

**LAPORAN
TUGAS KECIL 1 IF2211
STRATEGI ALGORITMA**

***Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma
Brute Force***



Nyoman Ganadipa Narayana

13522066

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG**

2023/2024

Daftar Isi

Bagian 1: Latar Belakang.....	3
Bagian 2: Algoritma <i>Brute Force</i>	5
Bagian 3: Implementasi dalam C++	6
Bagian 4: Uji Coba.....	14
A. Cara menjalankan program:	14
B. Uji coba program:	14
Lampiran	24

Bagian 1: Latar Belakang

Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video *Cyberpunk 2077*. Minigame ini merupakan simulasi peretasan jaringan local dari *ICE (Intrusion Countermeasures Electronics)* pada permainan *Cyberpunk 2077*. Komponen pada permainan ini antara lain adalah:

1. Token – terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks – terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.
3. Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
4. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Dengan aturan bermain dari permainan Breach Protocol adalah sebagai berikut:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
5. Setiap sekuens memiliki bobot hadiah atau *reward* yang variatif.
6. Sekuens memiliki panjang minimal berupa dua token

Contoh bermain:

Diberikan matriks sebagai berikut dan ukuran buffernya adalah tujuh.

7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

Dengan sekuens sebagai berikut:

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi optimal yang bisa didapatkan untuk konfigurasi matriks dan sekuens tersebut adalah sebagai berikut



Gambar 1 Salah satu solusi optimal
(Sumber: <https://cyberpunk-hacker.com/>)

Dengan keterangan:

1. Pada awal mula permainan, pemain memilih baris 1 kolom 1,
2. Kemudian pada kolom 1, pemain memilih baris 4,
3. Lalu pada baris 4, pemain memilih kolom 3,
4. Selanjutnya pada kolom 3, pemain memilih baris 5,
5. Dilanjutkan dengan memilih kolom 6 pada baris 5,
6. Kemudian pada kolom 6, pemain memilih baris 4,
7. Terakhir, pemain memilih kolom 5 pada baris 4.

Sehingga, total bobot optimal adalah 50 dengan jumlah langkah sebanyak 6 (diambil dari dokumen Tugas Kecil 1 IF2211 Strategi Algoritma 2023/2024: Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma *Brute Force*). Saya, sebagai mahasiswa yang mengikuti perkuliahan Strategi Algoritma diminta untuk menemukan solusi paling optimal dari permainan ini untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma *brute force*.

Bagian 2: Algoritma *Brute Force*

Sebelum dilakukan *brute force*, telah disiapkan informasi-informasi berupa matriks, ukuran buffer, sekuens dan bobotnya. Setelah itu, dilakukan bruteforce dengan langkah sebagai berikut:

1. Awal mulanya, perlu diketahui algoritma ini akan melangkah sejauh ukuran buffer yang telah diberikan. Jadi, langkah terakhir akan memiliki langkah sejumlah ukuran buffer. Selain itu, dibuat variabel urutan langkah dengan memanfaatkan ADT stack, variabel bobot maksimum yang dapat capai, dan variabel urutan langkah untuk mencapai bobot maksimum tersebut.
2. Pada langkah pertama, algoritma ini secara bergilir mengambil satu persatu kolom pada baris pertama. Saat suatu kolom dipilih, sel ini di-push ke dalam stack urutan langkah dan kemudian algoritma ini berlanjut ke langkah kedua. Setelah langkah kedua berhasil sepenuhnya dijalankan, urutan langkah di-pop kemudian lanjut bergilir mengambil kolom berikutnya.
3. Pada langkah genap – langkah ke-2, ke-4, dan seterusnya hingga langkah terakhir, algoritma ini secara bergilir mengambil satu persatu baris pada kolom yang telah diambil pada langkah tepat satu sebelumnya, dengan syarat sel tersebut belum ada pada stack urutan langkah. Saat suatu baris berhasil dipilih, sel ini di-push ke dalam stack urutan langkah, urutan langkah ini kemudian dievaluasi total bobotnya, kemudian variabel-variabel yang kita miliki diperbarui secara bersyarat*. Jika ini adalah langkah terakhir, maka tidak dilanjutkan ke langkah berikutnya. Jika tidak, dilanjutkan ke langkah berikutnya. Dalam kedua kasus, urutan langkah akan selalu di-pop pada akhirnya.
4. Pada langkah ganjil – langkah ke-3, ke-5, dan seterusnya hingga langkah terakhir, algoritma ini secara bergilir mengambil satu persatu kolom pada baris yang telah dipilih pada langkah tepat satu sebelumnya, dengan syarat sel tersebut belum ada pada stack urutan langkah. Saat suatu kolom berhasil dipilih, sel ini di-push ke dalam stack urutan langkah, urutan langkah ini kemudian dievaluasi total bobotnya, kemudian variabel-variabel yang kita miliki diperbarui secara bersyarat*. Jika ini adalah langkah terakhir, maka tidak dilanjutkan ke langkah berikutnya. Jika tidak, dilanjutkan ke langkah berikutnya. Dalam kedua kasus, urutan langkah akan selalu di-pop pada akhirnya.

