Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2023/2024

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Daniel Mulia Putra Manurung – 13522043

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

DAFTAR ISI

BAB I : 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.

Untuk menemukan solusi yang terbaik, yaitu solusi yang dapat menghasilkan nilai reward yang paling besar dengan batasan buffer yang telah ditentukan, akan dibuat sebuah program dalam Bahasa C++ dengan algoritma *Brute Force*. Laporan ini disusun dengan tujuan untuk membahas program yang dapat digunakan untuk mencari solusi dari cyberpunk breach protocol dengan algoritma *Brute Force*.

Gambar 1. Cyberpunk Breach Protoco

BAB 2 : ALGORITMA BRUTE FORCE UNTUK PENCARIAN SOLUSI PERMAINAN BREACH PROTOCOL

Pencarian solusi paling optimal dari permainan breach protocol dapat diselesaikan dengan algoritma brute force dengan cara mengiterasi seluruh kemungkinan urutan pengisian buffer atau sequence dengan panjang buffer. Dalam kasus ini, token – token yang akan dipilih untuk menyusun kode tersimpan dalam sebuah matrix, yang adalah list dari list dari elemen yang merepresentasikan token – token tiap baris dari matrix tersebut.

- **Langkah 1 (Pengisian Buffer)**

  Langkah ini bertujuan untuk mengisi buffer sementara yang merepresentasikan token – token yang akan dievaluasi dengan seluruh kemungkinan berdasarkan matrix yang tersedia. Pengisian ini dapat dilakukan dengan melakukan pengulangan sesuai dengan ukuran matriks yang tersedia menggunakan teknik rekursi.

  - **Basis :** List atau senarai token yang tidak memiliki slot kosong adalah list akhir yang akan dievaluasi.
  - **Rekurens :** Sesuai dengan parameter arah pencarian sekarang (yang akan direpresentasikan dengan variable) akan dilakukan pengulangan sebanyak ukuran dari arah pencarian sekarang untuk mengambil setiap elemen pada baris atau kolom tersebut sebagai elemen selanjutnya dari list. Perlu diingat bahwa elemen yang sebelumnya telah menjadi bagian dari list tidak akan diambil.

- **Langkah 2 (Stringmatching)**
  Langkah ini akan dilakukan untuk setiap pengulangan dari langkah 1, untuk mengekstrak total reward dari setiap list token yang telah diisi melalui langkah 1. Pengisian ini dapat dilakukan dengan melakukan pengulangan sesuai dengan ukuran token – token target menggunakan teknik iterative.

  Dalam langkah ini akan dilakukan pengulangan sebanyak jumlah sekuens target, dan pengulangan sebanyak panjang dari sekuens yang dicari. Untuk setiap sekuens target yang terdapat dalam buffer, akan ditambahkan poin sesuai dengan reward dari sekuens target yang terdapat dalam buffer.

- **Langkah 3 (Evaluasi)**
  Untuk setiap iterasi dari langkah 2, akan dievaluasi nilai dari total reward yang didapatkan. Jika nilai reward yang ditemukan lebih tinggi dari nilai reward yang dimiliki sekarang, maka buffer yang sednag dievaluasi akan menjadi buffer maksimum, dan nilai reward yang dievaluasi akan menjadi nilai reward yang dimiliki.

Ketiga langkah ini akan diulangi hingga setiap pengulangan selesai, dan untuk sebuah permasalahan dengan solusi yang tidak ditemukan, akan dikeluarkan output nilai maksimal 0.

BAB 3 : IMPLEMENTASI PROGRAM DENGAN BAHASA C++

Algoritma pada Bab 2 diimplementasikan dengan sebuah program C++. Program dengan 1 file realmain.cpp yang memuat program utama yang melingkupi seluruh fungsi dan proses penyelesaian permasalahan. Dalam realmain.cpp terdapat beberapa fungsi yang terbagi menjadi fungsi – fungsi pembantu atau utility, dan fungsi brute force utama.

