

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 BANDUNG

BAB 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. Lakukan operasi sesuai permutasi yang telah terbentuk. Jika membentuk angka 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 pengecekan 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 $T(n! * m * v^3)$ atau $T(n! * r * m * v^3)$ 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 ($n = 4$), $a \neq b \neq c \neq d$ 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>> dbsp permutation, 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

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : k
Masukkan 4 kartu : 8 7 3 A
=====
Kartu 8 7 3 A
9 Solution Found
8 * ((7 - 3) - A)
8 * (7 - (3 + A))
8 * ((7 - A) - 3)
8 * (7 - (A + 3))
((7 - 3) - A) * 8
(7 - (3 + A)) * 8
((7 - A) - 3) * 8
(7 - (A + 3)) * 8
3 / (A - (7 / 8))

Execution time : 3011 microseconds
Save solution to file [y/n]? default n
y
rename file [y/n]? default n
default file name : 873A
n
```

2. Test input file

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : f
Masukkan nama file : test_Q89K.txt
=====
Kartu Q 8 9 K
98 Solution Found
(Q + 8) - (9 - K)
((Q + 8) - 9) + K
(Q + (8 - 9)) + K
Q + ((8 - 9) + K)
K - ((9 - Q) - 8)
K - (9 - (Q + 8))
(K - 9) + (8 + Q)
((K - 9) + 8) + Q
(K - (9 - 8)) + Q
K - ((9 - 8) - Q)
K - (9 - (8 + Q))

Execution time : 4014 microseconds
Save solution to file [y/n]? default n
n
File not saved
```

3. Test random input (1: input r)

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : r
=====
Kartu 2 9 6 6
10 Solution Found
(2 * (9 + 6)) - 6
(2 + 6) * (9 - 6)
(2 * (6 + 9)) - 6
(9 - 6) * (2 + 6)
((9 + 6) * 2) - 6
(9 - 6) * (6 + 2)
((9 * 6) - 6) / 2
(6 + 2) * (9 - 6)
((6 + 9) * 2) - 6
((6 * 9) - 6) / 2
```

4. Test random input (2 : input apapun)

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type :
=====
Kartu 2 J 2 3
16 Solution Found
2 * ((J - 2) + 3)
2 * (J - (2 - 3))
2 * ((J + 3) - 2)
2 * (J + (3 - 2))
2 * ((3 + J) - 2)
2 * (3 + (J - 2))
2 * ((3 - 2) + J)
2 * (3 - (2 - J))
((J - 2) + 3) * 2
(J - (2 - 3)) * 2
((J + 3) - 2) * 2
(J + (3 - 2)) * 2
((3 - 2) + J) * 2
(3 - (2 - J)) * 2
((3 + J) - 2) * 2
(3 + (J - 2)) * 2
```

5. Test save file (1 : n solusi)

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : f
Masukkan nama file : test_Q89K.txt
=====
Kartu Q 8 9 K
98 Solution Found

Execution time : 3234 microseconds
Save solution to file [y/n]? default n
y
rename file [y/n]? default n
default file name : Q89K
n
```

test >	Q89K.txt
1	$(Q + 8) - (9 - K)$
2	$((Q + 8) - 9) + K$
3	$(Q + (8 - 9)) + K$
4	$Q + ((8 - 9) + K)$
5	$Q + (8 - (9 - K))$
6	$(Q + 8) + (K - 9)$
7	$(Q * 8) / (K - 9)$
8	$((Q + 8) + K) - 9$
9	$(Q + (8 + K)) - 9$
10	$Q + ((8 + K) - 9)$
90	$((K - 9) + Q) + 8$
91	$(K - (9 - Q)) + 8$
92	$K - ((9 - Q) - 8)$
93	$K - (9 - (Q + 8))$
94	$(K - 9) + (8 + Q)$
95	$((K - 9) + 8) + Q$
96	$(K - (9 - 8)) + Q$
97	$K - ((9 - 8) - Q)$
98	$K - (9 - (8 + Q))$
99	

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

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : f
Masukkan nama file : test_J53K.txt
=====
Kartu J 5 3 K
0 Solution Found

Execution time : 3457 microseconds
Save solution to file [y/n]? default n
y
rename file [y/n]? default n
default file name : J53K
y
input file name : 0Solusi
```

test >	0Solusi.txt
1	

7. Test error input keyboard

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : k
Masukkan 4 kartu : 1 2 3 4
Input tidak valid, Ulangi!
Masukkan 4 kartu : 8 9 10 11
Input tidak valid, Ulangi!
Masukkan 4 kartu : a 2 3 4
Input tidak valid, Ulangi!
Masukkan 4 kartu :
```

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

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : f
Masukkan nama file : gakada.txt
File tidak ditemukan
exiting program...
```

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

```
===== 24 Solver : Card Game =====
k : keyboard, f : file, r : random
(Masukkan apapun untuk randomize)
Input type : f
Masukkan nama file : isifilesalah.txt
Isi file tidak valid!
```

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