Tugas Kecil 1 IF2211 Strategi Algoritma
24 Game Solver dengan Algoritma Brute Force

Disusun oleh:

Farhan Nabil Suryono 13521114

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

1. Spesifikasi Tugas

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 (()).

2. Penjelasan Program

Algoritma *Brute Force* adalah salah satu strategi dalam pembuatan algoritma untuk pemecahan masalah. *Brute Force* memiliki sifat yaitu *straightforward* dan umumnya adalah solusi yang pertama kali kita terpikirkan. Algoritma *Brute Force* pada umumnya sederhana dan mudah dimengerti, namun membutuhkan *resource* yang banyak baik dalam segi memori maupun waktu eksekusi karena algoritma ini memeriksa semua kemungkinan. Oleh karena itu, Algoritma *Brute Force* biasanya hanya digunakan untuk permasalahan-permasalahan yang memiliki masukan tidak terlalu banyak atau ketika waktu eksekusi dan memori bukanlah hal yang perlu diutamakan.

Pada tugas kecil ini, permasalahan yang diselesaikan adalah mencari semua kombinasi operasi aritmatika yang diaplikasikan ke 4 nilai berdasarkan 4 kartu masukan agar menghasilkan nilai 24 dengan pendekatan algoritma *Brute Force*. Program akan mengeluarkan jumlah solusi, semua solusi, dan dapat menyimpan solusi tersebut pada suatu *file*.

Implementasi program "24 *Game Solver*" yang dibuat penulis menggunakan langkah-langkah sebagai berikut.

1. Program meminta metode *input* dari pengguna. Pengguna dapat memilih untuk meng-*input* 4 kartu secara manual maupun meminta program untuk men-*generate* 4 kartu secara acak.
2. Program mencari kartu-kartu yang muncul lebih dari sekali.
3. Program menghasilkan semua permutasi keempat kartu serta semua kemungkinan operator lalu menyelesaikkannya dengan memanggil prosedur solve.
4. Pada prosedur solve, program mengeevaluasi hasil operasi aritmatika berdasarkan 4 nilai dan 3 operator dalam 5 kemungkinan posisi kurung. Apabila hasil operasi bernilai 24, program akan menyimpan solusi tersebut sebagai suatu *string* dan memasukkannya ke dalam vector solutions.
5. Program meng-*output* jumlah solusi, solusi, serta waktu eksekusi program ketika mencari solusi.

6. Program meminta *input* dari pengguna berupa opsi apakah mau menyimpan solusi sebagai *file* atau tidak. Apabila pengguna ingin menyimpannya, program meminta *input* nama file lalu program akan membuat file berisi solusi dan menyimpannya di folder test dengan format .txt.

