

LAPORAN TUGAS KECIL
PENYELESAIAN MINIGAME BREACH PROTOCOL PADA
CYBERPUNK 2077 DENGAN ALGORITMA BRUTE FORCE
IF2211 STRATEGI ALGORITMA



Disusun oleh:

Ikhwan Al Hakim

(13522147)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023/2024

BAB I

DESKRIPSI MASALAH

Breach Protocol adalah salah satu minigame yang terdapat dalam permainan video Cyberpunk 2077. Minigame ini menghadirkan tantangan pemecahan kode dan hacking yang memungkinkan pemain untuk meretas sistem keamanan, mengakses informasi sensitif, atau mengambil alih kendali perangkat elektronik di dalam dunia game.

Algoritma brute force adalah metode pencarian solusi dengan menguji semua kemungkinan secara berurutan hingga menemukan solusi yang benar. Dalam konteks Breach Protocol pada permainan Cyberpunk 2077, algoritma brute force dapat digunakan untuk mencoba setiap kemungkinan kombinasi kode secara berturut-turut sampai kode yang benar ditemukan.

Dalam Breach Protocol, pemain diberi serangkaian kode yang terdiri dari simbol-simbol yang harus diurutkan dengan benar untuk meretas sistem. Jika pemain tidak memiliki informasi tambahan atau pola yang jelas untuk diikuti, mereka mungkin akan menggunakan algoritma brute force untuk mencoba semua kemungkinan kombinasi kode.

Namun, menggunakan algoritma brute force dalam Breach Protocol bisa menjadi strategi yang tidak efisien. Seiring dengan meningkatnya panjang kode, jumlah kemungkinan kombinasi yang harus diuji akan meningkat secara eksponensial. Ini bisa memakan waktu lama dan mungkin tidak praktis dalam situasi di mana pemain memiliki batas waktu atau risiko terdeteksi.

BAB II

TEORI SINGKAT

I. Breach Protocol

A. Pengertian Breach Protocol

Breach Protocol adalah sebuah *minigame* di Cyberpunk 2077 yang mensimulasikan proses memecahkan kode keamanan untuk mendapatkan akses ke sistem dalam dunia game tersebut. Dalam *minigame* ini, pemain dihadapkan pada sebuah grid yang terdiri dari rangkaian karakter atau simbol. Tugas pemain adalah menyusun serangkaian kode yang memungkinkan mereka untuk memperoleh akses ke berbagai informasi rahasia atau mengambil alih kontrol sistem.

B. Komponen pada Breach Protocol

1. Token

Token adalah karakter alfanumerik (seperti E9, BD, dan 55) yang berfungsi sebagai elemen pada matriks.

2. Matriks

Matriks adalah kumpulan dari token yang akan dipilih untuk menyusun urutan kode.

3. Sekuens

Sebuah rangkaian yang terdiri dari dua atau lebih token, memiliki bobot, dan harus dicocokkan pada matriks.

4. Buffer

Jumlah maksimal token yang dapat disusun secara sekuensial dari matriks.

II. Brute Force

Brute force adalah pendekatan sederhana dalam pemecahan masalah komputasi yang secara sistematis mencoba semua kemungkinan solusi secara berurutan untuk menemukan solusi optimal. Metode ini umumnya digunakan ketika ukuran masalah relatif kecil atau ketika tidak ada metode solusi yang lebih efisien.

Konsepnya adalah untuk mencoba semua kemungkinan solusi yang mungkin secara langsung, tanpa mempertimbangkan apakah solusi tersebut efisien atau tidak. Meskipun sederhana, algoritma brute force sering kali membutuhkan waktu komputasi yang sangat lama karena jumlah percobaan yang besar.

Dalam beberapa kasus, algoritma brute force mungkin merupakan satu-satunya cara untuk menyelesaikan masalah. Namun, dalam banyak kasus lainnya, algoritma ini tidak praktis karena waktu eksekusi yang tinggi. Oleh karena itu, sering kali sebaiknya dicari metode solusi lain yang lebih efisien.

