

Tugas Kecil IF2211 Strategi Algoritma
Penyelesaian Cyberpunk 2077 Breach Protocol
dengan Algoritma Brute Force

Semester I Tahun 2023/2024



Disusun oleh:

13522061-Maximilian Sulistiyo

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

Bab I Algoritma Bruteforce dan Deskripsi Permasalahan	3
1.1 Algoritma Bruteforce.....	3
1.2 Deskripsi Permasalahan Cyberpunk 2077 Breach Protocol	3
1.3 Penjelasan Implementasi Algoritma Bruteforce pada Cyberpunk 2077 Breach Protocol	3
Bab II Implementasi Dalam Bahasa C++	5
2.1 utilities.cpp	5
2.2 algorithm.cpp	11
2.3 main.cpp	14
Bab III Testing Input Output.....	17
3.1 Input Dari File	17
3.2 Input Dari CLI	19
Bab IV Lampiran	22

Bab I

Algoritma Bruteforce dan Deskripsi Permasalahan

1.1 Algoritma Bruteforce

Algoritma bruteforce dapat dideskripsikan sebagai algoritma yang menyelesaikan suatu masalah secara *straight forward* / lempang. Hal ini karena algoritma bruteforce memecahkan suatu masalah dengan cara mencoba semua solusi yang mungkin terjadi. Biasanya algoritma ini pun didasarkan pada pernyataan pada persoalan (*problem statement*) dan juga definisi/konsep yang dilibatkan.

Algoritma bruteforce memiliki beberapa ciri khas yaitu sederhana, jelas, dan langsung dimana algoritma yang diimplementasikan tidak memerlukan pemecahan masalah yang canggih. Namun karena hal tersebut maka algoritma menjadi sangat tidak efisien karena memerlukan banyak enumerasi untuk mencari semua solusi yang mungkin terjadi. Oleh karena itu juga dijaminakan bahwa jika terdapat sebuah solusi maka algoritma ini akan menemukannya.

1.2 Deskripsi Permasalahan Cyberpunk 2077 Breach Protocol

Cyberpunk 2077 Breach Protocol merupakan salah satu minigame dari game Cyberpunk 2077. Minigame ini mensimulasikan proses peretasan jaringan lokal dari *ICE (Intrusion Countermeasures Electronics)*. Dalam game ini terdapat beberapa komponen yaitu

- Token: Terdiri dari dua karakter alphanumerik
- Matriks: Terdiri dari kumpulan-kumpulan token
- Sekuens: Rangkaian token (dua atau lebih) yang harus dicocokkan
- Buffer: Jumlah maksimal dari token yang tersusun sekuensial

Permainan ini memiliki beberapa aturan yaitu:

1. Jalan yang ditempuh harus bergerak secara vertikal, horizontal, vertikal, horizontal bergantian hingga ukuran buffer penuh.
2. Jalan dimulai dari salah satu token yang berada di baris paling atas dan kemudian bergerak vertikal.
3. Setelah buffer ditemukan maka sequence akan dicocokkan untuk menghitung point yang didapat.
4. Token dapat diulang dalam suatu buffer.
5. Setiap sekuens memiliki point yang berbeda-beda.
6. Sekuens memiliki panjang minimal dua.

1.3 Penjelasan Implementasi Algoritma Bruteforce pada Cyberpunk 2077 Breach Protocol

Masalah *Cyberpunk 2077 Breach Protocol* pun dapat diselesaikan menggunakan strategi bruteforce. Algoritma penyelesaian sebagai berikut:

1. Pertama program menerima input matriks main dan juga sequence beserta pointnya.
2. Menginisialisasi matriks visited yang menandakan apakah sebuah cell sudah di lalui atau belum berdasarkan ukuran matriks main yang diterima.
3. Program meng-iterasi dari tiap elemen baris teratas untuk menjadikan masukkan pada fungsi rekursif findPath.

4. findPath merupakan fungsi rekursif dengan basis saat program memilih cell diluar batas row dan col atau cell sudah di visit berdasarkan matriks visited.
5. Setelah melewati kasus basis, program menandakan cell visited menggunakan matriks visited dan kemudian juga menyimpan path sementara dan juga koordinat path sementara.
6. Program akan menghitung point untuk path sementara, jika lebih besar maka program akan menyimpan maxPoints, path, dan pathCoordinates.
7. Program mengecek apakah remainingBuffer masih tersisa, jika iya, menggunakan bool isVertical akan menentukan apakah langkah selanjutnya vertical atau horizontal, program akan mengiterasi fungsi findPath pada setiap elemen pada vertical maupun horizontal tergantung dengan bool yang diterima.
8. Jika remainingBuffer sudah tidak ada sisa maka program akan menandakan cell unvisited dan juga mengurangi path dan pathCoordinates.
9. Setelah semua path dicari, nilai maxPoints, path, dan pathCoordinates akan ditampilkan, terdapat opsi untuk menyimpan hasil ke file txt di folder test.

