LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh:

Ahmad Thoriq Saputra (13522141)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

BANDUNG
2024

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Breach Protocol dalam permainan Cyberpunk 2077 adalah sebuah minigame yang menggambarkan proses meretas jaringan lokal dari ICE (Intrusion Countermeasures Electronics). Permainan ini terdiri dari beberapa komponen utama yang harus dimengerti pemain, yakni token, matriks, sekuens, dan buffer. Token merupakan representasi karakter alfanumerik seperti E9, BD, dan 55 yang digunakan untuk membangun urutan kode dalam meretas sistem. Matriks adalah kumpulan token yang tersedia untuk dipilih oleh pemain guna menyusun kombinasi kode. Sekuens adalah rangkaian token yang harus dicocokkan oleh pemain sesuai dengan instruksi yang diberikan, dengan setiap sekuens memiliki bobot hadiah yang berbeda-beda.

Aturan permainan Breach Protocol menetapkan langkah-langkah yang harus diikuti oleh pemain. Pemain diminta untuk bergerak secara bergantian, antara horizontal dan vertikal, hingga semua sekuens berhasil dicocokkan atau buffer mencapai kapasitas maksimum. Permainan dimulai dengan memilih satu token dari posisi baris paling atas dalam matriks. Satu token yang dipilih dapat digunakan dalam lebih dari satu sekuens. Namun, sekuens memiliki panjang minimal berupa dua token. Buffer, sementara itu, adalah batasan jumlah maksimum token yang dapat disusun secara sekuensial oleh pemain.

Tantangan utama dalam permainan ini adalah menemukan solusi yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer yang diberikan. Untuk mengatasi tantangan tersebut, pemain menggunakan algoritma brute force. Pendekatan ini melibatkan mencoba semua kemungkinan kombinasi token dalam matriks untuk mencocokkan sekuens yang diberikan atau mengisi buffer sesuai dengan ukuran yang ditentukan. Meskipun algoritma brute force membutuhkan waktu yang lebih lama untuk mengevaluasi semua kemungkinan, pendekatan ini dianggap efektif untuk menemukan solusi optimal dalam situasi yang kompleks. Dengan demikian, penggunaan algoritma brute force merupakan strategi yang efektif dalam menyelesaikan permainan Breach Protocol dalam Cyberpunk 2077.

BAB II
TEORI SINGKAT

Algoritma brute force, meskipun sederhana, adalah pendekatan yang kuat dalam menyelesaikan masalah dengan mencoba setiap kemungkinan solusi secara sistematis. Metode ini sangat bermanfaat dalam berbagai bidang, termasuk pemrograman, matematika, dan ilmu komputer. Salah satu keunggulan utamanya adalah kemampuannya untuk menemukan solusi optimal dari segi keakuratan. Dengan mempertimbangkan semua kemungkinan solusi, algoritma ini dapat memastikan bahwa solusi yang ditemukan adalah yang terbaik.

Namun, kelemahan utamanya adalah waktu eksekusi yang meningkat secara eksponensial seiring dengan bertambahnya ukuran masalah. Hal ini disebabkan oleh fakta bahwa algoritma brute force harus memeriksa setiap kemungkinan solusi secara berurutan. Oleh karena itu, untuk masalah yang sangat besar atau kompleks, metode ini mungkin tidak efisien.