BAB III IMPLEMENTASI PUSTAKA

Pada tugas kali ini, penulis menggunakan bahasa C++. Di dalam *root* folder, terdapat tiga folder yaitu:

1. bin

Berisi file *executable* yang bisa dijalankan di sistem operasi Windows dan Linux.

2. doc

Berisi file laporan ini.

3. src

Folder ini berisi program utama untuk tugas ini. Terdapat file *main.cpp* yang berfungsi sebagai *main driver* program. Berikut adalah isi dari file tersebut:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>
#include <time.h>
#include <memory>
#include "io/io.h"
using namespace std;

bool turnCheck(int buffer, vector<pair<int, int>> input) {
    for (int i = 1; i < buffer; i++) {
        if (i % 2 != 0) {
            if (input[i].second != input[i-1].second || input[i].first == input[i-1].first) {
                return false;
            }
        }
        else if (i % 2 == 0) {
            if (input[i].first != input[i-1].first || input[i].second == input[i-1].second) {
                return false;
            }
        }
    }

    if (input[0].first != 0) {
        return false;
    }

    return true;
}

bool checkDone(int *seqPointer, Sequence *seq, int idx) {
    return seqPointer[idx] == (seq[idx].length);
}

int calculateReward(int buffer, vector<pair<int, int>> input, int width, int height, string *matrix,
Sequence *seq, int seqAmount, int *seqPointer) {
    bool valid = turnCheck(buffer, input);

    if (!valid) {
        return 0;
    }

    int result = 0;
    for (int i = 0; i < buffer; i++) {
        for (int a = 0; a < seqAmount; a++) {
            if (matrix[input[i].first * width + input[i].second] == seq[a].set[seqPointer[a]]) {
                seqPointer[a]++;
            }
        }
    }
}
```

```

    }
    else {
        seqPointer[a] = 0;
    }

    if (checkDone(seqPointer, seq, a)) {
        result += seq[a].reward;
        seqPointer[a] = 0;
        // cout << "sequence yg ngasih nilai tuh " << a << endl;
        // cout << "current result " << result << endl;
        // cout << "current sequence ";
        // for (int i = 0; i < buffer; i++) {
        //     cout << matrix[input[i].first * width + input[i].second] << " ";
        // }
        // cout << endl << endl;
    }
}

// debug begin
// if (coor[0].x == 0 && coor[0].y == 0 && coor[1].x == 0 && coor[1].y == 3 && coor[2].x == 2 &&
coor[2].y == 3 && coor[3].x == 2 && coor[3].y == 4 && coor[4].x == 5 && coor[4].y == 4 && coor[5].x
== 5 && coor[5].y == 3 && coor[6].x == 4 && coor[6].y == 3) {
    // if (coor[0].x == 0 && coor[0].y == 0 && coor[1].x == 0 && coor[1].y == 3) {
    //     cout << input << endl;
    // }
    // debug end
    for (int i = 0; i < seqAmount; i++) {
        seqPointer[i] = 0;
    }

    return result;
}

vector<vector<pair<int, int>>> generateCombinations(int width, int height, int buffer_size,
pair<int, int> start={0,0}, bool is_horizontal=true) {
    int i = start.first;
    int j = start.second;

    vector<vector<int>> matrix(height, vector<int>(width));

    // Fill the matrix with values using nested loops
    int count = 1;
    for (int i = 0; i < height; ++i) {
        for (int j = 0; j < width; ++j) {
            matrix[i][j] = count++;
        }
    }

