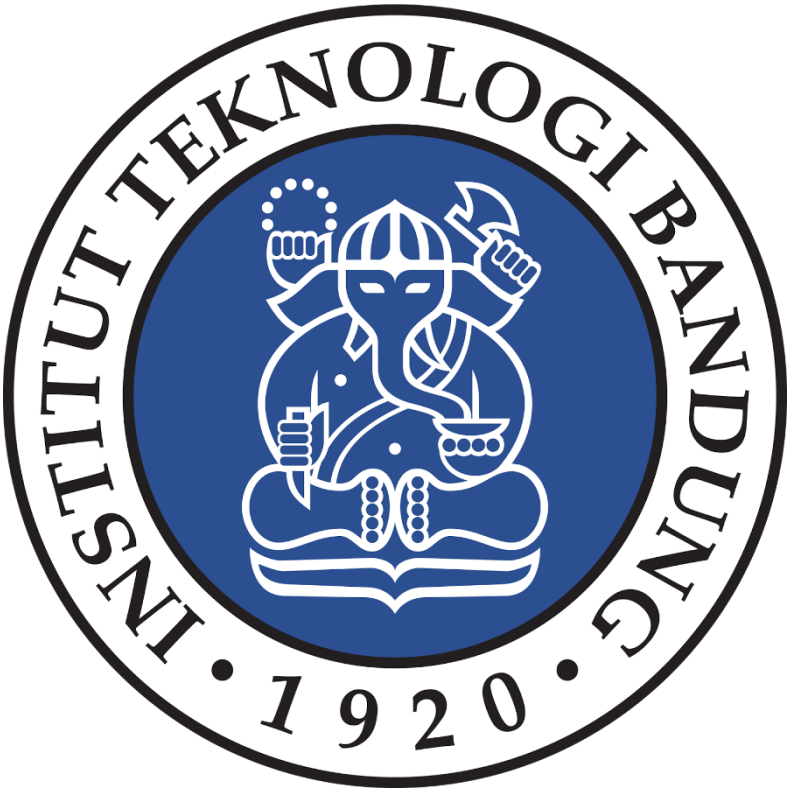
Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2022/2023

Algoritma Brute Force untuk Permainan "*24 Card Game*"

Disusun oleh:

Nathan Tenka / 13521172



PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG 2022

Bab I

Algoritma Brute Force Pencarian Solusi *24 Card Game*

Algoritmanya adalah sebagai berikut :

1. Terima masukan keempat jenis kartu.
2. Simpan semua permutasi keempat kartu dan operator yang mungkin. Untuk permutasi kartu, jangan simpan permutasi yang sudah pernah muncul sebelumnya (bisa terjadi saat ada 2 atau lebih kartu yang jenisnya sama).
3. Periksa hasil operasi semua permutasi jenis kartu, operator, dan operasi kurung yang mungkin. Apabila sama dengan 24, simpan solusinya. Pada program ini digunakan 5 jenis operasi kurung yang sudah merepresentasikan semua jenis pengelompokan (sama seperti di *website* http://24solver.us-west-2.elasticbeanstalk.com/) , yaitu :
    - ((A op B) op C) op D
    - (A op (B op C)) op D
    - A op (B op (C op D))
    - A op ((B op C) op D)
    - (A op B) op (C op D)

    Sifat komutatif tidak diperhitungkan, sehingga solusi yang mirip bisa muncul lebih dari sekali. Solusi dengan posisi kurung yang berbeda juga dianggap berbeda walau artinya sama dengan solusi lain.
4. Cetak semua solusi ke layar (jika ada) dan file (jika diminta).

Bab II
Source Code Program

Program ini menggunakan Bahasa C++ dan terdapat class Card untuk menyimpan jenis dan nilai kartu.

card.h :

```
#ifndef CARD_H
#define CARD_H

#include <iostream>
using namespace std;

class Card {
    /* Penampung jenis kartu */
    private :
        int number;
        string cardType;
    public :
        Card() {}; // Konstruktor
        // Getter dan setter
        int getNumber();
        string getType();
        void setNumber(int i);
        void setType(string t);
};
#endif
```

card.cpp :

```
#include "card.h"

using namespace std;

// Getter dan setter
int Card :: getNumber() {
    return number;
}
string Card :: getType() {
    return cardType;
}
void Card :: setNumber(int i) {
    number = i;
}
void Card :: setType(string t) {
    cardType = t;
}
```

