Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute-force

Ditujukan untuk memenuhi tugas kecil I mata kuliah IF2211 Strategi Algoritma Semester II tahun akademik 2022/2023

Disusun oleh:

Muhammad Zaki Amanullah – 13521146

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2023

A. Algoritma Brute-force

Algoritma brute-force adalah strategi pemecahan masalah dengan mencoba semua kemungkinan dari permasalahan. Karena “approach” dari algoritma ini yang relatif sederhana, yaitu dengan mencoba segala kombinasi kemungkinan, penyelesaian masalah dengan algoritma ini relatif “straight-forward”. Dalam tugas ini, saya akan mengimplementasikan algoritma brute-force untuk mencari penyelesaian atau solusi dari puzzle atau teka-teki 24.

Dalam teka-teki ini, diberikan empat buah angka, yang harus kita ubah menjadi angka 24 dengan menggunakan operasi penjumlahan, pengurangan, perkalian, atau pembagian. Dalam, masalah ini, angka pada teka-teki 24 yang digunakan diambil dari satu dek kartu yang berisikan A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K. Program yang saya buat, akan menghasilkan semua solusi yang mungkin menghasilkan angka 24 dengan menggunakan algoritma brute-force.

Alur yang dilalui oleh program yang dibuat adalah sebagai berikut.

1. Program menerima 4 buah angka yang dapat diambil dari input user atau secara random.
2. Program mencari kombinasi dari urutan angka tersebut dan kombinasi dari operator-operator yang mungkin serta semua kemungkinan *grouping* dan mengevaluasi hasilnya.
3. Apabila hasil yang didapat adalah 24, program akan memasukkan string berisikan salah satu solusi ke dalam vector.

Secara teori karena terdapat 4 buah angka yang dianggap berbeda, kita dapat mendapatkan 4! buah susunan angka yang berbeda. Operator dapat disusun menjadi 4.4.4 buah susunan yang berbeda karena setiap tempat menempatkan operator kita memiliki empat buah pilihan. Terdapat lima kemungkinan grouping yang mungkin sebagai berikut.

1. ((a op b) op c) op d
2. (a op (b op c)) op d
3. a op ((b op c) op d)
4. a op (b op (c op d))
5. (a op b) op (c op d)

Dapat diperhatikan, bahwa dengan mempertimbangkan semua kemungkinan, program akan mengevaluasi sebanyak 7680 buah kemungkinan yang relatif mudah untuk diselesaikan oleh komputer. Oleh karena itu, permasalahan ini cocok digunakan untuk algoritma brute-force. Berikut adalah approach saya dalam menyelesaikan masalah ini.

1. Mengubah string menjadi operasi

```
double getResult(double first, double second, char op) {
    if (first == 99999 || second == 99999) {
        return 99999;
    } else {
        if (op == '*' || op == 'x') {
            return first * second;
        } else if (op == '+') {
            return first + second;
        } else if (op == '-') {
            return first - second;
        } else if (op == '/' && second != 0) {
            return first / second;
        } else if (op == '/' && second == 0) {
            return 99999;
        } return -1;
    }
}
```

Karena program akan mendapatkan input berupa string, diperlukan sebuah fungsi yang melakukan operasi antara dua angka. Dapat dilihat pada fungsi getResult terdapat suatu metode untuk menghindari terjadinya error yang akan terjadi apabila terdapat pembagian dengan angka 0. Apabila terdapat kasus seperti itu, dikembalikan nilai 99999 yang dipilih karena nilai tersebut lebih tinggi dari semua angka yang akan dicapai pada permasalahan ini (28561). Apabila terdapat satu operasi yang menghasilkan 99999 maka operasi selanjutnya akan selalu menghasilkan 99999 dan tidak mencapai 24.