    if (buffer_size == 1) {
        if (is_horizontal) {
            vector<vector<pair<int, int>>> paths;
            for (int new_j = 0; new_j < matrix[0].size(); ++new_j) {
                if (new_j != j) {
                    paths.push_back({i, new_j});
                }
            }
            return paths;
        } else {
            vector<vector<pair<int, int>>> paths;
            for (int new_i = 0; new_i < matrix.size(); ++new_i) {
                if (new_i != i) {
                    paths.push_back({new_i, j});
                }
            }
            return paths;
        }
    } else {
        vector<vector<pair<int, int>>> paths;
        if (is_horizontal) {
            for (int new_j = 0; new_j < matrix[0].size(); ++new_j) {
                if (new_j != j) {
                    auto smaller_paths = generateCombinations(width, height, buffer_size - 1, {i,
new_j}, false);
                    for (auto path : smaller_paths) {
                        path.insert(path.begin(), {i, new_j});
                        paths.push_back(path);
                    }
                }
            }
        } else {
            for (int new_i = 0; new_i < matrix.size(); ++new_i) {
                if (new_i != i) {
                    auto smaller_paths = generateCombinations(width, height, buffer_size - 1, {new_i, j}, false);
                    for (auto path : smaller_paths) {
                        path.insert(path.begin(), {new_i, j});
                        paths.push_back(path);
                    }
                }
            }
        }
        return paths;
    }
}

```

```

    }
    }
    } else {
        for (int new_i = 0; new_i < matrix.size(); ++new_i) {
            if (new_i != i) {
                auto smaller_paths = generateCombinations(width, height, buffer_size - 1,
{new_i, j}, true);
                for (auto path : smaller_paths) {
                    path.insert(path.begin(), {new_i, j});
                    paths.push_back(path);
                }
            }
        }
    }
    return paths;
}
}

void fileOption() {
    string filename;
    cout << "Enter the file name (without format): ";
    cin >> filename;
    int buffer, width, height, seqAmount;
    string *matrix;
    Sequence *seq;
    readinput(filename, &buffer, &width, &height, &matrix, &seq, &seqAmount);

    unique_ptr<int[]> seqPointer(new int[seqAmount]);
    for (int i = 0; i < seqAmount; i++) {
        seqPointer[i] = 0;
    }

    clock_t start = clock();

    vector<vector<pair<int, int>>> combinations = generateCombinations(width, height, buffer);

    int max = 0, temp;
    vector<pair<int, int>> resultCoor;
    for (const auto &combination : combinations) {
        temp = calculateReward(buffer, combination, width, height, matrix, seq, seqAmount,
seqPointer.get());
        if (temp > max) {
            max = temp;
            resultCoor = combination;
        }
    }

    clock_t stop = clock();
    double timeTaken = (double)(stop - start) / CLOCKS_PER_SEC;
    cout << endl << "Done!" << endl;
    cout << "Time taken: " << timeTaken * 1000 << " ms" << endl;
    cout << "Max value: " << max << endl;

    if (max == 0) {
        cout << "There is no buffer solution because the max value is 0." << endl;
    }
    else {
        cout << "Buffer solution: ";
        for (int i = 0; i < buffer; i++) {
            cout << matrix[resultCoor[i].first * width + resultCoor[i].second] << " ";
        }
        cout << endl;
        cout << "Buffer solution's coordinate: ";
        for (int i = 0; i < buffer; i++) {
            cout << "(" << resultCoor[i].second << ", " << resultCoor[i].first << ") ";
        }
        cout << endl;
    }
    cout << endl;

    char yOrN;
    cout << "Do you want to save the solution into a file? (Y/N)" << endl;
    cin >> yOrN;
    if (yOrN == 'Y' || yOrN == 'y') {
        cout << "Enter the filename (without format): ";
        cin >> filename;
    }
}

```

```

        saveToFile(filename, buffer, height, width, matrix, seqAmount, seq, timeTaken, max,
resultCoor);
    }
}

void cliOption() {
    int n, buffer, width, height, seqAmount, intTemp;
    vector<string> token;
    string strTemp;
    cout << "Enter the number of token types: ";
    cin >> n;
    cout << "Enter the tokens:\nNote: The length must be 2." << endl;
    for (int i = 0; i < n; i++) {
        cin >> strTemp;
        token.push_back(strTemp);
    }
    cout << "Enter the buffer size: ";
    cin >> buffer;
    cout << "Enter the matrix's width: ";
    cin >> width;
    cout << "Enter the matrix's height: ";
    cin >> height;
    cout << "Enter the amount of sequences: ";
    cin >> seqAmount;
    cout << "Enter the maximum length of the sequence: ";
    cin >> n;

    srand(time(0));
    string* matrix = new string[width * height];
    for (int i = 0; i < height; ++i) {
        for (int j = 0; j < width; ++j) {
            int randomIndex = rand() % token.size();
            matrix[i * width + j] = token[randomIndex];
        }
    }