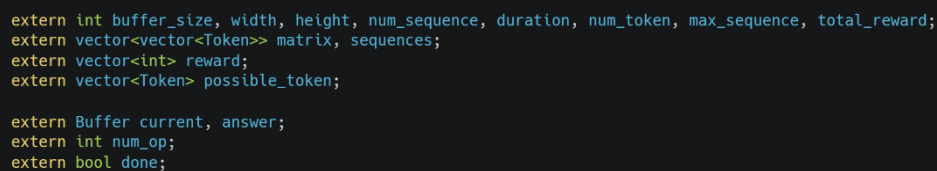
*Variabel diperbarui secara bersyarat dalam algoritma brute force ini dilakukan untuk memperbarui variabel bobot maksimum yang dapat dicapai dan urutan langkah yang mencapai bobot maksimum tersebut. Jika setelah stack urutan langkah setelah dievaluasi memiliki total bobot yang lebih besar daripada variabel bobot maksimum, maka variabel bobot maksimum dan urutan langkah untuk mencapai bobot maksimum juga diperbarui. Selain itu, jika bobotnya sama dan jumlah langkahnya lebih sedikit, maka variabel bobot maksimum dan urutan langkah untuk mencapai bobot maksimum juga diperbarui.

Bagian 3: Implementasi dalam C++

Algoritma pada Bab 2 diimplementasikan menggunakan Bahasa pemrograman C++ dengan mempertimbangkan kecepatan program serta kecepatan *development*. Program yang telah dibuat diimplementasikan secara modular ke dalam 4 *file*: main.cpp, utils.cpp, program.cpp, dan headers.hpp.

1. headers.hpp

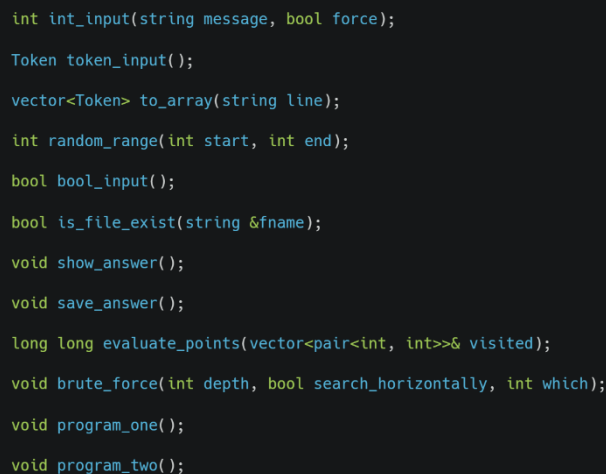
Berisi definisi-definisi fungsi yang digunakan di dalam program serta variabel global



```
extern int buffer_size, width, height, num_sequence, duration, num_token, max_sequence, total_reward;
extern vector<vector<Token>>> matrix, sequences;
extern vector<int> reward;
extern vector<Token> possible_token;

extern Buffer current, answer;
extern int num_op;
extern bool done;
```

Gambar 2 Variabel global



```
int int_input(string message, bool force);
Token token_input();
vector<Token> to_array(string line);
int random_range(int start, int end);
bool bool_input();
bool is_file_exist(string &fname);
void show_answer();
void save_answer();
long long evaluate_points(vector<pair<int, int>>& visited);
void brute_force(int depth, bool search_horizontally, int which);
void program_one();
void program_two();
```

Gambar 3 Definisi fungsi dan prosedur pada program.

