LAPORAN TUGAS KECIL 1

IF2211 - STRATEGI ALGORITMA

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

Dosen:

Ir. Rila Mandala, M.Eng., Ph.D.

Monterico Adrian, S.T., M.T.

Mahasiswa:

M. Zaidan Sa'dun R. (13522146)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER I TAHUN 2023/2024

DAFTAR ISI

BAB I

Algoritma *Brute Force*

Algoritma brute force merupakan metode penyelesaian masalah yang mengadopsi pendekatan langsung dengan mengeksplorasi semua solusi yang memungkinkan secara menyeluruh. Langkah-langkah implementasinya pada minigames breach protocol dalam game Cyberpunk 2077 adalah sebagai berikut:

1. Inisialisasi Matriks dan Sequence: Proses dimulai dengan mempersiapkan matriks yang merepresentasikan token-token dan sebuah sequence. Inisialisasi ini dapat dilakukan melalui masukan dari file atau dengan pembangkitan secara acak.
2. Eksplorasi Semua Kemungkinan: Algoritma brute force melakukan pencarian terhadap semua kemungkinan rute yang mungkin dimulai dari posisi awal. Proses pencarian ini menggunakan rekursi dan perulangan untuk mencoba semua kemungkinan rute.
3. Rekursi atau Perulangan: Selama proses pencarian, algoritma menggunakan langkah-langkah rekursif atau perulangan untuk mengeksplorasi semua kemungkinan langkah dari posisi saat ini. Misalnya, dari posisi (i, j), algoritma mencoba semua kemungkinan langkah ke sel-sel yang bersebelahan atau terhubung langsung.
4. Penandaan dan Penyimpanan Solusi: Setiap kali algoritma menemukan rute yang valid yang mencapai tujuan atau memenuhi kriteria tertentu, itu menandai rute tersebut dan mungkin menyimpannya untuk perbandingan lebih lanjut. Solusi-solusi yang ditemukan dapat disimpan dalam struktur data seperti array atau vektor.
5. Backtracking: Jika algoritma mengalami jalan buntu atau tidak dapat melanjutkan, maka dilakukan backtracking untuk kembali ke langkah sebelumnya dan mencoba rute atau langkah alternatif lainnya. Hal ini memastikan bahwa semua kemungkinan dieksplorasi dengan lengkap.
6. Penanganan Seluruh Matriks: Proses pencarian dilanjutkan hingga semua sel dalam matriks dieksplorasi dan semua kemungkinan rute yang valid telah diidentifikasi.

7. Output atau Penyimpanan Hasil: Akhirnya, algoritma menghasilkan output sesuai kebutuhan. Ini bisa berupa mencetak rute-rute yang ditemukan atau menyimpannya dalam struktur data untuk penggunaan lebih lanjut.

BAB II

Source Program

Pada program breach protocol ini saya menggunakan bahasa C++, berikut code nya

File input.cpp (berisi inputan user)

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <algorithm>

using namespace std;

struct Masukan {
    int bufferSize;
    int matrixHeight;
    int matrixWidth;
    int numberOfSequences;
    int maxSequenceSize;
    vector<vector<string>> matrix;
    vector<vector<string>> sequences;
    vector<int> rewards;
};

int findToken(const string &str) {
    stringstream ss(str);
    string word;
    int count = 0;

    while (ss >> word)
    {
        count++;
    }
    return count;
}
```

```
Masukan generateInput(int bufferSize, int matrixHeight, int matrixWidth, int
numberOfSequences, int maxSequenceSize, const vector<string>& token, int
jumlah_token_unik) {
    Masukan input;

    input.bufferSize = bufferSize;
    input.matrixHeight = matrixHeight;
    input.matrixWidth = matrixWidth;
    input.numberOfSequences = numberOfSequences;
    input.maxSequenceSize = maxSequenceSize;

    // Generate matrix
    input.matrix.resize(matrixHeight, vector<string>(matrixWidth));
    for (int i = 0; i < matrixHeight; i++) {
        for (int j = 0; j < matrixWidth; j++) {
            input.matrix[i][j] = token[rand() % jumlah_token_unik];
        }
    }

    // Generate sequences
    input.sequences.resize(numberOfSequences, vector<string>(maxSequenceSize));
    for (int i = 0; i < numberOfSequences; i++) {
        int seq_random = rand() % (maxSequenceSize - 2 + 1) + 2;
        for (int j = 0; j < seq_random; j++) {
            input.sequences[i][j] = token[rand() % jumlah_token_unik];
        }
    }

