

LAPORAN TUGAS KECIL 01

IF2211 STRATEGI ALGORITMA

“Penyelesaian Permainan Breach Protocol dengan Algoritma *Brute Force*”



Disusun oleh:

Novelya Putri Ramadhani K-02 13522096

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023-2024

DAFTAR ISI

DAFTAR ISI	2
BAB 1	3
BAB 2	4
BAB 3	5
BAB 4	13
BAB 5	19

BAB 1

DESKRIPSI MASALAH

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.

Aturan permainan Breach Protocol antara lain:

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

(Dikutip dari: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Tucil1-2024.pdf>)

BAB 2

TEORI SINGKAT

2.1 Algoritma Brute Force

Algoritma yang digunakan untuk menemukan solusi paling optimal dari permainan Breach Protocol adalah algoritma *brute force*.

Langkah-langkah algoritma yang digunakan dalam menyelesaikan permainan tersebut adalah sebagai berikut:

- 1. Mencari semua kemungkinan solusi**

Program menghasilkan semua kemungkinan solusi yang memiliki panjang sama dengan *buffer_size*. Solusi memenuhi ketentuan bahwa token pertama dimulai dari posisi baris paling atas matriks dan dilanjutkan dengan bergerak vertikal, horizontal, vertikal, horizontal (bergantian) hingga buffer penuh.

- 2. Mengecek sebuah sequence di sebuah solusi**

Untuk setiap solusi, program akan mengecek apakah suatu sequence tertentu ada di solusi tersebut. Jika iya, program akan mengembalikan nilai *true* dan menyimpan posisi token pertama dan terakhir dari sequence di buffer.

- 3. Mengevaluasi semua sequence yang ada di sebuah solusi**

Untuk setiap solusi, akan dicek sequence apa saja yang berada pada solusi tersebut dan total *reward*-nya disimpan. Kemudian, nilai posisi token terakhir tiap sequence akan dibandingkan dan solusi akan dipotong agar token terakhirnya adalah token terakhir dari sequence yang memenuhi.

- 4. Mendapatkan solusi optimal**

Terakhir, program akan memeriksa semua solusi yang telah dievaluasi dan membandingkan total *reward* juga panjang solusi. Solusi optimal adalah yang total *reward*-nya terbesar dengan panjang solusi terkecil.

BAB 3

IMPLEMENTASI PROGRAM

3.1 Membaca input

a. Input dari berkas

```
vector<string> readFromFile(string filename){
    ifstream file(filename);
    string line;
    vector<string> lines;

    if (file.is_open()){
        while (getline(file, line)){
            lines.push_back(line);
        }
        file.close();
    } else{
        cout << "Unable to open file";
    }

    return lines;
}
```

b. Input dari CLI

```
Data inputFromCLI(){
    int num_unique_tokens;
    cout << "\nEnter number of unique tokens: ";
    cin >> num_unique_tokens;

    vector<string> tokens(num_unique_tokens);
    cout << "Enter unique tokens: ";
    for (int i = 0; i < num_unique_tokens; ++i){
        cin >> tokens[i];
    }

    int buffer_size;
    cout << "Enter buffer size: ";
    cin >> buffer_size;
```

```

int matrix_width, matrix_height;
cout << "Enter matrix width and height: ";
cin >> matrix_width >> matrix_height;

int num_sequences;
cout << "Enter number of sequences: ";
cin >> num_sequences;

int max_sequence_length;
cout << "Enter max sequence length: ";
cin >> max_sequence_length;

Data gameData = generateGame(num_unique_tokens, tokens,
buffer_size, matrix_width, matrix_height, num_sequences,
max_sequence_length);

return gameData;
}

```

3.2 Menyimpan data

a. Input dari berkas

```

Data readDataFromFile(string filename){
    vector<string> lines = readFromFile(filename);
    Data data;
    int line_number = 1;
    for (const auto& line : lines){
        if (line_number == 1){
            data.buffer_size = stoi(line);
        } else if (line_number == 2){
            istringstream iss(line);
            iss >> data.matrix_width >> data.matrix_height;
        } else if (3 <= line_number && line_number <=
data.matrix_height + 2){
            istringstream iss(line);
            Token token;
            vector<Token> row;
            int col = 0;
            while (iss >> token.value){