2. main.cpp

main.cpp adalah file yang memuat kode utama untuk program dan memanggil fungsi/prosedur pembantu lainnya.

```

int main() {
    opening();
    cout << "Tipe input:\n 1. Input spesifikasi 1 (file)\n 2. Input spesifikasi 2 (CLI)\n";

    int type = int_input("Masukkan angka: ", false);

    try {
        switch (type) {
            case 1:
                program_one();
                break;
            case 2:
                program_two();
                cout << "\n\nMatriks yang digunakan: \n";
                for (auto &row: matrix) {
                    for (auto &col: row) {
                        cout << col << ' ';
                    } cout << '\n';
                }

                cout << "\n\nReward sekuens dan sekuens yang digunakan: \n";
                for (int i = 0; i < num_sequence; i++) {
                    cout << reward[i] << ": ";
                    for (auto &t: sequences[i]) cout << t << ' ';
                    cout << '\n';
                }

                break;
            default:
                cout << "Angka yang anda masukkan di luar yang diinginkan. Program
berhenti.\n";
        }
    } catch (const runtime_error& e) {
        cout << "\n\nError: " << e.what() << "\nProgram berhenti.\n";
        return 0;
    }

    cout << "\n\n";
    show_answer();
    cout << duration << " ms\n";
    cout << "\n\n";

    cout << "\nApakah ingin menyimpan solusi? [Y/n] ";
    bool save = bool_input();
    if (save) save_answer();

    return 0;
}

```

Gambar 4 main.cpp

3. utils.cpp

utils.cpp memuat kode program yang berkaitan dengan validator input, random number generator, dan fungsi/ prosedur lainnya yang bukan berkaitan erat dengan algoritma bruteforce program.

a. Masukan dengan *validator*

```
int int_input(string message, bool force) {
    int user_input;

    while (true) {
        try {
            cout << message;
            cin >> user_input;

            if (cin.fail()) {
                if (!force) throw runtime_error("Terdeteksi sebuah non-angka, input diharapkan angka.");
                else throw runtime_error("Input bukan berupa angka. Masukkanlah angka.");
            }

        } catch (const runtime_error& e) {
            cerr << "Message: " << e.what() << endl;
            if (force) continue;
        }

        break;
    }

    return user_input;
}

bool bool_input() {
    string ans;
    cin >> ans;

    if (ans == "Y" || ans == "y") return true;
    else if (ans == "n" || ans == "N") return false;
    else throw runtime_error("Input di luar jawaban yang disediakan.");
}
```

Gambar 5 Integer dan Boolean input

```
Token token_validator(string token) {
    if (token.size() != 2) throw runtime_error("Terdeteksi sebuah non-token pada masukkan yang seharusnya token.");

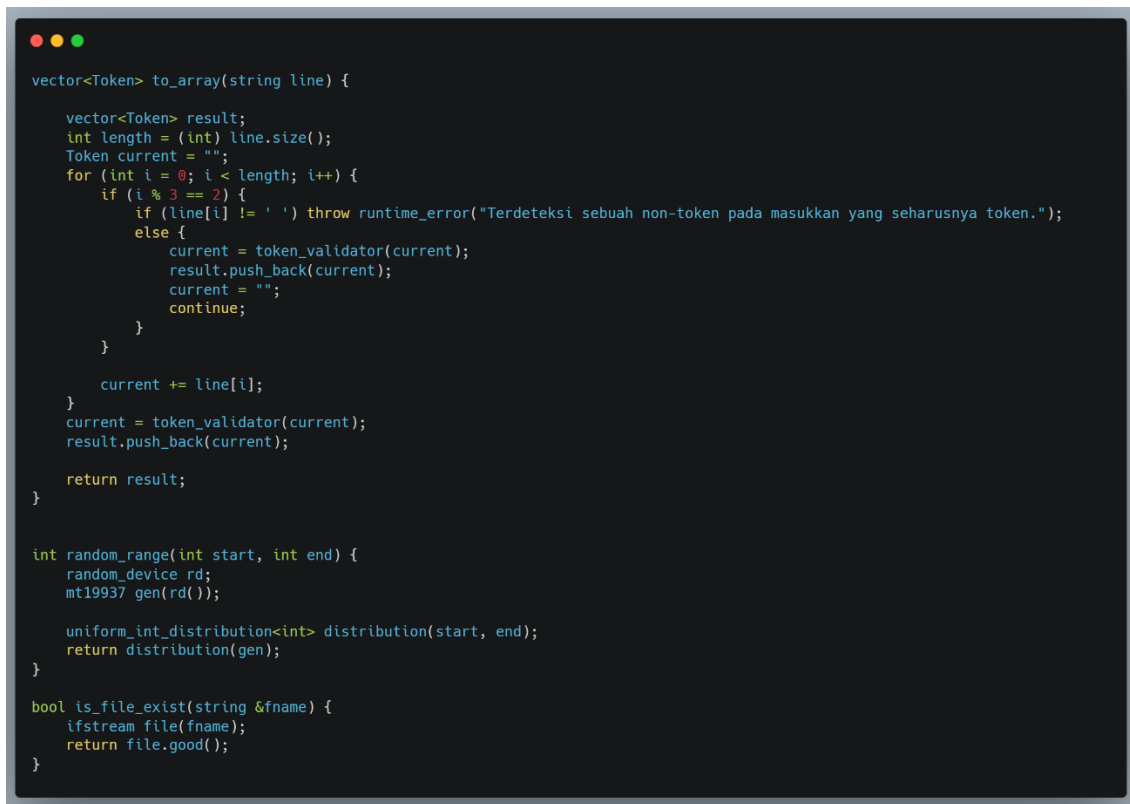
    for (int i = 0; i < 2; i++) {
        if (token[i] - 'A' <= 25 && token[i] - 'A' >= 0) continue;
        else if (token[i] - '0' <= 9 && token[i] - '0' >= 0) continue;
        else throw runtime_error("Terdeteksi sebuah non-token pada masukkan yang seharusnya token.");
    }

    return token;
}

Token token_input() {
    string token;
    cin >> token;
    token = token_validator(token);
    return token;
}
```