Bab II

Implementasi Dalam Bahasa C++

Dalam pengimplementasiannya, struktur program dibagi menjadi tiga file yaitu utilities.cpp, algorithm.cpp, dan main.cpp.

2.1 utilities.cpp

Function / procedure yang terdapat pada file ini berfungsi untuk membantu function lain di program, terdapat beberapa fungsi / procedure yaitu:

Fungsi / Procedure	Penjelasan
<pre>bool readFile string fileName, int &bufferSize, vector<vector<string>> &playMatrix, vector<vector<string>> &seqVec, vector<int> &rewardVec);</pre>	Fungsi ini berguna untuk membaca file yang sesuai dengan format tertulis pada spek tugas ini, setelah membaca fungsi ini akan menyimpan bufferSize, playMatrix, seqVec, dan rewardVec.
<pre>void readTerminal (vector<vector<string>> &playMatrix, vector<vector<string>> &seqVec, int &bufferSize, vector<int> &rewardVec);</pre>	Fungsi ini berguna untuk menerima inputan dari CLI, setelah membaca fungsi ini akan menyimpan bufferSize, playMatrix, seqVec, dan rewardVec.
<pre>void printMatrix (vector<vector<string>> matrix);</pre>	Fungsi in berguna untuk mencetak matrix ke terminal, berguna dalam debugging
<pre>void printStringVector (vector<string> vec);</pre>	Fungsi in berguna untuk mencetak vector string ke terminal.
<pre>void printCoordinates (vector<pair<int, int>> pathCoordinates);</pre>	Fungsi in berguna untuk mencetak vector pair coordinates ke terminal.
<pre>void printOutput(int maxPoints, vector<string> path, vector<pair<int, int>> pathCoordinates, long long runTime)</pre>	Fungsi ini digunakan untuk menampilkan hasil algoritma ke terminal
<pre>void saveFile(int maxPoints, vector<string> bestPath, vector<pair<int,int>> bestPathCoordinates, long long runTime);</pre>	Fungsi ini digunakan untuk mencetak hasil algoritma ke sebuah file txt

```
void saveFileOptions(int maxPoints,
vector<string> bestPath,
vector<pair<int,int>>
bestPathCoordinates, long long
runTime);
```

Fungsi ini digunakan untuk pengecekan I/O saat proses save

Berikut adalah code yang terdapat pada utilities.cpp

```
void printMatrix(vector<vector<string>> matrix){
    for(auto row : matrix){
        for (auto element : row){
            cout << element << " ";
        }
        cout << endl;
    }
}

void printStringVector(vector<string> vec){
    for(auto elmt : vec) {
        cout << elmt << " ";
    }
    cout << endl;
}

bool readFile(string fileName, int &bufferSize, vector<vector<string>>
&playMatrix, vector<vector<string>> &seqVec, vector<int> &rewardVec){
    ifstream file(fileName);

    if(!file.is_open()){
        cout << "File gagal dibuka / tidak ditemukan, silahkan coba lagi!" <<
endl;
        return false;
    }
    file.exceptions(ifstream::failbit | ifstream::badbit);
    try{
        int row, col, numOfSeq;
        string token;
        file >> bufferSize;
        file >> col >> row;
        playMatrix = vector<vector<string>>(row, vector<string>(col, ""));
        for(int i = 0; i < row; i++){
            for(int j = 0; j < col; j++){
                file >> token;
                playMatrix[i][j] = token;
            }
        }
        file >> numOfSeq;
        file.ignore(numeric_limits<streamsize>::max(), '\n');
```

```

        seqVec.resize(numOfSeq);
        string line;
        for (int i = 0; i < numOfSeq; i++){
            getline(file, line);
            stringstream strStream(line);
            while(strStream >> token){
                seqVec[i].push_back(token);
            }
            int temp;
            file >> temp;
            rewardVec.push_back(temp);
            file.ignore(numeric_limits<streamsize>::max(), '\n');
        }

    } catch(const std::ifstream::failure& e){
        if (!file.eof()) {
            std::cerr << "Error occurred during file operations: " << e.what()
<< std::endl;
            return false;
        }
    } catch(...){
        cerr << "Sebuah error terjadi saat pembacaan file!" << endl;
        return false;
    }
    return true;
}

void readTerminal(vector<vector<string>> &playMatrix, vector<vector<string>>
&seqVec, int &bufferSize, vector<int> &rewardVec){
    int totUniqueTokens, row, col, numOfSeq, maxSizeSeq;
    vector<string> tokenVec;
    random_device rd;
    mt19937 gen(rd());
    bool valid = false;
    while(!valid){
        cout << "Jumlah token unik: ";
        cin >> totUniqueTokens;
        if(cin.fail()){
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Input tidak valid. Silahkan coba lagi!" << endl;
            continue;
        }
        cout << "Token: ";
        if(cin.fail()){
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Input tidak valid. Silahkan coba lagi!" << endl;