```

```

        token.row = line_number - 3;
        token.col = col++;
        row.push_back(token);
    }
    data.matrix.push_back(row);
} else if (line_number == data.matrix_height + 3){
    data.num_of_seq = stoi(line);
} else{
    if (data.matrix_height % 2 == 0){
        static Sequence seq;
        if (line_number % 2 == 0){
            istringstream iss(line);
            Token token;
            while (iss >> token.value){
                seq.sequence.push_back(token);
            }
        } else{
            seq.reward = stoi(line);
            data.sequences.push_back(seq);
            seq = Sequence();
        }
    } else{
        static Sequence seq;
        if (line_number % 2 != 0){
            istringstream iss(line);
            Token token;
            while (iss >> token.value){
                seq.sequence.push_back(token);
            }
        } else{
            seq.reward = stoi(line);
            data.sequences.push_back(seq);
            seq = Sequence();
        }
    }
}

line_number++;
}

return data;
}

```

b. Input dari CLI

```
Data generateGame(int num_unique_tokens, vector<string>& tokens, int
buffer_size, int matrix_width, int matrix_height, int num_sequences,
int max_sequence_length){
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(0, num_unique_tokens - 1);
    uniform_int_distribution<> dis_seq(2, max_sequence_length);
    uniform_int_distribution<> dis_reward(1, 100);

    Data gameData;
    gameData.buffer_size = buffer_size;
    gameData.matrix_width = matrix_width;
    gameData.matrix_height = matrix_height;
    gameData.num_of_seq = num_sequences;

    gameData.matrix.resize(matrix_height, vector<Token>(matrix_width));
    for (int i = 0; i < matrix_height; ++i){
        for (int j = 0; j < matrix_width; ++j){
            gameData.matrix[i][j].value = tokens[dis(gen)];
            gameData.matrix[i][j].row = i;
            gameData.matrix[i][j].col = j;
        }
    }

    gameData.sequences.resize(num_sequences);
    for (int i = 0; i < num_sequences; ++i){
        int sequence_length = dis_seq(gen);
        gameData.sequences[i].sequence.resize(sequence_length);
        for (int j = 0; j < sequence_length; ++j){
            gameData.sequences[i].sequence[j].value = tokens[dis(gen)];
        }
        gameData.sequences[i].reward = dis_reward(gen);
    }

    return gameData;
}
```

3.3 Mencari semua kemungkinan solusi

```
vector<vector<Token>> generateAllPossibleSolutions(Data& data){
```



```

    vector<vector<Token>> solutions;

    function<void(vector<Token>, int, int, bool)> generateSolutions =
[&](vector<Token> current_solution, int current_row, int current_col, bool
is_horizontal){
        if (current_solution.size() == data.buffer_size){
            solutions.push_back(current_solution);
            return;
        }
        if (is_horizontal){
            for (int i = 0; i < data.matrix_width; i++){
                if (!data.matrix[current_row][i].is_selected){
                    data.matrix[current_row][i].is_selected = true;
                    data.matrix[current_row][i].position_in_buffer =
current_solution.size();
                    current_solution.push_back(data.matrix[current_row][i]);
                    generateSolutions(current_solution, current_row, i,
false);

                    current_solution.pop_back();
                    data.matrix[current_row][i].is_selected = false;
                }
            }
        } else{
            for (int i = 0; i < data.matrix_height; i++){
                if (!data.matrix[i][current_col].is_selected){
                    data.matrix[i][current_col].is_selected = true;
                    data.matrix[i][current_col].position_in_buffer =
current_solution.size();
                    current_solution.push_back(data.matrix[i][current_col]);
                    generateSolutions(current_solution, i, current_col, true);
                    current_solution.pop_back();
                    data.matrix[i][current_col].is_selected = false;
                }
            }
        }
    };

    generateSolutions({}, 0, 0, true);

    return solutions;
}

```

3.4 Mengecek sebuah sequence di sebuah solusi

```
bool checkSequenceInSolution(vector<Token>& solution, Sequence& sequence){
    for (int i = 0; i <= solution.size() - sequence.sequence.size(); i++){
        bool sequenceFound = true;
        for (int j = 0; j < sequence.sequence.size(); j++){
            if (solution[i + j].value != sequence.sequence[j].value){
                sequenceFound = false;
                break;
            }
        }
        if (sequenceFound){
            sequence.in_buffer = true;
            sequence.first_token_position_in_buffer = i;
            sequence.last_token_position_in_buffer = i +
sequence.sequence.size() - 1;
            return true;
        }
    }
    return false;
}
```