    // Generate rewards
    input.rewards.resize(numberOfSequences);
    for (int i = 0; i < numberOfSequences; i++) {
        input.rewards[i] = rand() % 100;
    }

    return input;
}

void printInputan(Masukan input) {
    cout << "\nData yang dihasilkan: \n\n";
    cout << "Buffer size: " << input.bufferSize << endl;
    cout << "Matrix Dimension: " << input.matrixHeight << "x" << input.matrixWidth <<
endl;
    cout << "\nMatrix: " << endl;
    for (int i = 0; i < input.matrixHeight; i++)
    {
        for (int j = 0; j < input.matrixWidth; j++)
        {
            cout << input.matrix[i][j] << " ";
        }
```

```
        cout << endl;
    }
    cout << "\nNumber of Sequences: " << input.numberOfSequences << endl;
    cout << "Sequence Length: " << input.maxSequenceSize << endl;
    for (int i = 0; i < input.numberOfSequences; i++)
    {
        cout << "Sequence " << i + 1 << ": ";
        for (int j = 0; j < input.sequences[i].size(); j++)
        {
            cout << input.sequences[i][j] << " ";
        }
        cout << "Reward: " << input.rewards[i] << endl;
    }
    cout << endl;
}

Masukan file() {
    Masukan input;
    string filename, line;
    int temp;

    cout << "Input file name (without .txt): ";
    cin >> filename;
    filename += ".txt";
    ifstream file(filename);

    while (!file.is_open())
    {
        cout << "File not found" << endl;
        cout << "Input file name (without .txt): ";
        cin >> filename;
        filename += ".txt";
        file.open(filename);
    }

    file >> input.bufferSize >> input.matrixWidth >> input.matrixHeight;

    input.matrix.resize(input.matrixHeight, vector<string>(input.matrixWidth));
    for (int i = 0; i < input.matrixHeight; i++)
    {
        for (int j = 0; j < input.matrixWidth; ++j)
        {
            file >> input.matrix[i][j];
        }
    }

    file >> input.numberOfSequences;
    getline(file, line);
    input.maxSequenceSize = 0;
```

```
    input.sequences.resize(input.numberOfSequences,
vector<string>(input.maxSequenceSize));
    for (int i = 0; i < input.numberOfSequences; i++)
    {
        getline(file, line);
        temp = findToken(line);
        if (temp > input.maxSequenceSize)
        {
            input.maxSequenceSize = temp;
            for (int i = 0; i < input.numberOfSequences; i++)
            {
                input.sequences[i].resize(input.maxSequenceSize);
            }
        }
        stringstream ss(line);
        string word;
        vector<string> words;
        int j = 0;
        while (ss >> word)
        {
            input.sequences[i][j] = word;
            j++;
        }
        getline(file, line);
        input.rewards.push_back(stoi(line));
    }

    file.close();
    return input;
}

Masukan cli() {
    Masukan input;
    string temp;
    int jumlah_token_unik;
    vector<string> token;

    cout << "Masukkan format CLI: " << endl;
    cin >> jumlah_token_unik;
    token.resize(jumlah_token_unik);
    for (int i = 0; i < jumlah_token_unik; ++i)
    {
        cin >> temp;
        token[i] = temp;
    }
    cin >> input.bufferSize;
    cin >> input.matrixHeight ;
    cin >> input.matrixWidth;
```

```
    cin >> input.numberOfSequences;
    cin >> input.maxSequenceSize;
    input = generateInput(input.bufferSize, input.matrixHeight, input.matrixWidth,
input.numberOfSequences, input.maxSequenceSize, token, jumlah_token_unik);
    return input;
}
```

File solver.cpp (berisi algoritma penyelesaian masalah)

```
#include "input.cpp"
#include <iostream>
#include <vector>
#include <utility>

using namespace std;

struct Location {
    int row;
    int col;

    Location(int r, int c) : row(r), col(c) {}
};

int calculateSequenceLength(Masukan input, int idx) {
    int length = 0;
    for (int i = 0; i < input.maxSequenceSize; i++) {
        if (input.sequences[idx][i] != "") {
            length++;
        }
    }
    return length;
}

int calculateScore(Masukan input, vector<string> tempRoute) {
    int score = 0;
    int size = tempRoute.size();
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < input.numberOfSequences; j++) {
            int seqLength = calculateSequenceLength(input, j);
            if (size - i >= seqLength) {
                int count = 0;
                for (int k = 0; k < seqLength; k++) {
                    if (tempRoute[i + k] == input.sequences[j][k]) {
                        count++;
                    }
                }
                if (count == seqLength) {
                    score += input.rewards[j];
```