```

```

        continue;
    }
    for (int i = 0; i < totUniqueTokens; i++){
        string token;
        cin >> token;
        tokenVec.push_back(token);
    }
    cout << "Ukuran buffer: ";
    cin >> bufferSize;
    if(cin.fail()){
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Input tidak valid. Silahkan coba lagi!" << endl;
        continue;
    }

    cout << "Ukuran matrix (row,col): ";
    cin >> row >> col;
    if(cin.fail()){
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Input tidak valid. Silahkan coba lagi!" << endl;
        continue;
    }

    cout << "Jumlah sequence: ";
    cin >> numOfSeq;
    if(cin.fail()){
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Input tidak valid. Silahkan coba lagi!" << endl;
        continue;
    }

    cout << "Ukuran maximal sequence: ";
    cin >> maxSizeSeq;
    if(cin.fail()){
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Input tidak valid. Silahkan coba lagi!" << endl;
        continue;
    }else{
        valid = true;
    }
}

seqVec.resize(numOfSeq);
for (int i = 0; i < numOfSeq; i++){

```



```

        uniform_int_distribution<> disSeqSize(2, maxSizeSeq);
        int seqSize = disSeqSize(gen);
        for(int j = 0; j < seqSize; j++){
            uniform_int_distribution<> disTokenIndex(0, totUniqueTokens-1);
            int randTokenIndex = disTokenIndex(gen);
            seqVec[i].push_back(tokenVec[randTokenIndex]);
        }
        uniform_int_distribution<> disReward(10, 50);
        int reward = disReward(gen);
        rewardVec.push_back(reward);
    }
    playMatrix = vector<vector<string>>(row, vector<string>(col, ""));
    for (int i = 0; i < row; i++){
        for(int j = 0; j < col; j++){
            uniform_int_distribution<> disMatrix(0, totUniqueTokens-1);
            int randTokenIndex = disMatrix(gen);
            playMatrix[i][j] = tokenVec[randTokenIndex];
        }
    }
    cout << endl << "Token Matrix: " << endl;
    printMatrix(playMatrix);
    for(int i = 0; i < numOfSeq; i++){
        cout << endl << "Sequence" << to_string(i+1) << ": ";
        printStringVector(seqVec[i]);
        cout << "Rewards: " << rewardVec[i] << endl;
    }
    cout << endl;
}

void printCoordinates(vector<pair<int, int>> pathCoordinates){
    for(auto coordinates : pathCoordinates){
        cout << coordinates.first << ", " << coordinates.second << endl;
    }
    cout << endl;
}

void printOutput(int maxPoints, vector<string> path, vector<pair<int, int>>
pathCoordinates, long long runTime){
    cout << maxPoints << endl;
    printStringVector(path);
    printCoordinates(pathCoordinates);
    cout << endl << runTime << "ms" << endl << endl;
}

void saveFile(int maxPoints, vector<string> bestPath, vector<pair<int,int>>
bestPathCoordinates, long long runTime){
    string outputString, extensionString, fileName;
    int count;