2. Mencari semua kombinasi

```
string solve(int numArray[4]) {
    auto start = chrono::high_resolution_clock::now();
    int count = 0;
    vector<string> solutions;
    for (size_t l = 0; l < 4; l++) {
        for (size_t m = 0; m < 4; m++) {
            if (m != l) {
                for (size_t n = 0; n < 4; n++) {
                    if (n != m && n != l) {
                        for (size_t o = 0; o < 4; o++) {
                            if (o != m && o != n && o != l) {
                                int permutatedNum[4] = {numArray[l], numArray[m], numArray[n], numArray[o]};
                                char tempOp[3];
                                for (int i = 0; i < 4; i++) {
                                    tempOp[0] = operatorList[i];
                                    for (int j = 0; j < 4; j++) {
                                        tempOp[1] = operatorList[j];
                                        for (int k = 0; k < 4; k++) {
                                            tempOp[2] = operatorList[k];
                                            double leftLeft, leftRight, rightLeft, rightRight, splitMiddle;
```

Untuk mendapatkan semua kemungkinan susunan operator dan susunan angka yang mungkin, digunakan nested for loop sebanyak 7 kali.

```
leftLeft = getResult(getResult(getResult(permutatedNum[0], permutatedNum[1], tempOp[0]), permutatedNum[2], tempOp[1]), permutatedNum[3], tempOp[2]);
leftRight = getResult(getResult(permutatedNum[0], getResult(permutatedNum[1], permutatedNum[2], tempOp[1]), tempOp[0]), permutatedNum[3], tempOp[2]);
rightLeft = getResult(permutatedNum[0], getResult(getResult(permutatedNum[1], permutatedNum[2], tempOp[1]), permutatedNum[3], tempOp[2]), tempOp[0]);
rightRight = getResult(permutatedNum[0], getResult(permutatedNum[1], getResult(permutatedNum[2], permutatedNum[3], tempOp[2]), tempOp[1]), tempOp[0]);
splitMiddle = getResult(getResult(permutatedNum[0], permutatedNum[1], tempOp[0]), getResult(permutatedNum[2], permutatedNum[3], tempOp[2]), tempOp[1]);
```

```
if (leftLeft == 24 && !exist(solutions, getLineString(permutatedNum, tempOp, 1))) {
    count++;
    solutions.push_back(getLineString(permutatedNum, tempOp, 1));
}
if (leftRight == 24 && !exist(solutions, getLineString(permutatedNum, tempOp, 2))) {
    count++;
    solutions.push_back(getLineString(permutatedNum, tempOp, 2));
}
if (rightLeft == 24 && !exist(solutions, getLineString(permutatedNum, tempOp, 3))) {
    count++;
    solutions.push_back(getLineString(permutatedNum, tempOp, 3));
}
if (rightRight == 24 && !exist(solutions, getLineString(permutatedNum, tempOp, 4))) {
    count++;
    solutions.push_back(getLineString(permutatedNum, tempOp, 4));
}
if (splitMiddle == 24 && !exist(solutions, getLineString(permutatedNum, tempOp, 5))) {
    count++;
    solutions.push_back(getLineString(permutatedNum, tempOp, 5));
}
```

Setelah didapatkan semua kombinasi dari angka dan operator, kita lakukan operasi berdasarkan semua kemungkinan grouping yang ada. Kemudian, kita akan evaluasi apakah hasil yang didapat adalah 24 dan apakah solusi yang didapat sudah terdapat pada vector of string yang digunakan untuk menyimpan solusi. Apabila memenuhi kedua hal tersebut, maka solusi akan dimasukkan ke dalam vector.

3. Mendapatkan string dari solusi

```
string getLineString(int numArray[4], char opArray[3], int format) {
    string result;
    if (format == 1) {
        result = "((" + cardName(numArray[0]) + " " + opArray[0] + " " + cardName(numArray[1]) + ")" + " " + opArray[1] + " " + cardName(numArray[2])+ ")" + " " + opArray
    } else if (format == 2) {
        result = "(" + cardName(numArray[0]) + " " + opArray[0] + " (" + cardName(numArray[1]) + " " + opArray[1] + " " + cardName(numArray[2])+ ")) " + opArray[2] + " "
    } else if (format == 3) {
        result = cardName(numArray[0]) + " " + opArray[0] + " ((" + cardName(numArray[1]) + " " + opArray[1] + " " + cardName(numArray[2])+ ")" + " " + opArray[2] + " " +
    } else if (format == 4) {
        result = cardName(numArray[0]) + " " + opArray[0] + " (" + cardName(numArray[1]) + " " + opArray[1] + " (" + cardName(numArray[2]) + " " + opArray[2] + " " + card
    } else {
        result = "(" + cardName(numArray[0]) + " " + opArray[0] + " " + cardName(numArray[1]) + ") " + opArray[1] + " (" + cardName(numArray[2]) + " " + opArray[2] + " "
    }
    return result;
}
```

Karena grouping yang memungkinkan hanya terdapat 5 buah, dapat dibuat sebuah fungsi yang akan mengembalikan sebuah string yang sesuai dengan solusi yang ada. Format yang dimasukkan merupakan angka dari 1—5 yang direpresentasikan oleh grouping yang telah dibahas sebelumnya.