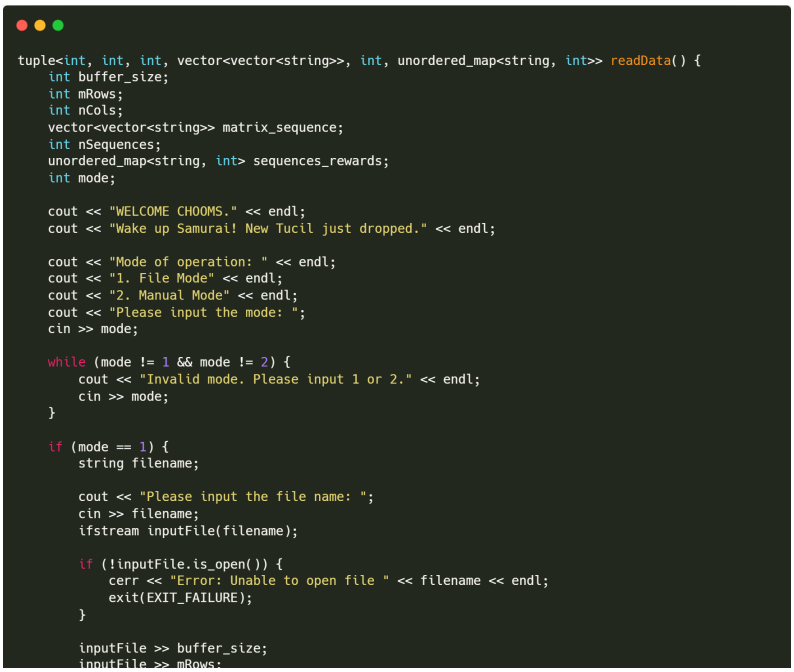
Meskipun demikian, algoritma brute force memiliki berbagai aplikasi yang penting. Selain pencarian solusi optimal, algoritma ini juga digunakan untuk validasi dan verifikasi data. Dalam konteks pengujian perangkat lunak, brute force digunakan untuk memeriksa semua kemungkinan input dan output guna memastikan bahwa perangkat lunak bekerja dengan benar. Selain itu, algoritma brute force sering digunakan dalam analisis kompleksitas algoritma, membantu mengidentifikasi waktu yang diperlukan untuk menyelesaikan masalah.

Meskipun ada keraguan tentang efisiensi waktu dalam kasus-kasus tertentu, algoritma brute force tetap menjadi salah satu pendekatan yang paling sederhana dan mudah dipahami dalam menyelesaikan berbagai masalah. Kelebihan utamanya adalah kemampuannya untuk menemukan solusi optimal tanpa memerlukan pengetahuan awal yang mendalam tentang masalah yang diberikan. Dengan demikian, meskipun terdapat keterbatasan, algoritma brute force tetap menjadi salah satu metode yang paling sederhana dan sering digunakan dalam menyelesaikan berbagai masalah.

BAB III
PENJELASAN PROGRAM

Program ini merupakan implementasi dari sebuah permainan atau masalah yang melibatkan manipulasi matriks. Program ini memungkinkan pengguna untuk memilih antara dua mode operasi: "File Mode" dan "Manual Mode". Dalam mode File, program akan membaca data dari sebuah file yang berisi informasi tentang ukuran buffer, ukuran matriks, matriks itu sendiri, daftar urutan kata, dan hadiah yang terkait dengan setiap urutan kata. Sedangkan dalam mode Manual, pengguna diminta untuk memasukkan informasi tersebut secara manual. Setelah membaca data, program akan memprosesnya untuk menemukan urutan kata yang memberikan poin tertinggi berdasarkan aturan yang telah ditentukan, menggunakan fungsi GetHighestScore. Hasilnya akan dicetak, termasuk poin maksimum, pola urutan kata, dan langkah-langkah yang diambil, serta waktu eksekusi. Program ini menggabungkan penggunaan tuple, vector, unordered_map, dan beberapa fungsi bawaan C++ lainnya untuk mengelola data dan melakukan operasi terkait matriks.