main.cpp :

```
#include <iostream>
#include <chrono>
#include <random>
#include <array>
#include <vector>
#include <fstream>
#include "card.h" // Menampung class untuk kartu

using namespace std;
using namespace chrono;

double operate(double i1, int op, double i2) {
    /* Melakukan operasi pada i1 dan i2 berdasarkan op */
    if (op == 1) {
        return i1+i2;
    } else if (op == 2) {
        return i1-i2;
    } else if (op == 3) {
        return i1*i2;
    } else {
        return i1/i2;
    }
}

bool samePermutation(vector<array<Card,4>> permutations, array<Card,4> arr,
int permSize) {
    /* Mengecek apakah permutasi arr sudah ada sebelumnya di daftar
permutations */
    int i = 0;
    bool same = false;

    while (i<permSize && !same) {
        same = true;
        for (int j=0; j<4; j++) {
            if (arr[j].getType() != permutations[i][j].getType()) {
                same = false;
                break;
            }
        }
        if (!same) {
            i++;
        }
    }
    return same;
}

vector<array<Card,4>> permuteCard(Card hand[], int *cardPermSize) {
```

```
    /* Mengembalikan semua permutasi Card dari array arr. Diasumsikan arr
hanya berisi 4 Card. */
    /* Pendekatan : permutasikan 3 elemen secara manual dengan elemen keempat
berbeda-beda */
    array<Card,4> copy, arr1, arr2, arr3, arr4, arr5, arr6;
    vector<array<Card,4>> cardPermutations;

    for (int i=0; i<4; i++) {
        int idx=0;
        copy[3] = hand[i];
        for (int k=0; k<4; k++) { // Salin isi arr yang lain
            if (k != i) {
                copy[idx] = hand[k];
                idx++;
            }
        }
        // Masukkan semua permutasi 3 elemen yang mungkin (hanya ada 6) dan
cek apakah permutasi sudah ada sebelumnya
        arr1 = {copy[0],copy[1],copy[2],copy[3]};
        arr2 = {copy[1],copy[0],copy[2],copy[3]};
        arr3 = {copy[1],copy[2],copy[0],copy[3]};
        arr4 = {copy[2],copy[1],copy[0],copy[3]};
        arr5 = {copy[2],copy[0],copy[1],copy[3]};
        arr6 = {copy[0],copy[2],copy[1],copy[3]};
        if (!samePermutation(cardPermutations, arr1, *cardPermSize)) {
            cardPermutations.push_back(arr1);
            *cardPermSize += 1;
        }
        if (!samePermutation(cardPermutations, arr2, *cardPermSize)) {
            cardPermutations.push_back(arr2);
            *cardPermSize += 1;
        }
        if (!samePermutation(cardPermutations, arr3, *cardPermSize)) {
            cardPermutations.push_back(arr3);
            *cardPermSize += 1;
        }
        if (!samePermutation(cardPermutations, arr4, *cardPermSize)) {
            cardPermutations.push_back(arr4);
            *cardPermSize += 1;
        }
        if (!samePermutation(cardPermutations, arr5, *cardPermSize)) {
            cardPermutations.push_back(arr5);
            *cardPermSize += 1;
        }
        if (!samePermutation(cardPermutations, arr6, *cardPermSize)) {
            cardPermutations.push_back(arr6);
            *cardPermSize += 1;
        }
```

```
    }
    return cardPermutations;
}

vector<array<int,3>> permuteOp() {
    /* Mengembalikan vektor berisi semua permutasi integer representasi
operator */
    vector<array<int,3>> opPermutation;

    for (int i=1; i<5; i++) { // Loop variasi operator pertama
        for (int j=1; j<5; j++) { // Loop variasi operator kedua
            for (int k=1; k<5; k++) { // Loop variasi operator ketiga
                opPermutation.push_back({i,j,k});
            }
        }
    }
    return opPermutation;
}

bool check24(array<Card,4> c, array<int,3> op, int groupingType) {
    /* Memeriksa apakah kombinasi kartu, operasi, dan groupingType yang
diberikan menghasilkan 24 */
    double result, result2;

    if (groupingType == 1) { // ((c[0] op c[1]) op c[2]) op c[3]
        result = operate(c[0].getNumber(), op[0], c[1].getNumber());
        result = operate(result, op[1], c[2].getNumber());
        result = operate(result, op[2], c[3].getNumber());
    } else if (groupingType == 2) { // (c[0] op (c[1] op c[2])) op c[3]
        result = operate(c[1].getNumber(), op[1], c[2].getNumber());