```

```

ifstream checkFile;

outputString = "output";
extensionString = ".txt";
count = 1;

do{
    checkFile.close();
    fileName = "test/" + outputString + to_string(count) +
extensionString;
    checkFile.open(fileName);
    count++;
}while(checkFile.is_open());

ofstream writeFile(fileName);
writeFile << maxPoints << endl;
for(auto token: bestPath){
    writeFile << token << " ";
}
writeFile << endl;
for(auto coordinates : bestPathCoordinates){
    writeFile << coordinates.first << ", " << coordinates.second << endl;
}
writeFile << endl;
writeFile << runTime << "ms" << endl;
writeFile.close();

cout << "File output berhasil dibuat, silahkan cek di: " << fileName <<
endl;
}

void saveFileOptions(int maxPoints, vector<string> bestPath,
vector<pair<int,int>> bestPathCoordinates, long long runTime){
    char saveOptions;
    bool valid, save;

    valid = false;
    save = false;
    while (!valid){
        cout << "Apakah ingin menyimpan solusi? (y/n)" << endl << endl <<
">>";
        cin >> saveOptions;
        if(cin.fail()){
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Input tidak valid. Silahkan coba lagi!" << endl;
        }else{
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

```

```

        if (saveOptions == 'y' || saveOptions == 'Y'){
            valid = true;
            save = true;
        }else if(saveOptions == 'n' || saveOptions == 'N'){
            valid = true;
        }else{
            cout << "Harap masukkan 'y' untuk ya atau 'n' untuk tidak." <<
endl;
        }
    }
}
if(save){
    saveFile(maxPoints, bestPath, bestPathCoordinates, runTime);
}
}

```

2.2 algorithm.cpp

Pada file ini algoritma utama berada, terdapat beberapa fungsi yaitu:

Fungsi / Procedure	Penjelasan
<pre> int stringMatchingPoint (vector<string> path, vector<vector<string>> seqVec, vector<int> rewardVec) </pre>	<p>Fungsi ini digunakan untuk menghitung point yang didapatkan dari sebuah buffer menggunakan sequence dan reward yang diberikan</p>
<pre> void findPath (vector<vector<string>> playMatrix, int currRow, int currCol, int remainingBuffer, bool isVertical, vector<string> &path, vector<vector<bool>> &visited, int &maxPoints, vector<string> &bestPath, vector<vector<string>> seqVec, vector<int> rewardVec, vector<pair<int, int>> &pathCoordinates, vector<pair<int, int>> &bestPathCoordinates); </pre>	<p>Fungsi ini merupakan fungsi rekursif untuk mencari semua kemungkinan path dan juga nilai maximum dari path.</p>

```
void bruteForceSolution
(vector<vector<string>> playMatrix,
int bufferSize,
vector<vector<string>> seqVec,
vector<int> rewardVec, int
&maxPoints, vector<string>
&bestPath, vector<pair<int, int>>
&bestPathCoordinates, long long
&runTime)
```

Fungsi ini berguna untuk mengiterasi fungsi findPath pada baris pertama suatu matriks token. Menyimpan bestPath, bestPathCoordinates, dan maxPoints

Berikut adalah code yang terdapat pada algorithm.cpp

```
int stringMatchingPoint(vector<string> path, vector<vector<string>> seqVec,
vector<int> rewardVec){
    int numOfSeq, i, j, k, points, pathSize, seqSize;
    vector<string> currSeq;
    bool found;

    numOfSeq = seqVec.size();
    pathSize = path.size();
    points = 0;

    for(i = 0; i < numOfSeq; i++){
        currSeq = seqVec[i];
        seqSize = currSeq.size();
        j = 0;
        found = false;

        while(j <= pathSize-seqSize && !found){
            k = 0;
            while(k <= seqSize && currSeq[k] == path[j+k]){
                k++;
            }
            if(k == seqSize){
                found = true;
            }else{
                j++;
            }
        }

        if(found){
            points += rewardVec[i];
        }
    }
    return points;
}
```

```

void findPath(vector<vector<string>> playMatrix, int currRow, int currCol, int
remainingBuffer, bool isVertical, vector<string> &path, vector<vector<bool>>
&visited, int &maxPoints, vector<string> &bestPath, vector<vector<string>>
seqVec, vector<int> rewardVec, vector<pair<int, int>> &pathCoordinates,
vector<pair<int, int>> &bestPathCoordinates){
    int row = playMatrix.size();
    int col = playMatrix[0].size();

    if(!(currRow >= 0 && currRow < row && currCol >= 0 && currCol < col &&
!visited[currRow][currCol])){
        return;
    }

    visited[currRow][currCol] = true;
    path.push_back(playMatrix[currRow][currCol]);
    pathCoordinates.push_back({currCol+1, currRow+1});

    int points = stringMatchingPoint(path, seqVec, rewardVec);
    if(points > maxPoints){
        maxPoints = points;
        bestPath = path;
        bestPathCoordinates = pathCoordinates;
    }

    if(remainingBuffer > 1){
        if(isVertical) {
            for(int nextRow = 0; nextRow < row; nextRow++){
                if(nextRow != currRow && !visited[nextRow][currCol]) {
                    findPath(playMatrix, nextRow, currCol, remainingBuffer -
1, false, path, visited, maxPoints, bestPath, seqVec, rewardVec,
pathCoordinates, bestPathCoordinates);
                }
            }
        }else{
            for(int nextCol = 0; nextCol < col; nextCol++){
                if(nextCol != currCol && !visited[currRow][nextCol]){
                    findPath(playMatrix, currRow, nextCol, remainingBuffer -
1, true, path, visited, maxPoints, bestPath, seqVec, rewardVec,
pathCoordinates, bestPathCoordinates);
                }
            }
        }
    }

    visited[currRow][currCol] = false;
    path.pop_back();
    pathCoordinates.pop_back();
}

