Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Semester II Tahun 2022/2023

Hanif Muhammad Zhafran

13521157

Teknik Informatika 2021

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

A. Langkah-langkah Algoritma Brute Force

1. Algoritma validasi input

- Di awal program, pengguna akan diminta untuk memilih salah satu dari dua jenis input. Input jenis pertama adalah input yang dilakukan oleh user, yang kedua adalah input random.
- Untuk input jenis pertama, pengguna diminta untuk menginput empat elemen yang masing-masing dipisah dengan spasi (Contoh: A 10 8 J). Jika input pengguna tidak sama dengan empat elemen atau ada elemen yang tidak diperbolehkan (elemen yang boleh hanya A, J, Q, K, dan angka 2-10), maka pengguna akan diminta input kembali sampai input dari pengguna benar.
- Tiap elemen dari input pengguna akan dimasukkan ke dalam suatu vector (vector STL C++) dalam bentuk double. Jika elemen input adalah A, J, Q, atau K, maka akan diubah dulu menjadi berturut-turut 1, 11, 12, dan 13.
- Untuk input jenis kedua, vector akan diisi dengan empat elemen secara acak yang memiliki range 1-13 dengan menggunakan fungsi srand() dan rand() dalam C++.
- Empat elemen yang dipilih secara acak akan ditampilkan ke layer dengan format X X X X (Contoh: K A 6 10).

2. Algoritma pencarian solusi

- Elemen pada vector diurutkan terlebih dahulu dari kecil ke besar dengan menggunakan algoritma selection sort.
- Algoritma selection sort memiliki langkah-langkah sebagai berikut:
  1) Cari elemen terkecil dalam vector, kemudian pindahkan elemen terkecil ke urutan pertama.
  2) Ulangi tahap 1 dengan range pencarian dimulai dari urutan kedua, ketiga, dst. Pemindahan elemen tersebut juga ditujukan ke urutan kedua, ketiga, dst. Tahap 2 dilakukan sampai pemrosesan dilakukan sampai diperoleh vector yang terurut
- Setelah diperoleh vector terurut, cari semua permutasi yang mungkin dari elemen-elemen pada vector.
- Pencarian permutasi akan dilakukan secara terurut sampai diperoleh permutasi urutan terakhir dengan urutan elemen vector adalah dari besar ke kecil. Contohnya adalah jika elemen vectornya adalah A 2 3 4, maka permutasi terakhirnya adalah 4 3 2 A.
- Algoritma untuk mencari permutasi selanjutnya dari suatu keadaan adalah sebagai berikut (akan dilengkapi dengan contoh vectornya):
  1) *Traverse* elemen vector dari elemen paling akhir ke awal. *Traverse* dihentikan ketika ditemukan elemen yang lebih kecil dari elemen sebelumnya. Kemudian tandai elemen tersebut. (Contoh: 4 3 8 5)

2) Cari nilai minimum dari nilai-nilai yang ada di kanan dari elemen yang sudah ditandai. (Contoh: 4 3 8 5).
3) Tukar elemen yang ditandai pada tahap 1 dengan elemen yang ditandai pada tahap 2. (Contoh: 4 5 8 3).
4) Balikkan urutan elemen-elemen yang ada di kanan elemen yang ditandai pada tahap 2. (Contoh: 4 5 3 8).

- Dari setiap permutasi elemen-elemen vector, akan dicoba setiap susunan tanda kurung sebagai berikut (11 susunan):

1) X op X op X op X
2) (X op X) op X op X
3) X op (X op X) op X
4) X op X op (X op X)
5) (X op X op X) op X
6) X op (X op X op X)
7) ((X op X) op X) op X
8) (X op (X op X)) op X
9) X op ((X op X) op X)
10) X op (X op (X op X))
11) (X op X) op (X op X)

- Dari setiap susunan tanda kurung, akan dicoba seluruh susunan operator yang mungkin dari empat operator yang ada (tambah, kurang, kali, bagi). Total susunan operator yang mungkin adalah $4^3 = 64$ susunan.
- Jika suatu ekspresi menghasilkan angka 24, maka ekspresi tersebut akan ditambahkan sebagai solusi. Solusi disimpan dalam bentuk string, kemudian variabel penyimpan jumlah solusi akan ditambah 1.
- Setelah semua kemungkinan sudah diperiksa, program akan menampilkan jumlah solusi yang ditemukan dan seluruh ekspresi yang termasuk ke dalam solusi. Program juga akan menampilkan waktu yang dibutuhkan dalam satuan *microseconds* untuk melakukan proses pencarian solusi.