1. Fungsi Pembantu

```
#include <iostream>
#include <string>
#include <fstream> // for txt io
#include <vector>  // for array and matrix
#include <chrono>  // for time
#include <random>
using namespace std;

// Mengembalikan nilai horizontal dari input txt
string getforhor(string in);

// Mengembalikan nilai vertikal dari input txt
string getforver(string in);

// Mengembalikan nilai baris matrix yang diakses sebagai array of tokens
vector<string> getperrow(string in, int how, int horizon);

// Mengembalikan jumlah token yang terdapat dalam string dari input txt
int findSize(string in);

// Mengembalikan jumlah slot kosong dalam array of tokens
int findEmpty(vector<string> in, int size);

// fungsi Stringmatching, mengembalikan nilai true juka terdapat string test
// sebagai substring dari check
bool inHere(vector<string> test, vector<string> check);

// I. S. in terdefinisi
// F. S. in akan tercetak di layar dengan tiap elemen dibatasi spasi
void printBufferv2(vector<string> in);

// Mengembalikan nilai true jika file terdapat pada path yang diinginkan
bool checkFExistence(string path);
```

Gambar 2. Fungsi utility

Fungsi getforhor(string in) dan getforver(string in) adalah fungsi string slicing untuk mengembalikan nilai dari parameter horizontal dan parameter vertical untuk menentukan dimensi dari matrix yang akan digunakan.

Fungsi getperrow(string in, int how, int horizon) adalah fungsi untuk yang digunakan

pada tiap baris matrix, mengekstrak nilai – nilai token pada baris matrix tersebut, dan mengembalikannya dalam bentuk array.

Fungsi findsize(string in) digunakan untuk mengembalikan jumlah token yang terdapat pada sebuah string.

Fungsi findEmpty(vector<string>) in, int size) digunakan untuk mengembalikan jumlah elemen buffer yang kosong.

Fungsi InHere(vector<string> test, vector<string> check) adalah fungsi stringmatching yang digunakan untuk menentukan keberadaan sequence test pada buffer check.

Prosedur printBufferv2(vector<string> in) digunakan untuk menampilkan buffer ke layer.

Fungsi checkFExistence(string path) akan mengembalikan nilai true path yang diinginkan terdapat pada device.

2. Fungsi Brute Force Utama

void recFind(vector<vector<string>> matric, vector<vector<string>> tempmatric, vector<int> curh, vector<int> curv, vector<string> curbuff, vector<vector<string>> tofind, vector<int> rew, int slot, int &max, vector<int> &finh, vector<int> &finv, vector<string> &maxbuff, int loch, int locv, int dir)

Dalam fungsi recFind terdapat 15 parameter yang berfungsi sebagai berikut:

- Matric, adalah matrix utama sebagai acuan matrix – matrix lainnya.
- Tempmatric, adalah matrix sementara untuk pengisian buffer tiap putaran.
- Curh, adalah array of int merepresentasikan koordinat horizontal buffer yang sedang ditrack
- Curv, adalah array of int merepresentasikan koordinat vertikal buffer yang sedang ditrack
- Curbuff, adalah array of string yang merepresentasikan buffer yang sedang ditrack
- Tofind, adalah array of array of string yang merepresentasikan target sequence yang ingin dicari
- Rew, adalah array of int yang merepresentasikan reward yang didapatkan untuk setiap target sequence
- Slot, adalah int yang merepresentasikan slot kosong pada array buffer
- Max, adalah int yang merepresentasikan nilai total poin tertinggi yang didapatkan selama proses fungsi
- Finh, adalah array of int yang merepresentasikan koordinat horizontal buffer max yang didapatkan selama proses fungsi

- Finv, adalah array of int yang merepresentasikan koordinat horizontal buffer max yang didapatkan selama proses fungsi
- Maxbuff, adalah array of string yang merepresentasikan buffer max yang didapatkan selama proses fungsi
- Loch, adalah letak koordinat horizontal yang ditrack sekarang
- Locv, adalah letak koordinat vertikal yang ditrack sekarang
- Dir, adalah int yang merepresentasikan arah gerak selanjutnya, 0 menandakan horizontal, 1 menandakan vertical, dan 99 menandakan inisialisasi

```
// FUNGSI UTAMA
void recFind(vector<vector<string>> matric, vector<vector<string>> tempmatric, vector<int> curh, vector<int>
{
    // BASIS
    if (slot <= 0)
    {
        // Calculation
    }

    // Recursion
    else
    {
        // Calculation
        if (dir == 99) // Initial
        {
            for (int i1 = 0; i1 < tempmatric[0].size(); i1++)
            {
                vector<vector<string>> temptemp1(tempmatric.size(), vector<string>(tempmatric[0].size()));
                temptemp1 = tempmatric;
                vector<string> prevbuff1(curbuff.size());
                prevbuff1 = curbuff;
                vector<int> prevh1(curbuff.size());
                prevh1 = curh;
                vector<int> prevv1(curbuff.size());
                prevv1 = curv;
                int prevloch = loch;
                int prevlocv = locv;
                int prevslot = slot;

                tempmatric = matric;
                curbuff[0] = tempmatric[0][i1];
                tempmatric[0][i1] = "XX";
```