```

```

}

void bruteForceSolution(vector<vector<string>> playMatrix, int bufferSize,
vector<vector<string>> seqVec, vector<int> rewardVec, int &maxPoints,
vector<string> &bestPath, vector<pair<int, int>> &bestPathCoordinates, long
long &runTime){
    auto start = chrono::high_resolution_clock::now();
    int row, col;
    vector<pair<int, int>> pathCoordinates;

    row = playMatrix.size();
    col = playMatrix[0].size();
    vector<vector<bool>> visited(row, vector<bool>(col, false));

    for(int currCol = 0; currCol < col; currCol++){
        vector<string> path;
        findPath(playMatrix, 0, currCol, bufferSize, true, path, visited,
maxPoints, bestPath, seqVec, rewardVec, pathCoordinates, bestPathCoordinates);
    }
    auto stop = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::milliseconds>(stop - start);
    runTime = duration.count();
}

```

2.3 main.cpp

File ini berisi gabungan-gabungan dari fungsi / prosedur sebelumnya untuk membuat program utama, terdapat beberapa variable yaitu:

Nama Variabel	Penjelasan
int bufferSize	Variabel ini menyimpan ukuran buffer yang diberikan pemain
int maxPoint	Variabel ini menyimpan nilai maksimum yang ditemukan
int options	Variabel ini menyimpan pilihan options yang dipilih oleh pengguna.
long long runTime	Variabel ini menyimpan waktu esekusi dari algoritma
vector<vector<string>> playMatrix	Variable ini menyimpan matriks token yang dijadikan papan bermain untuk mencari buffer dengan point tertinggi

<code>vector<vector<string>> seqVec</code>	Variabel ini menyimpan sequence yang diberikan oleh pengguna yang nanti nya akan di string matching
<code>vector<int> rewardVec</code>	Variabel ini menyimpan point / reward dari tiap sequence
<code>vector<pair<int, int>> bestPathCoordinates</code>	Variable ini menyimpan pair titik koordinat dari tiap cell yang dilewati oleh path terbaik
<code>vector<string> bestPath</code>	Variabel ini menyimpan rangkaian token buffer yang memiliki point terbesar
<code>bool running</code>	Variabel ini menyatakan kondisi bermain, sedang berjalan atau tidak
<code>string fileName</code>	Variable yang menyimpan nama file
<code>string filePath</code>	Variabel yang menyimpan path dari file

Berikut adalah code yang terdapat pada main.cpp

```
int main(){
    int bufferSize, maxPoints, options;
    long long runTime;
    vector<vector<string>> playMatrix, seqVec;
    vector<int> rewardVec;
    vector<pair<int, int>> bestPathCoordinates;
    vector<string> bestPath;
    bool running;

    maxPoints = 0;
    running = true;
    while(running){
```

```

        bufferSize = 0;
        maxPoints = 0;
        runTime = 0;
        playMatrix.clear();
        seqVec.clear();
        rewardVec.clear();
        bestPathCoordinates.clear();
        bestPath.clear();
        cout << "-----" <<
endl;
        cout << "Cyberpunk 2077 Breach Protocol with Brute Force Algorithm" <<
endl;
        cout << "-----" <<
endl;
        cout << "Pilih opsi input:                " <<
endl;
        cout << "1. File (txt)" << endl;
        cout << "2. Command Line Interface" << endl;
        cout << "3. Exit" << endl << endl << ">> ";
        cin >> options;

        if(cin.fail()){
            cin.clear();
            cin.ignore(numeric_limits<std::streamsize>::max(), '\n');
            cout << "Input harus berupa angka! Silahkan coba lagi!" << endl;
            continue;
        }

        switch(options){
            case 1: {
                string fileName, filePath;
                cout << "Masukkan nama file, pastikan berada di folder test:
" << endl << endl << ">> ";
                cin >> fileName;
                filePath = "test/" + fileName;
                if(readFile(filePath, bufferSize, playMatrix, seqVec,
rewardVec)){
                    if(maxPoints > 0){
                        printOutput(maxPoints, bestPath, bestPathCoordinates,
runTime);
                        saveFileOptions(maxPoints, bestPath,
bestPathCoordinates, runTime);
                    }else{
                        cout << "Tidak ada solusi!!" << endl;
                        cout << "0ms" << endl;
                    }
                }
                break;
            }