    Sequence *seq = new Sequence[seqAmount];
    for (int i = 0; i < seqAmount; i++) {
        int randomLength = 1 + rand() % n;
        int randomReward = 1 + rand() % 100;
        seq[i].set = new string[randomLength];
        for (int j = 0; j < randomLength; j++) {
            int randomIndex = rand() % token.size();
            seq[i].set[j] = token[randomIndex];
        }
        seq[i].reward = randomReward;
        seq[i].length = randomLength;
    }

    cout << endl << "Generated Matrix:" << endl;
    for (int i = 0; i < height; ++i) {
        for (int j = 0; j < width; ++j) {
            cout << matrix[i * width + j] << " ";
        }
        cout << endl;
    }

    cout << endl << "Generated sequence:" << endl;
    for (int i = 0; i < seqAmount; i++) {
        for (int j = 0; j < seq[i].length; j++) {
            cout << seq[i].set[j] << " ";
        }
        cout << endl << "Reward: " << seq[i].reward << endl;
    }

    unique_ptr<int[]> seqPointer(new int[seqAmount]);
    for (int i = 0; i < seqAmount; i++) {
        seqPointer[i] = 0;
    }

    clock_t start = clock();

    vector<vector<pair<int, int>>> combinations = generateCombinations(width, height, buffer);

    int max = 0, temp;
    vector<pair<int, int>> resultCoor;
    for (const auto &combination : combinations) {

```



```

        temp = calculateReward(buffer, combination, width, height, matrix, seq, seqAmount,
seqPointer.get());
        if (temp > max) {
            max = temp;
            resultCoor = combination;
        }
    }

    clock_t stop = clock();
    double timeTaken = (double)(stop - start) / CLOCKS_PER_SEC;
    cout << endl << "Done!" << endl;
    cout << "Time taken: " << timeTaken * 1000 << " ms" << endl;
    cout << "Max value: " << max << endl;

    if (max == 0) {
        cout << "There is no buffer solution because the max value is 0." << endl;
    }
    else {
        cout << "Buffer solution: ";
        for (int i = 0; i < buffer; i++) {
            cout << matrix[resultCoor[i].first * width + resultCoor[i].second] << " ";
        }
        cout << endl;
        cout << "Buffer solution's coordinate: ";
        for (int i = 0; i < buffer; i++) {
            cout << "(" << resultCoor[i].second << ", " << resultCoor[i].first << ") ";
        }
        cout << endl;
    }
    cout << endl;

    char yOrN;
    cout << "Do you want to save the solution into a file? (Y/N)" << endl;
    cin >> yOrN;
    if (yOrN == 'Y' || yOrN == 'y') {
        string filename;
        cout << "Enter the filename (without format): ";
        cin >> filename;
        saveToFile(filename, buffer, height, width, matrix, seqAmount, seq, timeTaken, max,
resultCoor);
    }
}

int main() {
    int opt;
    cout << "Welcome to Cyberpunk Breach Protocol Hacker!" << endl;
    cout << "Choose the input option:" << endl;
    cout << "[1] Input from file (put the file in src/input)" << endl;
    cout << "[2] Input from CLI (the matrix and the sequence will be randomized)" << endl;
    cin >> opt;

    if (opt == 1) {
        fileOption();
    }
    else if (opt == 2) {
        cliOption();
    }
}

```

Selain itu, juga terdapat folder berikut:

a. adt

Berisi file *sequence.h* yang berfungsi untuk menyimpan tipe data buatan *Sequence*. Berikut adalah isi dari file tersebut:

```

#ifndef SEQUENCE_H
#define SEQUENCE_H

#include <iostream>
#include <string>
using namespace std;

typedef struct {

```

```

    string *set;
    int length;
    int reward;
} Sequence;