1. Program Utama
    a. readData()



```
tuple<int, int, int, vector<vector<string>>, int, unordered_map<string, int>> readData() {
    int buffer_size;
    int mRows;
    int nCols;
    vector<vector<string>> matrix_sequence;
    int nSequences;
    unordered_map<string, int> sequences_rewards;
    int mode;

    cout << "WELCOME CHOOMS." << endl;
    cout << "Wake up Samurai! New Tucil just dropped." << endl;

    cout << "Mode of operation: " << endl;
    cout << "1. File Mode" << endl;
    cout << "2. Manual Mode" << endl;
    cout << "Please input the mode: ";
    cin >> mode;

    while (mode != 1 && mode != 2) {
        cout << "Invalid mode. Please input 1 or 2." << endl;
        cin >> mode;
    }

    if (mode == 1) {
        string filename;

        cout << "Please input the file name: ";
        cin >> filename;
        ifstream inputFile(filename);

        if (!inputFile.is_open()) {
            cerr << "Error: Unable to open file " << filename << endl;
            exit(EXIT_FAILURE);
        }

        inputFile >> buffer_size;
        inputFile >> mRows;
```

```
        ifstream inputFile(filename);

        if (!inputFile.is_open()) {
            cerr << "Error: Unable to open file " << filename << endl;
            exit(EXIT_FAILURE);
        }

        inputFile >> buffer_size;
        inputFile >> mRows;
        inputFile >> nCols;
        matrix_sequence.resize(mRows, vector<string>(nCols));
        for (int i = 0; i < mRows; i++) {
            for (int j = 0; j < nCols; j++) {
                inputFile >> matrix_sequence[i][j];
            }
        }
        inputFile >> nSequences;
        for (int i = 0; i < nSequences; i++) {
            inputFile.ignore();
            string Sequence;
            int reward;
            getline(inputFile, Sequence);
            Sequence.erase(remove_if(Sequence.begin(), Sequence.end(), ::isspace), Sequence.end());
            inputFile >> reward;
            sequences_rewards[Sequence] = reward;
        }

        inputFile.close();
    } else {
        vector<string> tokens;
        int nTokens;
        int maxLength;
        while (!(std::cout << "Please input the number of unique tokens: " && std::cin >> nTokens))
{
        std::cin.clear();
        std::cin.ignore();
        std::cout << "Invalid input. Please input a valid number.\n";
    }
        for (int i = 0; i < nTokens; i++) {
            string token;
            cout << "Please input token " << i+1 << ": ";
            cin >> token;
            while (token.size() != 2) {
                cout << "Invalid token. Please input a 2-character token: ";
                cin >> token;
            }
            tokens.push_back(token);
        }

        int* variables[] = {&buffer_size, &mRows, &nCols, &nSequences, &maxLength};
        const char* prompts[] = {"buffer size", "number of rows", "number of columns", "number of
sequences", "sequences max length"};

        for (int i = 0; i < 5; ++i) {
            while (!(std::cout << "Please input the " << prompts[i] << ": " && std::cin >>
*variables[i])) {
                std::cin.clear();
                std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                std::cout << "Invalid input. Please input a valid number.\n";
            }
        }

        matrix_sequence = randomMatrix(tokens);
        sequences_rewards = randomSequence(tokens, nSequences, maxLength);
    }

    cout << "Buffer Size: " << buffer_size << endl;
    cout << "Matrix Size: " << mRows << "x" << nCols << endl;
    cout << "Matrix: " << endl;
    for (const auto& i : matrix_sequence) {
        for (const auto& j : i) {
            cout << j << " ";
        }
        cout << endl;
    }
    cout << endl;
    cout << "Number of sequences: " << nSequences << endl;
    cout << "All sequences and rewards:" << endl;
    for (const auto& i : sequences_rewards) {
        cout << i.first << ": " << i.second << endl;
    }
    cout << endl;

    return make_tuple(buffer_size, mRows, nCols, matrix_sequence, nSequences, sequences_rewards);
}
```

Program ini adalah sebuah fungsi bernama “readData()” yang membaca data dari pengguna dalam dua mode: "File Mode" dan "Manual Mode". Dalam mode File, program meminta pengguna untuk memasukkan nama file yang berisi data yang akan dibaca. Data

tersebut mencakup ukuran buffer, ukuran matriks, isi matriks, jumlah urutan kata, dan hadiah yang terkait dengan setiap urutan kata. Setelah membaca file, data diproses dan disimpan dalam variabel yang sesuai. Dalam mode Manual, pengguna diminta untuk memasukkan informasi tersebut secara langsung melalui keyboard. Selanjutnya, program mencetak data yang telah dibaca untuk verifikasi. Data yang telah dibaca kemudian dikembalikan dalam bentuk tuple yang berisi semua informasi yang telah dibaca. Program ini menggunakan berbagai fitur C++ seperti input/output, manipulasi string, penggunaan file, serta penggunaan struktur data seperti vektor dan peta tidak berurutan (unordered_map).

