

Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Penyelesaian Cyberpunk 2077 Breach Protocol dengan
Algoritma Brute Force
Semester II Tahun 2023/2024

Disusun Oleh:
Randy Verdian 13522067



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Bab 1

Deskripsi Masalah



Gambar 1 Permainan Breach Protocol

(Sumber: <https://cyberpunk.fandom.com/wiki/Quickhacking>)

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.

Yang akan dilakukan pada tugas ini adalah menemukan solusi dari **permainan Breach Protocol** yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan ***algoritma brute force***.

Bab 2

Algoritma Brute Force pada Penyelesaian Cyberpunk 2077 Breach Protocol

Algoritma *brute force* adalah algoritma yang mencoba semua kemungkinan solusi untuk menemukan solusi yang optimal atau memenuhi kriteria tertentu. Program cyberpunk 2077 breach protocol solver menggunakan algoritma *brute force* untuk menemukan kombinasi token yang memberikan solusi optimal dengan hadiah terbesar dalam permainan breach protocol. Permainan ini melibatkan sebuah matriks kode dan beberapa sekuens token yang memiliki bobot hadiah tertentu. Kemudian program akan mengecek semua kemungkinan kombinasi token satu per satu hingga memperoleh solusi sequence yang paling optimal.

Program ini memanggil fungsi rekursif untuk menghasilkan semua kemungkinan kombinasi token yang memenuhi aturan permainan. Fungsi rekursif melakukan hal berikut:

- Fungsi ini menambahkan token yang dipilih ke dalam kombinasi token sementara, menandai token tersebut sebagai terpilih, dan menyimpan koordinatnya.
- Jika ukuran kombinasi token sementara masih kurang dari buffer size, fungsi ini mengubah arah pemilihan token (dari baris ke kolom atau sebaliknya) dan mencoba semua kemungkinan token yang belum dipilih di baris atau kolom yang sama dengan token terakhir yang dipilih, dengan memanggil dirinya sendiri secara rekursif.
- Jika ukuran kombinasi token sementara sudah sama dengan buffer size, fungsi ini memeriksa apakah kombinasi tersebut valid, yaitu mengandung setidaknya satu sekuens token yang ada. Jika valid, fungsi ini menghitung nilai hadiah dari kombinasi tersebut dan membandingkannya dengan nilai hadiah maksimal. Jika lebih besar, fungsi ini mengganti kombinasi token optimal, koordinat token optimal, dan nilai hadiah maksimal dengan kombinasi token sementara, koordinat token sementara, dan nilai hadiah sementara.

Program ini mengembalikan sequence optimal, koordinat token dari sequence optimal, nilai hadiah maksimal, dan waktu eksekusi sebagai solusi.

Bab 3

Implementasi Program

1. File main.cpp

```
1 #include <iostream>
2 #include "IO.cpp"
3 using namespace std;
4
5 int main() {
6     cout << "-----" << endl;
7     cout << "                                Cyberpunk 2077 Breach Protocol Solver                                " << endl;
8     cout << "-----" << endl;
9     int choice;
10    string continueChoice;
11
12    do {
13        cout << endl << "Pilih menu:" << endl;
14        cout << "1. Matriks dan sekuens permainan dibaca dari file .txt" << endl;
15        cout << "2. Matriks dan sekuens permainan dihasilkan otomatis oleh program" << endl;
16        cout << "Masukkan menu yang Anda pilih: ";
17        cin >> choice;
18
19        while (!(choice == 1 || choice == 2)) {
20            cout << "Pilihan tidak valid!" << endl;
21            cout << "Masukkan menu yang Anda pilih: ";
22            cin >> choice;
23        }
24
25        if (choice == 1) {
26            input_file();
27        } else {
28            input_automatic();
29        }
30
31        cout << "Apakah Anda ingin melanjutkan permainan? (y/n): ";
32        cin >> continueChoice;
33
34    } while (continueChoice == "y");
35
36    return 0;
37 }
38
```

Penjelasan :

Menampilkan antarmuka pengguna untuk menyelesaikan permainan Cyberpunk 2077 Breach Protocol dengan dua pilihan masukan, yaitu membaca dari file atau menghasilkan matriks dan sekuens permainan secara otomatis.