3. Algoritma penyimpanan solusi

- Setelah solusi ditampilkan di layer, pengguna akan diminta input sebuah karakter (y/n) apakah solusi ingin disimpan dalam file atau tidak.
- Jika iya (y), pengguna akan diminta nama file yang akan disimpan. Setelah itu, seluruh solusi disimpan dalam file tersebut beserta dengan susunan awal vector, waktu yang dibutuhkan, dan juga jumlah solusi yang ditemukan.

- Jika tidak (n), maka program akan langsung selesai.
- Jika input bukan merupakan karakter y atau n, pengguna akan diminta untuk melakukan input ulang.

B. Source Program

Struktur program dibagi menjadi 4 bagian, yaitu main program, header file, source code berisi implementasi fungsi-fungsi, dan source code untuk membuat potongan kode. Program menggunakan percabangan (if-else) pada proses pengecekan tanda kurung dan susunan operator. Sehingga, potongan kode yang dibutuhkan menjadi cukup panjang. Karena potongan kode yang dibutuhkan memiliki pola, dibuat program khusus untuk menuliskan kode program untuk pengecekan susunan tanda kurung dan susunan operator.

1. Main program

```cpp
#include "24game.hpp"
#include <sstream>
#include <cstdlib>

int main() {

    // Inisiasi variabel
    int n;
    std::string card = "";
    std::vector<double> input;
    std::cout << "Welcome to 24 Game Solver." << std::endl;
    std::cout << "Choose input:" << std::endl;
    std::cout << "1. User input" << std::endl;
    std::cout << "2. Random" << std::endl;
    while (true) {
        std::cin >> n;
        if (n == 1) {   // Input dari pengguna
            std::cin.clear();
            fflush(stdin);
            std::cout << "Type in the input with the format below.\nX X X X" << std::endl;
```

```
        while (true) {

            std::getline(std::cin, card);
            std::stringstream ss(card);
            std::string str_num;

            bool valid = true;
            while (ss >> str_num) {
                if (str_num == "A") {
                    input.push_back(1);
                } else if (str_num == "J") {
                    input.push_back(11);
                } else if (str_num == "Q") {
                    input.push_back(12);
                } else if (str_num == "K") {
                    input.push_back(13);
                } else if (str_num == "2" || str_num == "3" || str_num == "4" ||
                           str_num == "5" || str_num == "6" || str_num == "7" ||
                           str_num == "8" || str_num == "9" || str_num == "10") {
                    input.push_back(stoi(str_num));
                } else {
                    std::cout << "Input not valid, please try again." << std::endl;
                    valid = false;
                    ss.clear();
                    input.clear();
                    break;
                }
            }

            if (!valid) {
                continue;
            }

            if (input.size() != 4) {
                std::cout << "Input not valid, please try again." << std::endl;
                ss.clear();
                input.clear();
                continue;
            }

            break;
        }
        break;