4. Menghindari hasil kembar

```
bool exist(vector<string> stringList, string element) {
    for (size_t i = 0; i < stringList.size(); i++) {
        if (stringList[i] == element) {
            return true;
        }
    }
    return false;
}
```

Karena terdapat kemungkinan apabila terdapat solusi yang sama maka dibuatlah fungsi exist yang mengembalikan true apabila vector of string yang digunakan untuk menyimpan solusi sudah memiliki solusi tersebut dan false apabila tidak.

B. Source Code Program

Semua program yang dibuat diletakkan di dalam file main.cpp yang berisi semua fungsionalitas dari pembacaan input, pencarian angka secara random, algoritma brute force, dan penulisan ke file txt.

1. Pembacaan input.

Angka yang didapat dapat berasal dari input pengguna atau berasal dari program. Untuk mendapatkan angka dari pengguna, pengguna diharapkan untuk menuliskan 4 buah nomor kartu yang valid yang dipisahkan oleh white space dan diakhiri oleh enter.

```
string removeTrailingAndLeadingSpace(string stringInput) {
    int trailingSpaceCount = 0;
    for (size_t i = stringInput.size() - 1; i >= 0; i--) {
        if (stringInput[i] == ' ') {
            trailingSpaceCount++;
        } else {
            break;
        }
    }
    int leadingSpaceCount = 0;
    for (size_t i = 0; i < stringInput.size(); i++) {
        if (stringInput[i] == ' ') {
            leadingSpaceCount++;
        } else {
            break;
        }
    }
    return stringInput.substr(leadingSpaceCount, stringInput.size() - trailingSpaceCount - leadingSpaceCount);
}
```

Untuk memudahkan parsing kartu, dibuatlah fungsi diatas yang akan menghapus leading dan trailing white space yang mungkin akan ada saat pengguna memasukkan input.

```
bool isValidCardInput(string cardInput) {
    vector<string> cardList;
    string word = "";
    for (size_t i = 0; i < cardInput.size(); i++) {
        if (cardInput[i] == ' ') {
            cardList.push_back(word);
            word = "";
        } else {
            word += cardInput[i];
        }
    }
    cardList.push_back(word);
    if (cardList.size() != 4) {
        return false;
    }
    for (int i = 0; i < 4; i++) {
        if (!isElementOf(cardList[i], validCard)) {
            return false;
        }
    }
    return true;
}
```

Untuk mengetahui apakah kartu yang dimasukkan merupakan kartu yang valid, kita lakukan pengecekan satu per satu akan setiap ‘kata’ yang dimasukkan pengguna. Apabila setelah parsing terdapat kurang dari atau lebih dari 4 buah kartu, maka input tidak valid. Apabila terdapat sebuah kartu yang tidak terdapat dalam dek, maka masukkan akan dianggap tidak valid dan pengguna akan diminta untuk memasukkan kembali 4 buah kartu yang valid.

```
void parseCards(string cards, int* cardValues[]) {
    string currentCard = "";
    *cardValues = new int[4];
    int j = 0;
    for (size_t i = 0; i < cards.size(); i++) {
        if (cards[i] == ' ') {
            (*cardValues)[j] = cardValue(currentCard);
            currentCard = "";
            j++;
        } else {
            currentCard += cards[i];
        }
    }
    (*cardValues)[j] = cardValue(currentCard);
}
```

Setelah semua input dipastikan sudah valid, akan dijalankan fungsi berikut yang akan mengubah variabel cardValues menjadi array yang berisikan angka yang dimasukkan pengguna.

```
string cardInput;
cout << "Please input 4 valid cards seperated by white spaces\n-> ";
getline(cin, cardInput);
while (!isValidCardInput(removeTrailingAndLeadingSpace(cardInput))) {
    cout << "Invalid input. Please input 4 valid cards seperated by white spaces\n-> ";
    getline(cin, cardInput);
}
parseCards(removeTrailingAndLeadingSpace(cardInput), &numValues);
```