b. main()

```cpp
int main() {

    int buffer_size, mRows, nCols, nSequences;
    vector<vector<string>> matrix_sequence;
    unordered_map<string, int> sequences_rewards;
    pair<int, int> start;
    bool isHorizontal;
    PossibleSequence MaxPossible;

    tie(buffer_size, mRows, nCols, matrix_sequence, nSequences, sequences_rewards) = readData();

    auto begin = chrono::high_resolution_clock::now();

    PossibleSequence result = GetHighestScore(start, isHorizontal, matrix_sequence, buffer_size,
MaxPossible, sequences_rewards);

    auto end = chrono::high_resolution_clock::now();

    auto duration = chrono::duration_cast<chrono::milliseconds>(end - begin);

    cout << "Max Point: " << result.Point << endl;
    cout << "Pattern: " << addSpace(result.Pattern) << endl;
    for (const auto& point : result.steps) {
        cout << "<" << point.first+1 << "," << point.second+1 << "> " << endl;
    }
    cout << endl;
    cout << "Execution time: " << duration.count() << " ms" << endl;

    cout << "Do you want to save the result to a file? (y/n): ";
    char choice;
    cin >> choice;
    while (choice != 'y' && choice != 'n') {
        cout << "Invalid input. Please input y or n: ";
        cin >> choice;
    }

    if (choice == 'y') {
        cout << "Please input the filename: ";
        string filename;
        cin >> filename;
        ofstream outputFile("../test/"+ filename +".txt");
        if (!outputFile.is_open()) {
            cerr << "Error: Unable to open file " << "test/"+filename+".txt" << endl;
            exit(EXIT_FAILURE);
        }
        outputFile << "Max Point: " << result.Point << endl;
        outputFile << "Pattern: " << addSpace(result.Pattern) << endl;
        for (const auto& point : result.steps) {
            outputFile << "<" << point.first+1 << "," << point.second+1 << "> " << endl;
        }
        outputFile << endl;
        outputFile << "Execution time: " << duration.count() << " ms" << endl;
        outputFile.close();
        cout << "Result has been saved to " << "test/"+filename+".txt" << endl;
    } else {
        cout << "Result has not been saved." << endl;
    }

    return 0;
}
```

Program ini merupakan bagian utama dari program yang menggunakan beberapa fungsi yang telah dijelaskan sebelumnya. Pertama, program membaca data dari pengguna

atau dari file menggunakan fungsi readData(). Setelah itu, program menghitung hasil terbaik menggunakan fungsi GetHighestScore(). Waktu yang diperlukan untuk eksekusi juga diukur menggunakan chrono. Hasilnya, seperti poin maksimum, pola urutan kata, dan langkah-langkah yang diambil, dicetak ke layar. Pengguna juga diberi pilihan untuk menyimpan hasil ke dalam file. Jika pengguna memilih untuk menyimpan, program akan meminta nama file dan menyimpan hasil tersebut ke dalam file teks. Jika tidak, program akan mencetak bahwa hasil tidak disimpan.

2. Fungsi

a. findAllPossiblePaths()

```
vector<vector<pair<int, int>>> findAllPossiblePaths(const vector<vector<string>>& matrix, int
bufferSize, pair<int, int> start, bool isHorizontal) {
    int x = start.first;
    int y = start.second;

    vector<vector<pair<int, int>>> paths;
    if (bufferSize == 1) {
        if (isHorizontal) {
            for (int new_y = 0; new_y < matrix[0].size(); ++new_y) {
                if (new_y != y) {
                    paths.push_back({{x, new_y}});
                }
            }
        } else {
            for (int new_x = 0; new_x < matrix.size(); ++new_x) {
                if (new_x != x) {
                    paths.push_back({{new_x, y}});
                }
            }
        }
    } else {
        if (isHorizontal) {
            for (int new_y = 0; new_y < matrix[0].size(); ++new_y) {
                if (new_y != y) {
                    auto smallerPaths = findAllPossiblePaths(matrix, bufferSize - 1, {x, new_y},
false);

                    for (const auto& path : smallerPaths) {
                        vector<pair<int, int>> newPath = {{x, new_y}};
                        newPath.insert(newPath.end(), path.begin(), path.end());
                        paths.push_back(newPath);
                    }
                }
            }
        } else {
            for (int new_x = 0; new_x < matrix.size(); ++new_x) {
                if (new_x != x) {
                    auto smallerPaths = findAllPossiblePaths(matrix, bufferSize - 1, {new_x, y},
true);

                    for (const auto& path : smallerPaths) {
                        vector<pair<int, int>> newPath = {{new_x, y}};
                        newPath.insert(newPath.end(), path.begin(), path.end());
                        paths.push_back(newPath);
                    }
                }
            }
        }
    }
    return paths;
}
```

