

LAPORAN TUGAS KECIL 1 IF2211
STRATEGI ALGORITMA

*Penyelesaian Cyberpunk 2077 Breach Protocol
dengan Algoritma Brute Force*



Dosen Pengampu: Dr. Nur Ulfa Maulidevi, S.T, M.Sc

Disusun oleh:

Amalia Putri

(13522042)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Allah SWT, karena atas rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan dengan judul "Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force". Laporan ini disusun sebagai salah satu tugas mata kuliah Strategi Algoritma.

Penyusunan makalah ini tidak lepas dari bantuan berbagai pihak. Penulis mengucapkan terima kasih kepada dosen beserta asisten pembimbing mata kuliah Strategi Algoritma yang telah memberikan bimbingan dan panduan selama proses pembuatan laporan ini.

Semoga laporan ini dapat memberikan kontribusi positif dalam pemahaman dan pengembangan algoritma *brute force*. Akhir kata, penulis menyampaikan permohonan maaf atas segala keterbatasan dalam laporan ini, dan penulis menerima dengan terbuka segala kritik dan saran yang bersifat membangun.

Bandung, 11 Februari 2024

A handwritten signature in black ink, appearing to be 'Amalia', written in a cursive style.

Amalia Putri (13522042)

DAFTAR ISI

KATA PENGANTAR.....	1
DAFTAR ISI.....	2
PENGECEKAN PROGRAM.....	3
DESKRIPSI MASALAH.....	4
A. Algoritma Brute Force.....	5
B. Source Program C++.....	7
C. Input dan Output.....	12
D. Repository.....	14

PENGECEKAN 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		✓

Tabel 1. Tabel Pengecekan Program

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:

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.

Terdapat aturan permainan Breach Protocol pada minigame Cyberpunk 2077 ini, yakni sebagai berikut.

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.

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

Gambar 1. Matriks berisi token alfanumerik

Dengan sekuens sebagai berikut.

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi yang optimal untuk matriks dan sekuens yang diberikan adalah sebagai berikut.

Total bobot hadiah : 50 poin

Total langkah : 6 langkah

Objektifnya adalah menemukan solusi dari permainan Breach Protocol yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma brute force.

A. Algoritma Brute Force

Dalam pembuatan program ini, terdapat langkah-langkah yang dilakukan untuk mengimplementasikannya ke dalam program, yakni membagi modul dan eksekusi program utama secara terpisah sehingga terdapat 3 files: **solver.hpp**, **solver.cpp**, dan **main.cpp**.

1. **solver.hpp**

Penulis membuat 2 tipe data buatan khusus untuk **token** dan **sekuens**. Suatu token berisi 2 karakter alfanumerik dan koordinat. Sedangkan sekuens berisi 2 karakter alfanumerik dan nilai bobot hadiah.

Setelah itu, karena suatu nilai-nilai, seperti token yang sedang ditunjuk, nilai bobot hadiah, dan koordinat serta beberapa variabel lainnya yang membantu melakukan *tracking* pada proses *brute force* dibuat secara global sehingga dapat diakses di file mana pun. Semua *attribute* dan *method* disimpan pada *class* **BreachProtocolSolver**.

2. **solver.cpp**

Dalam pengimplementasian *header* dari **solver.hpp**, maka terdapat langkah-langkah utama yang perlu diimplementasikan, yakni sebagai berikut.

- 1) Mengonversikan seluruh masukan berupa matriks menjadi vektor yang berisi vektor token (*nested vector of token*);

- 2) Penulis menggunakan konsep *depth first search* (DFS), seperti pengimplentasian pada teori graf sehingga memerlukan rekursi. Namun, untuk mengecek seluruh kemungkinan kecocokan antara *buffer* dan sekuens, diperlukan juga suatu fungsi yang mengecek bahwa suatu simpul (*buffer*) dan/atau lintasan (antar koordinat) tersebut sudah dilewati;
- 3) Untuk menyesuaikan aturan, yakni harus mulai dari baris pertama, maka pada fungsi **solve()** dan pergerakan vertikal serta horizontal secara bergantian akan diatur pada fungsi **dfs()**; dan
- 4) Mengeluarkan *output* berupa solusi, yakni bobot hadiah maksimum, isi *buffer*, koordinat, dan waktu eksekusi program.

