**LAPORAN TUGAS KECIL 1**

**IF2211 STRATEGI ALGORITMA**

**Penyelesaian Permainan Kartu 24 dengan Algoritme Brute Force**

****

Disusun oleh

Akmal Mahardika Nurwahyu Pratama 13521070

**SEMESTER I TAHUN 2022/2023**

**JURUSAN TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNGBAB I**

**ALGORITME BRUTE FORCE**

Algoritme *brute force* adalah algoritma yang menyelesaikan masalah secara terus terang. Algoritme ini memecahkan persoalan komputasi dengan langsung dan jelas caranya. Algoritme ini lebih mudah dipahami karena lebih menggunakan tenaga dari pada otak. Penyebutan tersebut terjadi karena pada beberapa kasus penerapan algoritme *brute force* akan menemukan sebuah komputasi yang sebenarnya tidak diperlukan.

Permainan kartu 24 menggunakan algoritme *brute force*. Permainan ini bertujuan untuk menemukan kombinasi matematika dari empat kartu remi, dengan rantang angka 1 – 13, sehingga menghasilkan angka 24. Algoritme *brute force* pada permainan Kartu 24 adalah sebagai berikut

1. Pilih 4 kartu, misal a b c d
2. Sesuai angka, tentukan bentuk atau peletakan tanda kurung
3. Sesuai tanda kurung tersebut, lakukan permutasi dari 4 buah operasi, ”+-\*/”, pilih 3 buah untuk diletakan diantara a b c d.
4. Lakuukan operasi sesuai permutasi yang telah terbentuk. Jika membentuk angkaa 24 maka menjadi hasil. (opsional) simpan urutan angka pada sebuah larik
5. Ulangi langkah 2-4 dengan melakukan permutasi pada susunan angka a b c d. (opsional) lakukan pengecekkan apakah urutan empat angka sudah terjadi atau belum; jika belum maka lanjut, jika sudah maka cari urutan lain

Kompleksitas waktu algoritme berkisar O(n!), dengan waktu operasi T(n! \* m \* v3) atau T(n! \* r \* m \* v3) dengan n adalah banyak angka, m adalah banyak bentuk tanda kurung yang digunakan, v adalah banyak operasi 2 bilangan yang dilakukan, dan r (opsional) adalah panjang larik dengan r dipengaruhi kesamaan angka dan banyak angkanya, yang menyimpan permutasi dari urutan angka. Sehingga banyak operasi yang mungkin terjadi jika memisalkan terdapat 4 angka (n = 4), a ≠ b ≠ c ≠ d sehingga r tidak dipakai, 5 bentuk tanda kurung, 4 buah operasi sehingga terdapat sekitar 7680 operasi. Tanda kurung terdapat 5 karena memperjelas urutan operasi seperti ((a b) c) d dari pada (a b c) d.

**BAB II**

**SOURCE PROGRAM**

Program dibuat dengan menggunakan bahasa C++ dengan menggunakan *library*

* iostream
* vector
* sstream
* istream
* fstream
* cstdlib
* chrono

Dengan menggunakan fungsi / prosedur:

// input

void inputHandle(vector<*string*> \**sCards*);

bool isCard(*string* *card*);

void cardStrToDouble(vector<*string*> *strCard*, vector<double> \**dblCard*);

vector<*string*> cardRandTranslator(vector<*string*> *strCard*);

// proses

void bruteForceSolver24(vector<double> *cardList*, *string* \**result*, int \**countResult*);

bool isHaveSamePermutation(vector<vector<double>> *dbspermutation*, double *a*, double *b*, double *c*, double *d*);

void permutationBrackets(double *a*, double *b*, double *c*, double *d*, *string* \**result*, int \**countResult*, bool \**is24*);

void permutationOp(double *a*, double *b*, double *c*, double *d*, *string* \**result*, int \**countResult*, bool \**is24*, int *brType*);

double evalOp(double *a*, double *b*, char *op*);

*string* doubleToCard(double *a*);

// output

void displayCard(vector<*string*> *v*);

void saveFile(*string* *result*, vector<*string*> *sCards*);

sehingga terbentuk *source code* berikut:

int main()

{

    // KAMUS

    vector<*string*> sCards;

    vector<double> dCards;

*string* result;

    double executionTime;

    int countResult;