        result = operate(c[0].getNumber(), op[0], result);
        result = operate(result, op[2], c[3].getNumber());
    } else if (groupingType == 3) { // c[0] op (c[1] op (c[2] op c[3]))
        result = operate(c[2].getNumber(), op[2], c[3].getNumber());
        result = operate(c[1].getNumber(), op[1], result);
        result = operate(c[0].getNumber(), op[0], result);
    } else if (groupingType == 4) { // c[0] op ((c[1] op c[2]) op c[3])
        result = operate(c[1].getNumber(), op[1], c[2].getNumber());
        result = operate(result, op[2], c[3].getNumber());
        result = operate(c[0].getNumber(), op[0], result);
    } else { // (c[0] op c[1]) op (c[2] op c[3])
        result = operate(c[0].getNumber(), op[0], c[1].getNumber());
        result2 = operate(c[2].getNumber(), op[2], c[3].getNumber());
        result = operate(result, op[1], result2);
    }
    return (abs(result-24) < 1e-10);
}
```

```
vector<string> findSolutions(Card hand[], int *solSize) {
    /* Cari semua solusi yang mungkin dari keempat nilai kartu (disimpan di
array hand) */
    vector<string> solutions;
    /* Permutasikan semua kemungkinan kartu dan operator */
    int cardPermSize = 0;
    int opPermSize = 64;
    vector<array<int,3>> opPermutations = permuteOp();
    const string operations[4] = {"+","-","*","/"};
    vector<array<Card,4>> cardPermutations = permuteCard(hand,&cardPermSize);

    for (int i=0; i<cardPermSize; i++) {
        // Simpan semua jenis kartu dalam array
        string handTypes[4] =
{cardPermutations[i][0].getType(),cardPermutations[i][1].getType(),
                                cardPermutations[i][2].getType(),
cardPermutations[i][3].getType()};
        for (int j=0; j<opPermSize; j++) {
            int currOps[3] = {opPermutations[j][0]-1,opPermutations[j][1]-
1,opPermutations[j][2]-1};
            if (check24(cardPermutations[i], opPermutations[j], 1)) {
                solutions.push_back("(("+handTypes[0]+operations[currOps[0]]+h
andTypes[1]+")"+operations[currOps[1]]
                                    + handTypes[2] + ")" +
operations[currOps[2]] + handTypes[3]);
                *solSize += 1;
            }
            if (check24(cardPermutations[i], opPermutations[j], 2)) {
                solutions.push_back("("+handTypes[0]+operations[currOps[0]]+"(
"+handTypes[1]+operations[currOps[1]]
                                    + handTypes[2] + "))" +
operations[currOps[2]] + handTypes[3]);
                *solSize += 1;
            }
            if (check24(cardPermutations[i], opPermutations[j], 3)) {
                solutions.push_back(handTypes[0]+operations[currOps[0]]+"("+ha
ndTypes[1]+operations[currOps[1]]
                                    + "(" + handTypes[2] +
operations[currOps[2]] + handTypes[3] + "))");
                *solSize += 1;
            }
            if (check24(cardPermutations[i], opPermutations[j], 4)) {
                solutions.push_back(handTypes[0]+operations[currOps[0]]+"(("+h
andTypes[1]+operations[currOps[1]]
                                     + handTypes[2] + ")" +
operations[currOps[2]] + handTypes[3] + ")");
                *solSize += 1;
            }
```

```
            if (check24(cardPermutations[i], opPermutations[j], 5)) {
                solutions.push_back("("+handTypes[0]+operations[currOps[0]]+ha
ndTypes[1]+")"+ operations[currOps[1]]
                                        + "(" + handTypes[2] +
operations[currOps[2]] + handTypes[3] + ")");
                *solSize += 1;
            }
        }
    }
    return solutions;
}

void inputHand(Card hand[], array<string,13> cardTypes) {
    /* Menerima input jenis kartu dari pengguna dan memasukkannya ke hand */
    Card temp;
    string input, singleInput;
    bool valid = false;
    int count,i;

    cin.ignore();
    while (!valid) {
        valid = true;
        singleInput = "";
        count = 0;
        cout << "Masukkan keempat kartu" << endl;
        getline(cin,input);

        i = 0;
        while(input[i]) {
            if (count >= 4 && input[i] != ' ') {