Gambar 3. Fungsi recFind bagian inisialisasi

Fungsi recFind diawali dengan basis yang menyatakan fungsi untuk berhenti ketika slot dalam buffer sudah mencapai 0 atau sudah tidak tersedia slot kosong dalam buffer. Kemudian pada bagian rekursi, terdapat kondisional yang jika menerima input 99, yang mengindikasikan pencarian pertama. Pada bagian ini dibentuk sebuah matrix temporary yang akan digunakan selama tiap perulangan (sebanyak dimensi horizontal matrix). Dibentuk juga buffer temporary, array tracking koordinat, dan variable untuk melakukan track letak koordinat yang sedang dicari. Dapat dilihat untuk setiap perulangan, matrix pada baris pertama dan kolom ke-i1 akan menjadi elemen pertama buffer, dan pada matrix elemen tersebut diubah menjadi “XX” yang adalah MARK pada kasus ini.

```
            // Setup
            loch = i1;
            locv = 0;
            curh[curbuff.size() - slot] = i1;
            curv[curbuff.size() - slot] = 0;
            slot -= 1;

            // Stringmatching
            int points = 0;
            for (int i2 = 0; i2 < tofind.size(); i2++)
            {
                if (inHere(tofind[i2], curbuff))
                {
                    points += rew[i2];
                    return;
                }
            }

            // Compare to Highest
            if (points > max)
            {
                maxbuff = curbuff;
                finh = curh;
                finv = curv;
                max = points;
                break;
            }

            // printBuffer(curbuff);

            // Next Recursion
            dir = 1;
            recFind(matric, tempmatric, curh, curv, curbuff, tofind, rew, slot, max, finh, finv, maxbuff, loch, locv, dir);

            tempmatric = temptemp1;
            curh = prevh1;
            curv = prevv1;
            curbuff = prevbuff1;
            slot = prevslot;
            loch = prevloch;
            locv = prevlocv;
        }
    }
```

Gambar 4. Fungsi recFind bagian proses dan rekursi

Selanjutnya terdapat bagian setup untuk mengubah tracking koordinat sesuai dengan letak koordinat baru sesuai dengan loop, diikuti dengan bagian stringmatching dan pengambilan total poin, lalu diikuti dengan perbandingan dengan nilai maksimum yang dimiliki, untuk mencari nilai maksimum dari keseluruhan kemungkinan buffer. Bagian ini diakhiri dengan pemanggilan fungsi recFind kembali dengan parameter yang telah diubah, dan pengembalian parameter yang telah diubah, agar dapat melakukan proses kembali, jika layer proses selanjutnya sudah selesai. Dengan kata lain, jika program telah menemukan seluruh kemungkinan dari buffer 1C – BD – 55 – “XX”, maka akan melakukan ‘backtrack’ menjadi 1C – BD – BD – “XX” dan mengulangi prosesnya kembali.

Kemudian, proses yang sama diulangi untuk arah secara horizontal dan vertical dengan sedikit perubahan di bagian tracking koordinat, sesuai dengan gerakan tracking yang dilakukan selanjutnya.

```
else if (dir == 0) // Horizontal
{
    for (int i3 = 0; i3 < tempmatric[0].size(); i3++)
    {
```

```
else if (dir == 1) // Vertical
{
    for (int i5 = 0; i5 < tempmatric.size(); i5++)
    {
```

Gambar 4. Fungsi recFind bagian pencarian dengan horizontal dan vertical

Selanjutnya adalah program utama, program utama terbagi menjadi 2 cara, yaitu melalui txt, dan melalui CLI.

```
// MAIN PROGRAM
int main()
{
    int method = 0;
    cout << "========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========" << endl;
    cout << "1. TXT input." << endl;
    cout << "2. CLI input." << endl;
    cout << "Select entry method: (1/2) ";
    cin >> method;
```