    // ALGORITME

    // Input

    cout << "==================== 24 Solver : Card Game ====================" << endl;

    inputHandle(&sCards);

    cardStrToDouble(sCards, &dCards);

    cout << "===============================================================" << endl;

    displayCard(sCards);

    // Proses

    countResult = 0;

    auto start = chrono::*high\_resolution\_clock*::now();

    bruteForceSolver24(dCards, &result, &countResult);

    auto end = chrono::*high\_resolution\_clock*::now();

    executionTime = chrono::duration\_cast<chrono::*microseconds*>(end - start).count();

    // Output

    cout << countResult << " Solution Found" << endl;

    cout << result << endl;

    cout << "Execution time : " << executionTime << " microseconds" << endl;

    saveFile(result, sCards);

    return 0;

}

// Input

void inputHandle(vector<*string*> \**strCards*)

{

    // KAMUS LOKAL

*string* inputType;

*string* lineInput, card, fileName, stemp;

    int i;

    int cnt;

    bool isPass;

*ifstream* fin;

    // ALGORITMA

    cout << "k : keyboard, f : file, r : random" << endl;

    cout << "(Masukkan apapun untuk randomize)" << endl;

    cout << "Input type : ";

    getline(cin, inputType);

    if (inputType == "k")

    {

        isPass = false;

        do

        {

            // input from Keyboard

            cout << "Masukkan 4 kartu : ";

            getline(cin, lineInput);

*stringstream* c(lineInput);

            cnt = 0; card.clear();

            while (c >> card)

            {

                if (isCard(card))

                {

*strCards*->push\_back(card);

                    cnt++;

                }

                else

                {

                    break;

                }

            }

            if (cnt == 4)

            {

                isPass = true;

            }

            else

            {

                cout << "Input tidak valid, Ulangi!" << endl;

            }

        } while (!isPass);

    }

    else if (inputType == "f")

    {

        // input from File

        cout << "Masukkan nama file : ";

        getline(cin, stemp);

        fileName = "../test/" + stemp;

        cout << fileName << endl;

        fin.open(fileName);

        if (fin.fail())

        {

            fin.clear();

            cout << "File tidak ditemukan\n" << "exiting program..." << endl;

            exit(1); // belum tau cara kerjanya file scan

        }

        getline(fin, lineInput);

        fin.close();

*stringstream* c(lineInput);

        cnt = 0;

        while(c >> card)

        {

            if (isCard(card))

            {

*strCards*->push\_back(card);

                cnt++;

            }

            else

            {

                break;

            }

        }

        if (cnt != 4)

        {

            cout << "Isi file tidak valid!" << endl;

            exit(1);

        }

    }

    else

    {

        // input from Random

        srand(time(NULL));

        for (i = 0; i < 4; i++)

        {

*strCards*->push\_back(to\_string(rand() % 13 + 1));

        }

        \**strCards* = cardRandTranslator(\**strCards*);

    }

}

bool isCard(*string* *card*)

{

    // KAMUS LOKAL

    int i;

    bool isCard;

    // ALGORITMA

    vector<*string*> cardList = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

    isCard = false;

    i = 0;

    do

    {

        isCard = (*card* == cardList[i]);

        i++;

    } while (i < 13 && !isCard);

    // card == cardList[i] ~ isCard || i == 13

    return isCard;

}

void cardStrToDouble(vector<*string*> *strCard*, vector<double> \**dblCard*)

{

    // KAMUS LOKAL

    int i;

*string* card;

    // ALGORITME

    i = 0;

    do

    {

        if (*strCard*[i] == "A")

        {

*dblCard*->push\_back(1);

        }

        else if (*strCard*[i] == "J")

        {

*dblCard*->push\_back(11);

        }

        else if (*strCard*[i] == "Q")

        {

*dblCard*->push\_back(12);

        }

        else if (*strCard*[i] == "K")

        {

*dblCard*->push\_back(13);

        }

        else

        {

*dblCard*->push\_back(stod(*strCard*[i]));

        }

        i++;

    } while (i < 4);

}

vector<*string*> cardRandTranslator(vector<*string*> *strCard*)

{

    // KAMUS LOKAL

    int i;

*string* card;

    // ALGORITMA

    for (i = 0; i < 4; i++)

    {

        if (*strCard*[i] == "1")

        {

*strCard*[i] = "A";

        }

        else if (*strCard*[i] == "11")

        {

*strCard*[i] = "J";