```

```
64
65          } else if (n == 2) { // Input random
66
67              srand(time(0));
68              for (int i = 0; i < 4; i++) {
69                  input.push_back((rand() % 13) + 1);
70                  if (input[i] == 1) {
71                      card += "A ";
72                  } else if (input[i] == 11) {
73                      card += "J ";
74                  } else if (input[i] == 12) {
75                      card += "Q ";
76                  } else if (input[i] == 13) {
77                      card += "K ";
78                  } else {
79                      card += std::to_string((int)input[i]) + " ";
80                  }
81              }
82              card += "\n";
83              std::cout << card;
84              break;
85
86          } else {
87              std::cout << "Input not valid, please try again." << std::endl;
88          }
89
90      }
```

```
92      // Waktu awal
93      auto time1 = std::chrono::high_resolution_clock::now();
94
95      selectionSort(input);
96      int count = 0;
97      std::string solution = "";
98
99      // Pencarian solusi
100     do {
101         computeSolution(input, count, solution);
102     } while (nextPermutation(input));
103
104     // Waktu akhir
105     auto time2 = std::chrono::high_resolution_clock::now();
106     auto duration = std::chrono::duration_cast<std::chrono::microseconds>(time2 - time1);
107
108     // Pencetakan solusi, jumlah solusi, dan waktu ke layar
109     if (count == 0) {
110         std::cout << "No solution found." << std::endl;
111     } else {
112         std::cout << count << " solution(s) found." << std::endl;
113     }
114     std::cout << solution;
115     std::cout << "Time (microseconds): " << duration.count() << std::endl;
```

```cpp
    // Penyimpanan solusi dalam file
    char save;
    std::cout << "Store the solution in a .txt file?(y/n) ";
    while (true) {
        std::cin >> save;
        if (save == 'y') {

            std::string name;
            std::cout << "Input solution file name: ";
            std::cin >> name;
            std::ofstream FileSolution("test/" + name + ".txt");

            FileSolution << card;
            if (count == 0) {
                FileSolution << "No solution found." << std::endl;
            } else {
                FileSolution << count << " solution(s) found." << std::endl;
            }
            FileSolution << "Time (microseconds): " << duration.count() << std::endl;
            FileSolution << solution;
            std::cout << "Success." << std::endl;
            break;
        } else if (save != 'y' && save != 'n') {

            std::cout << "Input not valid, please try again." << std::endl;
            continue;
        }
        break;
    }

    std::cout << "Program Finished." << std::endl;
}
```

2. Header file

src > 24game.hpp > ...

```cpp
#ifndef HEADER_24GAME
#define HEADER_24GAME

#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <fstream>
#include <chrono>

void swap(double& a, double& b);

void selectionSort(std::vector<double>& numVec);

void reverseSubVec(std::vector<double>& numVec, int idxFirst, int idxLast);

bool nextPermutation(std::vector<double>& numVec);

void computeSolution(std::vector<double> input, int& count, std::string& solution);

#endif
```