Gambar 5. Main Program

Dengan memilih pilihan 1, maka program akan meminta file txt yang akan dibaca sesuai dengan format yang telah ditentukan.

```
if (method == 1)
{
    ifstream testcase;
    string fileName;
    cout << "Enter file name: ";
    cin >> fileName;
    string fileAdd = "../test/" + fileName + ".txt";

    while (!checkFExistence(fileAdd))
    {
        cout << "File does not exist, please enter an existing file" << endl;
        cout << "Enter file name: ";
        cin >> fileName;
        fileAdd = "../test/" + fileName + ".txt";
    }
    testcase.open(fileAdd);
```

Gambar 6. Program dengan input txt

Dengan memilih pilihan 2, maka program akan meminta serangkaian input melalui CLI, mulai dari jumlah token unik, ukuran buffer, ukuran matrix, jumlah target sequences, dan panjang maksimal target sequence. Dalam input CLI ini, akan digunakan pembangkit acak, untuk menentukan matrix yang digunakan sesuai dengan dimensinya, dan begitu pula dengan target sequence yang akan dicari.

```
else
{
    int unique;
    int buff_size;
    int m_h, m_v;
    string temptoken;
    cout << "Enter unique token type amount: ";
    cin >> unique;
    vector<string> token_pool(unique);
    cout << "Enter tokens for the token pool: ";
    for (int i = 0; i < unique; i++)
    {
        cin >> temptoken;
        token_pool[i] = temptoken;
    }

    int min = 0;
    int max = unique - 1;

    cout << "Enter buffer size: ";
    cin >> buff_size;
    vector<string> buffer(buff_size);
    cout << "Enter matrix dimensions (horizontal vertical): ";
    for (int i = 0; i < 2; i++)
    {
        if (i == 0)
        {
            cin >> m_h;
        }
        else
        {
            cin >> m_v;
        }
    }
```

```
vector<vector<string>> matrix(m_v, vector<string>(m_h));
int rng;

// RNG - 1 (token pool)
random_device mach;
mt19937 generator(mach());
uniform_int_distribution<> rendo(min, max);

for (int i = 0; i < m_v; i++)
{
    for (int j = 0; j < m_h; j++)
    {
        rng = rendo(generator);
        matrix[i][j] = token_pool[rng];
        cout << matrix[i][j] + ' ';
    }
    cout << endl;
}

int seqnum;
cout << "Enter target sequences amount: ";
cin >> seqnum;

vector<vector<string>> sequences(seqnum);
vector<int> rewards(seqnum);

int max_size;
cout << "Enter maximum sequence length: ";
cin >> max_size;

// RNG - 2 (sequence length)
random_device mach2;
mt19937 generator2(mach2());
uniform_int_distribution<> rendo2(2, max_size);

for (int i = 0; i < seqnum; i++)
{
    int tempsize = rendo2(generator2);
    sequences[i].resize(tempsize);
    rewards[i] = tempsize * 15;
    cout << "Target Sequence " << i + 1 << ": ";
    cout << "(Value: " << rewards[i] << ") " << endl;
    for (int j = 0; j < tempsize; j++)
    {
        rng = rendo(generator);
        sequences[i][j] = token_pool[rng];
        cout << sequences[i][j] + ' ';
    }
```

Gambar 7. Program dengan input CLI

Bagian akhir dari program adalah output hasil dan penyimpanan hasil.

```
// Results
cout << "======= COMPLETED =======" << endl;
cout << maxval << endl;
printBufferv2(buffer);

if (maxval == 0)
{
    cout << "NO SOLUTION FOUND" << endl;
}
else
{
    for (int i = 0; i < buff_size; i++)
    {
        if (buffer[i] != "XX")
        {
            cout << finhor[i] + 1;
            cout << ", ";
            cout << finver[i] + 1 << endl;
        }
    }
}
cout << "Time taken: " << timeTaken.count() << "seconds." << endl;
```