3. **main.cpp**

Pada program utama, semua program lainnya, yakni **solver.hpp** dan **solver.cpp** dirakit dan disatukan pada **main.cpp**. Terdapat langkah-langkah utama yang perlu diimplementasikan, yakni sebagai berikut.

- 1) Melakukan *write file* untuk menyimpan *file* yang berekstensi **.txt** sehingga akan berguna pada saat pengguna ingin melakukan pengacakan pada matriks dan sekuens serta menyimpan hasil *output*;
- 2) Membuat fungsi yang dapat mengacak atau mengombinasikan token alfanumerik menjadi suatu matriks dan sekuens, serta terdapat pula pengacakan angka, penulis memilih angka acak dari -100 sampai dengan 100; dan
- 3) Membuat opsi penyimpanan hasil *output* kepada pengguna, yang berisi: bobot hadiah, isi *buffer*, koordinat, dan waktu eksekusi program.

B. Source Program C++

1. solver.hpp

```
1  #ifndef SOLVER_HPP
2  #define SOLVER_HPP
3
4  #include <vector>
5  #include <string>
6
7  using namespace std;
8
9  struct Token
10 {
11     string value;
12     int x, y;
13     Token(string val, int xCoord, int yCoord) : value(val), x(xCoord), y(yCoord) {}
14 };
15
16 struct Sequence
17 {
18     vector<string> tokens;
19     int reward;
20     Sequence(vector<string> tok, int rew) : tokens(tok), reward(rew) {}
21 };
```

Gambar 1.1. Source Code solver.hpp

```
22
23 class BreachProtocolSolver
24 {
25 public:
26     BreachProtocolSolver(int bufferSize, const vector<vector<string>> &matrix, const vector<Sequence> &sequences);
27     void solve();
28     void printSolution();
29     bool hasBeenVisited(vector<Token> &visited, Token &token);
30     int bufferSize;
31     vector<vector<Token>> matrix;
32     vector<Sequence> sequences;
33     vector<Token> buffer;
34     int maxReward;
35     vector<Token> solutionBuffer;
36     void dfs(int x, int y, int direction, int currentReward, vector<bool> &sequenceMatched, vector<Token> &visited);
37     bool isSequenceMatched(const Sequence &sequence, int &currentReward, const vector<Token> &visited);
38 };
39
40 #endif
```

Gambar 1.2. Source Code solver.hpp

2. solver.cpp

```
1  #include "solver.hpp"
2  #include <iostream>
3  #include <vector>
4  #include <string>
5
6  using namespace std;
7
8  BreachProtocolSolver::BreachProtocolSolver(int bufferSize, const vector<vector<string>> &matrixInput, const vector<Sequence> &sequencesInput)
9      : bufferSize(bufferSize), sequences(sequencesInput), maxReward(0)
10 {
11     // Konversi matrix input menjadi vector of vector of Token
12     for (int i = 0; i < matrixInput.size(); i++)
13     {
14         vector<Token> row;
15         for (int j = 0; j < matrixInput[i].size(); j++)
16         {
17             row.emplace_back(matrixInput[i][j], i, j);
18         }
19         matrix.push_back(row);
20     }
21 }
22
23 bool BreachProtocolSolver::hasBeenVisited(vector<Token> &visited, Token &token)
24 {
25     for (auto &visitedToken : visited)
26     {
27         if (visitedToken.x == token.x && visitedToken.y == token.y)
28         {
29             return true;
30         }
31     }
32     return false;
33 }
```