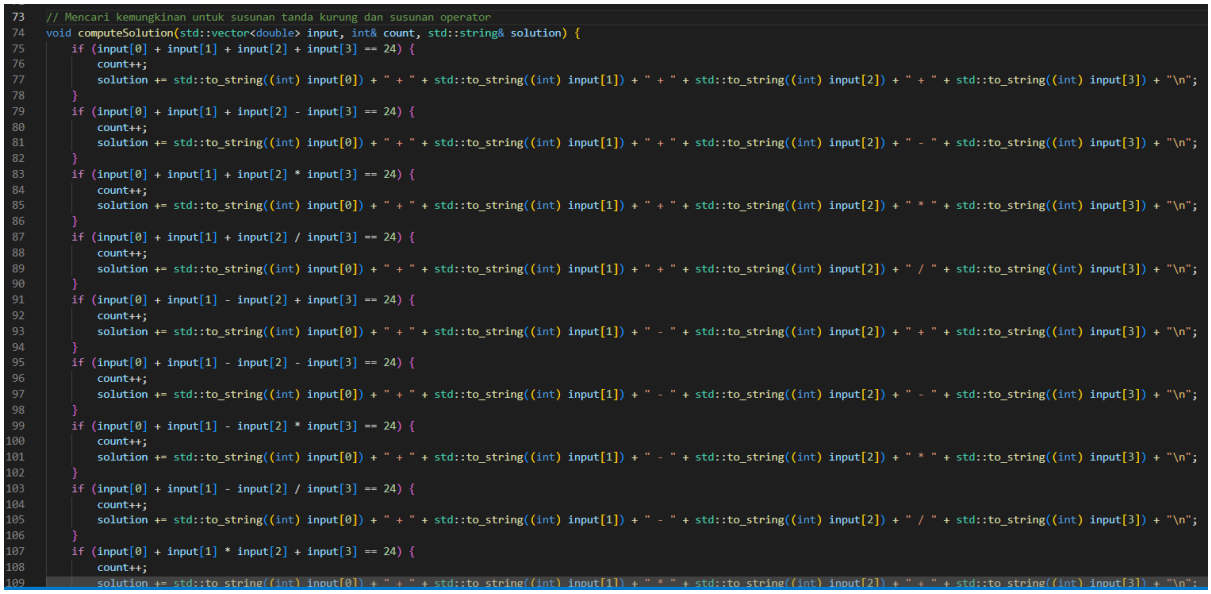
3. Implementasi fungsi

src > 24game.cpp > ...

```cpp
#include "24game.hpp"

// Menukar elemen
void swap(double& a, double& b) {
    double temp = a;
    a = b;
    b = temp;
}

// Algoritma pengurutan selection sort
void selectionSort(std::vector<double>& numVec) {
    int idxMin;
    double min;

    for (int i = 0; i < 4; i++) {
        idxMin = i;
        min = numVec[i];
        for (int j = i + 1; j < 4; j++) {
            if (min > numVec[j]) {
                min = numVec[j];
                idxMin = j;
            }
        }
        swap(numVec[i], numVec[idxMin]);
    }
}
```

src > 24game.cpp > ...

```cpp
// Membalikkan urutan suatu sub bagian dari suatu vector
void reverseSubVec(std::vector<double>& numVec, int idxFirst, int idxLast) {
    while (idxFirst < idxLast) {
        swap(numVec[idxFirst], numVec[idxLast]);
        idxFirst++;
        idxLast--;
    }
}

// Pencarian urutan permutasi berikutnya
bool nextPermutation(std::vector<double>& numVec) {
    int idxSwap;
    bool lastPermutation = true;
    for (int i = 3; i > 0; i--) {
        if (numVec[i] > numVec[i - 1]) {
            idxSwap = i - 1;
            lastPermutation = false;
            break;
        }
    }

    if (lastPermutation) {
        return false;
    }

    bool found = false;
    int min, idxMin;
    for (int i = 3; i > idxSwap; i--) {
        if (!found && numVec[idxSwap] < numVec[i]) {
            found = true;
            min = numVec[i];
            idxMin = i;
        } else if (found) {
            if (min > numVec[i]) {
                min = numVec[i];
                idxMin = i;
            }
        }
    }

    swap(numVec[idxMin], numVec[idxSwap]);
    reverseSubVec(numVec, idxSwap + 1, 3);
    return true;
}
```



```
// Mencari kemungkinan untuk susunan tanda kurung dan susunan operator
void computeSolution(std::vector<double> input, int& count, std::string& solution) {
    if (input[0] + input[1] + input[2] + input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " + " + std::to_string((int) input[2]) + " + " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] + input[2] - input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " + " + std::to_string((int) input[2]) + " - " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] + input[2] * input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " + " + std::to_string((int) input[2]) + " * " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] + input[2] / input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " + " + std::to_string((int) input[2]) + " / " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] - input[2] + input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " - " + std::to_string((int) input[2]) + " + " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] - input[2] - input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " - " + std::to_string((int) input[2]) + " - " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] - input[2] * input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " - " + std::to_string((int) input[2]) + " * " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] - input[2] / input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " - " + std::to_string((int) input[2]) + " / " + std::to_string((int) input[3]) + "\n";
    }
    if (input[0] + input[1] * input[2] + input[3] == 24) {
        count++;
        solution += std::to_string((int) input[0]) + " + " + std::to_string((int) input[1]) + " * " + std::to_string((int) input[2]) + " + " + std::to_string((int) input[3]) + "\n";
```

Untuk gambar source code dalam pencarian solusi untuk kemungkinan susunan tanda kurung dan susunan operator tidak ditampilkan secara keseluruhan karena source code cukup panjang.

