("((" + num1 + " " + op[i] + " " + num2
+ ") " + op[j] + " " + num3 + ") " + op[k] + " " + num4);
                }
            }
        }
    }
}

void kurung3(int a, int b, int c, int d, char op[4])
{
    double solve1, solve2, solve3;
    int i;
    string num1, num2, num3, num4;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                solve1 = count(b, c, op[j]);
                solve2 = count(a, solve1, op[i]);
                solve3 = count(solve2, d, op[k]);
                if (24 - solve3 <= 0.00001 && 24 - solve3 >= -0.00001)
                {
                    num1 = to_string(a);
                    num2 = to_string(b);
                    num3 = to_string(c);
                    num4 = to_string(d);
                    solution.insert("(" + num1 + " " + op[i] + " (" + num2
+ " " + op[j] + " " + num3 + ")) " + op[k] + " " + num4);
                }
            }
        }
    }
}

void kurung4(int a, int b, int c, int d, char op[4])
{
    double solve1, solve2, solve3;
    int i;
    string num1, num2, num3, num4;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                solve1 = count(b, c, op[j]);
                solve2 = count(solve1, d, op[k]);
                solve3 = count(a, solve2, op[i]);
                if (24 - solve3 <= 0.00001 && 24 - solve3 >= -0.00001)
                {
```

```
                    num1 = to_string(a);
                    num2 = to_string(b);
                    num3 = to_string(c);
                    num4 = to_string(d);
                    solution.insert(num1 + " " + op[i] + " ((" + num2 + " "
+ op[j] + " " + num3 + ") " + op[k] + " " + num4 + ")");
                }
            }
        }
    }
}

void kurung5(int a, int b, int c, int d, char op[4])
{
    double solve1, solve2, solve3;
    int i;
    string num1, num2, num3, num4;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                solve1 = count(c, d, op[k]);
                solve2 = count(b, solve1, op[j]);
                solve3 = count(a, solve2, op[i]);
                if (24 - solve3 <= 0.00001 && 24 - solve3 >= -0.00001)
                {
                    num1 = to_string(a);
                    num2 = to_string(b);
                    num3 = to_string(c);
                    num4 = to_string(d);
                    solution.insert(num1 + " " + op[i] + " (" + num2 + " "
+ op[j] + " (" + num3 + " " + op[k] + " " + num4 + "))");
                }
            }
        }
    }
}

int main()
{
    char op[4] = {'+', '-', '*', '/'};
    int inp[4];
    int count;
    int j;
    srand(time(NULL));
    inp_way(inp);

    string inp_card[4];
    inp_card[0] = numToString(inp[0]);
    inp_card[1] = numToString(inp[1]);
    inp_card[2] = numToString(inp[2]);
    inp_card[3] = numToString(inp[3]);

    auto start = std::chrono::high_resolution_clock::now();
```

```
    for (int a = 0; a < 4; a++)
    {
        for (int b = 0; b < 4; b++)
        {
            for (int c = 0; c < 4; c++)
            {
                for (int d = 0; d < 4; d++)
                {
                    if ((a != b) && (a != c) && (a != d) && (b != c) && (b
!= d) && (c != d))
                    {
                        kurung1(inp[a], inp[b], inp[c], inp[d], op);
                        kurung2(inp[a], inp[b], inp[c], inp[d], op);
                        kurung3(inp[a], inp[b], inp[c], inp[d], op);
                        kurung4(inp[a], inp[b], inp[c], inp[d], op);
                        kurung5(inp[a], inp[b], inp[c], inp[d], op);
                    }
                }
            }
        }
    }
    auto end = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<chrono::microseconds>(end -
start);
    double duration_sec = (double)duration.count() / 1000000;

    cout << "\nKartu yang terpilih/dipilih:\n";
    cout << inp_card[0] << " " << inp_card[1] << " " << inp_card[2] << " "
<< inp_card[3] << endl;
    count = solution.size();

    if (count > 0)
    {
        cout << count << " solutions found\n";
    }
    else
    {
        cout << "Tidak ada solusi\n";
    }

    for (auto j = solution.begin(); j != solution.end(); ++j)
    {
        cout << *j << endl;
    }
    cout << "\nExecution time: " << duration_sec << " seconds\n";

    while (true)
    {
        cout << "Apakah Anda ingin menyimpan solusi? (y/N) ";
        char choseSave;
        cin >> choseSave;
        cin.ignore();
        if (choseSave != 'Y' && choseSave != 'y' && choseSave != 'N' &&
choseSave != 'n')
        {
            cout << "Masukan tidak valid! Silahkan coba lagi.\n\n";
            continue;
```

```
        }

        if (choseSave == 'Y' || choseSave == 'y')
        {
            cout << "Masukkan nama file: ";
            string fileName;
            cin >> fileName;
            ofstream File("test/" + fileName + ".txt");

            File << inp_card[0] << " " << inp_card[1] << " " << inp_card[2]
<< " " << inp_card[3] << endl;

            if (count > 0)
            {
                File << count << " solutions found\n";
            }
            else
            {
                File << "Tidak ada solusi\n";
            }

            for (auto i = solution.begin(); i != solution.end(); ++i)
            {
                File << *i << endl;
            }