Gambar 2.1. Source Code solver.cpp


```

35 void BreachProtocolSolver::dfs(int x, int y, int direction, int currentReward, vector<bool> &sequenceMatched, vector<token> &visited)
36 {
37     if (buffer.size() > bufferSize || hasBeenVisited(visited, matrix[x][y]))
38         return;
39     visited.push_back(matrix[x][y]);
40     // Tambahkan token ke buffer
41     buffer.push_back(matrix[x][y]);
42     // Cek apakah buffer cocok dengan sequence
43     for (int i = 0; i < sequences.size(); ++i)
44     {
45         if (!sequenceMatched[i] && isSequenceMatched(sequences[i], currentReward, visited))
46         {
47             sequenceMatched[i] = true;
48         }
49     }
50     // Bergerak Rekursif (DFS) Vertikal atau Horizontal
51     if (direction == 0)
52     {
53         // Vertikal
54         for (int newX = 0; newX < matrix.size(); ++newX)
55         {
56             if (newX != x)
57             {
58                 dfs(newX, y, 1, currentReward, sequenceMatched, visited);
59             }
60         }
61     }
62     else
63     {
64         // Horizontal
65         for (int newY = 0; newY < matrix[0].size(); ++newY)
66         {
67             if (newY != y)
68             {
69                 dfs(x, newY, 0, currentReward, sequenceMatched, visited);
70             }
71         }
72     }
73     // Backtracking
74     buffer.pop_back();
75     visited.pop_back();
76     for (int i = 0; i < sequences.size(); ++i)
77     {
78         sequenceMatched[i] = false; // Reset sequenceMatched
79     }
80 }
81 void BreachProtocolSolver::solve()
82 {
83     vector<bool> sequenceMatched(sequences.size(), false);
84     vector<token> visited;
85     // Berjalan ke setiap token di baris pertama
86     for (int y = 0; y < matrix[0].size(); ++y)
87     {
88         dfs(0, y, 0, 0, sequenceMatched, visited);
89     }
90 }

```

Gambar 2.2. Source Code solver.cpp

```

99 bool BreachProtocolSolver::isSequenceMatched(const Sequence &sequence, int &currentReward, const vector<token> &visited)
100 {
101     // Cek apakah buffer cocok dengan sequence
102     if (buffer.size() < sequence.tokens.size())
103         return false;
104     bool matched = true;
105     for (int i = 0; i < sequence.tokens.size(); ++i)
106     {
107         if (sequence.tokens[i] != buffer[buffer.size() - sequence.tokens.size() + i].value)
108         {
109             matched = false;
110             break;
111         }
112     }
113     if (matched)
114     {
115         currentReward += sequence.reward;
116         // Max reward yang didapatkan
117         if (currentReward > maxReward)
118         {
119             maxReward = currentReward;
120             solutionBuffer = buffer;
121         }
122         return true;
123     }
124     return false;
125 }
126 void BreachProtocolSolver::printSolution()
127 {
128     if (!solutionBuffer.empty())
129     {
130         cout << maxReward << endl;
131         for (const auto &token : solutionBuffer)
132         {
133             cout << token.value << " ";
134         }
135         cout << endl;
136         for (const auto &token : solutionBuffer)
137         {
138             cout << token.y + 1 << ", " << token.x + 1 << endl;
139         }
140     }
141     else
142     {
143         cout << "Tidak ada buffer dan sequence yang cocok. Tidak ada solusi yang ditemukan!" << endl;
144         return;
145     }
146 }
147 }
148 }
149 }
150 }
151 }

```

Gambar 2.3. Source Code solver.cpp

3. main.cpp

```
1 #include "solver.hpp"
2 #include <iostream>
3 #include <fstream>
4 #include <sstream>
5 #include <vector>
6 #include <string>
7 #include <cstdlib>
8 #include <ctime>
9 #include <chrono>
10
11 using namespace std;
12 using namespace std::chrono;
13
14 void writeToFile(const string &filename,
15                 int buffer_size,
16                 int matrix_width,
17                 int matrix_height,
18                 const vector<vector<string>> &matrix,
19                 int number_of_sequences,
20                 const vector<vector<string>> &sequences,
21                 const vector<vector<int>> &sequence_rewards)
22 {
23     ofstream file(filename);
24     if (!file.is_open())
25     {
26         cerr << "Gagal membuka file." << endl;
27         return;
28     }
29
30     file << buffer_size << endl;
31     file << matrix_width << " " << matrix_height << endl;
32
33     for (const auto &row : matrix)
34     {
35         for (const auto &token : row)
36         {
37             file << token << " ";
38         }
39         file << endl;
40     }
41
42     file << number_of_sequences << endl;
43     for (int i = 0; i < number_of_sequences; ++i)
44     {
45         for (const auto &token : sequences[i])
46         {
47             file << token;
48             if (&token != &sequences[i].back())
49             {
50                 file << " ";
51             }
52         }
53         file << endl;
54
55         // Menulis reward sequence (sekali doang), angka acak [-100,100]
56         int random_reward = rand() % 201 - 100;
57
58         file << random_reward;
59
60         if (i < number_of_sequences - 1)
61         {
62             file << endl;
63         }
64     }
65
66     file.close();
67 }
```