4. Source code untuk membuat potongan kode untuk kemungkinan susunan tanda kurung dan susunan operator.

src > codeGenerator.cpp > main()

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <fstream>

using namespace std;
int main() {
    vector<char> op = {'+','+','+'};
    // 0+, 1-, 2*, 3/
    ofstream MyFile("script.txt");
    for (int a = 0; a < 11; a++) {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                for (int k = 0; k < 4; k++) {
                    if (i == 0) {
                        op[0] = '+';
                    }
                    if (i == 1) {
                        op[0] = '-';
                    }
                    if (i == 2) {
                        op[0] = '*';
                    }
                    if (i == 3) {
                        op[0] = '/';
                    }
                    if (j == 0) {
                        op[1] = '+';
                    }
                    if (j == 1) {
                        op[1] = '-';
                    }
                    if (j == 2) {
                        op[1] = '*';
                    }
                    if (j == 3) {
                        op[1] = '/';
                    }
                    if (k == 0) {
                        op[2] = '+';
                    }
                    if (k == 1) {
                        op[2] = '-';
                    }
                    if (k == 2) {
                        op[2] = '*';
                    }
                    if (k == 3) {
                        op[2] = '/';
                    }
```

```
                switch (a) {
                    case 0:
                        // A B C D
                        MyFile << "\t\tif (input[0] " << op[0] << " input[1] " << op[1] << " input[2] " << op[2] << " input[3] == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += std::to_string((int) input[0]) + \" " <<
                        op[0] << " \" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \" " << op[2] << " \" + std::to_string((int) input[3]) + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 1:
                        // (A B) C D
                        MyFile << "\t\tif ((input[0] " << op[0] << " input[1]) " << op[1] << " input[2] " << op[2] << " input[3] == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += \"(\" + std::to_string((int) input[0]) + \" " <<
                        op[0] << " \" + std::to_string((int) input[1]) + \")\" + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \" " << op[2] << " \" + std::to_string((int) input[3]) + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 2:
                        // A (B C) D
                        MyFile << "\t\tif (input[0] " << op[0] << " (input[1] " << op[1] << " input[2]) " << op[2] << " input[3] == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + \"(\" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \")\" + \" " << op[2] << " \" + std::to_string((int) input[3]) + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 3:
                        // A B (C D)
                        MyFile << "\t\tif (input[0] " << op[0] << " input[1] " << op[1] << " (input[2] " << op[2] << " input[3]) == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + std::to_string((int) input[1]) + \" " << op[1] << " \" + \"(\" + std::to_string((int) input[2]) + \" " <<
                        op[2] << " \" + std::to_string((int) input[3]) + \")\" + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 4:
                        // (A B C) D
                        MyFile << "\t\tif ((input[0] " << op[0] << " input[1] " << op[1] << " input[2]) " << op[2] << " input[3] == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += \"(\" + std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \")\" + \" " << op[2] <<
                        " \" + std::to_string((int) input[3]) + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 5:
                        // A (B C D)
                        MyFile << "\t\tif (input[0] " << op[0] << " (input[1] " << op[1] << " input[2] " << op[2] << " input[3]) == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + \"(\" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \" " << op[2] << " \" + std::to_string((int) input[3]) + \")\" + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 6:
                        // ((A B) C) D
                        MyFile << "\t\tif (((input[0] " << op[0] << " input[1]) " << op[1] << " input[2]) " << op[2] << " input[3] == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += \"((\" + std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + std::to_string((int) input[1]) + \")\"+ \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \")\" + \" " << op[2] << " \" + std::to_string((int) input[3]) + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 7:
                        // (A (B C)) D
                        MyFile << "\t\tif ((input[0] " << op[0] << " (input[1] " << op[1] << " input[2])) " << op[2] << " input[3] == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += \"(\" + std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + \"(\" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \"))\" + \" " << op[2] << " \" + std::to_string((int) input[3]) + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 8:
                        // A ((B C) D)
                        MyFile << "\t\tif (input[0] " << op[0] << " ((input[1] " << op[1] << " input[2]) " << op[2] << " input[3]) == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + \"((\" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + std::to_string((int) input[2]) + \")\" + \" " << op[2] << " \" + std::to_string((int) input[3]) + \")\" + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 9:
                        // A (B (C D))
                        MyFile << "\t\tif (input[0] " << op[0] << " (input[1] " << op[1] << " (input[2] " << op[2] << " input[3])) == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + \"(\" + std::to_string((int) input[1]) + \" " << op[1] <<
                        " \" + \"(\" + std::to_string((int) input[2]) + \" " << op[2] << " \" + std::to_string((int) input[3]) + \"))\" + \"\\n\";\n\t\t}" << endl;
                        break;
                    case 10:
                        // (A B) (C D)
                        MyFile << "\t\tif ((input[0] " << op[0] << " input[1]) " << op[1] << " (input[2] " << op[2] << " input[3]) == 24) {" << endl;
                        MyFile << "\t\t\tcount++;" << endl;
                        MyFile << "\t\t\tsolution += \"(\" + std::to_string((int) input[0]) + \" " << op[0] <<
                        " \" + std::to_string((int) input[1]) + \")\" + \" " << op[1] <<
                        " \" + \"(\" + std::to_string((int) input[2]) + \" " << op[2] << " \" + std::to_string((int) input[3]) + \")\" + \"\\n\";\n\t\t}" << endl;
                        break;
                }
            }
        }
    }
}
```

C. Test Case

<table>
<tr><th>Output layar</th><th>Output teks</th></tr>
<tr><td>