2. Mendapatkan angka secara acak.

```
numValues = new int[4];
srand(time(0));
cout << "Cards:\n";
for (int i = 0; i < 4; i++) {
    numValues[i] = rand() % 13 + 1;
    cout << cardType(numValues[i]) << " ";
}
cout << endl;
```

Untuk mendapatkan angka secara acak, digunakan fungsi dari standard library c++ srand dan rand. srand merupakan fungsi yang digunakan untuk mendapatkan seed untuk memastikan bahwa angka acak yang didapatkan berbeda setiap waktunya. Lalu untuk mendapatkan angka acaknya

digunakan rand yang dilanjutkan dengan operasi modulo untuk memastikan bahwa angka acak yang didapat dalam kisaran 0-13.

3. Mendapatkan pilihan dari user.
4. Algoritma brute-force

```
string solve(int numArray[4]) {
    auto start = chrono::high_resolution_clock::now();
    int count = 0;
    vector<string> solutions;
    for (size_t l = 0; l < 4; l++) {
        for (size_t m = 0; m < 4; m++) {
            if (m != l) {
                for (size_t n = 0; n < 4; n++) {
                    if (n != m && n != l) {
                        for (size_t o = 0; o < 4; o++) {
                            if (o != m && o != n && o != l) {
                                int permutatedNum[4] = {numArray[l], numArray[m],
                                char tempOp[3];
                                for (int i = 0; i < 4; i++) {
                                    tempOp[0] = operatorList[i];
                                    for (int j = 0; j < 4; j++) {
                                        tempOp[1] = operatorList[j];
                                        for (int k = 0; k < 4; k++) {
                                            tempOp[2] = operatorList[k];
                                            double leftLeft, leftRight, rightLeft
                                            leftLeft = getResult(getResult(getRes
                                            leftRight = getResult(getResult(permu
                                            rightLeft = getResult(permutatedNum[0
                                            rightRight = getResult(permutatedNum[
                                            splitMiddle = getResult(getResult(per
                                            if (leftLeft == 24 && !exist(solution
                                                count++;
                                                solutions.push_back(getLineString
                                            }
                                            if (leftRight == 24 && !exist(solutio
                                                count++;
                                                solutions.push_back(getLineString
```

```
                                                    }
                                                    if (leftRight == 24 && !exist(solut
                                                        count++;
                                                        solutions.push_back(getLineStri
                                                    }
                                                    if (rightLeft == 24 && !exist(solut
                                                        count++;
                                                        solutions.push_back(getLineStri
                                                    }
                                                    if (rightRight == 24 && !exist(solu
                                                        count++;
                                                        solutions.push_back(getLineStri
                                                    }
                                                    if (splitMiddle == 24 && !exist(sol
                                                        count++;
                                                        solutions.push_back(getLineStri
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    auto end = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::microseconds>(end - start);
    string result = getOneContinuousString(solutions);
    result += "Process finished in " + to_string(duration.count()) + " microsec
    return result;
}
```

Terdapat penggunaan library chrono untuk mendapatkan waktu eksekusi program dalam microsecond lalu dimasukkan ke dalam solusi untuk selanjutnya dimasukkan ke dalam file txt dengan mudah.

5. Penulisan ke file txt

```
cout << "Do you want to store the result in a txt file? (Y/N): ";
char store;
cin >> store;
while (store != 'y' && store != 'Y' && store != 'N' && store != 'n') {
    cout << "Sorry, that input was invalid.\n(Y/N): ";
    cin >> store;
}
if (store == 'Y' || store == 'y') {
    cout << "Please enter the designated file name: ";
    string fileName;
    cin >> fileName;
    while (isFileExist(fileName)) {
        cout << "Sorry, that file name is already taken.\nPlease enter another file name: ";
        cin >> fileName;
    }
    string filePath = "../test/" + fileName + ".txt";
    ofstream tempFile(filePath);
    tempFile << result;
    tempFile.close();
}
```

Program dapat menaruh hasil yang didapat ke dalam text file. Apabila user memasukkan ‘Y’ atau ‘y’ maka pengguna akan diminta untuk memasukkan nama file yang valid, yaitu nama file yang belum ada. Apabila nama file sudah ada, maka pengguna akan diminta untuk memasukkan kembali nama file.

```
bool isFileExist(string name) {
    string testFilePath = "../test/" + name + ".txt";
    ifstream f(testFilePath.c_str());
    return f.good();
}
```

Untuk mengetahui apakah sebuah file sudah ada, dibuat fungsi isFileExist yang memanfaatkan library fstream dari c++. Untuk menulis hasil ke file text digunakan kelas ofstream dari library fstream.