```



```

    }
    case 2:
        readTerminal(playMatrix, seqVec, bufferSize, rewardVec);
        bruteForceSolution(playMatrix, bufferSize, seqVec, rewardVec,
maxPoints, bestPath, bestPathCoordinates, runTime);
        if(maxPoints > 0){
            printOutput(maxPoints, bestPath, bestPathCoordinates,
runTime);
            saveFileOptions(maxPoints, bestPath, bestPathCoordinates,
runTime);
        }else{
            cout << "Tidak ada solusi!!" << endl;
            cout << "0ms" << endl;
        }
        break;
    case 3:
        running = false;
        break;
    default:
        cout << "Pastikan input antara 1-3! Silahkan coba kembali" <<
endl;

        break;
    }
}
return 0;
}



```

Bab III

Testing Input Output

3.1 Input Dari File

Berikut adalah beberapa contoh input dari file:

test >  contoh.txt	PS C:\Users\Maximilian\Documents\Github\tucili_13522061> ./run.bat	test >  output1.txt
1 7	C:\Users\Maximilian\Documents\Github\tucili_13522061>g++ -o bin/main src/main.cpp	1 50
2 6 6	C:\Users\Maximilian\Documents\Github\tucili_13522061>bin/main.exe	2 7A BD 7A BD 1C BD 55
3 7A 55 E9 E9 1C 55	Cyberpunk 2077 Breach Protocol with Brute Force Algorithm	3 1, 1
4 55 7A 1C 7A E9 55	Pilih opsi input:	4 1, 4
5 55 1C 1C 55 E9 BD	1. File (txt)	5 3, 4
6 BD 1C 7A 1C 55 BD	2. Command Line Interface	6 3, 5
7 BD 55 BD 7A 1C 1C	3. Exit	7 6, 5
8 1C 55 55 7A 55 7A	>> 1	8 6, 3
9 3	Masukkan nama file, pastikan berada di folder test:	9 1, 3
10 BD E9 1C	>> contoh.txt	10
11 15	50	11 396ms
12 BD 7A BD	7A BD 7A BD 1C BD 55	12
13 20	1, 1	
14 BD 1C BD 55	1, 4	
15 30	3, 4	
	3, 5	
	6, 5	
	6, 3	
	3, 3	
	396ms	
	Apakah ingin menyimpan solusi? (y/n)	
	>>y	
	File output berhasil dibuat, silahkan cek di: test/output1.txt	

Gambar 1 Contoh input dari file, kanan isi contoh.txt, tengah luaran terminal, kiri luaran txt

<pre>test > ≡ contoh.txt 1 8 2 7 6 3 CD 1F BC TR AB CD TR 4 CD TR TR AB 1F BC CD 5 1F 1F BC AB CD BC BC 6 CD BC AB AB TR TR AB 7 1F 1F CD BC AB BC AB 8 CD TR TR BC 1F TR CD 9 4 10 TR AB CD BC 11 35 12 TR 1F AB 1F 13 40 14 BC CD BC 15 25 16 BC TR TR CD 1F 17 50</pre>	<pre>----- Cyberpunk 2077 Breach Protocol with Brute Force Algorithm ----- Pilih opsi input: 1. File (txt) 2. Command Line Interface 3. Exit >> 1 Masukkan nama file, pastikan berada di folder test: >> contoh.txt 75 BC CD BC TR TR CD 1F 3, 1 3, 5 4, 5 4, 1 7, 1 7, 2 5, 2 4560ms Apakah ingin menyimpan solusi? (y/n) >>y File output berhasil dibuat, silahkan cek di: test/output3.txt</pre>	<pre>test > ≡ output3.txt 1 75 2 BC CD BC TR TR CD 1F 3 3, 1 4 3, 5 5 4, 5 6 4, 1 7 7, 1 8 7, 2 9 5, 2 10 11 4560ms 12</pre>
---	---	---