Gambar 3.1. Source Code main.cpp

```
69 void randomizer(int jumlah_token_unik,
70                 const vector<string> &token,
71                 int ukuran_buffer,
72                 int width,
73                 int height,
74                 int jumlah_sekuens,
75                 int ukuran_maksimal_sekuens)
76 {
77     // Random number generator
78     srand(time(NULL));
79
80     // Matriks random
81     vector<vector<string>> matrix(height, vector<string>(width));
82     for (int i = 0; i < height; i++)
83     {
84         for (int j = 0; j < width; j++)
85         {
86             matrix[i][j] = token[rand() % jumlah_token_unik];
87         }
88     }
89
90     // Sekuens random
91     vector<vector<string>> sequences(jumlah_sekuens, vector<string>(ukuran_maksimal_sekuens));
92     vector<vector<int>> sequence_rewards(jumlah_sekuens, vector<int>(ukuran_maksimal_sekuens));
93     for (int i = 0; i < jumlah_sekuens; i++)
94     {
95         for (int j = 0; j < ukuran_maksimal_sekuens; j++)
96         {
97             sequences[i][j] = token[rand() % jumlah_token_unik];
98         }
99     }
100
101     writeToFile("../test/random.txt", ukuran_buffer, width, height, matrix, jumlah_sekuens, sequences, sequence_rewards);
102 }
```

Gambar 3.2. Source Code main.cpp

```

1 void randomMatrixSequence()
2 {
3     int jumlah_token_unik;
4
5     // cout << "Masukkan jumlah token unik: ";
6     cin >> jumlah_token_unik;
7
8     // cout << "Masukkan token: ";
9     vector<string> tokens(jumlah_token_unik);
10    for (int i = 0; i < jumlah_token_unik; i++)
11    {
12        cin >> tokens[i];
13    }
14
15    int ukuran_buffer, width, height, jumlah_sekuens, ukuran_maksimal_sekuens;
16
17    // cout << "Masukkan ukuran buffer: ";
18    cin >> ukuran_buffer;
19
20    // cout << "Masukkan ukuran matriks (width height): ";
21    cin >> width >> height;
22
23    // cout << "Masukkan jumlah sekuens: ";
24    cin >> jumlah_sekuens;
25
26    // cout << "Masukkan ukuran maksimal sekuens: ";
27    cin >> ukuran_maksimal_sekuens;
28
29    randomizer(jumlah_token_unik, tokens, ukuran_buffer, width, height, jumlah_sekuens, ukuran_maksimal_sekuens);
30 }
31
32 vector<vector<string>> readMatrix(istream &inputFile, int width, int height)
33 {
34     vector<vector<string>> matrix;
35     string line;
36     for (int i = 0; i < height; ++i)
37     {
38         getline(inputFile, line);
39         istringstream iss(line);
40         vector<string> row;
41         for (int j = 0; j < width; ++j)
42         {
43             string token;
44             iss >> token;
45             row.push_back(token);
46         }
47         matrix.push_back(row);
48     }
49     return matrix;
50 }
51
52 vector<Sequence> readSequences(istream &inputFile, int numberOfSequences)
53 {
54     vector<Sequence> sequences;
55     for (int i = 0; i < numberOfSequences; ++i)
56     {
57         string line, token;
58         int reward;
59         getline(inputFile, line);
60         istringstream iss(line);
61         vector<string> tokens;
62         while (iss >> token)
63         {
64             tokens.push_back(token);
65         }
66         inputFile >> reward;
67         inputFile.ignore();
68         sequences.emplace_back(tokens, reward);
69     }
70     return sequences;
71 }