3. Source Code Program

```
#include <iostream>
#include <fstream>
#include <vector>
#include <chrono>
#include <time.h>

using namespace std;

// Variabel Global
int solCount = 0;
vector<double> cards;
vector<int> isDupe(4, -1);
vector<string> solutions, operators = {"+", "-", "*", "/"};

bool isCardValid(string s) {
    // Fungsi memeriksa apakah s merupakan sebuah nilai kartu yang
valid (2-10, J, Q, K, A)
    if (s.length() > 1) {
        if (s == "10") {
            return true;
        }
    } else {
        if (s == "A" || s == "J" || s == "Q" || s == "K" || (s >= "2"
&& s <= "9")) return true;
    }

    return false;
}

bool isInputValid(string s) {
    // Fungsi memeriksa apakah s merupakan input yang valid yaitu
berisi 4 kartu valid
    if (s.length() > 11) return false;

    int spaceCount = 0;
    string temp = "";
    for (auto x : s) {
        if (x == ' ') {
            spaceCount++;
            if (!isCardValid(temp)) return false;
            temp.clear();
        } else {
```

```
            temp.push_back(x);
        }
    }

    if (!isCardValid(temp)) return false;
    return (spaceCount == 3);
}

double getCardValue(string s) {
    // Mengembalikan nilai kartu pada game 24
    if (s.length() > 1) {
        if (s == "10") return 10.0;
    } else {
        if (s == "A") return 1.0;
        else if (s == "J") return 11.0;
        else if (s == "Q") return 12.0;
        else if (s == "K") return 13.0;
        else if (s >= "2" && s <= "9") return (double) (s[0] - '0');
    }

    return -1;
}

string getCardName(int val) {
    // Mengembalikan nama kartu berdasarkan nilai kartu pada game 24
    if (val == 1) return "A";
    else if (val == 11) return "J";
    else if (val == 12) return "Q";
    else if (val == 13) return "K";
    else return to_string(val);
}

double getOperationResult(string op, double leftval, double rightval) {
    // Mengembalikan hasil operasi aritmatika dengan masukan 2 nilai
dan operator
    double ret = 0;
    if (op == "+") {
        ret = leftval + rightval;
    } else if (op == "-") {
        ret = leftval - rightval;
    } else if (op == "*") {
        ret = leftval * rightval;
    } else if (op == "/") {
        ret = leftval / rightval;
    }
    return ret;
}
```

```
void assignCard(string s) {
    // Memasukkan kartu hasil input pada vector cards
    cards.clear();

    string temp = "";
    for (auto x : s) {
        if (x == ' ') {
            cards.push_back(getCardValue(temp));
            temp.clear();
        } else {
            temp.push_back(x);
        }
    }

    cards.push_back(getCardValue(temp));
    return;
}

void inputUser() {
    // Prosedur meminta input 4 kartu kepada user
    string s;

    cin.ignore();
    while (true) {
        getline(cin, s);
        if (isInputValid(s)) {
            break;
        } else {
            cout << "Masukan tidak sesuai. Ulangi!" << endl;
        }
    }
    assignCard(s);

    return;
}

void generateRandomCards() {
    // Prosedur menghasilkan 4 kartu secara random
    srand(time(0));
    for (int i = 0; i < 4; i++) cards.push_back((double) (rand() % 13 +
1));

    cout << "Kartu yang dihasilkan yaitu: ";
    for (auto nums : cards) cout << getCardName((int) nums) << " ";
    cout << endl;
}

bool isSolution(double val) {
```

```
    // Memeriksa apakah val merupakan 24. Untuk menghindari kemungkinan
bug compare pada double.
    return abs(val - 24) <= 1e-9;
}

void saveSolution(string a, string b, string c, string d, string e,
vector<double> val) {
    // Meng-increment solCount dan memformat serta menyimpan solusi
    solCount++;
    solutions.push_back(a + to_string((int) val[0]) + b +
to_string((int) val[1]) + c + to_string((int) val[2]) + d +
to_string((int) val[3]) + e);
}

void solve(vector<double> nums, vector<string> op) {
    // Mengecek semua kemungkinan solusi 24
    double op1, op2, op3;

    // ((num1 op1 num2) op2 num3) op3 num4
    op1 = getOperationResult(op[0], nums[0], nums[1]);
    op2 = getOperationResult(op[1], op1, nums[2]);
    op3 = getOperationResult(op[2], op2, nums[3]);

    if (isSolution(op3)) {
        saveSolution("((", " " + op[0] + " ", ") " + op[1] + " ", ") "
+ op[2] + " ", "", nums);
    }

    // (num1 op1 (num2 op2 num3)) op3 num4
    op1 = getOperationResult(op[1], nums[1], nums[2]);
    op2 = getOperationResult(op[0], nums[0], op1);
    op3 = getOperationResult(op[2], op2, nums[3]);

    if (isSolution(op3)) {
        saveSolution("(", " " + op[0] + " (", " " + op[1] + " ", ")) "