            File.close();

            cout << "Solusi berhasil disimpan dalam folder test\n";
        }
        break;
    }

    return 0;
}
```

BAGIAN IV
CONTOH PROGRAM

1. Contoh dari Spek (input dan beberapa output)

```
Pilih cara untuk memasukkan 4 kartu:
1. Dipilih dengan input manual
2. Dipilih secara random

Ketik 1 atau 2: 1
Kartu ke-1 : A
Kartu ke-2 : 8
Kartu ke-3 : 9
Kartu ke-4 : Q

Kartu yang terpilih/dipilih:
A 8 9 Q
49 solutions found
((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
(8 * 1) * (12 - 9)
(8 / 1) * (12 - 9)
(9 + (1 - 8)) * 12
(9 - (8 - 1)) * 12
1 * ((12 - 9) * 8)
1 * (8 * (12 - 9))
```

2. Contoh apabila tidak ada solusi

```
Kartu yang terpilih/dipilih:
9 5 A A
Tidak ada solusi

Time taken: 0.282 miliseconds
Apakah Anda ingin menyimpan solusi? (y/N)
```

3. Salah satu test case dengan solusi yang banyak (lihat TestCase3.txt untuk sisa solusinya)

```
Kartu yang terpilih/dipilih:
A 4 8 Q
335 solutions found
((1 * 12) * 8) / 4
((1 * 12) + 4) + 8
((1 * 12) + 8) + 4
((1 * 12) / 4) * 8
((1 * 4) + 12) + 8
((1 * 4) + 8) + 12
((1 * 8) * 12) / 4
((1 * 8) + 12) + 4
((1 * 8) + 4) + 12
((1 * 8) / 4) * 12
((1 + 8) * 4) - 12
((1 / 4) * 12) * 8
((1 / 4) * 8) * 12
((12 * 1) * 8) / 4
((12 * 1) + 4) + 8
((12 * 1) + 8) + 4
((12 * 1) / 4) * 8
((12 * 8) * 1) / 4
((12 * 8) / 1) / 4
((12 * 8) / 4) * 1
((12 * 8) / 4) / 1
((12 + 4) * 1) + 8
((12 + 4) + 8) * 1
((12 + 4) + 8) / 1
((12 + 4) / 1) + 8
((12 + 8) * 1) + 4
((12 + 8) + 4) * 1
```

4. Menyimpan solusi menjadi file txt

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

Time taken: 0.33 miliseconds
Apakah Anda ingin menyimpan solusi? (y/N) y
Masukkan nama file: TestCase4
Solusi berhasil disimpan dalam folder test
```

5. Kartu dipilih secara random

```
Pilih cara untuk memasukkan 4 kartu:
1. Dipilih dengan input manual
2. Dipilih secara random

Ketik 1 atau 2: 2

Kartu yang terpilih/dipilih:
10 9 A K
4 solutions found
(1 - 9) * (10 - 13)
(10 - 13) * (1 - 9)
(13 - 10) * (9 - 1)
(9 - 1) * (13 - 10)

Time taken: 0.253 miliseconds
Apakah Anda ingin menyimpan solusi? (y/N) y
Masukkan nama file: TestCase5
Solusi berhasil disimpan dalam folder test
```

6. Apabila masukan user tidak sesuai

```
Pilih cara untuk memasukkan 4 kartu:
1. Dipilih dengan input manual
2. Dipilih secara random

Ketik 1 atau 2: 4
Masukan tidak sesuai! Silahkan coba lagi.

Pilih cara untuk memasukkan 4 kartu:
1. Dipilih dengan input manual
2. Dipilih secara random

Ketik 1 atau 2: 2

Kartu yang terpilih/dipilih:
J 2 9 J
22 solutions found
((11 * 2) + 11) - 9
((11 * 2) - 9) + 11
((11 - 9) * 11) + 2
((2 * 11) + 11) - 9
((2 * 11) - 9) + 11
(11 * (11 - 9)) + 2
(11 * 2) + (11 - 9)
(11 * 2) - (9 - 11)
(11 + (11 * 2)) - 9
(11 + (2 * 11)) - 9
(11 - 9) + (11 * 2)
(11 - 9) + (2 * 11)
(2 * 11) + (11 - 9)
(2 * 11) - (9 - 11)
11 + ((11 * 2) - 9)
11 + ((2 * 11) - 9)
11 - (9 - (11 * 2))
11 - (9 - (2 * 11))
2 + ((11 - 9) * 11)
2 + (11 * (11 - 9))
2 - ((9 - 11) * 11)
2 - (11 * (9 - 11))

Time taken: 0.23 miliseconds
Apakah Anda ingin menyimpan solusi? (y/N) y
Masukkan nama file: TestCase6
```

BAGIAN V
LINK REPOSITORY

Link repository GitHub : https://github.com/Mifkiyan/Tucil1_13521075

BAGIAN VI
CHECK LIST

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	✓	