Gambar 6 Token input

b. Lainnya

A screenshot of a C++ code editor with a dark background and light-colored text. The code defines three utility functions: `to_array`, `random_range`, and `is_file_exist`. The `to_array` function takes a string and returns a vector of tokens, validating each character. The `random_range` function uses a random device and a uniform distribution to generate a random integer. The `is_file_exist` function checks if a file exists and is readable.

```
vector<Token> to_array(string line) {  
    vector<Token> result;  
    int length = (int) line.size();  
    Token current = "";  
    for (int i = 0; i < length; i++) {  
        if (i % 3 == 2) {  
            if (line[i] != ' ') throw runtime_error("Terdeteksi sebuah non-token pada masukkan yang seharusnya token.");  
            else {  
                current = token_validator(current);  
                result.push_back(current);  
                current = "";  
                continue;  
            }  
        }  
        current += line[i];  
    }  
    current = token_validator(current);  
    result.push_back(current);  
    return result;  
}  
  
int random_range(int start, int end) {  
    random_device rd;  
    mt19937 gen(rd());  
    uniform_int_distribution<int> distribution(start, end);  
    return distribution(gen);  
}  
  
bool is_file_exist(string &fname) {  
    ifstream file(fname);  
    return file.good();  
}
```

Gambar 7 Utilitas lainnya

4. program.cpp

Berisi algoritma bruteforce dan fungsi/prosedur pendukungnya, seperti input dari text ataupun dari CLI untuk menyiapkan informasi-informasi yang dibutuhkan.

a. Persiapan sebelum melakukan bruteforce

Prosedur `program_one` adalah prosedur yang menjalankan program yang meminta input berupa file (.txt).

```

void program_one() {
    // Inisialisasi, membaca input dari file
    cout << "Masukkan nama file input yang berada di folder test: ";
    string fname; cin >> fname;
    fname = "test/" + fname;

    // Pastikan ada filenya
    ifstream file(fname);
    if(!file.is_open()) throw runtime_error("File tidak ditemukan, contoh pemakaian: input.txt");

    // Input dari file di mask jadi cin.
    auto origin = cin.rdbuf();
    cin.rdbuf(file.rdbuf());

    // Insert input to global variable
    buffer_size = int_input("", false);
    width = int_input("", false);
    height = int_input("", false);

    matrix.resize(height);
    for (int row = 0; row < height; row++) {
        matrix[row].resize(width);
        for (int col = 0; col < width; col++) {
            matrix[row][col] = token_input();
        }
    }

    num_sequence = int_input("", false);
    // Insert sequence and its reward
    sequences.resize(num_sequence);
    reward.resize(num_sequence);

    string line;
    cin.ignore(1000, '\n');
    for (int i = 0; i < num_sequence; i++) {
        getline(cin, line);
        line = trim(line, " \t");

        sequences[i] = (to_array(line));
        reward[i] = int_input("", false);
    }
    cin.ignore(1000, '\n');

    // bruteforcing and measure the duration
    auto start = chrono::high_resolution_clock::now();

    brute_force(buffer_size, true, 0);
    auto end = chrono::high_resolution_clock::now();
    duration = static_cast<int>(chrono::duration_cast<chrono::milliseconds>(end - start).count());

    // Go back to use CLI as its input.
    cin.rdbuf(origin);
}