Program ini adalah sebuah fungsi bernama `findAllPossiblePaths` yang bertujuan untuk menemukan semua jalur yang mungkin dari posisi awal dalam matriks tertentu, dengan mempertimbangkan batasan ukuran buffer dan arah pergerakan (horizontal atau vertikal). Fungsi ini menggunakan rekursi untuk

mengeksplorasi semua kemungkinan langkah yang dapat diambil dari posisi awal yang diberikan. Pada setiap langkah, fungsi mempertimbangkan apakah buffer masih tersedia untuk pergerakan tambahan. Jika buffer masih tersedia, fungsi akan memanggil dirinya sendiri dengan buffer yang dikurangi satu, mempertimbangkan langkah tambahan yang mungkin diambil, dan menambahkan langkah-langkah tersebut ke dalam jalur-jalur yang telah ditemukan sebelumnya. Setiap jalur yang berhasil ditemukan akan dimasukkan ke dalam vektor paths dan kemudian dikembalikan sebagai hasil dari fungsi ini. Ini adalah pendekatan rekursif yang memungkinkan untuk penjelajahan lengkap semua kemungkinan jalur yang mungkin dengan memperhatikan batasan buffer.

b. addSpace()

```
string addSpace(const string& input) {
    string spacedString = "";
    for (size_t i = 0; i < input.length(); i += 2) {
        spacedString += input.substr(i, 2) + " ";
    }
    if (!spacedString.empty()) {
        spacedString.pop_back();
    }
    return spacedString;
}
```

Fungsi `addSpace` ini bertujuan untuk menambahkan spasi setiap dua karakter dalam string yang diberikan. Ini berguna untuk memformat string menjadi bentuk yang lebih mudah dibaca atau ditampilkan. Fungsi ini menggunakan loop for untuk mengiterasi melalui string input, dan setiap dua karakter diambil dengan menggunakan fungsi `substr` dan ditambahkan spasi di belakangnya sebelum ditambahkan ke dalam string `spacedString`. Setelah selesai iterasi, spasi terakhir yang tidak dibutuhkan dihapus menggunakan `pop_back` jika `spacedString` tidak kosong. Akhirnya, string yang telah diformat kembali dikembalikan sebagai hasil fungsi. Ini adalah pendekatan sederhana yang efektif untuk menambahkan spasi pada setiap pasangan karakter dalam string.

c. GetHighestScore()

```cpp
const PossibleSequence &GetHighestScore(std::pair<int, int> &start, bool &isHorizontal,
std::vector<std::vector<std::__cxx11::string>> &matrix_sequence, int buffer_size, PossibleSequence
&MaxPossible, std::unordered_map<std::__cxx11::string, int> &sequences_rewards)
{
    start = {0, -1};
    isHorizontal = true;
    auto paths = findAllPossiblePaths(matrix_sequence, buffer_size, start, isHorizontal);

    MaxPossible.Point = 0;
    MaxPossible.Pattern = "";
    for (const auto &path : paths)
    {
        string sequence = "";
        int poin = 0;
        for (const auto &point : path)
        {
            sequence += matrix_sequence[point.first][point.second];
        }
        for (const auto &i : sequences_rewards)
        {
            size_t found = sequence.find(i.first);
            while (found != string::npos)
            {
                poin += i.second;
                found = sequence.find(i.first, found + 1);
            }
        }
        if (poin > MaxPossible.Point)
        {
            MaxPossible.Point = poin;
            MaxPossible.steps = path;
            MaxPossible.Pattern = sequence;
        }
    }

    return MaxPossible;
}
```

Fungsi `GetHighestScore` bertujuan untuk mencari urutan kata yang memberikan poin tertinggi berdasarkan aturan tertentu di dalam matriks. Pertama-tama, posisi awal dan arah pergerakan diatur, lalu semua kemungkinan jalur dari posisi awal dengan memperhatikan batasan ukuran buffer dan arah pergerakan ditemukan menggunakan fungsi `findAllPossiblePaths`. Selanjutnya, variabel `MaxPossible` yang merepresentasikan hasil terbaik yang telah ditemukan diinisialisasi dengan poin 0 dan pola urutan kata kosong. Selama iterasi melalui semua jalur yang mungkin, fungsi ini membangun string urutan kata dari setiap jalur, dan kemudian menghitung poin untuk urutan kata tersebut berdasarkan hadiah yang telah ditentukan. Jika poin yang dihitung lebih besar dari poin maksimum yang telah ada sebelumnya, `MaxPossible` diperbarui dengan nilai yang lebih tinggi, dan jalur serta pola urutan kata dari urutan tersebut juga diperbarui. Setelah iterasi selesai, `MaxPossible` dikembalikan sebagai hasil, yang berisi urutan kata dengan poin tertinggi yang telah ditemukan. Ini adalah pendekatan yang efisien untuk mencari solusi terbaik dalam kumpulan jalur yang mungkin, dengan mempertimbangkan setiap urutan kata dan menghitung poinnya.