```
            }
        }
    }
}

if (score == 0) {
    score = -1;
}

return score;
}

bool isSameSequence(vector<string> route, vector<string> correctRoute) {
    if (route.size() != correctRoute.size()) {
        return false;
    } else {
        bool isCorrect = true;
        for (int i = 0; i < correctRoute.size(); ++i) {
            if (route[i] != correctRoute[i]) {
                isCorrect = false;
                break;
            }
        }
        return isCorrect;
    }
}

// Global variables to store the maximum reward and corresponding route
int maxReward = 0;
vector<string> maxRoute;
vector<Location> maxRouteLocations; // Store the locations of the max route

void exploreRoutes(Masukan input, vector<vector<string>> &matrix, int row, int col,
vector<string> &route, vector<Location> &routeLocations, vector<vector<bool>>
&visited, int bufferSize, bool isVertical) {
    // Add current point to the route
    route.push_back(matrix[row][col]);
    routeLocations.push_back(Location(row, col));
    visited[row][col] = true;
    int tempReward = calculateScore(input, route);
    if (tempReward > maxReward) {
        maxReward = tempReward;
        maxRoute = route;
        maxRouteLocations = routeLocations; // Update the max route locations
    }

    // Base case: if the route size equals buffer
    if (route.size() == bufferSize) {
        // Backtrack
```

```
        route.pop_back();
        routeLocations.pop_back();
        visited[row][col] = false;
        return;
    }

    // Explore vertically
    if (isVertical) {
        for (int i = 0; i < input.matrixHeight; ++i) {
            if (!visited[i][col]) {
                exploreRoutes(input, matrix, i, col, route, routeLocations, visited, bufferSize,
!isVertical);
            }
        }
    }

    // Explore horizontally
    else {
        for (int j = 0; j < input.matrixWidth; ++j) {
            if (!visited[row][j]) {
                exploreRoutes(input, matrix, row, j, route, routeLocations, visited, bufferSize,
!isVertical);
            }
        }
    }

    // Backtrack
    route.pop_back();
    routeLocations.pop_back();
    visited[row][col] = false;
}
```

File main,cpp (dimana program dapat dijalankan)

```
#include "solver.cpp"
#include <iostream>
#include <string>
#include <chrono> // Include the chrono library for time measurement
using namespace std;
using namespace std::chrono; // Use the chrono namespace

int main() {
    string inputType;
    cout << "Input type: \n1. cli\n2. file .txt\nselect: ";
    cin >> inputType;
    Masukan input;
    while (inputType != "1" && inputType != "2") {
```

```
        cout << "Invalid input type." << endl;
        cout << "Input type: \n1. cli\n2. file .txt\nselect: ";
        cin >> inputType;
    }
    if (inputType == "1") {
        input = cli();
        printInputan(input);
    } else if (inputType == "2") {
        input = file();
        printInputan(input);
    }

    // Measure the processing time for counting
    auto start = high_resolution_clock::now(); // Get the current time before counting starts

    vector<string> route;
    vector<Location> routePoints;
    vector<vector<bool>> visited(input.matrixHeight, vector<bool>(input.matrixWidth,
false)); // Initialize visited matrix
    for (int i = 0; i < input.matrixWidth; ++i) {
        exploreRoutes(input, input.matrix, 0, i, route, routePoints, visited, input.bufferSize,
true);
    }

    auto stop = high_resolution_clock::now(); // Get the current time after counting finishes
    auto duration = duration_cast<milliseconds>(stop - start); // Calculate the duration of
counting

    cout << "Max sequence: ";
    for (int i = 0; i < maxRoute.size(); ++i){
        cout << maxRoute[i] << " ";
    }
    cout << endl << "Location: "<< endl;
    for (int i = 0; i < maxRoute.size(); ++i){
        cout << maxRouteLocations[i].col+1 <<","<<maxRouteLocations[i].row+1 << endl;
    }
    cout << "Reward: " << maxReward << endl;
    cout << "Processing time: " << duration.count() << " milliseconds" << endl; // Print the
processing time

    string n;
    cout << "\nApakah ingin menyimpan solusi? (y/n)" << endl;
    cin >> n;

    while (n != "y" && n != "n") {
        cout << "Pilihan tidak valid\n"<< endl;
        cout << "Apakah ingin menyimpan solusi? (y/n)" << endl;
        cin >> n;
    }
```