```

Gambar 8 Program yang menerima input file

Sementara prosedur program_two adalah prosedur yang menjalankan program yang meminta input dari CLI, namun untuk menyederhanakan input, program harus mengenerate matriks dan sekuensnya terlebih dahulu.

```

void program_two() {
    cout << "\nJumlah token unik: ";
    num_token = int_input("", false);
    cin.ignore(1, '\n');

    cout << "\nToken-token unik: ";
    string line;
    getline(cin, line);
    possible_token = to_array(line);

    cout << "\nUkuran buffer: ";
    buffer_size = int_input("", false);

    cout << "\nTinggi matriks (banyaknya baris): ";
    height = int_input("", false);

    cout << "\nLebar matriks (banyaknya kolom): ";
    width = int_input("", false);

    cout << "\nJumlah sekuens: ";
    num_sequence = int_input("", false);

    cout << "Panjang maksimum sekuens: ";
    max_sequence = int_input("", false);

    // Generate random matrix, sequence, and also random reward for each sequence;
    bool generate_another = false;
    while (!generate_another) {
        // 1. Generate random matrix
        matrix.clear();
        matrix.resize(height);
        for (int row = 0; row < height; row++) {
            matrix[row].resize(width);
            for (int col = 0; col < width; col++) {
                int random_token_index = random_range(0, num_token - 1);
                matrix[row][col] = possible_token[random_token_index];
            }
        }

        // 2. Generate random sequences
        sequences.clear();
        sequences.resize(num_sequence);
        for (int i = 0; i < num_sequence; i++) {
            int length = random_range(2, max_sequence);
            for (int j = 0; j < length; j++) {
                int random_token_index = random_range(0, num_token - 1);
                Token token = possible_token[random_token_index];
                sequences[i].push_back(token);
            }
        }

        // 3. Generate random reward
        reward.clear();
        reward.resize(num_sequence);
        for (int i = 0; i < num_sequence; i++) {
            reward[i] = random_range(15, 40);
        }

        // Output generated.
        cout << "\nMatriks: \n";
        for (int row = 0; row < height; row++) {
            for (int col = 0; col < width; col++) {
                cout << matrix[row][col] << ' ';
            }
            cout << endl;
        }

        cout << "\nPoint dan sequence-nya: \n";
        for (int i = 0; i < num_sequence; i++) {
            cout << reward[i] << ": ";

            for (auto &token: sequences[i]) {
                cout << token << ' ';
            }
            cout << '\n';
        }

        cout << "\nApakah ingin menggunakan konfigurasi matriks dan \nsequence yang telah digenerate di atas?\n";
        cout << "[Y/n] ";
        generate_another = bool_input();
    }

    auto start = chrono::high_resolution_clock::now();
    brute_force(buffer_size, true, 0);
    auto end = chrono::high_resolution_clock::now();
    duration = static_cast<int>(chrono::duration_cast<chrono::milliseconds>(end - start).count());
}