2. File IO.cpp

```
1  #include <iostream>
2  #include <sstream>
3  #include <fstream>
4  #include "solve.cpp"
5  using namespace std;
6
7  void print_solution(vector<vector<string>>& arr, int buffer_size, vector<sequences>& seq, int matrix_width) {
8      cout << endl << "----- SOLUSI OPTIMAL -----" << endl;
9      vector<string> temp, optimal;
10     vector<pair<int, int>> optimal_coord, current_cord;
11     vector<vector<bool>> taken;
12     int time, max_reward;
13
14     solve(arr, buffer_size, seq, temp, optimal, optimal_coord, current_cord, matrix_width, max_reward, time);
15
16     if(optimal.size() != 0){
17         cout << "Reward terbesar: " << max_reward << endl;
18         cout << "Buffer: ";
19         print_vector(optimal);
20         print_coordinates(optimal_coord);
21     }
22     else{
23         cout << "Tidak ada solusi yang memenuhi." << endl;
24     }
25
26     cout << "Waktu eksekusi: " << time << " ms" << endl;
27     cout << "-----" << endl;
28
29     string s;
30     cout << endl << "Apakah Anda ingin menyimpan solusi? (y/n): ";
31     cin >> s;
32     if(s == "y"){
33         string outputFileName;
34         cout << "Masukkan nama file untuk menyimpan solusi: ";
35         cin >> outputFileName;
36
37         string filePath = "../test/output/" + outputFileName;
38
39         ofstream outputFile(filePath);
40         if (!outputFile.is_open()) {
41             cerr << "Tidak dapat membuat file." << endl;
42             return;
43         }
44         if(optimal.size() != 0){
45             outputFile << max_reward << endl;
46             for (string s : optimal) {
47                 outputFile << s << " ";
48             }
49             outputFile << endl;
50             for (const auto& coord : optimal_coord) {
51                 outputFile << coord.second + 1 << " ", " << coord.first + 1 << endl;
52             }
53             outputFile << endl;
54         }
55         else{
56             outputFile << "Tidak ada solusi yang memenuhi." << endl << endl;
57         }
58
59         outputFile << time << " ms";
60
61         outputFile.close();
62         cout << "Penyimpanan berhasil!" << endl;
63     }
64 }
65
```

```

1 void input_file() {
2     string filename;
3     ifstream inputFile;
4     bool is_valid_token = true;
5     bool is_valid_buffer_size = true;
6
7     do {
8         cout << "Masukkan nama file yang ingin dibaca: ";
9         cin >> filename;
10
11         string filePath = "../test/input/" + filename;
12
13         inputFile.open(filePath);
14
15         if (!inputFile.is_open()) {
16             cerr << "File tidak ditemukan." << endl;
17         }
18     } while (!inputFile.is_open());
19
20     int buffer_size;
21     int matrix_width, matrix_height;
22     inputFile >> buffer_size;
23     if(buffer_size <= 0){
24         is_valid_buffer_size = false;
25     }
26     inputFile >> matrix_width >> matrix_height;
27
28     vector<vector<string>> arr(matrix_height, vector<string>(matrix_width));
29
30     for (int i = 0; i < matrix_height; i++) {
31         for (int j = 0; j < matrix_width; j++) {
32             inputFile >> arr[i][j];
33             if(!is_two_char(arr[i][j])){
34                 is_valid_token = false;
35             }
36         }
37     }
38
39     int num_of_seq;
40     inputFile >> num_of_seq;
41     inputFile.ignore();
42
43     vector<sequences> seq;
44
45     for (int i = 0; i < num_of_seq; i++) {
46         string line;
47         getline(inputFile, line);
48         stringstream ss(line);
49         string seq_token;
50         sequences temp;
51         while (ss >> seq_token) {
52             temp.token.push_back(seq_token);
53         }
54         int seq_reward;
55         inputFile >> seq_reward;
56         temp.reward = seq_reward;
57         seq.push_back(temp);
58         inputFile.ignore();
59     }
60
61     if(!is_valid_token || !is_valid_buffer_size){
62         if(!is_valid_token && is_valid_buffer_size){
63             cout << "Token tidak valid, harus terdiri dari 2 karakter." << endl;
64         }
65         else if(is_valid_token && !is_valid_buffer_size){
66             cout << "Ukuran buffer harus lebih besar dari 0." << endl;
67         }
68         else{
69             cout << "Ada token yang tidak valid, setiap token harus terdiri dari 2 karakter." << endl;
70             cout << "Ukuran buffer harus lebih besar dari 0." << endl;
71         }
72         input_file();
73     }
74     else{
75         print_solution(arr, buffer_size, seq, matrix_width);
76     }
77 }
78