C. Penggunaan program dan contoh input/output

Contoh input/output:

1. Kartu masukkan
    a. Dari pengguna
        - Input valid

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 2
Please input 4 valid cards seperated by white spaces
-> A 2 3 4
242 solution(s) found
((1 * 2) * 3) * 4
(1 * (2 * 3)) * 4
1 * ((2 * 3) * 4)
1 * (2 * (3 * 4))
(1 * 2) * (3 * 4)
((1 + 2) + 3) * 4
(1 + (2 + 3)) * 4
((1 * 2) * 4) * 3
(1 * (2 * 4)) * 3
1 * ((2 * 4) * 3)
1 * (2 * (4 * 3))
```

        - Input tidak valid (jumlah kartu tidak sesuai)

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 2
Please input 4 valid cards seperated by white spaces
-> 2 3 3 4 5
Invalid input. Please input 4 valid cards seperated by white spaces
```

- Input tidak valid (kartu tidak sesuai)

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 2
Please input 4 valid cards seperated by white spaces
-> 1 2 3 4
Invalid input. Please input 4 valid cards seperated by white spaces
```

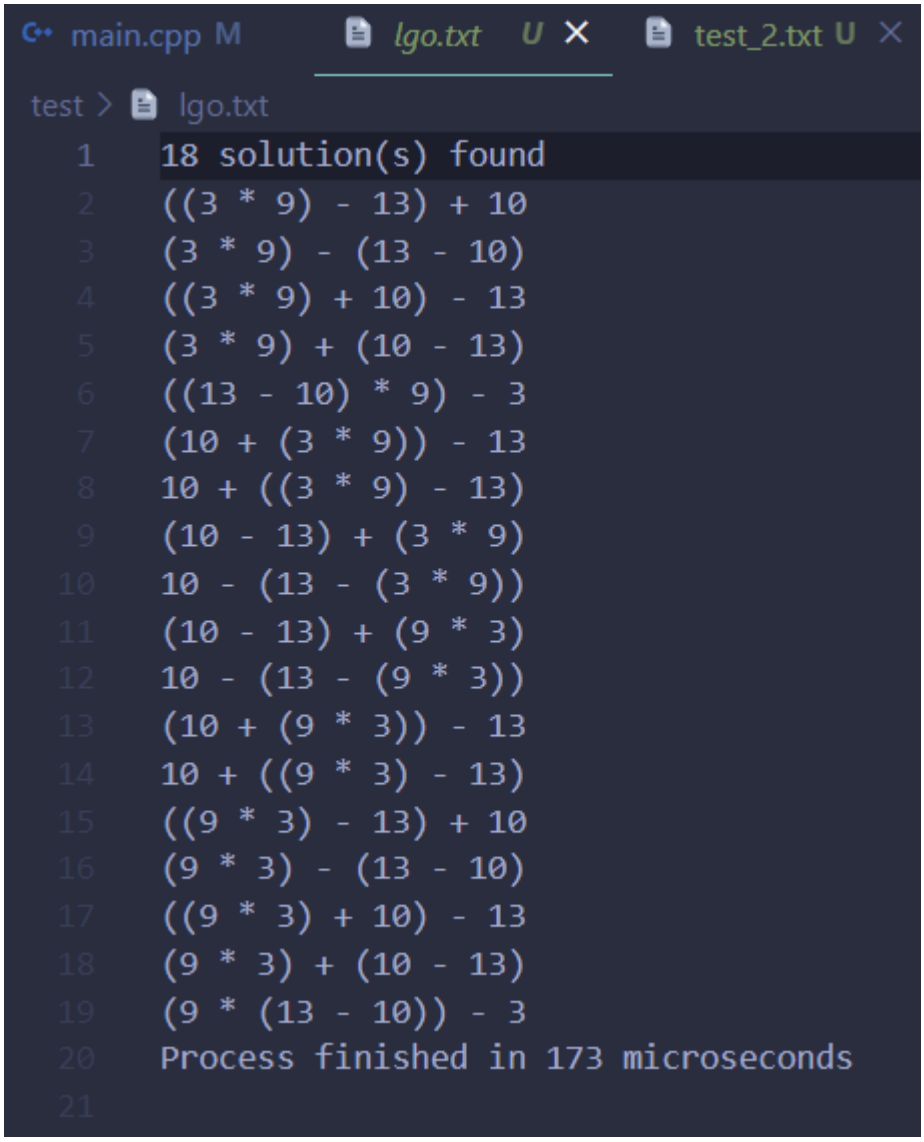
b. Secara acak

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 1
Cards:
5 J 3 6
26 solution(s) found
((5 * 3) - 11) * 6
((11 - 5) * 3) + 6
(11 - 5) + (3 * 6)
11 - (5 - (3 * 6))
(11 - 5) + (6 * 3)
11 - (5 - (6 * 3))
(11 + (3 * 6)) - 5
11 + ((3 * 6) - 5)
(11 + (6 * 3)) - 5
11 + ((6 * 3) - 5)
((3 * 5) - 11) * 6
(3 * (11 - 5)) + 6
((3 * 6) - 5) + 11
(3 * 6) - (5 - 11)
((3 * 6) + 11) - 5
(3 * 6) + (11 - 5)
6 - ((5 - 11) * 3)
6 * ((5 * 3) - 11)
6 + ((11 - 5) * 3)
6 * ((3 * 5) - 11)
((6 * 3) - 5) + 11
(6 * 3) - (5 - 11)
6 - (3 * (5 - 11))
((6 * 3) + 11) - 5
(6 * 3) + (11 - 5)
6 + (3 * (11 - 5))
Process finished in 230 microseconds
Do you want to store the result in a txt file? (Y/N): n
Would you like to use this program again? (Y/N): y
```