```
Welcome to 24 Game Solver.
Choose input:
1. User input
2. Random
2
Q 7 J 4
24 solution(s) found.
4 * (7 + 11 - 12)
4 * ((7 + 11) - 12)
4 * (7 + (11 - 12))
4 * (7 - 12 + 11)
4 * ((7 - 12) + 11)
4 * (7 - (12 - 11))
4 * (11 + 7 - 12)
4 * ((11 + 7) - 12)
4 * (11 + (7 - 12))
4 * (11 - 12 + 7)
4 * ((11 - 12) + 7)
4 * (11 - (12 - 7))
(7 + 11 - 12) * 4
((7 + 11) - 12) * 4
(7 + (11 - 12)) * 4
(7 - 12 + 11) * 4
((7 - 12) + 11) * 4
(7 - (12 - 11)) * 4
(11 + 7 - 12) * 4
((11 + 7) - 12) * 4
(11 + (7 - 12)) * 4
(11 - 12 + 7) * 4
((11 - 12) + 7) * 4
(11 - (12 - 7)) * 4
Time (microseconds): 1842
Store the solution in a .txt file?(y/n) y
Input solution file name: sol2
Success.
Program Finished.
```

</td><td>

test > sol2.txt

```
1   Q 7 J 4
2   24 solution(s) found.
3   Time (microseconds): 1842
4   4 * (7 + 11 - 12)
5   4 * ((7 + 11) - 12)
6   4 * (7 + (11 - 12))
7   4 * (7 - 12 + 11)
8   4 * ((7 - 12) + 11)
9   4 * (7 - (12 - 11))
10  4 * (11 + 7 - 12)
11  4 * ((11 + 7) - 12)
12  4 * (11 + (7 - 12))
13  4 * (11 - 12 + 7)
14  4 * ((11 - 12) + 7)
15  4 * (11 - (12 - 7))
16  (7 + 11 - 12) * 4
17  ((7 + 11) - 12) * 4
18  (7 + (11 - 12)) * 4
19  (7 - 12 + 11) * 4
20  ((7 - 12) + 11) * 4
21  (7 - (12 - 11)) * 4
22  (11 + 7 - 12) * 4
23  ((11 + 7) - 12) * 4
24  (11 + (7 - 12)) * 4
25  (11 - 12 + 7) * 4
26  ((11 - 12) + 7) * 4
27  (11 - (12 - 7)) * 4
28
```

</td></tr>
<tr><td>

```
Welcome to 24 Game Solver.
Choose input:
1. User input
2. Random
1
Type in the input with the format below.
X X X X
6 7 J 3
No solution found.
Time (microseconds): 1531
Store the solution in a .txt file?(y/n) y
Input solution file name: sol1
Success.
Program Finished.
```

</td><td>

test > sol1.txt