```

```

1 void input_automatic(){
2     int num_token, buffer_size, matrix_height, matrix_width, num_sequence, max_sequence_length;
3     vector<string> tokens;
4     vector<vector<string>> arr;
5     vector<sequences> seq;
6     mt19937 g(time(0));
7
8     cout << "Matriks dan sekuens permainan akan dibuat otomatis oleh program." << endl;
9     cout << "Jumlah token unik: ";
10    cin >> num_token;
11    bool is_valid_token = true;
12    do{
13        is_valid_token = true;
14        cout << "Token: ";
15        for(int i = 0; i < num_token; i++){
16            string token;
17            cin >> token;
18            if(!is_two_char(token)){
19                is_valid_token = false;
20            }
21            else{
22                tokens.push_back(token);
23            }
24        }
25        if(!is_valid_token){
26            cout << "Token tidak valid, harus terdiri dari 2 karakter." << endl;
27            tokens.clear();
28        }
29    } while(!is_valid_token);
30
31    cout << "Ukuran buffer: ";
32    cin >> buffer_size;
33    while (buffer_size <= 0) {
34        cout << "Ukuran buffer harus lebih besar dari 0. Silakan input kembali: ";
35        cin >> buffer_size;
36    }
37
38    cout << "Ukuran matriks (kolom, baris): ";
39    cin >> matrix_width >> matrix_height;
40    while (matrix_width <= 0 || matrix_height <= 0) {
41        cout << "Ukuran matriks harus lebih besar dari 0. Silakan input kembali: ";
42        cin >> matrix_width >> matrix_height;
43    }
44
45    cout << "Jumlah sekuens: ";
46    cin >> num_sequence;
47    while (num_sequence <= 0) {
48        cout << "Jumlah sekuens harus lebih besar dari 0. Silakan input kembali: ";
49        cin >> num_sequence;
50    }
51
52    cout << "Ukuran maksimal sekuens: ";
53    cin >> max_sequence_length;
54    while (max_sequence_length <= 1) {
55        cout << "Ukuran maksimal sekuens harus lebih besar dari 1. Silakan input kembali: ";
56        cin >> max_sequence_length;
57    }
58
59    seq = generate_random_sequences(num_sequence, max_sequence_length, tokens, g);
60    arr = generate_random_matrix(matrix_width, matrix_height, num_token, tokens);
61    print_matrix(arr);
62    print_sequences(seq);
63
64    print_solution(arr, buffer_size, seq, matrix_width);
65
66 }

```


Penjelasan :

1. `Print_solution` : Mencetak solusi optimal dari permainan yaitu reward terbesar, buffer, koordinat optimal, dan waktu eksekusi. Selain itu, memberikan opsi untuk menyimpan solusi ke dalam file.
2. `Input_file` : Meminta pengguna untuk memasukkan nama file yang berisi data input untuk permainan, lalu menjalankan fungsi `print_solution` dengan data yang dibaca.
3. `Input_automatic` : Menghasilkan matriks dan sekuens permainan secara otomatis berdasarkan input dari pengguna seperti jumlah token, ukuran buffer, ukuran matriks, jumlah sekuens, dan ukuran maksimal sekuens. Program kemudian mencetak matriks dan sekuens yang dihasilkan serta mengeksekusi fungsi `print_solution` dengan data yang telah dibuat otomatis.