#endif

```

b. input

Folder ini menyimpan file yang digunakan sebagai masukan program dari file berformat *.txt*. Struktur dari file tersebut adalah sebagai berikut:

```

ukuran_buffer
lebar_matriks tinggi_matriks
matrix
jumlah_sekuens
sekuens_1
bobot_sekuens_1
sekuens_2
bobot_sekuens_2
. . .
sekuens_n
bobot_sekuens_n

```

c. io

Folder ini berisi fungsionalitas *input/output* program untuk membaca dan mencetak ke file berformat *.txt*. Folder ini berisi dua kode, yaitu *io.h* yang berisi:

```

#ifndef SEQUENCE_H
#define SEQUENCE_H

#include <iostream>
#include <string>
using namespace std;

typedef struct {
    string *set;
    int length;
    int reward;
} Sequence;

#endif

```

dan *io.cpp* yang berisi:

```

#include "io.h"

void readinput(string filename, int *buffer, int *width, int *height, string **matrix,
Sequence **seq, int *seqAmount) {
    string temp;
    ifstream file ("../src/input/" + filename + ".txt");
    if (file.is_open()) {
        // dapetin buffer di baris pertama
        getline(file, temp);
        *buffer = stoi(temp);

        // dapetin ukuran matriks di baris kedua
        getline(file, temp);
        int i = 0;
        while (temp[i] != ' ') {
            i++;
        }
        *width = stoi(temp.substr(0,i));
        i++;
        int j = 0;
        while (temp[i+j] != '\0') {
            j++;
        }
    }
}

```

```
}  
*height = stoi(temp.substr(i,j));  
  
// ngebaca matriksnya  
*matrix = new string[*width*(*height)];  
for (int y = 0; y < *height; y++) {
```

d. output

Folder ini berfungsi untuk menyimpan hasil keluaran program ke dalam file berformat *.txt*.

BAB IV PROGRAM UTAMA

Berikut adalah fungsi-fungsi yang dibuat penulis untuk program ini:

1. `readInput`
Digunakan untuk membaca masukan dari file berformat *.txt*.
2. `saveToFile`
Digunakan untuk menyimpan keluaran program ke file berformat *.txt*.
3. `turnCheck`
Digunakan untuk memeriksa apakah kombinasi susunan buffer memenuhi aturan pada spesifikasi tugas.
4. `checkDone`
Digunakan untuk memeriksa apakah suatu sekuens sudah terpenuhi pada suatu buffer.
5. `calculateReward`
Digunakan untuk menghitung total bobot dari kombinasi susunan buffer.
6. `generateCombinations`
Digunakan untuk menghasilkan semua kombinasi buffer yang mungkin.
7. `fileOption`
Fungsi ini dipanggil jika pengguna memilih untuk menjalankan program dengan membaca input dari file.
8. `cliOption`
Fungsi ini dipanggil jika pengguna memilih untuk menjalankan program dengan membaca input dari user.

Alur *brute force* dari program ini adalah, kode akan menghasilkan semua kombinasi susunan buffer yang mungkin menggunakan fungsi *generateCombinations* berdasarkan ukuran matriks yang dimasukkan. Setelah itu, setiap kombinasi akan diperiksa validitasnya dengan fungsi *turnCheck*. Jika kombinasi tersebut valid, maka bobotnya akan dihitung dengan fungsi *calculateReward* dan *checkDone*. Setelah bobotnya didapatkan, bobotnya akan disimpan ke dalam suatu variabel dan akan digantikan jika ada didapatkan bobot yang lebih besar.

BAB V

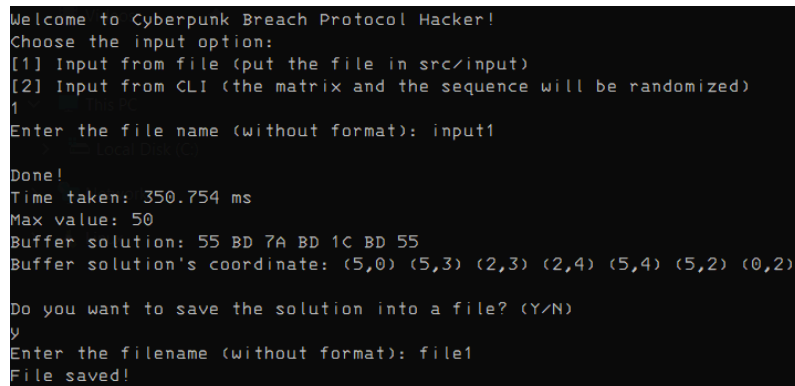
PENGUJIAN PROGRAM

Penulis melakukan dua kali pengujian program. Pertama untuk mencoba input dari file dan kedua untuk mencoba input dari CLI. Berikut adalah variasi test case untuk pengujian pertama:

1. input1.txt