                cout << "Masukan tidak sesuai" << endl;
                valid = false;
                break;
            }
            if (input[i] != ' ') {
                singleInput += input[i];
            } else if (count < 4) {
                int k = 0;
                while (k < 13 && cardTypes[k] != singleInput) {
                    k++;
                }
                if (k >= 13) {
                    cout << "Masukan tidak sesuai" << endl;
                    valid = false;
                    break;
                } else {
                    temp.setNumber(k+1);
                    temp.setType(singleInput);
```

```
                    hand[count] = temp;
                    count++;
                }
                singleInput = "";
            }
            i++;
        }
        if (singleInput != "" && valid) {
            int k = 0;
            while (k < 13 && cardTypes[k] != singleInput) {
                k++;
            }
            if (k >= 13) {
                cout << "Masukan tidak sesuai" << endl;
                valid = false;
            } else {
                temp.setNumber(k+1);
                temp.setType(singleInput);
                hand[count] = temp;
                count++;
            }
            singleInput = "";
        }
        if (count != 4 && valid) {
            cout << "Masukan tidak sesuai" << endl;
            valid = false;
        }
    }
}

void randomizeHand (Card hand[], const array<string,13> cardTypes) {
    /* Mengisi hand secara acak */
    Card temp;

    cout << "Kartu dipilih secara acak" << endl;
    srand(time(0));
    for (int i=0; i<4; i++) {
        temp.setNumber((rand() % 13) + 1);
        temp.setType(cardTypes[temp.getNumber()-1]);
        hand[i] = temp;
    }
}

void outputSolution (vector<string> solutions, auto executionTime, int
solCount) {
    /* Mencetak semua solusi ke terminal */

    if (solCount > 0) {
```

```
        cout << "Banyak solusi : " << solCount << endl;
        for (int i=0; i<solCount; i++) {
            cout << solutions[i] << endl;
        }
    } else {
        cout << "Tidak ada solusi" << endl;
    }
    cout << "Waktu eksekusi : " << executionTime.count() << " detik" << endl;
}

void saveSolution (vector<string> solutions, auto executionTime, int solCount)
{
    /* Menyimpan semua solusi ke file */
    string fileName;
    ofstream f;

    cout << "Masukkan nama / path file untuk penyimpanan solusi (ketikkan juga
ekstensi .txt setelah nama file) : ";
    cin.ignore();
    getline(cin, fileName);
    f.open(fileName);
    if (solCount > 0) {
        f << "Banyak solusi : " << solCount << endl;
        for (int i=0; i<solCount; i++) {
            f << solutions[i] << endl;
        }
    }
    else {
        f << "Tidak ada solusi" << endl;
    }
    f << "Waktu eksekusi : " << executionTime.count() << " detik" << endl;
    f.close();
}

int main() {
    char choice;
    const array<string,13> cardTypes = {"A", "2", "3", "4", "5", "6", "7",
"8", "9", "10", "J", "Q", "K"};
    vector<string> solutions;
    int solSize = 0;
    Card hand[4]; // Kumpulan 4 kartu

    cout << "Selamat datang di permainan kartu 24 !" << endl << "Masukan kartu
dari pengguna/acak ? Ketik P atau A untuk memilih" << endl;
    cin >> choice;
    while (choice != 'P' && choice != 'A') {
        cout << "Masukan tidak sesuai" << endl;
        cin >> choice;
```

```
    }
    if (choice == 'P') {
        inputHand(hand, cardTypes);
    } else {
        randomizeHand(hand, cardTypes);
    }
    /* Cari solusi */
    auto startTime = high_resolution_clock::now();
    solutions = findSolutions(hand, &solSize);
    auto endTime = high_resolution_clock::now();
    auto executionTime = duration<long double>(endTime-startTime);

    outputSolution(solutions,executionTime,solSize);

    cout << "Apakah ingin menyimpan solusi ? Y/N" << endl;
    cin >> choice;
    while (choice != 'Y' && choice != 'N') {
        cout << "Masukan tidak sesuai" << endl;
        cin >> choice;
    }
    if (choice == 'Y') {
        saveSolution(solutions,executionTime,solSize);
    }

    return 0;
}
```