Gambar 2 Contoh input dari file, kanan isi contoh.txt, tengah luaran terminal, kiri luaran txt

<pre>test > ≡ contoh.txt 1 10 2 6 6 3 BD 1C BD E9 55 BD 4 1C 7A 1C 1C 1C 55 5 1C 1C 1C E9 55 1C 6 7A 55 BD 7A 55 E9 7 55 7A 55 E9 55 7A 8 7A BD BD 55 1C 55 9 3 10 1C BD E9 11 45 12 7A BD 1C 13 15 14 7A 1C E9 55 15 50</pre>	<pre>----- Cyberpunk 2077 Breach Protocol with Brute Force Algorithm ----- Pilih opsi input: 1. File (txt) 2. Command Line Interface 3. Exit >> 1 Masukkan nama file, pastikan berada di folder test: >> contoh.txt 75 BC CD BC TR TR CD 1F 3, 1 3, 5 4, 5 4, 1 7, 1 7, 2 5, 2 4546ms Apakah ingin menyimpan solusi? (y/n) >>y File output berhasil dibuat, silahkan cek di: test/output4.txt</pre>	<pre>test > ≡ output4.txt 1 75 2 BC CD BC TR TR CD 1F 3 3, 1 4 3, 5 5 4, 5 6 4, 1 7 7, 1 8 7, 2 9 5, 2 10 11 4546ms 12</pre>
---	---	---

Gambar 3 Contoh input dari file, kanan isi contoh.txt, tengah luaran terminal, kiri luaran txt

<pre>test > ≡ contoh.txt 1 7 2 6 6 3 7A BD 1C 1C BD E9 4 7A 1C E9 7A E9 E9 5 1C 7A 7A 1C 7A 55 6 55 BD BD 7A 1C 55 7 55 1C 55 55 BD 55 8 E9 55 BD 1C 7A 1C 9 3 10 E9 1C 11 13 12 BD BD BD 13 32 14 55 55 1C 7A 15 INI SALAH</pre>	<pre>----- Cyberpunk 2077 Breach Protocol with Brute Force Algorithm ----- Pilih opsi input: 1. File (txt) 2. Command Line Interface 3. Exit >> 1 Masukkan nama file, pastikan berada di folder test: >> contoh.txt Error occurred during file operations: basic_ios::clear: iostream error</pre>
--	---

Gambar 4 Contoh input dari file, kanan isi contoh.txt, kiri luaran terminal

<pre>test > ≡ contoh.txt 1 7 2 1 1 3 7A 4 3 5 E9 1C 6 13 7 BD BD BD 8 32 9 55 55 1C 7A 10 35</pre>	<pre>----- Cyberpunk 2077 Breach Protocol with Brute Force Algorithm ----- Pilih opsi input: 1. File (txt) 2. Command Line Interface 3. Exit >> 1 Masukkkan nama file, pastikan berada di folder test: >> contoh.txt Tidak ada solusi!! 0ms</pre>
---	---

Gambar 5 Contoh input dari file, kanan isi contoh.txt, kiri luaran terminal

3.2 Input Dari CLI

Berikut adalah beberapa contoh input dari CLI:

<pre>----- Cyberpunk 2077 Breach Protocol with Brute Force Algorithm ----- Pilih opsi input: 1. File (txt) 2. Command Line Interface 3. Exit >> 2 Jumlah token unik: 5 Token: BD 1C 7A 55 E9 Ukuran buffers: 7 Ukuran matrix (row,col): 6 6 Jumlah sequence: 3 Ukuran maximal sequence: 4 Token Matrix: BD 1C 55 7A E9 BD E9 E9 1C E9 BD BD E9 55 1C E9 BD 1C E9 BD BD E9 55 7A 1C 55 BD 55 BD BD 1C BD E9 BD E9 7A Sequence1: 1C E9 E9 Rewards: 40 Sequence2: 1C E9 Rewards: 50 Sequence3: E9 E9 Rewards: 42 132 BD E9 1C E9 1C E9 E9 1, 1 1, 2 3, 2 3, 6 1, 6 1, 3 4, 3 397ms Apakah ingin menyimpan solusi? (y/n) >>Y File output berhasil dibuat, silahkan cek di: test/output2.txt</pre>	<pre>test > ≡ output2.txt 1 132 2 BD E9 1C E9 1C E9 E9 3 1, 1 4 1, 2 5 3, 2 6 3, 6 7 1, 6 8 1, 3 9 4, 3 10 11 397ms 12</pre>
---	---