```
1   6 7 J 3
2   No solution found.
3   Time (microseconds): 1531
4
```

</td></tr>
</table>

```
Welcome to 24 Game Solver.
Choose input:
1. User input
2. Random
1
Type in the input with the format below.
X X X X
6 6 6 6
17 solution(s) found.
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)
(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)
6 + (6 + (6 + 6))
(6 + 6) + (6 + 6)
(6 * 6) - (6 + 6)
Time (microseconds): 753
Store the solution in a .txt file?(y/n) y
Input solution file name: sol3
Success.
Program Finished.
```

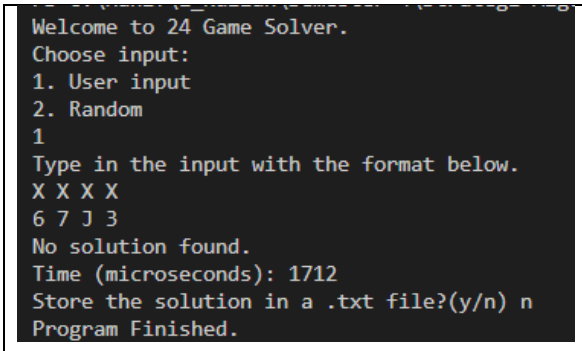
test > sol3.txt

```
6 6 6 6
17 solution(s) found.
Time (microseconds): 753
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)
(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)
6 + (6 + (6 + 6))
(6 + 6) + (6 + 6)
(6 * 6) - (6 + 6)

```

```
PS C:\Hanif\1_Kuliah\Semester 4\Strategi Algo
Welcome to 24 Game Solver.
Choose input:
1. User input
2. Random
1
Type in the input with the format below.
X X X X
A Q K 7
24 solution(s) found.
(1 + 13) / 7 * 12
((1 + 13) / 7) * 12
(1 + 13) / (7 / 12)
(1 + 13) * 12 / 7
((1 + 13) * 12) / 7
(1 + 13) * (12 / 7)
12 * (1 + 13) / 7
(12 * (1 + 13)) / 7
12 * ((1 + 13) / 7)
12 / 7 * (1 + 13)
12 / (7 / (1 + 13))
(12 / 7) * (1 + 13)
12 / 7 * (13 + 1)
12 / (7 / (13 + 1))
(12 / 7) * (13 + 1)
12 * (13 + 1) / 7
(12 * (13 + 1)) / 7
12 * ((13 + 1) / 7)
(13 + 1) / 7 * 12
((13 + 1) / 7) * 12
(13 + 1) / (7 / 12)
(13 + 1) * 12 / 7
((13 + 1) * 12) / 7
(13 + 1) * (12 / 7)
Time (microseconds): 1664
Store the solution in a .txt file?(y/n) y
Input solution file name: sol4
Success.
Program Finished.
```

test > sol4.txt

```
A Q K 7
24 solution(s) found.
Time (microseconds): 1634
(1 + 13) / 7 * 12
((1 + 13) / 7) * 12
(1 + 13) / (7 / 12)
(1 + 13) * 12 / 7
((1 + 13) * 12) / 7
(1 + 13) * (12 / 7)
12 * (1 + 13) / 7
(12 * (1 + 13)) / 7
12 * ((1 + 13) / 7)
12 / 7 * (1 + 13)
12 / (7 / (1 + 13))
(12 / 7) * (1 + 13)
12 / 7 * (13 + 1)
12 / (7 / (13 + 1))
(12 / 7) * (13 + 1)
12 * (13 + 1) / 7
(12 * (13 + 1)) / 7
12 * ((13 + 1) / 7)
(13 + 1) / 7 * 12
((13 + 1) / 7) * 12
(13 + 1) / (7 / 12)
(13 + 1) * 12 / 7
((13 + 1) * 12) / 7
(13 + 1) * (12 / 7)

```

	-

D. Checklist Status 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	✓	

E. Link Repository

- https://github.com/hanifmz07/Tucil1_13521157