```

Gambar 9 Program yang menerima input dari CLI

b. Algoritma brute force

```
long long evaluate_points(vector<pair<int, int>>& visited) {
    long long result = 0;

    int length = (int) visited.size();
    vector<int> pointer(num_sequence);
    for (int i = 0; i < length; i++) {
        Token current = matrix[visited[i].second][visited[i].first];

        for (int j = 0; j < num_sequence; j++) {

            if (pointer[j] != -1) {
                if (current == sequences[j][pointer[j]]) pointer[j]++;
                else pointer[j] = 0;
            }

            if (pointer[j] == (int) sequences[j].size()) {
                result += reward[j];
                pointer[j] = -1;
            }
        }
    }

    return result;
}

void brute_force(int depth, bool search_horizontally, int which) {
    ++num_op;
    if (depth == 0) return;

    if (search_horizontally) {
        for (int col = 0; col < width; col++) {

            // check whether it is possible to visit here.
            bool possible = true;
            int length = (int) current.visited.size(), i = 0;
            while (possible && i < length) {
                if (current.visited[i].first == col && current.visited[i].second == which) possible = false;
                i++;
            }

            if (!possible) continue;

            // If it is possible, try and evaluate the points gained.
            current.visited.push_back({col, which});

            // If points is greater than the computed answer, update.
            current.points = evaluate_points(current.visited);

            if (current.points >= answer.points) {
                if (current.visited.size() < answer.visited.size() || current.points > answer.points) {
                    answer = current;
                };
            }

            // Try to go more deep with the current configuration.
            brute_force(depth - 1, !search_horizontally, col);

            // Try another configuration.
            current.visited.pop_back();
        }
    } else {
        // This is just a mirror case of the above procedure.

        for (int row = 0; row < height; row++) {

            // check whether it is possible to visit here.
            bool possible = true;
            int length = (int) current.visited.size(), i = 0;
            while (possible && i < length) {
                if (current.visited[i].first == which && current.visited[i].second == row) possible = false;
                i++;
            }

            if (!possible) continue;

            // We can go here, so try and evaluate the points gained.
            current.visited.push_back({which, row});

            current.points = evaluate_points(current.visited);

            if (current.points >= answer.points) {
                if (current.visited.size() < answer.visited.size() || current.points > answer.points) {
                    answer = current;
                };
            }

            brute_force(depth - 1, !search_horizontally, row);
            current.visited.pop_back();
        }
    }
}
```

Gambar 10 Algoritma *brute force*

c. Output jawaban dan penyimpanan jawaban

```
void show_answer() {
    // output
    cout << answer.points << " points.\n";

    if (answer.points > 0) {
        for (auto &p: answer.visited) {
            cout << matrix[p.second][p.first] << ' ';
        }
        cout << '\n';

        for (auto &p: answer.visited) {
            cout << p.first + 1 << ", " << p.second + 1 << '\n';
        }
    } else cout << "Solusi memiliki panjang 0.\n";
}

void save_answer() {
    cout << "Nama file: ";
    string name; cin >> name;

    bool rewrite = false;
    string path = "test/" + name;
    while (is_file_exist(path) && !rewrite) {
        cout << "\nWarning: Sudah ada file bernama " << name << ". Overwrite saja? Ketik nama baru jika tidak.\n";
        cout << "[Y/<nama-file>] ";
        string ans; cin >> ans;

        if (ans == "Y" || ans == "y") rewrite = true;
        else {
            name = ans;
            path = "test/" + name;
        }
    }

    ofstream output(path);
    if (output.is_open()) {
        auto origin = cout.rdbuf();
        cout.rdbuf(output.rdbuf());

        show_answer();
        cout.rdbuf(origin);
    } else {
        throw runtime_error("Terjadi suatu kesalahan saat file ingin disimpan.");
    }

    cout << "\n\nBerhasil menyimpan jawaban pada file " << name << " yang terletak di folder test.\n";
}
```

Gambar 11 Output dan menyimpan solusi optimal

Bagian 4: Uji Coba

A. Cara menjalankan program:

1. Compile dan run aplikasi,
2. Program ini memiliki 2 opsi:
 - a. Pengguna memasukkan masukan menggunakan file,
 - b. Pengguna memasukkan masukan melalui *Command Line Interface*,
3. Untuk masukan menggunakan file, pengguna harus mengikuti format:

```
buffer_size
matrix_width matrix_height
matrix
number_of_sequences
sequences_1
sequences_1_reward
sequences_2
sequences_2_reward
...
sequences_n
sequences_n_reward
```

Sementara untuk masukan melalui *CLI*, pengguna akan diarahi untuk memasuki input berdasarkan program.

4. Saat berhasil memasukkan masukan ke dalam program, program akan memberikan solusi optimal yang kemudian dapat disimpan pada file (.txt).