        }

        else if (*strCard*[i] == "12")

        {

*strCard*[i] = "Q";

        }

        else if (*strCard*[i] == "13")

        {

*strCard*[i] = "K";

        }

    }

    return *strCard*;

}

// Proses

void bruteForceSolver24(vector<double> *cardList*, *string* \**result*, int \**countResult*)

{

    // KAMUS LOKAL

    vector<vector<double>> dbsCardsPermutation;

    double a, b, c, d; // variabel that will be use to store the value of card

    int i, j, k, l;

    bool is24;

    int cnt = 0;

    // ALGORITMA

    for (i = 0; i < 4; i++)

    {

        for (j = 0; j < 4; j++)

        {

            for (k = 0; k < 4; k++)

            {

                for (l = 0; l < 4; l++)

                {

                    if (i != j && i != k && i != l && j != k && j != l && k != l)

                    {

                        if (isHaveSamePermutation(dbsCardsPermutation, *cardList*[i], *cardList*[j], *cardList*[k], *cardList*[l]))

                        {

                            a = *cardList*[i];

                            b = *cardList*[j];

                            c = *cardList*[k];

                            d = *cardList*[l];

                            permutationBrackets(a, b, c, d, *result*, *countResult*, &is24);

                            if (is24)

                            {

                                dbsCardsPermutation.push\_back({a, b, c, d});

                            }

                        }

                    }

                }

            }

        }

    }

}

bool isHaveSamePermutation(vector<vector<double>> *dbspermutation*, double *a*, double *b*, double *c*, double *d*)

{

    // KAMUS LOKAL

    int i, sizeRow;

    // ALGORITME

    sizeRow = *dbspermutation*.size();

    for (i = 0; i < sizeRow; i++)

    {

        if (*dbspermutation*[i][0] == *a* && *dbspermutation*[i][1] == *b* && *dbspermutation*[i][2] == *c* && *dbspermutation*[i][3] == *d*)

        {

            return false;

        }

    }

    return true;

}

void permutationBrackets(double *a*, double *b*, double *c*, double *d*, *string* \**result*, int \**countResult*, bool \**is24*)

{

    // KAMUS LOKAL

    int i = 1;

    // ALGORITMA

    for (i = 1; i <= 5; i++)

    {

        permutationOp(*a*, *b*, *c*, *d*, *result*, *countResult*, *is24*, i);

    }

    //1 (a b) (c d)

    //2 ((a b) c) d

    //3 (a (b c)) d

    //4 a ((b c) d)

    //5 a (b (c d))

}

void permutationOp(double *a*, double *b*, double *c*, double *d*, *string* \**result*, int \**countResult*, bool \**is24*, int *brType*)

{

    // KAMUS LOKAL

*string* op = "+-\*/";

*string* tempResult;

    int i, j, k;

    double ab, bc, cd, dblresult;

    // ALGORITMA

    for (i = 0; i < 4; i++)

    {

        for (j = 0; j < 4; j++)

        {

            for (k = 0; k < 4; k++)

            {

                dblresult = 0;

                ab = evalOp(*a*, *b*, op[i]);

                bc = evalOp(*b*, *c*, op[j]);

                cd = evalOp(*c*, *d*, op[k]);

                switch (*brType*)

                {

                case 1:

                    dblresult = evalOp(ab, cd, op[j]);

                    tempResult = "(" + doubleToCard(*a*) + " " + op[i] + " " + doubleToCard(*b*) + ") " + op[j] + " (" + doubleToCard(*c*) + " " + op[k] + " " + doubleToCard(*d*) + ")";

                    break;

                case 2:

                    dblresult = evalOp(evalOp(ab, *c*, op[j]), *d*, op[k]);

                    tempResult = "((" + doubleToCard(*a*) + " " + op[i] + " " + doubleToCard(*b*) + ") " + op[j] + " " + doubleToCard(*c*) + ") " + op[k] + " " + doubleToCard(*d*);

                    break;

                case 3:

                    dblresult = evalOp(evalOp(*a*, bc, op[i]), *d*, op[k]);

                    tempResult = "(" + doubleToCard(*a*) + " " + op[i] + " (" + doubleToCard(*b*) + " " + op[j] + " " + doubleToCard(*c*) + ")) " + op[k] + " " + doubleToCard(*d*);

                    break;

                case 4:

                    dblresult = evalOp(*a*, evalOp(bc, *d*, op[k]), op[i]);

                    tempResult = doubleToCard(*a*) + " " + op[i] + " ((" + doubleToCard(*b*) + " " + op[j] + " " + doubleToCard(*c*) + ") " + op[k] + " " + doubleToCard(*d*) + ")";

                    break;

                case 5:

                    dblresult = evalOp(*a*, evalOp(*b*, cd, op[j]), op[i]);

                    tempResult = doubleToCard(*a*) + " " + op[i] + " (" + doubleToCard(*b*) + " " + op[j] + " (" + doubleToCard(*c*) + " " + op[k] + " " + doubleToCard(*d*) + "))";

                    break;

                }

                if (dblresult == 24)

                {

                    \**result* += tempResult + "\n";

                    \**countResult* += 1;

                    \**is24* = true;

                }

            }

        }

    }

}

double evalOp(double *a*, double *b*, char *op*)

{

    // KAMUS LOKAL

    double result;

    // ALGORITMA

    switch (*op*)

    {

    case '+':

        result = *a* + *b*;

        break;

    case '-':

        result = *a* - *b*;

        break;

    case '\*':

        result = *a* \* *b*;

        break;

    case '/':

        if (*b* == 0)

        {

            result = 0;

        }

        else

        {

            result = *a* / *b*;

        }

        break;

    }

    return result;

}

*string* doubleToCard(double *a*)

{

    // KAMUS LOKAL

*string* card;

    // ALGORITMA

    if (*a* == 1)

    {

        card = "A";

    }

    else if (*a* == 11)

    {

        card = "J";

    }

    else if (*a* == 12)

    {

        card = "Q";

    }

    else if (*a* == 13)

    {

        card = "K";

    }

    else

    {

        card = to\_string((int)*a*);

    }

    return card;

}

// output

void displayCard(vector<*string*> *v*)

{

    // KAMUS LOKAL

    int i;

    // ALGORITMA

    cout << "Kartu";

    for (i = 0; i < *v*.size(); i++)

    {

        cout << " " << *v*[i];

    }

    cout << endl;

}

void saveFile(*string* *reslut*, vector<*string*> *sCards*)

{

    // KAMUS LOKAL

*string* fileName;

*string* save, rename;

    for (int i = 0; i < 4; i++)

    {

        fileName += *sCards*[i];

    }

    cout << "Save solution to file [y/n]? default n " << endl;

    cin >> save;

    if (save == "y")

    {

        cout << "rename file [y/n]? default n " << endl;

        cout << "default file name : " << fileName << endl;

        cin >> rename;

        if (rename == "y")

        {

            cout << "input file name : ";

            cin >> fileName;

        }

*ofstream* fileOut("../test/"+ fileName + ".txt");

        fileOut << *reslut*;

        fileOut.close();

    }

    else

    {

        cout << "File not saved" << endl;

    }

}

**BAB III**

**TEST PROGRAM**

Berikut merupakan beberapa test program yang dilakukan :

1. Test input keyboard

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

1. Test input file

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

1. Test random input (1: input r)

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

1. Test random input (2 : input apapun)

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

1. Test save file (1 : n solusi)

Sebuah gambar berisi teks

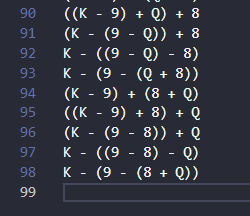
Deskripsi dibuat secara otomatis

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

Sebuah gambar berisi teks, elektronik, keyboard, hitam

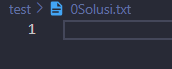
Deskripsi dibuat secara otomatis



1. Tesst save file (2 : 0 solusi dan rename)

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis



1. Test error input keyboard

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

1. Test error input file (1: File tidak ada, exit)

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

1. Test error input file (2: Isi file salah, exit)

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

**BAB IV**

**LAMPIRAN**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| Program berhasil dikompilasi tanpa kesalahan | ✓ |  |
| Program berhasil running | ✓ |  |
| Program dapat membaca input / generate sendiri dan memberikan luaran | ✓ |  |
| Solusi yang diberikan program memenuhi (berhasil mencapai 24) | ✓ |  |
| Program dapat menyimpan solusi dalam file teks | ✓ |  |

Repository Github

<https://github.com/akmaldika/Tucil1_IF2211_24CardGame.git>