3. File solve.cpp

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <chrono>
5  #include <random>
6  #include <ctime>
7  using namespace std;
8
9  struct sequences {
10     vector<string> token;
11     int reward;
12 };
13
14 bool is_two_char(string str) {
15     return str.length() == 2;
16 }
17
18 // Fungsi untuk mencetak vektor string
19 void print_vector(vector<string> v) {
20     for (string s : v) {
21         cout << s << " ";
22     }
23     cout << endl;
24 }
25
26 void print_coordinates(vector<pair<int, int>>& coordinates) {
27     cout << "Koordinat: " << endl;
28     for (const auto& coord : coordinates) {
29         cout << "(" << coord.second + 1 << ", " << coord.first + 1 << ") " << endl;
30     }
31     cout << endl;
32 }
33
34 bool contains_sequence(const vector<string>& v, const sequences& s) {
35     if(v.size() >= s.token.size()){
36         for (size_t i = 0; i <= v.size() - s.token.size(); ++i) {
37             auto it = search(v.begin() + i, v.end(), s.token.begin(), s.token.end());
38             if (it != v.end()) {
39                 return true;
40             }
41         }
42         return false;
43     }
44     return false;
45 }
46
47
48 // Fungsi untuk menghitung total bobot hadiah dari sebuah vektor string berdasarkan sekuens-sekuens yang ada
49 int calculate_reward(vector<string>& v, vector<sequences>& seq) {
50     int total = 0;
51     for (sequences s : seq) {
52         if (contains_sequence(v, s)) {
53             total += s.reward;
54         }
55     }
56     return total;
57 }
```

```

1 // Fungsi untuk menghasilkan semua kemungkinan kombinasi token yang memenuhi aturan permainan
2 void generate_combinations(vector<vector<string>>& arr, int& buffer_size, vector<sequences>& seq, vector<string>& temp, vector<string>& optimal,
3     vector<pair<int, int>>& optimal_coord, vector<pair<int, int>>& current_coord, vector<vector<bool>>& taken, int row, int col, int dir, int& max_reward) {
4
5     temp.push_back(arr[row][col]);
6     taken[row][col] = true;
7     current_coord.push_back(make_pair(row, col));
8
9     if (temp.size() < buffer_size){
10
11         dir = 1 - dir;
12         // Jika horizontal, coba semua kemungkinan kolom di baris yang sama
13         if (dir == 0) {
14             for (int j = 0; j < arr[0].size(); j++) {
15                 if (j != col && !taken[row][j]) {
16                     generate_combinations(arr, buffer_size, seq, temp, optimal, optimal_coord, current_coord, taken, row, j, dir, max_reward);
17                 }
18             }
19         }
20         // Jika vertikal, coba semua kemungkinan baris di kolom yang sama
21         if (dir == 1) {
22             for (int i = 0; i < arr.size(); i++) {
23                 if (i != row && !taken[i][col]) {
24                     generate_combinations(arr, buffer_size, seq, temp, optimal, optimal_coord, current_coord, taken, i, col, dir, max_reward);
25                 }
26             }
27         }
28     }
29 }
30
31 // Jika size temp udah maksimal
32 else if (temp.size() == buffer_size) {
33     bool valid = false;
34     for(int j = 0; j < seq.size(); j++){
35         if(contains_sequence(temp, seq[j])){
36             valid = true;
37             break;
38         }
39     }
40     if(valid){
41         if(calculate_reward(temp, seq) > max_reward){
42             max_reward = calculate_reward(temp, seq);
43             optimal = temp;
44             optimal_coord = current_coord;
45         }
46     }
47 }
48 }
49
50 taken[row][col] = false;
51 temp.pop_back();
52 current_coord.pop_back();
53 }
54 }
55
56
57 void solve(vector<vector<string>>& arr, int& buffer_size, vector<sequences>& seq, vector<string>& temp, vector<string>& optimal,
58     vector<pair<int, int>>& optimal_coord, vector<pair<int, int>>& current_coord, int matrix_width, int& max_reward, int& time){
59     max_reward = 0;
60     vector<vector<bool>> taken(arr.size(), vector<bool>(arr[0].size(), false));
61
62     auto start_time = chrono::high_resolution_clock::now();
63
64     // mulai dari baris pertama
65     for (int j = 0; j < matrix_width; j++) {
66         generate_combinations(arr, buffer_size, seq, temp, optimal, optimal_coord, current_coord, taken, 0, j, 0, max_reward);
67     }
68
69     auto end_time = chrono::high_resolution_clock::now();
70     auto duration = chrono::duration_cast<chrono::milliseconds>(end_time - start_time);
71     time = duration.count();
72
73 }
74 }

```