Bab III
Hasil Uji

Masukan 2 A 3 4

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
P
Masukkan keempat kartu
2 A 3 4
Banyak solusi : 242
(A+3)*(4+2)
((A*3)*4)*2
(A*(3*4))*2
A*(3*(4*2))
A*((3*4)*2)
(A*3)*(4*2)
(3+A)*(4+2)
((3*A)*4)*2
(3*(A*4))*2
3*(A*(4*2))
3*((A*4)*2)
(3*A)*(4*2)
((3/A)*4)*2
(3/A)*(4*2)
(3/(A/4))*2
3/(A/(4*2))
3/((A/4)/2)
```

… sampai 242 solusi

```
2*(3*(A*4))
2*((3*A)*4)
(2*3)*(A*4)
((2*3)/A)*4
(2*(3/A))*4
2*((3/A)*4)
2*(3/(A/4))
(2*3)/(A/4)
Waktu eksekusi : 0.005991 detik
Apakah ingin menyimpan solusi ? Y/N
Y
Masukkan nama / path file untuk penyimpanan solusi (ketikkan juga ekstensi .txt setelah nama file) : test2A34.txt
```

Test2A34.txt :

```
Banyak solusi : 242
(A+3)*(4+2)
((A*3)*4)*2
(A*(3*4))*2
A*(3*(4*2))
A*((3*4)*2)
(A*3)*(4*2)
(3+A)*(4+2)
((3*A)*4)*2
(3*(A*4))*2
3*(A*(4*2))
3*((A*4)*2)
(3*A)*(4*2)
((3/A)*4)*2
(3/A)*(4*2)
(3/(A/4))*2
3/(A/(4*2))
3/((A/4)/2)
((3*4)*A)*2
(3*(4*A))*2
3*(4*(A*2))
3*((4*A)*2)
(3*4)*(A*2)
((3*4)/A)*2
(3*(4/A))*2
3*((4/A)*2)
3*(4/(A/2))
```

… sampai 242 solusi

```
2*(3*(A*4))
2*((3*A)*4)
(2*3)*(A*4)
((2*3)/A)*4
(2*(3/A))*4
2*((3/A)*4)
2*(3/(A/4))
(2*3)/(A/4)
Waktu eksekusi : 0.005991 detik
```

Masukan A 2 3 (masukan kurang dari 4)

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
P
Masukkan keempat kartu
A 2 3
Masukan tidak sesuai
Masukkan keempat kartu
```

Masukan A 2 3 4 5 (masukan lebih dari 4)

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
P
Masukkan keempat kartu
A 2 3 4 5
Masukan tidak sesuai
Masukkan keempat kartu
```

Masukan A 22 3 45 (jenis kartu tidak ada)

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
P
Masukkan keempat kartu
A 22 3 45
Masukan tidak sesuai
Masukkan keempat kartu
```

Masukan 6 6 6 6

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
P
Masukkan keempat kartu
6 6 6 6
Banyak solusi : 7
((6+6)+6)+6
(6+(6+6))+6
6+(6+(6+6))
6+((6+6)+6)
(6+6)+(6+6)
(6*6)-(6+6)
((6*6)-6)-6
Waktu eksekusi : 0.001001 detik
Apakah ingin menyimpan solusi ? Y/N
Y
Masukkan nama / path file untuk penyimpanan solusi (ketikkan juga ekstensi .txt setelah nama file) : test6666.txt
```

Masukan 7 8 7 7

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
P
Masukkan keempat kartu
7 8 7 7
Tidak ada solusi
Waktu eksekusi : 0.002003 detik
Apakah ingin menyimpan solusi ? Y/N
Y
Masukkan nama / path file untuk penyimpanan solusi (ketikkan juga ekstensi .txt setelah nama file) : test7877.txt
```

Test7877.txt :

```
Tidak ada solusi
Waktu eksekusi : 0.002003 detik
```

Pemilihan kartu acak (didapat 2 Q J 10)

```
Selamat datang di permainan kartu 24 !
Masukan kartu dari pengguna/acak ? Ketik P atau A untuk memilih
A
Kartu dipilih secara acak
Banyak solusi : 42
(Q*(J-10))*2
Q*((J-10)*2)
(Q/(J-10))*2
Q/((J-10)/2)
(J*(Q-10))+2
((J-10)*Q)*2
(J-10)*(Q*2)
(Q-10)+(J*2)
2+((Q-10)*J)
(Q-10)+(2*J)
Q-(10-(2*J))
2-((10-Q)*J)
2*(Q*(J-10))
(2*Q)*(J-10)
2*(Q/(J-10))
(2*Q)/(J-10)
(Q+(2*J))-10
Q+((2*J)-10)
Q*(2*(J-10))
(Q*2)*(J-10)
Q*(2/(J-10))
(Q*2)/(J-10)
(Q+(J*2))-10
Q+((J*2)-10)
((J*2)+Q)-10
(J*2)+(Q-10)
2+(J*(Q-10))
((2*J)+Q)-10
(2*J)+(Q-10)
Waktu eksekusi : 0.006996 detik
Apakah ingin menyimpan solusi ? Y/N
Y
Masukkan nama / path file untuk penyimpanan solusi (ketikkan juga ekstensi .txt setelah nama file) : test2QJ10.txt
```

Lampiran

Checklist program

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	✓	

Link repository : https://github.com/Nat10k/Tucil1_13521172