```
    if (n == "y") {
        string filename;
        string path = "../test/";
        cout << "Masukkan nama file tanpa (.txt): ";
        cin >> filename;
        filename += ".txt";
        ofstream file(path + filename);

        if (!file.is_open()) {
            cout << "Gagal membuka file untuk penulisan" << endl;
            return 1;
        }

        file << maxReward << endl;

        for (int i = 0; i < maxRoute.size(); ++i) {
            file << maxRoute[i] << " ";
        }

        file << endl;

        for (int i = 0; i < maxRouteLocations.size(); ++i) {
            file << maxRouteLocations[i].col +1 << ", " << maxRouteLocations[i].row + 1 <<
endl;
        }

        file << endl << duration.count() << " ms";
        file.close();
    }
    return 0;
}
```

BAB III

Input & Output

INPUT CLI

```
Input type:
1. cli
2. file .txt
select: 1
Masukkan format CLI:
5
BD 1C 7A 55 E9
7
6 6
3
4
```

INPUT 1 **OUTPUT 1**

```
Buffer size: 7
Matrix Dimension: 6x6

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

Number of Sequences: 3
Sequence Length: 4
Sequence 1: E9 BD E9 55 Reward: 11
Sequence 2: 7A 55   Reward: 53
Sequence 3: E9 1C   Reward: 68
```

```
Max sequence: E9 1C E9 1C BD E9 1C
Location:
3,1
3,4
4,4
4,3
5,3
5,1
1,1
Reward: 204
Processing time: 13291 milliseconds

Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file tanpa (.txt): output1
```

INPUT 2 **OUTPUT 2**

```
Buffer size: 6
Matrix Dimension: 5x5

Matrix:
1C 55 7A BD 1C
BD 7A 7A 7A BD
1C 1C 1C 55 1C
55 55 7A 55 BD
55 BD 7A 1C BD

Number of Sequences: 3
Sequence Length: 4
Sequence 1: 1C BD 7A  Reward: 99
Sequence 2: 55 7A 55 7A Reward: 35
Sequence 3: BD 55   Reward: 94
```

```
Max sequence: 1C BD 7A 1C BD 7A
Location:
1,1
1,2
4,2
4,5
2,5
2,2
Reward: 198
Processing time: 647 milliseconds
```

INPUT 3 **OUTPUT 3**

```
Buffer size: 6
Matrix Dimension: 3x4

Matrix:
1C 55 7A BD
1C BD 7A 7A
7A BD 1C 1C

Number of Sequences: 3
Sequence Length: 4
Sequence 1: 55 1C 55  Reward: 53
Sequence 2: 7A 55 BD  Reward: 92
Sequence 3: BD 7A   Reward: 82
```

```
Max sequence: BD 7A 7A 7A 55 BD
Location:
4,1
4,2
3,2
3,1
2,1
2,2
Reward: 174
Processing time: 26 milliseconds
```

INPUT FILE

```
Input type:
1. cli
2. file .txt
select: 2
Input file name (without .txt): input
```

INPUT 4 **OUTPUT 4**

```
Buffer size: 9
Matrix Dimension: 6x3

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

Number of Sequences: 3
Sequence Length: 4
Sequence 1: 1C 1C   Reward: 18
Sequence 2: 7A 55   Reward: 95
Sequence 3: 7A 1C 1C  Reward: 47
```

```
Max sequence: E9 7A 55 7A 1C 1C 7A 1C 1C
Location:
3,1
3,3
2,3
2,1
1,1
1,6
3,6
3,5
1,5
Reward: 225
Processing time: 2510 milliseconds
```

INPUT 5 **OUTPUT 5**

```
Buffer size: 4
Matrix Dimension: 2x1

Matrix:
1C
1C

Number of Sequences: 2
Sequence Length: 3
Sequence 1: BD 1C  Reward: 62
Sequence 2: BD BD  Reward: 64
```

OUTPUT TIDAK MUNGKIN

INPUT 6 **OUTPUT 6**

```
Buffer size: 5
Matrix Dimension: 5x3

Matrix:
1C 1C BD
BD 1C BD
BD BD BD
BD 1C 1C
1C 1C 1C

Number of Sequences: 3
Sequence Length: 4
Sequence 1: 1C BD 1C BD Reward: 21
Sequence 2: BD BD   Reward: 16
Sequence 3: BD BD   Reward: 18
```

```
Max sequence: BD BD BD BD BD
Location:
3,1
3,2
1,2
1,3
2,3
Reward: 136
Processing time: 27 milliseconds
```

PRANALA

Link GITHUB:

zaidanav/Tucil1_13522146 (github.com)

Tabel kebenaran:

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membacamasukan 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		✓