3.5 Mengevaluasi semua sequence yang ada di sebuah solusi

```
Solution evaluateSolutionAndCalculateReward(vector<Token>& solution,
vector<Sequence>& sequences){
    Solution sol;
    sol.solution = solution;
    for (auto& sequence : sequences){
        if (checkSequenceInSolution(solution, sequence)){
            sol.reward += sequence.reward;
            if (sol.last_token_position_in_buffer <
sequence.last_token_position_in_buffer){
                sol.last_token_position_in_buffer =
sequence.last_token_position_in_buffer;
            }
        }
    }
    if (sol.last_token_position_in_buffer != -1){
        sol.solution.erase(sol.solution.begin() +
sol.last_token_position_in_buffer + 1, sol.solution.end());
    } else{
        sol.last_token_position_in_buffer = sol.solution.size();
    }
}
```

```

    }
    return sol;
}

```

3.6 Mendapatkan solusi optimal

```

Solution getOptimalSolution(vector<vector<Token>>& solutions, Data& data){
    Solution optimal_solution;
    optimal_solution.last_token_position_in_buffer = data.buffer_size;
    for (auto& solution : solutions){
        Solution sol = evaluateSolutionAndCalculateReward(solution,
data.sequences);
        if (sol.reward > optimal_solution.reward){
            optimal_solution = sol;
        } else if (sol.reward == optimal_solution.reward &&
sol.last_token_position_in_buffer <
optimal_solution.last_token_position_in_buffer){
            optimal_solution = sol;
        }
    }
    if (optimal_solution.last_token_position_in_buffer == data.buffer_size){
        optimal_solution.solution = {};
    }
    return optimal_solution;
}

```

3.7 Menampilkan output

```

void displaySolution(const Solution& solution){
    cout << "\n----- OPTIMAL SOLUTION ----- \n";
    cout << "Reward: " << solution.reward << endl;
    cout << "Solution: ";
    printArray(solution.solution);
    cout << "Coordinates:\n";
    for (const auto& token : solution.solution){
        cout << token.col + 1 << ", " << token.row + 1 << endl;
    }
}

```

3.8 Menyimpan output ke berkas

```

void saveSolutionToFile(const Solution& solution){
    cout << "Enter filename: ";
}

```

```

string filename;
cin >> filename;
string filepath = "../test/" + filename;
ofstream file(filepath);
if (file.is_open()){
    file << "Reward: " << solution.reward << endl;
    file << "Solution: ";
    for (const auto& token : solution.solution){
        file << token.value << " ";
    }
    file << "\nCoordinates:\n";
    for (const auto& token : solution.solution){
        file << token.col + 1 << ", " << token.row + 1 << endl;
    }
    file.close();
    cout << "\nSolution saved to " << filepath << endl;
    cout << "\nThanks for playing. Goodbye!" << endl;
} else{
    cout << "\nUnable to open file";
}
}

```

BAB 4

EKSPERIMEN

4.1 Input dari berkas

4.1.1

```
src > input.txt
You, 8 seconds ago | 1 author (You)
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 BD E9 1C
11 15
12 BD 7A BD
13 20
14 BD 1C BD 55
15 30
```

```
Welcome to 096 Breach Protocol Game!

Choose input mode:
1. Input through Command Line Interface (CLI)
2. Input through file
Enter your choice (1 or 2): 2

Enter filename: input.txt

----- OPTIMAL SOLUTION -----
Reward: 50
Solution: 7A BD 7A BD 1C BD 55
Coordinates:
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

Execution time: 121 ms

Do you want to save the solution to a .txt file? (y/n): y
Enter filename: sol11.txt

Solution saved to ../test/sol11.txt

Thanks for playing. Goodbye!
```

4.1.2

```
src > input.txt
You, 1 second ago | 1 author (You)
1 9
2 6 5
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 3
9 BD 7A 1C
10 10
11 BD 7A 55
12 20
13 BD 1C BD 55
14 30
```

```
Welcome to 096 Breach Protocol Game!

Choose input mode:
1. Input through Command Line Interface (CLI)
2. Input through file
Enter your choice (1 or 2): 2

Enter filename: input.txt

----- OPTIMAL SOLUTION -----
Reward: 50
Solution: E9 BD 7A 55 BD 1C BD 55
Coordinates:
3, 1
3, 5
4, 5
4, 3
6, 3
6, 5
1, 5
1, 2

Execution time: 1797 ms

Do you want to save the solution to a .txt file? (y/n): y
Enter filename: sol12.txt

Solution saved to ../test/sol12.txt

Thanks for playing. Goodbye!
```

4.1.3

```
src > input.txt
You, 1 second ago | 1 author (You)
1 5
2 4 5
3 7A 55 E9 E9
4 55 7A 1C 7A
5 55 1C 1C 55
6 BD 1C 7A 1C
7 BD 55 BD 7A
8 3
9 BD 7A 1C
10 10
11 BD 7A 55
12 20
13 1C BD 55
14 30 You, 4 days ago • fea
```

Welcome to 096 Breach Protocol Game!