Gambar 8. Program output pada CLI

```
cout << "Do you want to save the solution? (Y/N): ";
cin >> confirm;
if (confirm == "Y" || confirm == "y")
{
    cout << "Enter your solution file name: ";
    cin >> saveFile;
    saveFilefin = "../test/" + saveFile + ".txt";

    if (checkFExistence(saveFilefin))
    {
        cout << "Filename exist, overwriting file" << endl;
    }
    else
    {
        cout << "Creating " << saveFile << ".txt" << endl;
    }

    saveStream.open(saveFilefin);

    if (!saveStream.is_open())
    {
        cerr << "Unable to open file" << endl;
        return 0;
    }

    // Output to file
    saveStream << to_string(maxval) << endl;
    if (maxval == 0)
    {
        saveStream << "NO SOLUTION FOUND" << endl;
    }
    else
    {
        for (int i = 0; i < buffer.size(); i++)
        {
            if (buffer[i] != "XX")
            {
                saveStream << buffer[i] + ' ';
            }
        }
        saveStream << endl;
        for (int i = 0; i < buff_size; i++)
        {
            saveStream << finhor[i] + 1;
            saveStream << ", ";
            saveStream << finver[i] + 1 << endl;
        }
```

Gambar 9. Program penyimpanan ke file

BAB 4 : EKSPERIMEN

Run Program

```
========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========
1. TXT input.
2. CLI input.
Select entry method: (1/2)
```

Gambar 10. Main

Contoh 1: (Input TXT dengan tidak menyimpan solusi)

```
Select entry method: (1/2) 1
Enter file name: testcase-100
File does not exist, please enter an existing file
Enter file name: testcase-1
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
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
======= COMPLETED =======
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3
Time taken: 0.85891seconds.
Do you want to save the solution? (Y/N):
```

Gambar 11. Contoh 1 (1)

```
1, 3
Time taken: 0.85891seconds.
Do you want to save the solution? (Y/N): N
```

Gambar 12. Contoh 1 (2)

Contoh 2: (Input TXT dengan menyimpan solusi)

```
========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========
1. TXT input.
2. CLI input.
Select entry method: (1/2) 1
Enter file name: testcase-2
7A 55 E9 E9
55 7A 1C 7A
55 1C 1C 55
BD 1C 7A 1C
BD E9
15
BD 7A
20
BD 1C BD
30
======= COMPLETED =======
20
7A BD 7A
1, 1
1, 4
3, 4
Time taken: 0.0040985seconds.
Do you want to save the solution? (Y/N): Y
Enter your solution file name: testcase-2-solution
Creating testcase-2-solution.txt
```

Gambar 13. Contoh 2

test > testcase-2-solution.txt

```
1   20
2   7A BD 7A
3   1, 1
4   1, 4
5   3, 4
6   1, 1
7   1, 1
8   Time taken: 0.0040985 seconds.
9
```

Gambar 14. Hasil penyimpanan solusi contoh 2

Contoh 3: (Input TXT tanpa solusi)

```
========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========
1. TXT input.
2. CLI input.
Select entry method: (1/2) 1
Enter file name: testcase-3
CC EE DD
BB AA EE
DD BB AA
CC DD CC CC BB EE BB EE
120
CC CC BB CC
60
CC AA
30
======= COMPLETED =======
0

NO SOLUTION FOUND
Time taken: 0.0007423seconds.
Do you want to save the solution? (Y/N): Y
Enter your solution file name: testcase-3-solution
Filename exist, overwriting file
```

Gambar 15. Contoh 3

```
test > testcase-3-solution.txt
1   0
2   NO SOLUTION FOUND
3   Time taken: 0.0007423 seconds.
```

Gambar 16. Hasil penyimpanan solusi contoh 3

Contoh 4: (Input CLI dengan menyimpan solusi)

```
========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========
1. TXT input.
2. CLI input.
Select entry method: (1/2) 2
Enter unique token type amount: 5
Enter tokens for the token pool: BD 1C 7A 55 E9
Enter buffer size: 7
Enter matrix dimensions (horizontal vertical): 6 6
55 1C 1C E9 7A E9
7A 55 BD BD 1C 1C
E9 7A 7A E9 1C 55
7A BD 55 55 55 55
55 E9 E9 BD E9 E9
1C BD E9 55 BD 1C
Enter target sequences amount: 3
Enter maximum sequence length: 4
Target Sequence 1: (Value: 60)
1C 55 7A E9
Target Sequence 2: (Value: 60)
E9 7A 7A 1C
Target Sequence 3: (Value: 60)
BD BD 1C BD
======= COMPLETED =======
60
55 7A 55 BD BD 1C BD
1, 1
1, 2
2, 2
2, 6
5, 6
5, 2
3, 2
Time taken: 0.877825seconds.
Do you want to save the solution? (Y/N): Y
Enter your solution file name: testcase-4-solution
Creating testcase-4-solution.txt
```