d. randomSequence()

```
unordered_map<string, int> randomSequence(const vector<string>& tokens, int numSequences, int
maxLength) {
    unordered_map<string, int> sequences_rewards;

    random_device rd;
    mt19937 g(rd());

    for (int i = 0; i < numSequences; ++i) {
        int sequenceLength = uniform_int_distribution<>(2, maxLength)(g);
        int randomNumber = uniform_int_distribution<>(1, 20)(g) * 5;
        string sequence;

        for (int j = 0; j < sequenceLength; ++j) {
            int index = uniform_int_distribution<>(0, tokens.size() - 1)(g);
            sequence += tokens[index];
        }

        sequences_rewards[sequence] = randomNumber;
    }

    return sequences_rewards;
}
```

Fungsi `randomSequence` bertujuan untuk menghasilkan urutan kata secara acak berserta poin hadiah yang terkait. Ini dilakukan dengan menggunakan vektor `tokens` yang berisi token-token yang mungkin digunakan dalam pembuatan urutan kata, jumlah urutan kata yang ingin dibuat (`numSequences`), dan panjang maksimum yang diizinkan untuk setiap urutan kata (`maxLength`). Dalam setiap iterasi, fungsi ini memilih secara acak panjang urutan kata antara 2 dan `maxLength`, kemudian menghasilkan sebuah urutan kata dengan memilih secara acak token dari vektor `tokens` sebanyak panjang urutan kata yang telah ditentukan. Setelah itu, fungsi ini juga menghasilkan secara acak sebuah nilai poin hadiah antara 5 dan 100, dan kemudian memetakan urutan kata yang telah dihasilkan ke nilai poin hadiah tersebut dalam sebuah `unordered_map` bernama `sequences_rewards`. Setelah semua iterasi selesai, `unordered_map` ini dikembalikan sebagai hasil dari fungsi. Dengan demikian, fungsi ini menghasilkan sejumlah urutan kata acak beserta poin hadiah yang terkait untuk digunakan dalam program lainnya.

e. randomMatrix()

```
vector<vector<string>> randomMatrix(const vector<string>& tokens) {
    random_device rd;
    mt19937 gen(rd());

    vector<string> shuffledTokens = tokens;
    shuffle(shuffledTokens.begin(), shuffledTokens.end(), gen);

    int numRows = shuffledTokens.size();
    vector<vector<string>> matrix(numRows, vector<string>(numRows));

    int tokenIndex = 0;
    for (int i = 0; i < numRows; i++) {
        for (int j = 0; j < numRows; j++) {
            matrix[i][j] = shuffledTokens[tokenIndex];
            tokenIndex = (tokenIndex + 1) % shuffledTokens.size();
        }
    }

    shuffle(matrix.begin(), matrix.end(), gen);
    for (int i = 0; i < numRows; i++) {
        shuffle(matrix[i].begin(), matrix[i].end(), gen);
    }

    return matrix;
}
```

Fungsi `randomMatrix` bertujuan untuk membuat matriks acak yang berisi token-token dari vektor `tokens`. Pertama-tama, vektor `tokens` disalin ke dalam vektor baru `shuffledTokens`, dan kemudian diacak menggunakan fungsi `shuffle`. Setelah itu, jumlah baris matriks diatur sama dengan ukuran `shuffledTokens`, dan matriks dengan jumlah baris tersebut dan setiap barisnya berisi token-token yang telah diacak secara acak dari `shuffledTokens` dibuat. Proses pengisian matriks dilakukan dengan cara menempatkan token dari `shuffledTokens` sesuai dengan indeks yang telah ditentukan sebelumnya. Setelah matriks terisi penuh, baik baris maupun kolom dari matriks juga diacak menggunakan fungsi `shuffle`. Akhirnya, matriks yang telah terbentuk dikembalikan sebagai hasil dari fungsi. Dengan demikian, fungsi ini menghasilkan matriks acak yang berisi token-token yang telah diacak dari vektor `tokens`, dan matriks ini dapat digunakan dalam program untuk tujuan tertentu seperti simulasi atau pengujian.