Choose input mode:

1. Input through Command Line Interface (CLI)

2. Input through file

Enter your choice (1 or 2): 2

Enter filename: input.txt

----- OPTIMAL SOLUTION -----

Reward: 30

Solution: 55 1C BD 55

Coordinates:

2, 1

2, 4

1, 4

1, 2

Execution time: 4 ms

Do you want to save the solution to a .txt file? (y/n): y

Enter filename: sol13.txt

Solution saved to ../test/sol13.txt

Thanks for playing. Goodbye!

4.2 Input dari CLI

4.2.1

```
Welcome to 096 Breach Protocol Game!

Choose input mode:
1. Input through Command Line Interface (CLI)
2. Input through file
Enter your choice (1 or 2): 1

Enter number of unique tokens: 3
Enter unique tokens: AA BB CC
Enter buffer size: 4
Enter matrix width and height: 4 4
Enter number of sequences: 3
Enter max sequence length: 3

Matrix:
CC BB BB CC
CC BB BB BB
BB BB BB CC
BB AA AA CC

Sequences and Rewards:
CC CC : 84
CC AA : 63
CC AA BB : 9

----- OPTIMAL SOLUTION -----
Reward: 156
Solution: CC CC AA BB
Coordinates:
4, 1
4, 4
2, 4
2, 1

Execution time: 1 ms

Do you want to save the solution to a .txt file? (y/n): y
Enter filename: sol21.txt

Solution saved to ../test/sol21.txt

Thanks for playing. Goodbye!
```


4.2.2

```
Welcome to 096 Breach Protocol Game!

Choose input mode:
1. Input through Command Line Interface (CLI)
2. Input through file
Enter your choice (1 or 2): 1

Enter number of unique tokens: 4
Enter unique tokens: AA BB CC DD
Enter buffer size: 5
Enter matrix width and height: 5 6
Enter number of sequences: 4
Enter max sequence length: 4

Matrix:
DD AA BB BB DD
BB BB BB AA BB
AA BB DD AA BB
DD AA AA CC BB
AA CC BB CC BB
AA DD CC AA BB

Sequences and Rewards:
CC AA DD DD : 4
BB BB AA : 37
CC BB : 50
AA AA CC : 49

----- OPTIMAL SOLUTION -----
Reward: 99
Solution: AA AA CC BB
Coordinates:
2, 1
2, 4
4, 4
4, 1

Execution time: 17 ms

Do you want to save the solution to a .txt file? (y/n): y
Enter filename: sol22.txt

Solution saved to ../test/sol22.txt

Thanks for playing. Goodbye!
PS: C:\Users\Lenovo\Project\Tuxi11_12522006\src>
```

4.2.3

```
Welcome to 096 Breach Protocol Game!

Choose input mode:
1. Input through Command Line Interface (CLI)
2. Input through file
Enter your choice (1 or 2): 1

Enter number of unique tokens: 4
Enter unique tokens: AA BB CC DD
Enter buffer size: 7
Enter matrix width and height: 6 6
Enter number of sequences: 4
Enter max sequence length: 4

Matrix:
CC CC BB AA DD CC
CC BB CC AA DD BB
AA BB AA BB BB DD
BB DD CC DD BB DD
BB CC AA CC CC DD
AA BB AA AA CC CC

Sequences and Rewards:
AA AA DD BB : 75
AA CC : 86
AA AA BB DD : 67
BB BB DD DD : 42

----- OPTIMAL SOLUTION -----
Reward: 161
Solution: AA AA DD BB AA CC
Coordinates:
4, 1
4, 2
5, 2
5, 3
1, 3
1, 1

Execution time: 139 ms

Do you want to save the solution to a .txt file? (y/n): y
Enter filename: sol23.txt

Solution saved to ../test/sol23.txt

Thanks for playing. Goodbye!
```

BAB 5

LAMPIRAN

5.1 Link Repository

Link repository berisi kode program untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut:

Link : https://github.com/novelxv/Tucil1_13522096

5.2 Tabel Checkpoint Program

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		✓