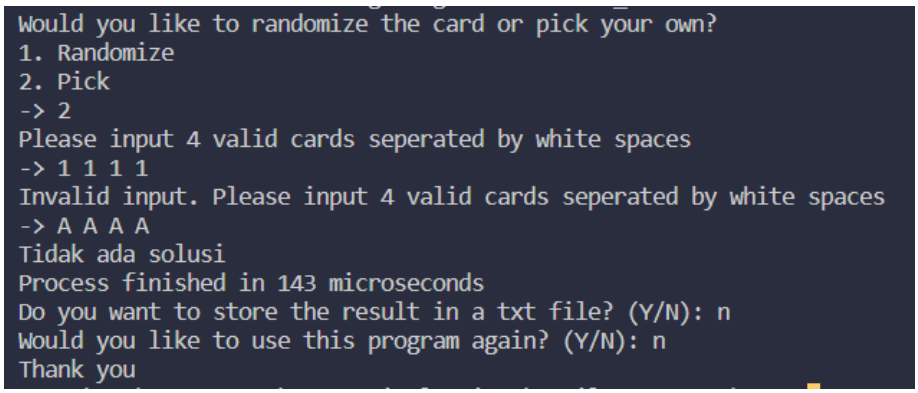
2. File txt

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 1
Cards:
3 K 10 9
18 solution(s) found
((3 * 9) - 13) + 10
(3 * 9) - (13 - 10)
((3 * 9) + 10) - 13
(3 * 9) + (10 - 13)
((13 - 10) * 9) - 3
(10 + (3 * 9)) - 13
10 + ((3 * 9) - 13)
(10 - 13) + (3 * 9)
10 - (13 - (3 * 9))
(10 - 13) + (9 * 3)
10 - (13 - (9 * 3))
(10 + (9 * 3)) - 13
10 + ((9 * 3) - 13)
((9 * 3) - 13) + 10
(9 * 3) - (13 - 10)
((9 * 3) + 10) - 13
(9 * 3) + (10 - 13)
(9 * (13 - 10)) - 3
Process finished in 173 microseconds
Do you want to store the result in a txt file? (Y/N): y
Please enter the designated file name: test_2
Sorry, that file name is already taken.
Please enter another file name: hehe
Sorry, that file name is already taken.
Please enter another file name: lgo
Would you like to use this program again? (Y/N): y
```

File txt yang dihasilkan:



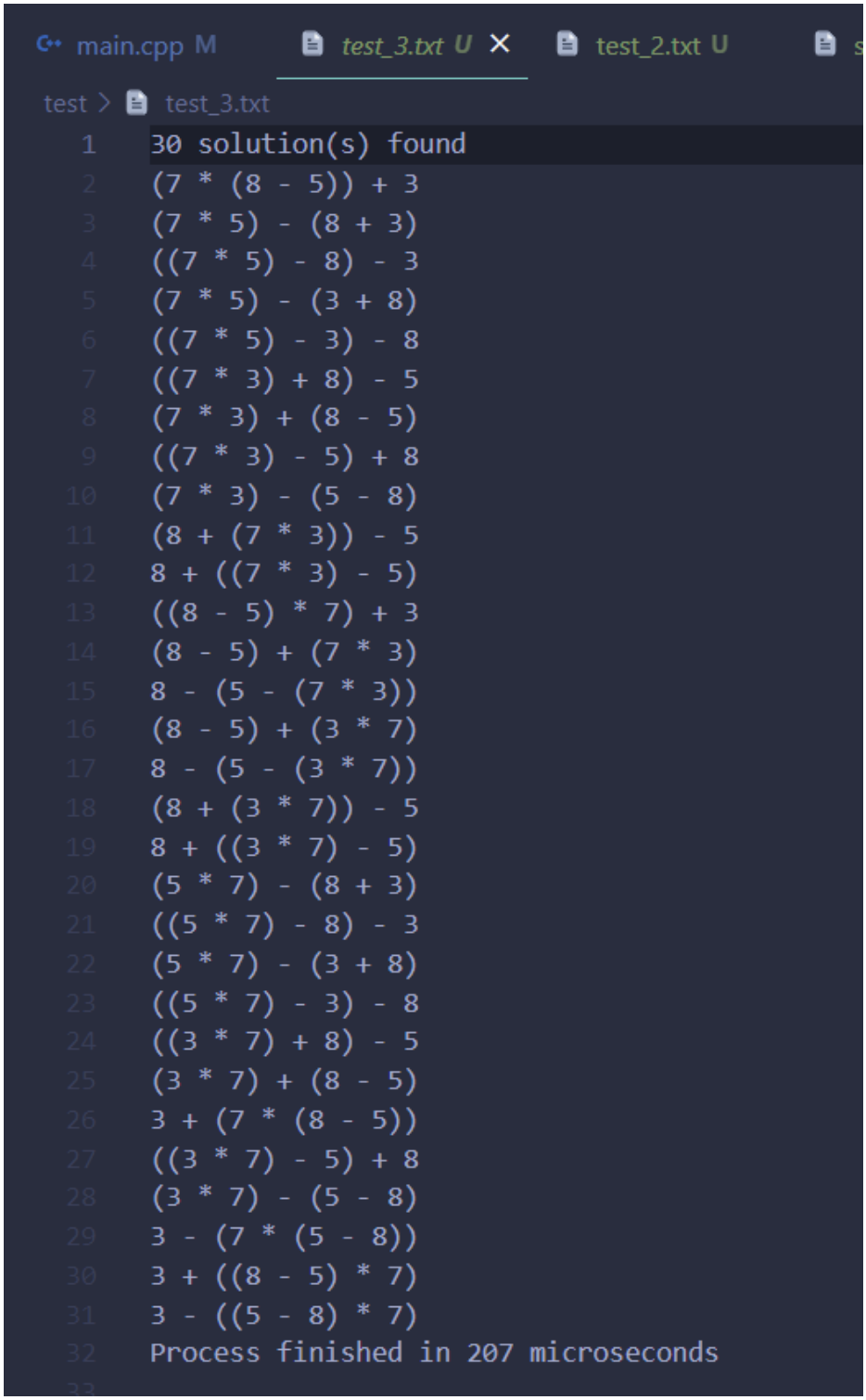