```
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
```

Berikut adalah hasilnya:



```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
[1] Input from file (put the file in src/input)
[2] Input from CLI (the matrix and the sequence will be randomized)
1
Enter the file name (without format): input1

Done!
Time taken: 350.754 ms
Max value: 50
Buffer solution: 55 BD 7A BD 1C BD 55
Buffer solution's coordinate: (5,0) (5,3) (2,3) (2,4) (5,4) (5,2) (0,2)

Do you want to save the solution into a file? (Y/N)
y
Enter the filename (without format): file1
File saved!
```

Gambar 1. Solusi buffer dari *input1.txt*

2. input2.txt

```
4
4 4
7A 55 E9 E9
55 7A 1C 7A
55 1C 1C 55
BD 1C 7A 1C
2
1C 1C 7A
```

```
50
7A 55 7A 1C
10
```

Berikut adalah hasilnya:

```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
[1] Input from file (put the file in src/input)
[2] Input from CLI (the matrix and the sequence will be randomized)
1
Enter the file name (without format): input2

Done!
Time taken: 0.267 ms
Max value: 50
Buffer solution: 55 1C 1C 7A
Buffer solution's coordinate: (1,0) (1,2) (2,2) (2,3)

Do you want to save the solution into a file? (Y/N)
y
Enter the filename (without format): file2
File saved!
```

Gambar 2. Solusi buffer dari *input2.txt*

3. input3.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
1
7A 55 1C 7A 55 1C
50
```

Berikut adalah hasilnya:

```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
[1] Input from file (put the file in src/input)
[2] Input from CLI (the matrix and the sequence will be randomized)
1
Enter the file name (without format): input3

Done!
Time taken: 69.023 ms
Max value: 50
Buffer solution: E9 7A 55 1C 7A 55 1C
Buffer solution's coordinate: (2,0) (2,3) (4,3) (4,0) (0,0) (0,1) (2,1)

Do you want to save the solution into a file? (Y/N)
y
Enter the filename (without format): file3
File saved!
```

Gambar 3. Solusi buffer dari *input3.txt*

4. input4.txt

```
1
4 4
```

```
7A 55 E9 E9
55 7A 1C 7A
55 1C 1C 55
BD 1C 7A 1C
2
1C 1C 7A
50
7A 55 7A 1C
10
```

Berikut adalah hasilnya:

```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
[1] Input from file (put the file in src/input)
[2] Input from CLI (the matrix and the sequence will be randomized)
1
Enter the file name (without format): input4
Done!
Time taken: 0.009 ms
Max value: 0
There is no buffer solution because the max value is 0.

Do you want to save the solution into a file? (Y/N)
Y
Enter the filename (without format): file4
File saved!
```

Gambar 4. Solusi buffer dari *input4.txt*

Berikut adalah variasi test case untuk pengujian kedua:

1. Buffer berukuran 6 dengan matriks berukuran 7x7

```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
[1] Input from file (put the file in src/input)
[2] Input from CLI (the matrix and the sequence will be randomized)
2
Enter the number of token types: 6
Enter the tokens:
Note: The length must be 2.
1C 55 7A BD E9 FF
Enter the buffer size: 6
Enter the matrix's width: 7
Enter the matrix's height: 7
Enter the amount of sequences: 4
Enter the maximum length of the sequence: 6

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

Generated sequence:
E9 BD BD FF
Reward: 73
FF
Reward: 60
55 7A FF
Reward: 28
7A
Reward: 61

Done!
Time taken: 183.047 ms
Max value: 365
Buffer solution: FF 7A 7A 7A 7A 7A
Buffer solution's coordinate: (6,0) (6,4) (2,4) (2,5) (0,5) (0,3)

Do you want to save the solution into a file? (Y/N)
Y
Enter the filename (without format): cli1
File saved!
```