Gambar 17. Contoh 4

```
test > testcase-4-solution.txt
 1   60
 2   55 7A 55 BD BD 1C BD
 3   1, 1
 4   1, 2
 5   2, 2
 6   2, 6
 7   5, 6
 8   5, 2
 9   3, 2
10   Time taken: 0.877825 seconds.
```

Gambar 18. Hasil penyimpanan solusi contoh 4

Contoh 5: (Input CLI dengan buffer dan matrix yang lebih besar)

```
========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========
1. TXT input.
2. CLI input.
Select entry method: (1/2) 2
Enter unique token type amount: 6
Enter tokens for the token pool: 1A 2B 3C 4D 5E 6F
Enter buffer size: 8
Enter matrix dimensions (horizontal vertical): 7 7
2B 4D 3C 4D 5E 4D 1A
2B 4D 5E 2B 2B 2B 3C
4D 6F 4D 4D 1A 1A 5E
3C 1A 1A 1A 4D 6F 2B
5E 3C 4D 3C 2B 5E 5E
4D 6F 1A 3C 1A 1A 3C
6F 3C 5E 6F 3C 4D 4D
Enter target sequences amount: 5
Enter maximum sequence length: 4
Target Sequence 1: (Value: 45)
6F 4D 4D
Target Sequence 2: (Value: 60)
6F 4D 6F 4D
Target Sequence 3: (Value: 45)
4D 6F 3C
Target Sequence 4: (Value: 45)
1A 4D 6F
Target Sequence 5: (Value: 30)
1A 3C
======= COMPLETED =======
150
4D 6F 3C 6F 4D 6F 4D 4D
4, 1
4, 7
2, 7
2, 3
1, 3
1, 7
6, 7
6, 1
Time taken: 23.4416seconds.
Do you want to save the solution? (Y/N): Y
Enter your solution file name: testcase-5-solution
Creating testcase-5-solution.txt
```

Gambar 19. Contoh 5

```
test > testcase-5-solution.txt
1   150
2   4D 6F 3C 6F 4D 6F 4D 4D
3   4, 1
4   4, 7
5   2, 7
6   2, 3
7   1, 3
8   1, 7
9   6, 7
10  6, 1
11  Time taken: 23.4416 seconds.
```

Gambar 19. Hasil penyimpanan solusi contoh 5

Contoh 6: (Input CLI tanpa solusi)

```
========== WELCOME TO CYBERPUNK BREACH PROTOCOL SOLVER ==========
1. TXT input.
2. CLI input.
Select entry method: (1/2) 2
Enter unique token type amount: 5
Enter tokens for the token pool: 1A 2B 3C 4D 5E
Enter buffer size: 3
Enter matrix dimensions (horizontal vertical): 3 3
2B 2B 1A
3C 3C 3C
5E 2B 4D
Enter target sequences amount: 3
Enter maximum sequence length: 10
Target Sequence 1: (Value: 90)
1A 5E 3C 3C 1A 4D
Target Sequence 2: (Value: 90)
3C 3C 2B 3C 5E 5E
Target Sequence 3: (Value: 135)
4D 1A 4D 2B 3C 1A 1A 2B 3C
======= COMPLETED =======
0

NO SOLUTION FOUND
Time taken: 0.0003058seconds.
Do you want to save the solution? (Y/N): Y
Enter your solution file name: testcase-6-solution
Creating testcase-6-solution.txt
```

Gambar 20. Contoh 6

```
test > testcase-6-solution.txt
1   0
2   NO SOLUTION FOUND
3   Time taken: 0.0003058 seconds.
```

Gambar 21. Hasil penyimpanan solusi contoh 6

LAMPIRAN

Repository Githu

https://github.com/Gryphuss/Tucil1_13522043

Tabel Spesifikasi

No	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		✓