Gambar 6 Contoh input dari CLI, kanan isi terminal, kiri luaran txt

```

File output berhasil dibuat, silahkan cek di: test/output4.txt
-----
Cyberpunk 2077 Breach Protocol with Brute Force Algorithm
-----
Pilih opsi input:
1. File (txt)
2. Command Line Interface
3. Exit

>> 2
Jumlah token unik: 6
Token: AB BC CD DE EF FG
Ukuran buffer: 5
Ukuran matrix (row,col): 6 8
Jumlah sequence: 5
Ukuran maximal sequence: 4

Token Matrix:
DE FG DE EF CD AB DE FG
BC CD FG CD EF DE BC BC
CD BC DE CD CD CD AB BC
CD CD CD DE EF AB EF DE
CD EF FG CD EF BC DE BC
FG AB CD BC FG FG CD AB

Sequence1: FG AB BC EF
Rewards: 10

Sequence2: DE BC CD
Rewards: 26

Sequence3: CD BC EF FG
Rewards: 14

Sequence4: FG CD AB
Rewards: 25

Sequence5: EF DE
Rewards: 22

48
FG EF DE BC CD
2, 1
2, 5
7, 5
7, 2
2, 2

55ms

Apakah ingin menyimpan solusi? (y/n)
>>Y
File output berhasil dibuat, silahkan cek di: test/output5.txt

```

```

test > ≡ output5.txt

1      48
2      FG EF DE BC CD
3      2, 1
4      2, 5
5      7, 5
6      7, 2
7      2, 2
8
9      55ms
10

```

Gambar 7 Contoh input dari CLI, kanan isi terminal, kiri luaran txt

```

Cyberpunk 2077 Breach Protocol with Brute Force Algorithm
-----
Pilih opsi input:
1. File (txt)
2. Command Line Interface
3. Exit

>> 2
Jumlah token unik: 4
Token: AB BC CD DF
Ukuran buffer: 10
Ukuran matrix (row,col): 5 9
Jumlah sequence: 3
Ukuran maximal sequence: 4

Token Matrix:
CD DF AB BC AB BC CD CD
CD DF CD DF AB CD AB AB DF
CD BC CD CD BC DF DF BC CD
DF DF BC CD BC AB DF BC DF
CD AB DF CD AB CD BC CD CD

Sequence1: AB DF CD
Rewards: 39

Sequence2: AB AB BC CD
Rewards: 48

Sequence3: DF BC AB BC
Rewards: 46

94
CD CD DF BC AB BC AB BC CD
1, 1
1, 2
4, 2
4, 1
7, 1
7, 5
5, 5
5, 1
6, 1
6, 2

135677ms

Apakah ingin menyimpan solusi? (y/n)
>>y
File output berhasil dibuat, silahkan cek di: test/output6.txt

```

```

test > ≡ output6.txt

1      94
2      CD CD DF BC AB BC AB AB BC CD
3      1, 1
4      1, 2
5      4, 2
6      4, 1
7      7, 1
8      7, 5
9      5, 5
10     5, 1
11     6, 1
12     6, 2
13
14     135677ms
15

```

Gambar 8 Contoh input dari CLI, kanan isi terminal, kiri luaran txt

```

-----
Cyberpunk 2077 Breach Protocol with Brute Force Algorithm
-----
Pilih opsi input:
1. File (txt)
2. Command Line Interface
3. Exit

>> 2
Jumlah token unik: INI JUGA SALAH!
Input tidak valid. Silahkan coba lagi!
Jumlah token unik: █

```

Gambar 9 Contoh input dari CLI

```

-----
Cyberpunk 2077 Breach Protocol with Brute Force Algorithm
-----
Pilih opsi input:
1. File (txt)
2. Command Line Interface
3. Exit

>> 2
Jumlah token unik: 3
Token: AB BC CD
Ukuran buffer: 2
Ukuran matrix (row,col): 1 1
Jumlah sequence: 2
Ukuran maximal sequence: 2

Token Matrix:
CD

Sequence1: BC BC
Rewards: 34

Sequence2: AB AB
Rewards: 38

Tidak ada solusi!!
0ms

```

Gambar 10 Contoh input dari CLI

Bab IV

Lampiran

Link Repository: https://github.com/riyorax/Tucil1_13522061

Checklist:

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 GUI		√