B. Uji coba program:

```
dP""b8 Yb dP 88""Yb 888888 88""Yb 88""Yb 88 88 88b 88 88 dP oP""Yb. dP""Yb 888888P 888888P
dP ``" YbdP 88__dP 88__ 88__dP 88__dP 88 88 88Yb88 88odP "' dP' dP Yb dP dP
Yb 8P 88""Yb 88"" 88""Yb 88"" Y8 8P 88 Y88 88""Yb dP' Yb dP dP dP
YboodP dP 88oodP 888888 88 Yb 88 `YbodP' 88 Y8 88 Yb .d8888 YbodP dP dP

Tipe input:
1. Input spesifikasi 1 (file)
2. Input spesifikasi 2 (CLI)
Masukkan angka: |
```

Gambar 12 Pembukaan program

1. Opsi 1
 - a. Isi file 2.txt

```
1 7
2 6 6
3 FF FF FF FF FF 55
4 FF FF FF FF FF FF
5 FF FF FF 55 FF BD
6 FF FF 7A FF FF BD
7 FF FF BD FF FF 1C
8 FF FF FF FF FF FF
9 2
10 BD 7A BD
11 5
12 BD 1C BD
13 10
14
```

Gambar 13 Input 1

Output program

```
15 points.
55 BD 7A BD 1C BD
6, 1
6, 4
3, 4
3, 5
6, 5
6, 3
```

15 ms

Apakah ingin menyimpan solusi? [Y/n] |

Gambar 14 Output 1

b. Isi file 3.txt

```
1 7
2 6 6
3 55 55 55 55 55 55
4 55 55 55 55 55 55
5 55 55 55 55 55 55
6 55 55 55 55 55 55
7 55 55 55 55 55 55
8 55 55 55 55 55 55
9 3
10 55 55 55 55
11 -100
12 55 55 55
13 15
14 55 55 55 55 55
15 30
```

Gambar 15 Input 2

Output program

```
15 points.
55 55 55
1, 1
1, 2
2, 2

44 ms

Apakah ingin menyimpan solusi? [Y/n] |
```

Gambar 16 Output 2

c. Isi file 4.txt

```
st 7 5.txt
1 7
2 7 7
3 PG D4 PG 7C 7C 7C 7C
4 PG PG D4 7C 7C 1A 7C
5 PG D4 1A 1A 1A D4 D4
6 7C 7C D4 7C D4 7C 7C
7 D4 7C PG PG 7C 1A 1A
8 D4 1A PG 1A D4 D4 1A
9 D4 7C 7C 1A 7C PG 7C
10 6
11 D4 7C 1A
12 211
13 PG D4 7C
14 125
15 7C PG D4 PG
16 220
17 1A PG
18 272
19 1A 7C 1A
20 282
21 PG 1A 7C
22 253
```

Gambar 17 Input 3

Output program

```
1018 points.
D4 7C 1A PG 1A 7C 1A
2, 1
2, 5
6, 5
6, 7
4, 7
4, 2
6, 2

90 ms

Apakah ingin menyimpan solusi? [Y/n] |
```

Gambar 18 Output 3

d. Isi file 5.txt

```
11
2 20
BD E9
1C 1C
BD 55
55 55
55 7A
7A 1C
7A 55
7A 55
E9 BD
E9 55
7A BD
1C 1C
55 55
55 1C
BD 55
1C 1C
BD E9
7A 1C
E9 BD
55 BD
1
BD 55 BD 1C 1C E9 BD 55 BD 1C 1C
10
```

Gambar 19 input 4

Output program

```
10 points.
BD 55 BD 1C 1C E9 BD 55 BD 1C 1C
1, 1
1, 20
2, 20
2, 2
1, 2
1, 9
2, 9
2, 3
1, 3
1, 12
2, 12

1287 ms

Apakah ingin menyimpan solusi? [Y/n] |
```

Gambar 20 Output 4

e. Isi file 6.txt

```
19
2 10
BD 1C
BD 7A
BD 55
BD E9
1C 7A
1C 55
1C E9
7A 55
7A E9
55 E9
1
1C E9 55 BD 7A E9 7A BD 55 55 7A BD E9 E9 1C 1C 7A 55 1C
10
```

Gambar 21 Input 5

Output program

```
10 points.
1C E9 55 BD 7A E9 7A BD 55 55 7A BD E9 E9 1C 1C 7A 55 1C
2, 1
2, 10
1, 10
1, 2
2, 2
2, 9
1, 9
1, 3
2, 3
2, 8
1, 8
1, 4
2, 4
2, 7
1, 7
1, 5
2, 5
2, 6
1, 6

1455 ms

Apakah ingin menyimpan solusi? [Y/n] |
```