BAB IV
EKSPERIMEN

Input	Output
7 6 6 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 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	WELCOME CHOOMS. Wake up Samurai! New Tucil just dropped. Mode of operation: 1. File Mode 2. Manual Mode Please input the mode: 1 Please input the file name: input1 Buffer Size: 7 Matrix Size: 6x6 Matrix: 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 Number of sequences: 3 All sequences and rewards: BD 1C BD 55: 30 BD E9 1C: 15 BD 7A BD: 20 Max Point: 50 Pattern: 7A BD 7A BD 1C BD 55 <1,1> <4,1> <4,3> <5,3> <5,6> <3,6> <3,1> Execution time: 766 ms Do you want to save the result to a file? (y/n): y Please input the filename: test1 Result has been saved to test/test1.txt

5 5 5 7A 55 E9 E9 1C 55 7A 1C 7A E9 55 1C 1C 55 E9 BD 1C 7A 1C 55 BD 55 BD 7A 1C 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	Mode of operation: 1. File Mode 2. Manual Mode Please input the mode: 1 Please input the file name: input2 Buffer Size: 5 Matrix Size: 5x5 Matrix: 7A 55 E9 E9 1C 55 7A 1C 7A E9 55 1C 1C 55 E9 BD 1C 7A 1C 55 BD 55 BD 7A 1C Number of sequences: 3 All sequences and rewards: BD 1C BD 55: 30 BD E9 1C: 15 BD 7A BD: 20 Max Point: 20 Pattern: 7A BD 7A BD BD <1,1> <4,1> <4,3> <5,3> <5,1> Execution time: 7 ms Do you want to save the result to a file? (y/n): y Please input the filename: output2 Result has been saved to test/output2.txt

7 5 5 7A 55 E9 E9 1C 55 7A 1C 7A E9 55 1C 1C 55 E9 BD 1C 7A 1C 55 BD 55 BD 7A 1C 4 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30 7A E9 BD 50	WELCOME CHOOMS. Wake up Samurai! New Tucil just dropped. Mode of operation: 1. File Mode 2. Manual Mode Please input the mode: 1 Please input the file name: input3 Buffer Size: 7 Matrix Size: 5x5 Matrix: 7A 55 E9 E9 1C 55 7A 1C 7A E9 55 1C 1C 55 E9 BD 1C 7A 1C 55 BD 55 BD 7A 1C Number of sequences: 4 All sequences and rewards: 7A E9 BD: 50 BD 1C BD 55: 30 BD E9 1C: 15 BD 7A BD: 20 Max Point: 50 Pattern: 55 7A 55 7A E9 BD BD <1,2> <2,2> <2,1> <1,1> <1,3> <5,3> <5,1> Execution time: 169 ms
Please input the number of unique tokens: 4 Please input token 1: BA Please input token 2: 7A Please input token 3: OU Please input token 4: E9 Please input the buffer size: 6 Please input the number of rows: 6 Please input the number of columns: 6 Please input the number of sequences: 4 Please input the sequences max length: 4	

```
WELCOME CHOOMS.
Wake up Samurai! New Tucil just dropped.
Mode of operation:
1. File Mode
2. Manual Mode
Please input the mode: 2
Please input the number of unique tokens: 4
Please input token 1: BA
Please input token 2: 7A
Please input token 3: OU
Please input token 4: E9
Please input the buffer size: 6
Please input the number of rows: 6
Please input the number of columns: 6
Please input the number of sequences: 4
Please input the sequences max length: 4
Buffer Size: 6
Matrix Size: 6x6
Matrix:
7A OU E9 BA
E9 OU BA 7A
E9 7A OU BA
7A OU E9 BA

Number of sequences: 4
All sequences and rewards:
7A BA OU 7A: 80
7A OU: 70
7A E9 BA: 75
OU E9 E9: 90

Max Point: 165
Pattern: 7A E9 BA OU E9 E9
```

```
<1,1>
<2,1>
<2,3>
<3,3>
<3,1>
<2,1>

Execution time: 6 ms

Do you want to save the result to a file? (y/n): y
Please input the filename: output4
Result has been saved to test/output4.txt
```