```

Gambar 3.3. Source Code main.cpp

```

1  int main(int argc, char *argv[])
2  {
3      string filename;
4
5      cout << "Apakah ingin menggunakan randomizer untuk matrix dan sequences? (y/n): ";
6
7      string isRandom;
8      cin >> isRandom;
9
10     if (isRandom == "y" || isRandom == "Y")
11     {
12         randomMatrixSequence();
13         filename = "../test/random.txt";
14     }
15     else
16     {
17         cout << "Masukkan nama file: ";
18         cin >> filename;
19         filename = "../test/" + filename;
20     }
21
22     if (argc > 1)
23     {
24         filename = argv[1];
25     }
26
27     ifstream inputFile(filename);
28     if (!inputFile.is_open())
29     {
30         cerr << "Failed to open file: " << filename << endl;
31         return 1;
32     }
33
34     int bufferSize, width, height, numberOfSequences;
35     inputFile >> bufferSize >> width >> height;
36     inputFile.ignore();
37
38     auto matrix = readMatrix(inputFile, width, height);
39
40     inputFile >> numberOfSequences;
41     inputFile.ignore();
42
43     auto sequences = readSequences(inputFile, numberOfSequences);
44
45     BreachProtocolSolver solver(bufferSize, matrix, sequences);
46
47     auto start = high_resolution_clock::now();
48
49     solver.solve();
50     solver.printSolution();
51
52     auto end = high_resolution_clock::now();
53     auto elapsed = duration_cast<milliseconds>(end - start).count();
54     cout << endl
55          << elapsed << " ms" << endl;
56
57     // Simpan hasil ke file .txt
58     cout << endl
59          << "Apakah ingin menyimpan hasil ke file? (y/n): ";
60     string saveToFile;
61     cin >> saveToFile;
62     if (saveToFile == "y" || saveToFile == "Y")
63     {
64         cout << "Masukkan nama file: ";
65         string outputFile;
66         cin >> outputFile;
67         outputFile = "../test/" + outputFile;
68
69         ofstream outputFile(outputFile);
70         if (!outputFile.is_open())
71         {
72             cerr << "Gagal membuka file." << endl;
73             return 1;
74         }

```

```

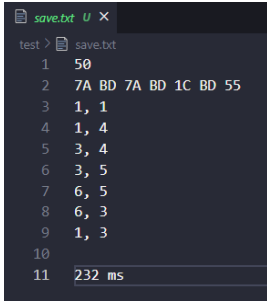
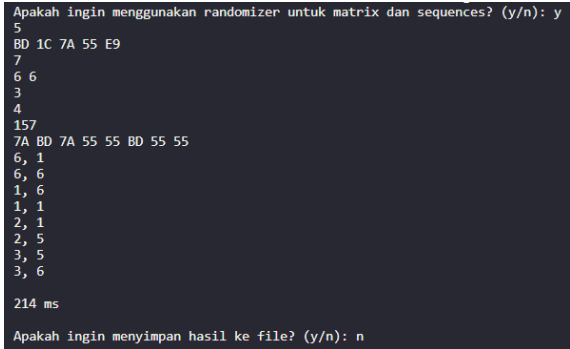
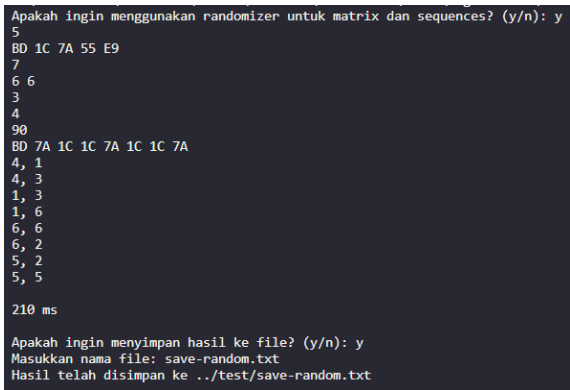
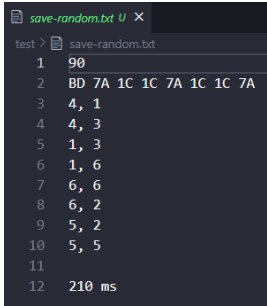
75
76 // Menyimpan maxReward dan solusi ke file
77 outputFile << solver.maxReward << endl;
78 for (const auto &token : solver.solutionBuffer)
79 {
80     outputFile << token.value << " ";
81 }
82
83 outputFile << endl;
84 for (const auto &token : solver.solutionBuffer)
85 {
86     outputFile << token.y + 1 << ", " << token.x + 1 << endl;
87 }
88
89 // Menambahkan waktu eksekusi ke akhir file
90 outputFile << endl
91     << elapsed << " ms";
92
93 outputFile.close();
94 cout << "Hasil telah disimpan ke " << outputFilename << endl;
95 }
96
97 return 0;
98 }

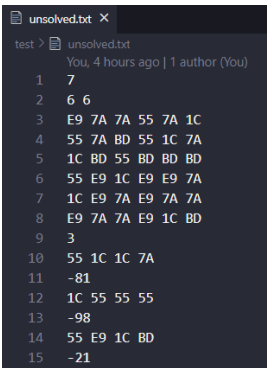
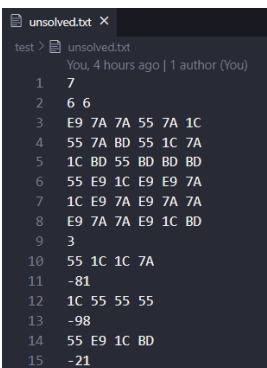
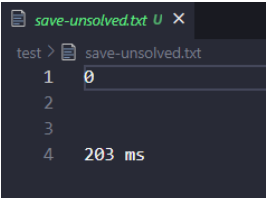
```