3. Full program flow
    a. Tidak ada solusi



    b. Solusi beragam

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 1
Cards:
7 8 5 3
30 solution(s) found
(7 * (8 - 5)) + 3
(7 * 5) - (8 + 3)
((7 * 5) - 8) - 3
(7 * 5) - (3 + 8)
((7 * 5) - 3) - 8
((7 * 3) + 8) - 5
(7 * 3) + (8 - 5)
((7 * 3) - 5) + 8
(7 * 3) - (5 - 8)
(8 + (7 * 3)) - 5
8 + ((7 * 3) - 5)
((8 - 5) * 7) + 3
(8 - 5) + (7 * 3)
8 - (5 - (7 * 3))
(8 - 5) + (3 * 7)
8 - (5 - (3 * 7))
(8 + (3 * 7)) - 5
8 + ((3 * 7) - 5)
(5 * 7) - (8 + 3)
((5 * 7) - 8) - 3
(5 * 7) - (3 + 8)
((5 * 7) - 3) - 8
((3 * 7) + 8) - 5
(3 * 7) + (8 - 5)
3 + (7 * (8 - 5))
((3 * 7) - 5) + 8
(3 * 7) - (5 - 8)
3 - (7 * (5 - 8))
3 + ((8 - 5) * 7)
3 - ((5 - 8) * 7)
Process finished in 207 microseconds
Do you want to store the result in a txt file? (Y/N): y
Please enter the designated file name: test_3
Would you like to use this program again? (Y/N): y
```