Please input the mode: 2
Please input the number of unique tokens: 5
Please input token 1: B9
Please input token 2: HU
Please input token 3: LO
Please input token 4: &A
Please input token 5: 3A
Please input the buffer size: 10
Please input the number of rows: 10
Please input the number of columns: 10
Please input the number of sequences: 5
Please input the sequences max length: 6

```
WELCOME CHOOMS.
Wake up Samurai! New Tucil just dropped.
Mode of operation:
1. File Mode
2. Manual Mode
Please input the mode: 2
Please input the number of unique tokens: 5
Please input token 1: B9
Please input token 2: HU
Please input token 3: LO
Please input token 4: &A
Please input token 5: 3A
Please input the buffer size: 10
Please input the number of rows: 10
Please input the number of columns: 10
Please input the number of sequences: 5
Please input the sequences max length: 6
Buffer Size: 10
Matrix Size: 10x10
Matrix:
&A 3A B9 HU LO
3A HU B9 &A LO
LO HU &A B9 3A
HU &A B9 3A LO
3A &A B9 LO HU

Number of sequences: 5
All sequences and rewards:
HU B9 LO HU HU: 80
&A LO: 70
B9 B9 3A 3A: 25
HU 3A B9 LO: 75
3A 3A B9 &A HU LO: 55
```

```
Execution time: 14858 ms

Do you want to save the result to a file? (y/n): y
Please input the filename: output5
Result has been saved to test/output5.txt
```

5 7 7 7A 55 E9 E9 1C 55 55 55 7A 1C 7A E9 55 BD 55 1C 1C 55 E9 BD 1C BD 1C 7A 1C 55 BD 7A BD 55 BD 7A 1C 1C E9 1C 55 55 7A 55 7A 55 1C 55 55 7A 55 7A 55 3 BD E9 1C 15 BD 7A BD BD 20 BD 1C BD 55 30 55 1C BD BD 100	Please input the mode: 1 Please input the file name: input6 Buffer Size: 5 Matrix Size: 7x7 Matrix: 7A 55 E9 E9 1C 55 55 55 7A 1C 7A E9 55 BD 55 1C 1C 55 E9 BD 1C BD 1C 7A 1C 55 BD 7A BD 55 BD 7A 1C 1C E9 1C 55 55 7A 55 7A 55 1C 55 55 7A 55 7A 55 Number of sequences: 3 All sequences and rewards: BD 1C BD 55: 30 BD E9 1C: 15 BD 7A BD BD: 20 Max Point: 30 Pattern: 7A BD 1C BD 55 <1,1> <5,1> <5,6> <3,6> <3,1> Execution time: 53 ms Do you want to save the result to a file? (y/n): y Please input the filename: output6 Result has been saved to test/output6.txt

BAB V
KESIMPULAN

1. Kesimpulan

Program ini mengimplementasikan algoritma brute force untuk mencari urutan kata dengan poin tertinggi dalam sebuah matriks. Pendekatan brute force mencoba semua kemungkinan jalur yang mungkin dari posisi awal, dengan mempertimbangkan batasan ukuran buffer dan arah pergerakan. Namun, pendekatan brute force memakan waktu yang cukup lama karena harus mengeksplorasi seluruh ruang pencarian tanpa memanfaatkan struktur atau informasi lain yang mungkin mempercepat proses pencarian.

2. Saran

Untuk meningkatkan efisiensi, dapat dipertimbangkan untuk mengimplementasikan algoritma pencarian yang lebih canggih, seperti algoritma backtracking atau algoritma greedy. Algoritma-algoritma ini memanfaatkan heuristik atau pemangkasan ruang pencarian untuk mempercepat proses pencarian, terutama untuk input yang lebih besar. Selain itu, pengoptimalan kode dan pemilihan struktur data yang sesuai juga dapat membantu meningkatkan kinerja program. Selain itu, jika memungkinkan, memanfaatkan paralelisme atau komputasi terdistribusi juga dapat menjadi solusi untuk meningkatkan kecepatan pencarian.

DAFTAR PUSTAKA

Nizar GG. (2021). *Cyberpunk 2077 Easiest Way to Breach Protocols & Access Points WHILE Cracking ALL Sequences* [Video]. YouTube. https://www.youtube.com/watch?v=75b4IUBZ5jI. Diakses pada 12 Februari pukul 23:55.

Lamanit. (2023, 22 Februari). *Algoritma Brute Force*: Pengertian, Fungsi dan Implementasi. Diakses pada 13 Februari pukul 5:30 dari https://lamanit.com/algoritma-brute-force/#Pengertian_Algoritma_Brute_Force.

LAMPIRAN

1. Github Link: https://github.com/thoriqsaputra/Tucil1_13522141.git
2.

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