```

1 // Fungsi untuk menghasilkan matriks secara acak
2 vector<vector<string>> generate_random_matrix(int matrix_width, int matrix_height, int num_token, vector<string> tokens) {
3
4     srand(time(0));
5     random_shuffle(tokens.begin(), tokens.end());
6
7     vector<vector<string>> matrix(matrix_height, vector<string>(matrix_width));
8     int token_index = 0;
9     for (int i = 0; i < matrix_height; ++i) {
10         for (int j = 0; j < matrix_width; ++j) {
11             matrix[i][j] = tokens[token_index];
12             ++token_index;
13             if (token_index == num_token) {
14                 token_index = 0;
15                 random_shuffle(tokens.begin(), tokens.end());
16             }
17         }
18     }
19
20     return matrix;
21 }
22
23
24 vector<sequences> generate_random_sequences(int num_sequence, int max_sequence_length, vector<string> tokens, mt19937& g) {
25     vector<sequences> random_sequences;
26
27     // Buat distribusi untuk panjang sekuens dan hadiah
28     uniform_int_distribution<int> num_tokens_distribution(2, max_sequence_length);
29     uniform_int_distribution<int> reward_distribution(10, 50);
30     uniform_int_distribution<int> token_distribution(0, tokens.size() - 1);
31
32     for (int i = 0; i < num_sequence; ++i) {
33         sequences seq;
34         int seq_length = num_tokens_distribution(g);
35
36         // Ambil token secara acak dari vektor tokens
37         for (int j = 0; j < seq_length; ++j) {
38             int token_index = token_distribution(g);
39             seq.token.push_back(tokens[token_index]);
40         }
41
42         seq.reward = reward_distribution(g);
43         random_sequences.emplace_back(seq);
44     }
45
46     return random_sequences;
47 }
48
49
50 // Fungsi untuk mencetak matriks
51 void print_matrix(const vector<vector<string>>& matrix) {
52     cout << "Matriks:" << endl;
53     for (const auto& row : matrix) {
54         for (const string& cell : row) {
55             cout << cell << " ";
56         }
57         cout << endl;
58     }
59 }
60
61 // Fungsi untuk mencetak sequence
62 void print_sequences(const vector<sequences>& seq) {
63     cout << "Sequence dan reward: " << endl;
64     for (const sequences& s : seq) {
65         print_vector(s.token);
66         cout << s.reward << endl;
67     }
68 }

```

Penjelasan :

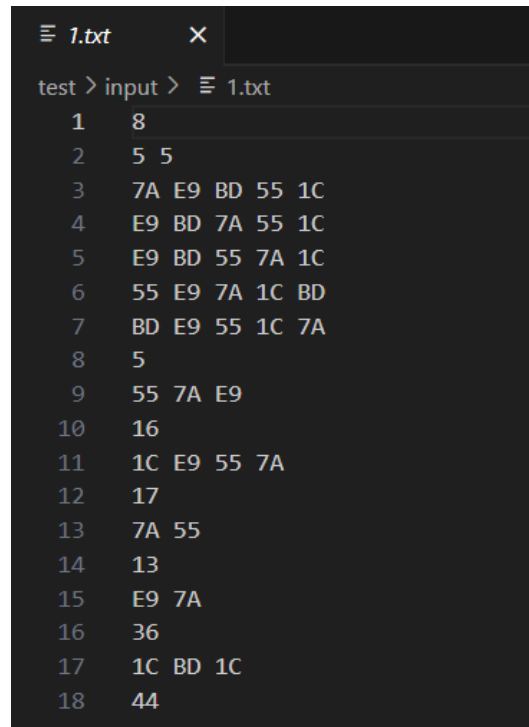
1. `print_vector`: Mencetak elemen-elemen vektor string.
2. `print_coordinates`: Mencetak koordinat yang disimpan dalam vektor pasangan bilangan bulat.
3. `contains_sequence`: Memeriksa keberadaan sekuens dalam vektor string.
4. `calculate_reward`: Menghitung total hadiah berdasarkan sekuens yang ada dalam vektor string.
5. `generate_combinations`: Menentukan solusi sequence optimal berdasarkan aturan permainan dengan menggunakan algoritma brute force.
6. `solve`: Menyelesaikan permainan dengan mengembalikan solusi optimal, hadiah maksimal, koordinat, dan waktu eksekusi.
7. `generate_random_matrix`: Membuat matriks secara acak.
8. `generate_random_sequences`: Membuat sequence secara acak.
9. `print_matrix`: Mencetak elemen-elemen matriks.
10. `print_sequences`: Mencetak sekuens beserta hadiahnya.

Bab 4

Uji Coba Program

1. Input dari file dan menyimpan ke file

- Input