+ op[2] + " ", "", nums);
    }

    // (num1 op1 num2) op2 (num3 op3 num4)
    op1 = getOperationResult(op[0], nums[0], nums[1]);
    op2 = getOperationResult(op[2], nums[2], nums[3]);
    op3 = getOperationResult(op[1], op1, op2);

    if (isSolution(op3)) {
        saveSolution("(", " " + op[0] + " ", ") " + op[1] + " (", " " +
op[2] + " ", ")", nums);
    }
```

```
    // num1 op1 ((num2 op2 num3) op4 num4)
    op1 = getOperationResult(op[1], nums[1], nums[2]);
    op2 = getOperationResult(op[2], op1, nums[3]);
    op3 = getOperationResult(op[0], nums[0], op2);

    if (isSolution(op3)) {
        saveSolution("", " " + op[0] + " ((", " " + op[1] + " ", ") " +
op[2] + " ", ")", nums);
    }

    // num1 op1 (num2 op2 (num3 op3 num4))
    op1 = getOperationResult(op[2], nums[2], nums[3]);
    op2 = getOperationResult(op[1], nums[1], op1);
    op3 = getOperationResult(op[0], nums[0], op2);

    if (isSolution(op3)) {
        saveSolution("", " " + op[0] + " (", " " + op[1] + " (", " " +
op[2] + " ", "))", nums);
    }

    return;
}

void generatePerm(vector<double> perms, vector<bool> isTaken, int
level) {
    // Menghasilkan permutasi dari cards dan memanggil solve untuk
setiap kemungkinan operator
    if (level == 4) {
        for (auto op1 : operators) {
            for (auto op2 : operators) {
                for (auto op3 : operators) {
                    solve(perms, {op1, op2, op3});
                }
            }
        }
        return;
    }

    for (int i = 0; i < 4; i++) {
        if (!isTaken[i]) {
            if (isDupe[i] != -1 && !isTaken[isDupe[i]]) continue;
            perms.push_back(cards[i]);
            isTaken[i] = true;
            generatePerm(perms, isTaken, level + 1);
            perms.pop_back();
            isTaken[i] = false;
        }
    }
```

```
}

void printSolution() {
    // Prosedur output solusi
    cout << endl;
    if (solCount > 0) {
        cout << "Terdapat " << solCount << " solusi, yaitu:" << endl;
        for (auto x : solutions) cout << x << endl;
    } else {
        cout << "Tidak terdapat solusi untuk kombinasi kartu di atas"
<< endl;
    }
}

void writeSolutionToFile() {
    // Prosedur menyimpan solusi
    string filename;

    cout << "File akan disimpan dalam bentuk file .txt pada folder
test" << endl << "Masukkan nama file: ";
    cin >> filename;

    ofstream SaveFile("../test/" + filename + ".txt");
    SaveFile << "Kartu: ";
    for (auto nums : cards) SaveFile << getCardName((int) nums) << " ";
    SaveFile << endl << endl;
    if (solCount > 0) {
        SaveFile << "Terdapat " << solCount << " solusi, yaitu:" <<
endl;
        for (auto x : solutions) SaveFile << x << endl;
    } else {
        SaveFile << "Tidak terdapat solusi untuk kombinasi kartu di
atas" << endl;
    }

    SaveFile.close();
}

int main() {
    // Main Program

    // Meminta masukan dari pengguna berupa opsi input kartu manual
atau random
    while (true) {
        cout << "Opsi Input:\n" << "1. Masukan dari Pengguna\n" << "2.
Random\n" << "Pilihan Pengguna: ";
        int opt;
        cin >> opt;
```

```
        if (opt == 1) {
            inputUser();
        } else if (opt == 2) {
            generateRandomCards();
        } else {
            cout << "Masukan tidak sesuai. Ulangi!" << endl << endl;
            continue;
        }
        break;
    }

    // Memulai penghitungan waktu eksekusi
    auto start = chrono::steady_clock::now();

    // Memeriksa kartu yang merupakan duplikat
    for (int i = 1; i < 4; i++) {
        for (int j = 0; j < i; j++) {
            if (cards[i] == cards[j]) isDupe[i] = j;
        }
    }

    // Mencari solusi
    generatePerm({}, {false, false, false, false}, 0);

    // Mengakhiri penghitungan waktu eksekusi
    auto end = chrono::steady_clock::now();

    // Output solusi
    printSolution();
    cout << endl << "Waktu eksekusi (dalam ms) : " <<
chrono::duration_cast<chrono::milliseconds>(end - start).count() << "
ms" << endl;

    // Opsi save solusi pada file
    while (true) {
        string pilihan;
        cout << endl << "Apakah hasil mau disimpan pada file (Y/N)? ";
        cin >> pilihan;

        if (pilihan == "Y") {
            writeSolutionToFile();
            break;
        } else if (pilihan == "N") {
            break;
        } else {
            cout << "Masukan tidak sesuai. Ulangi!" << endl << endl;
        }
```

```
    }

    cout << "Program Selesai" << endl;
}
```