Gambar 22 Output 6

f. Isi file 7.txt

```
8
5 10
FF FF AA BB DD
BB FF EE EE DD
AA DD DD BB AA
EE FF CC FF EE
AA DD CC BB DD
EE AA CC CC DD
FF FF FF FF FF
EE CC EE FF EE
BB EE AA DD AA
AA DD DD DD EE
3
EE CC AA
48
EE CC
-20
AA BB FF AA BB
10
```

Gambar 23 Input 6

Output program

```
28 points.
FF EE CC AA
1, 1
1, 4
3, 4
3, 1

413 ms

Apakah ingin menyimpan solusi? [Y/n] |
```

Gambar 24 Output 6

2. Opsi 2:

```
Tipe input:
  1. Input spesifikasi 1 (file)
  2. Input spesifikasi 2 (CLI)
Masukkan angka: 2

Jumlah token unik: 5

Token-token unik: BD 1C 7A 55 E9

Ukuran buffer: 7

Tinggi matriks (banyaknya baris): 5

Lebar matriks (banyaknya kolom): 5

Jumlah sekuens: 3
Panjang maksimum sekuens: 4

Matriks:
55 BD 55 1C 1C
7A E9 E9 1C 7A
E9 55 55 1C 7A
55 7A E9 E9 1C
55 BD BD 55 BD

Point dan sequence-nya:
15: 7A 1C BD 55
25: 55 BD 55 55
16: 7A BD 55

Apakah ingin menggunakan konfigurasi matriks dan
sequence yang telah digenerate di atas?
[Y/n] |
```

Gambar 25 Input 1 CLI

```
Apakah ingin menggunakan konfigurasi matriks dan
sequence yang telah digenerate di atas?
[Y/n] n

Matriks:
1C 1C BD 1C 1C
E9 E9 E9 7A 1C
7A E9 BD 7A 7A
E9 1C E9 BD BD
7A 1C BD BD E9

Point dan sequence-nya:
33: 7A 7A BD BD
21: 1C 1C 55 1C
32: BD 55 55

Apakah ingin menggunakan konfigurasi matriks dan
sequence yang telah digenerate di atas?
[Y/n] |
```

Gambar 26 Kasus ingin men-generate yang lain

```

Apakah ingin menggunakan konfigurasi matriks dan
sequence yang telah digenerate di atas?
[Y/n] Y

Matriks yang digunakan:
1C 1C BD 1C 1C
E9 E9 E9 7A 1C
7A E9 BD 7A 7A
E9 1C E9 BD BD
7A 1C BD BD E9

Reward sekuens dan sekuens yang digunakan:
33: 7A 7A BD BD
21: 1C 1C 55 1C
32: BD 55 55

33 points.
1C 7A 7A BD BD
1, 1
1, 3
4, 3
4, 4
5, 4

3 ms

Apakah ingin menyimpan solusi? [Y/n] |

```

Gambar 27 Output 1 CLI

3. Simpan solusi

```

Apakah ingin menyimpan solusi? [Y/n] Y
Nama file: 1.txt

Warning: Sudah ada file bernama 1.txt. Overwrite saja? Ketik nama baru jika tidak.
[Y/<nama-file>] █

```

Gambar 28 Kasus sudah ada file dengan nama yang sama

```

Warning: Sudah ada file bernama 1.txt. Overwrite saja? Ketik nama baru jika tidak.
[Y/<nama-file>] Y

Berhasil menyimpan jawaban pada file 1.txt yang terletak di folder test/output.

```

Gambar 29 Overwrite

```

Apakah ingin menyimpan solusi? [Y/n] Y
Nama file: 1.txt

Berhasil menyimpan jawaban pada file 1.txt yang terletak di folder test/output.

```

Gambar 30 Kasus tidak ada nama yang sama

```
Apakah ingin menyimpan solusi? [Y/n] Y
Nama file: 1.txt

Warning: Sudah ada file bernama 1.txt. Overwrite saja? Ketik nama baru jika tidak.
[Y/<nama-file>] 2.txt

Berhasil menyimpan jawaban pada file 2.txt yang terletak di folder test/output.
```

Gambar 31 Tidak Overwrite

Lampiran

Github Repository:

https://github.com/ganadipa/Tucil1_13522066

Checklist:

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	V	
2. Program berhasil dijalankan	V	
3. Program dapat membaca masukan berkas .txt	V	
4. Program dapat menghasilkan masukan secara acak	V	
5. Solusi yang diberikan program optimal	V	
6. Program dapat menyimpan solusi dalam berkas .txt	V	
7. Program memiliki GUI		V