```
test > input > 1.txt
1 8
2 5 5
3 7A E9 BD 55 1C
4 E9 BD 7A 55 1C
5 E9 BD 55 7A 1C
6 55 E9 7A 1C BD
7 BD E9 55 1C 7A
8 5
9 55 7A E9
10 16
11 1C E9 55 7A
12 17
13 7A 55
14 13
15 E9 7A
16 36
17 1C BD 1C
18 44
```

- Output

```
Pilih menu:
1. Matriks dan sekuens permainan dibaca dari file .txt
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program
Masukkan menu yang Anda pilih: 1
Masukkan nama file yang ingin dibaca: 1.txt

----- SOLUSI OPTIMAL -----
Reward terbesar: 109
Buffer: 1C BD 1C 55 7A E9 7A 55
Koordinat:
(5, 1)
(5, 4)
(4, 4)
(4, 1)
(1, 1)
(1, 2)
(3, 2)
(3, 3)

Waktu eksekusi: 677 ms

-----
Apakah Anda ingin menyimpan solusi? (y/n): y
Masukkan nama file untuk menyimpan solusi: 1_sol.txt
Penyimpanan berhasil!
Apakah Anda ingin melanjutkan permainan? (y/n): ☐
```

2. Input dari file dan menyimpan ke file
 - Input

```
≡ 2.txt ×
test > input > ≡ 2.txt
1 9
2 3 8
3 66 88 XY
4 Y4 AB 9E
5 XY AB Y4
6 88 66 9E
7 AB 66 XY
8 9E 88 Y4
9 XY 9E 88
10 Y4 66 AB
11 6
12 AB 9E 66
13 45
14 66 88
15 46
16 AB 9E
17 11
18 Y4 9E Y4
19 45
20 66 66
21 40
22 AB 66 66
23 44
```

- Output