4. Screenshot Contoh Input/Output

1. Contoh Input Benar

```
Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 1
6 6 6 6

Terdapat 7 solusi, yaitu:
((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 (dalam ms) : 0 ms

Apakah hasil mau disimpan pada file (Y/N)? Y
File akan disimpan dalam bentuk file .txt pada folder test
Masukkan nama file: result1
Program Selesai
```

2. Contoh Input Kartu Salah

```
Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 1
1 2 3 4
Masukan tidak sesuai. Ulangi!
8 9 10 11
Masukan tidak sesuai. Ulangi!
A B C D
Masukan tidak sesuai. Ulangi!
2 2 2 2

Tidak terdapat solusi untuk kombinasi kartu di atas

Waktu eksekusi (dalam ms) : 0 ms

Apakah hasil mau disimpan pada file (Y/N)? Y
File akan disimpan dalam bentuk file .txt pada folder test
Masukkan nama file: result2
Program Selesai
```

3. Contoh Random dan Opsi Salah

```
Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 3
Masukan tidak sesuai. Ulangi!

Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 0
Masukan tidak sesuai. Ulangi!

Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 2
Kartu yang dihasilkan yaitu: 4 J 10 9

Terdapat 6 solusi, yaitu:
4 + ((11 - 9) * 10)
4 + (10 * (11 - 9))
4 - (10 * (9 - 11))
4 - ((9 - 11) * 10)
((11 - 9) * 10) + 4
(10 * (11 - 9)) + 4

Waktu eksekusi (dalam ms) : 2 ms

Apakah hasil mau disimpan pada file (Y/N)? F
Masukan tidak sesuai. Ulangi!


Apakah hasil mau disimpan pada file (Y/N)? Y
File akan disimpan dalam bentuk file .txt pada folder test
Masukkan nama file: result3
Program Selesai
```

4. Contoh Random dan tidak ada solusi

```
Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 2
Kartu yang dihasilkan yaitu: 6 K K 6

Tidak terdapat solusi untuk kombinasi kartu di atas

Waktu eksekusi (dalam ms) : 0 ms

Apakah hasil mau disimpan pada file (Y/N)? Y
File akan disimpan dalam bentuk file .txt pada folder test
Masukkan nama file: result4
Program Selesai
```

5. Contoh Random

```
Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 2
Kartu yang dihasilkan yaitu: 7 8 5 6

Terdapat 28 solusi, yaitu:
((7 - 8) + 5) * 6
(7 - (8 - 5)) * 6
((7 + 5) - 8) * 6
(7 + (5 - 8)) * 6
(7 + 5) * (8 - 6)
(8 / (7 - 5)) * 6
8 / ((7 - 5) / 6)
(8 - 6) * (7 + 5)
(8 * 6) / (7 - 5)
8 * (6 / (7 - 5))
(8 - 6) * (5 + 7)
((5 + 7) - 8) * 6
(5 + (7 - 8)) * 6
(5 + 7) * (8 - 6)
((5 - 8) + 7) * 6
(5 - (8 - 7)) * 6
6 * ((7 - 8) + 5)
6 * (7 - (8 - 5))
6 * ((7 + 5) - 8)
6 * (7 + (5 - 8))
(6 / (7 - 5)) * 8
6 / ((7 - 5) / 8)
(6 * 8) / (7 - 5)
6 * (8 / (7 - 5))
6 * ((5 + 7) - 8)
6 * (5 + (7 - 8))
6 * ((5 - 8) + 7)
6 * (5 - (8 - 7))

Waktu eksekusi (dalam ms) : 2 ms

Apakah hasil mau disimpan pada file (Y/N)? Y
File akan disimpan dalam bentuk file .txt pada folder test
Masukkan nama file: result5
Program Selesai
```

6. Contoh Random

```
Opsi Input:
1. Masukan dari Pengguna
2. Random
Pilihan Pengguna: 2
Kartu yang dihasilkan yaitu: A K K Q

Terdapat 4 solusi, yaitu:
(1 + (13 / 13)) * 12
((13 / 13) + 1) * 12
12 * (1 + (13 / 13))
12 * ((13 / 13) + 1)

Waktu eksekusi (dalam ms) : 0 ms

Apakah hasil mau disimpan pada file (Y/N)? Y
File akan disimpan dalam bentuk file .txt pada folder test
Masukkan nama file: result6
Program Selesai
```

5. Lampiran

Link Repository Implementasi Program:
https://github.com/Altair1618/Tucil1_13521114