Gambar 3.4. Source Code main.cpp

C. Input dan Output

No.	Input	Output
1.		
2.		 <p>Hasil penyimpanan, save.txt:</p>

		 <pre> save.txt U X test > save.txt 1 50 2 7A BD 7A BD 1C BD 55 3 1, 1 4 1, 4 5 3, 4 6 3, 5 7 6, 5 8 6, 3 9 1, 3 10 11 232 ms </pre>
3.	<p>Jumlah token unik: 5</p> <p>Token: BD 1C 7A 55 E9</p> <p>Ukuran buffer: 7</p> <p>Ukuran matriks: 6 6</p> <p>Jumlah sekuens: 3</p> <p>Ukuran maksimal sekuens: 4</p>	 <pre> Apakah ingin menggunakan randomizer untuk matrix dan sequences? (y/n): y 5 BD 1C 7A 55 E9 7 6 6 3 4 157 7A BD 7A 55 55 BD 55 55 6, 1 6, 6 1, 6 1, 1 2, 1 2, 5 3, 5 3, 6 214 ms Apakah ingin menyimpan hasil ke file? (y/n): n </pre>
4.	<p>Jumlah token unik: 5</p> <p>Token: BD 1C 7A 55 E9</p> <p>Ukuran buffer: 7</p> <p>Ukuran matriks: 6 6</p> <p>Jumlah sekuens: 3</p> <p>Ukuran maksimal sekuens: 4</p>	 <pre> Apakah ingin menggunakan randomizer untuk matrix dan sequences? (y/n): y 5 BD 1C 7A 55 E9 7 6 6 3 4 90 BD 7A 1C 1C 7A 1C 1C 7A 4, 1 4, 3 1, 3 1, 6 6, 6 6, 2 5, 2 5, 5 210 ms Apakah ingin menyimpan hasil ke file? (y/n): y Masukkan nama file: save-random.txt Hasil telah disimpan ke ../test/save-random.txt </pre> <p>Hasil penyimpanan, save-random.txt:</p>  <pre> save-random.txt U X test > save-random.txt 1 90 2 BD 7A 1C 1C 7A 1C 1C 7A 3 4, 1 4 4, 3 5 1, 3 6 1, 6 7 6, 6 8 6, 2 9 5, 2 10 5, 5 11 12 210 ms </pre>

5.		<p>Apakah ingin menggunakan randomizer untuk matrix dan sequences? (y/n): n Masukkan nama file: unsolved.txt Tidak ada buffer dan sequence yang cocok. Tidak ada solusi yang ditemukan!</p> <p>211 ms</p> <p>Apakah ingin menyimpan hasil ke file? (y/n): n</p>
6.		<p>Apakah ingin menggunakan randomizer untuk matrix dan sequences? (y/n): n Masukkan nama file: unsolved.txt Tidak ada buffer dan sequence yang cocok. Tidak ada solusi yang ditemukan!</p> <p>203 ms</p> <p>Apakah ingin menyimpan hasil ke file? (y/n): y Masukkan nama file: save-unsolved.txt Hasil telah disimpan ke ../test/save-unsolved.txt</p> <p>Hasil penyimpanan, save-unsolved.txt:</p> 

Tabel 2. Input dan Output Program

D. Repository

Link Repository dari Tugas Kecil 01 IF2211 Strategi Algoritma adalah sebagai berikut.

https://github.com/amaliap21/Tucil1_13522042.git