```

Pilih menu:
1. Matriks dan sekuens permainan dibaca dari file .txt
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program
Masukkan menu yang Anda pilih: 1
Masukkan nama file yang ingin dibaca: 2.txt

----- SOLUSI OPTIMAL -----
Reward terbesar: 186
Buffer: 66 AB 66 66 AB 9E 66 88 XY
Koordinat:
(1, 1)
(1, 5)
(2, 5)
(2, 8)
(3, 8)
(3, 4)
(2, 4)
(2, 1)
(3, 1)

waktu eksekusi: 1129 ms

-----
Apakah Anda ingin menyimpan solusi? (y/n): y
Masukkan nama file untuk menyimpan solusi: 2_sol.txt
Penyimpanan berhasil!
Apakah Anda ingin melanjutkan permainan? (y/n): 

```

3. Input dari file dengan validasi kesalahan

- Input

```

≡ 3.txt ×
test > input > ≡ 3.txt
1 10
2 8 4
3 B F A D E C C A
4 E B F D B F D C
5 A E B E D C F A
6 F D C E B A F A
7 4
8 E E A D D
9 -23
10 C F
11 -38
12 A B B A C
13 29
14 A F
15 16

```

- Output

```
Pilih menu:
1. Matriks dan sekuens permainan dibaca dari file .txt
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program
Masukkan menu yang Anda pilih: 1
Masukkan nama file yang ingin dibaca: 3.txt
Token tidak valid, harus terdiri dari 2 karakter.
Masukkan nama file yang ingin dibaca: 
```

4. Input dari file dan tidak menghasilkan solusi optimal dan menyimpan ke file

- Input

```
test > input > 4.txt
1 6
2 6 6
3 TY QW P9 87 87 QW
4 P9 TY QW P9 87 TY
5 QW TY 87 P9 TY 87
6 P9 QW 87 P9 TY QW
7 QW P9 TY 87 QW TY
8 87 P9 TY QW P9 87
9 4
10 P9 87 TY P9 P9 TY QW
11 46
12 P9 P9 QW TY 87 87 87
13 40
14 TY QW QW P9 TY TY P9
15 11
16 QW QW TY TY 87 87 QW
17 37
```

- Output

```
Pilih menu:
1. Matriks dan sekuens permainan dibaca dari file .txt
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program
Masukkan menu yang Anda pilih: 1
Masukkan nama file yang ingin dibaca: 4.txt

----- SOLUSI OPTIMAL -----
Tidak ada solusi yang memenuhi.
Waktu eksekusi: 13 ms
-----

Apakah Anda ingin menyimpan solusi? (y/n): 
```

5. Input dari CLI dan tidak menyimpan ke file

- Input

```
Pilih menu:  
1. Matriks dan sekuens permainan dibaca dari file .txt  
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program  
Masukkan menu yang Anda pilih: 2  
Matriks dan sekuens permainan akan dibuat otomatis oleh program.  
Jumlah token unik: 6  
Token: AB CD EF GH IJ KL  
Ukuran buffer: 7  
Ukuran matriks (kolom, baris): 6 6  
Jumlah sekuens: 30  
Ukuran maksimal sekuens: 4
```

- Output

```
Matriks:  
GH CD KL IJ EF AB  
CD KL GH AB IJ EF  
AB KL CD EF IJ GH  
AB IJ EF KL GH CD  
AB GH KL EF IJ CD  
GH IJ AB EF CD KL
```