File txt yang dihasilkan:

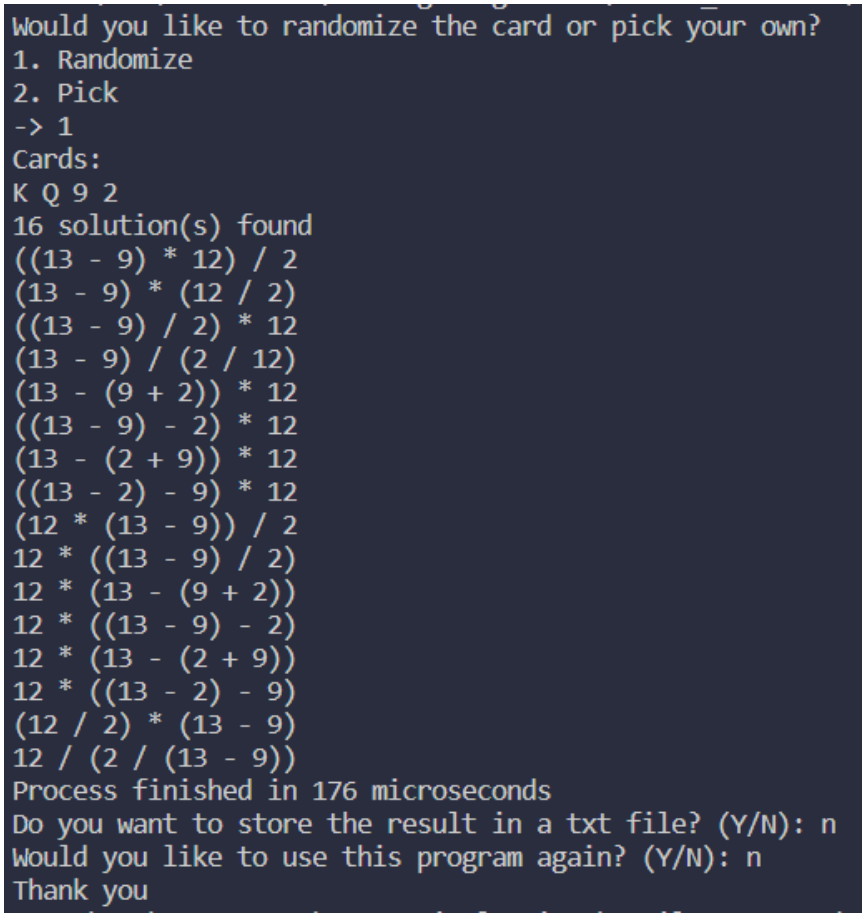


- Semua angka sama

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 2
Please input 4 valid cards seperated by white spaces
-> 6 6 6 6
7 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)
Process finished in 327 microseconds
Do you want to store the result in a txt file? (Y/N): n
Would you like to use this program again? (Y/N): y
```

- Randomize

```
Would you like to randomize the card or pick your own?
1. Randomize
2. Pick
-> 1
Cards:
8 3 2 K
4 solution(s) found
((3 + 13) * 2) - 8
(2 * (3 + 13)) - 8
(2 * (13 + 3)) - 8
((13 + 3) * 2) - 8
Process finished in 132 microseconds
Do you want to store the result in a txt file? (Y/N): n
Would you like to use this program again? (Y/N): n
Thank you
```



D. Link Repository

https://github.com/zakia215/Tucil1_13521146

E. Tabel

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	✓	