Gambar 5. Solusi buffer berukuran 6 dari matriks berukuran 7x7

2. Buffer berukuran 7 dengan matriks berukuran 8x8

```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
(1) Input from file (put the file in src/input)
(2) Input from CLI (the matrix and the sequence will be randomized)
2
Enter the number of token types: 6
Enter the tokens:
Note: The length must be 2.
1C 55 7A BD E9 FF
Enter the buffer size: 7
Enter the matrix's width: 8
Enter the matrix's height: 8
Enter the amount of sequences: 4
Enter the maximum length of the sequence: 7

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

Generated sequence:
7A FF
Reward: 92
7A
Reward: 41
E9 55 E9 7A 7A 7A
Reward: 61
1C BD
Reward: 92

Done!
Time taken: 3710.53 ms
Max value: 440
Buffer solution: 7A FF 7A FF 7A FF 7A
Buffer solution's coordinate: (3,0) (3,4) (1,4) (1,0) (3,0) (3,4) (1,4)

Do you want to save the solution into a file? (Y/N)
y
Enter the filename (without format): cli2
File saved!
```

Gambar 6. Solusi buffer berukuran 7 dari matriks berukuran 8x8

3. Buffer berukuran 8 dengan matriks berukuran 9x9

```
Welcome to Cyberpunk Breach Protocol Hacker!
Choose the input option:
(1) Input from file (put the file in src/input)
(2) Input from CLI (the matrix and the sequence will be randomized)
2
Enter the number of token types: 6
Enter the tokens:
Note: The length must be 2.
1C 55 7A BD E9 FF
Enter the buffer size: 8
Enter the matrix's width: 9
Enter the matrix's height: 9
Enter the amount of sequences: 4
Enter the maximum length of the sequence: 8

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

Generated sequence:
E9 BD FF FF
Reward: 48
E9 1C 1C 7A 55 FF
Reward: 72
1C 7A 1C
Reward: 79
BD 1C 7A
Reward: 63

Done!
Time taken: 87479.6 ms
Max value: 221
Buffer solution: 55 1C 7A 1C BD 1C 7A 1C
Buffer solution's coordinate: (1,0) (1,3) (7,3) (7,5) (0,5) (0,2) (4,2) (4,0)

Do you want to save the solution into a file? (Y/N)
y
Enter the filename (without format): cli3
File saved!
```

Gambar 7. Solusi buffer berukuran 8 dari matriks berukuran 9x9

BAB VI

KESIMPULAN DAN SARAN

A. Kesimpulan

Metode *brute force* adalah salah satu metode paling serbaguna karena bisa menyelesaikan hampir semua masalah terkait pemrograman, termasuk masalah *Breach Protocol* ini. Namun, kelemahannya adalah metode *brute force* dapat memakan waktu yang lama untuk ukuran data yang besar. Dapat dilihat pada gambar 6 dimana hanya dibutuhkan sekitar 3700 ms untuk matriks berukuran 8x8. Sementara pada gambar 7, dibutuhkan sekitar 87000 ms untuk matriks berukuran 9x9. Terjadi peningkatan sekitar 23 kali lipat. Dapat dibayangkan untuk ukuran-ukuran lain yang lebih besar pasti terjadi peningkatan yang lebih besar daripada ini.

B. Saran

Karena tujuan dari program ini adalah membuat program untuk menyelesaikan *minigame Breach Protocol*, maka sebaiknya dipilih algoritma yang lebih sangkil dan lebih mangkus daripada algoritma *brute force* karena *brute force* memakan waktu yang sangat lama untuk data berukuran besar.

LAMPIRAN

Source code: https://github.com/Nerggg/Tucil1_13522147