```
Sequence dan reward:  
CD EF  
50  
KL KL AB  
17  
CD EF EF AB  
34  
IJ KL  
32  
IJ EF CD GH  
15  
AB KL EF CD  
14  
IJ KL AB  
21  
IJ EF KL AB  
41  
IJ GH AB  
36  
IJ AB  
45  
IJ AB  
48
```

IJ GH EF
14
EF GH
22
AB IJ
39
GH AB IJ KL
25
EF EF AB AB
24
KL GH CD GH
42
KL IJ EF
17
EF GH
39
KL EF CD EF
37
IJ IJ IJ
27
KL IJ
25
AB KL CD
30
CD KL
45
AB AB KL AB
14
CD IJ AB CD
43
KL AB EF CD
28
KL EF IJ
21
GH KL CD
29
AB AB EF GH
18

```

----- SOLUSI OPTIMAL -----
Reward terbesar: 249
Buffer: CD KL CD EF GH IJ AB
Koordinat:
(2, 1)
(2, 3)
(3, 3)
(3, 4)
(5, 4)
(5, 2)
(4, 2)

Waktu eksekusi: 6655 ms
-----

Apakah Anda ingin menyimpan solusi? (y/n): n
Apakah Anda ingin melanjutkan permainan? (y/n): █

```

6. Input dari CLI dan menyimpan ke file

- Input

```

Pilih menu:
1. Matriks dan sekuens permainan dibaca dari file .txt
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program
Masukkan menu yang Anda pilih: 2
Matriks dan sekuens permainan akan dibuat otomatis oleh program.
Jumlah token unik: 5
Token: 11 22 33 44 55
Ukuran buffer: 7
Ukuran matriks (kolom, baris): 7 7
Jumlah sekuens: 7
Ukuran maksimal sekuens: 6

```

- Output

```

Matriks:
55 44 11 22 33 22 55
33 44 11 33 22 55 11
44 33 22 55 44 11 55
33 11 44 22 22 33 11
44 55 22 33 55 44 11
22 33 44 11 55 55 11
33 22 44 11 44 55 33
Sequence dan reward:
22 11 44 11
29
44 33 55
40
11 55 33 44 55 55
11
22 22 33 33 44 22
34
33 11
35
11 22 22 22 33 44
20
11 33
46

```

```

----- SOLUSI OPTIMAL -----
Reward terbesar: 121
Buffer: 55 33 11 33 44 33 55
Koordinat:
(1, 1)
(1, 2)
(7, 2)
(7, 7)
(5, 7)
(5, 1)
(7, 1)

Waktu eksekusi: 4530 ms
-----

Apakah Anda ingin menyimpan solusi? (y/n): y
Masukkan nama file untuk menyimpan solusi: 5_sol.txt
Penyimpanan berhasil!
Apakah Anda ingin melanjutkan permainan? (y/n): 

```

7. Input dari CLI dengan validasi kesalahan

```

Pilih menu:
1. Matriks dan sekuens permainan dibaca dari file .txt
2. Matriks dan sekuens permainan dihasilkan otomatis oleh program
Masukkan menu yang Anda pilih: 2
Matriks dan sekuens permainan akan dibuat otomatis oleh program.
Jumlah token unik: 9
Token: BJ HI KO PQ N 8U 9W 9K OK
Token tidak valid, harus terdiri dari 2 karakter.
Token: BJ HI KO PQ NW 8U 9W 9K OK
Ukuran buffer: -8
Ukuran buffer harus lebih besar dari 0. Silakan input kembali: 8
Ukuran matriks (kolom, baris): 6 6
Jumlah sekuens: 

```

Bab 5

Penutup

5.1. Kesimpulan

Penggunaan algoritma *brute force* dalam Cyberpunk 2077 Breach Protocol Solver menunjukkan pendekatan yang sederhana namun efektif untuk menemukan solusi optimal. Meskipun algoritma ini dapat menangani berbagai kasus permainan, termasuk matriks dan sekuens permainan yang kompleks, namun perlu diingat bahwa algoritma *brute force* cenderung memiliki kompleksitas waktu yang tinggi, terutama saat mencari kombinasi sekuens optimal dalam ruang solusi yang besar.

5.2. Lampiran

- Link *repository* github : https://github.com/randyver/Tucil1_13522067.git
- *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		✓