

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma Semester  
II tahun 2022/2023

# **Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force**



Elbert Chailes – 13522045

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

# DAFTAR ISI

DAFTAR ISI.....	2
BAB I : DESKRIPSI MASALAH.....	3
BAB II : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL.....	4
BAB III : IMPLEMENTASI PROGRAM DENGAN PYTHON DAN JAVASCRIPT.....	7
1. Framework Backend Flask ( Bahasa Pemrograman Python ).....	7
a. app.py.....	7
b. functions.py.....	12
2. Framework Frontend ReactJS ( Bahasa Pemrograman Javascript ).....	15
a. App.jsx.....	15
BAB IV : EKSPERIMEN.....	17
a. Masukan acak oleh program.....	17
i. Contoh 1.....	17
ii. Contoh 2.....	18
iii. Contoh 3.....	19
iv. Contoh 4.....	20
v. Contoh 5.....	21
vi. Contoh 6.....	22
b. Masukan dengan file .txt.....	24
i. Contoh 7.....	24
ii. Contoh 8.....	25
LAMPIRAN.....	27
Github Repository.....	27
Tabel Spesifikasi.....	27

# BAB I : DESKRIPSI MASALAH

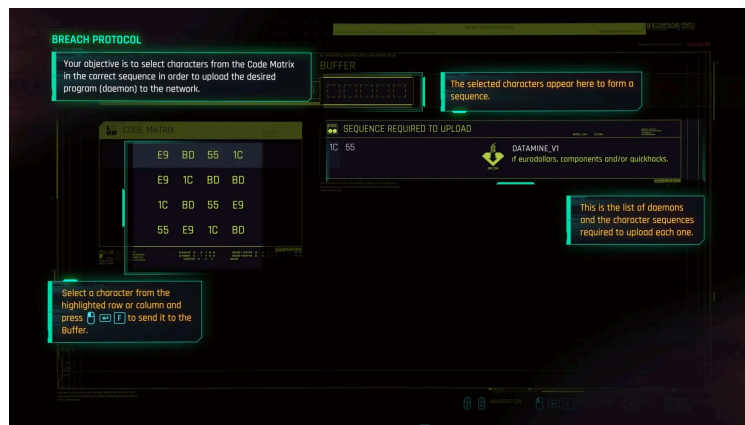
**Cyberpunk 2077 Breach Protocol** adalah minigame meretas pada permainan video Cyberpunk 2077. Minigame ini merupakan simulasi peretasan jaringan local dari ICE (Intrusion Countermeasures Electronics) pada permainan Cyberpunk 2077. Komponen pada permainan ini antara lain adalah:

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.

Laporan ini akan membahas program yang dapat digunakan untuk mencari solusi optimal untuk menyelesaikan permainan *Breach Protocol* ini dengan algoritma bruteforce.



Gambar 1.1. Permainan Cyberpunk 2077 Breach Protocol

## BAB II : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL

Algoritma bruteforce dapat digunakan untuk menyelesaikan permainan Cyberpunk 2077 Breach Protocol dengan mencoba semua kemungkinan yang mungkin terjadi pada sebuah matriks dengan panjang *buffer* yang telah ditentukan oleh pemain. Solusi atas algoritma bruteforce ini merupakan solusi dengan *sequence* yang memiliki *reward* dengan jumlah paling besar dan langkah yang ditempuh paling sedikit (optimal). Langkah-langkah penerapan algoritma bruteforce adalah sebagai berikut.

1. Pastikan terdapat informasi yang digunakan sebagai parameter pada fungsi bruteforce, yaitu panjang *buffer* maksimal, matriks yang akan dilakukan bruteforce, penandaan pada fungsi agar perlakuan bruteforce pertama dilakukan secara vertikal yang mana ditandai dengan boolean True (gerakan vertikal) dan False (gerakan horizontal), *sequence-sequence* yang hendak dicari beserta dengan *reward* yang diberikan untuk setiap *sequence*, panjang matriks, dan lebar matriks.
2. Kemudian, dilakukan algoritma bruteforce terhadap informasi yang diterima dengan menggunakan fungsi rekursif dengan tujuan agar semua kemungkinan dapat dicapai.
  - a. Basis : Jika ukuran buffer yang dicari sudah bernilai nol, maka artinya *sequence* yang dicari sudah sepanjang buffer yang diminta oleh user. Pada basis ini pula, *reward* dihitung untuk setiap kemungkinan yang sudah sepanjang panjang buffer yang diminta user dan kemudian dibandingkan satu antar lainnya. Selain itu, *reward* tersebut dijadikan sebagai perbandingan, serta indeks terakhir langkah yang ditempuh algoritma yang mengandung *sequence* juga dibandingkan, agar jalan tempuh algoritma dengan *reward* tertinggi dan langkah dengan indeks terkecil disimpan pada global variabel sehingga dapat ditemukan kemungkinan gerakan yang paling optimal.
  - b. Rekurens :
    - i. Jika sebelumnya telah melakukan gerakan horizontal atau sedang merupakan gerakan pertama, maka sekarang algoritma akan melakukan

jalur vertikal. Kemudian, algoritma akan melakukan bruteforce untuk setiap kemungkinan yang terdapat pada kolom vertikal. Jika titik sebelumnya pernah dilalui oleh algoritma, maka titik tersebut akan disimpan di sebuah variabel sehingga tidak akan dilewati lagi oleh algoritma.

- ii. Jika sebelumnya telah melakukan gerakan vertikal, maka sekarang algoritma akan melakukan jalur horizontal. Kemudian, algoritma akan melakukan bruteforce untuk setiap kemungkinan yang terdapat pada baris horizontal. Jika titik sebelumnya sudah pernah dilalui oleh algoritma, maka titik tersebut akan disimpan di sebuah variabel sehingga tidak akan dilewati lagi oleh algoritma.
  - iii. Untuk kedua kasus jika-maka pada rekurens (i) dan (ii), kedua ukuran buffer yang menjadi parameter pada fungsi rekursif akan berkurang satu pada setiap rekursif agar pada akhirnya basisnya (parameter ukuran buffer) menjadi 0 sehingga langkah yang ditempuh algoritma menjadi sepanjang ukuran buffer.
3. Setiap kali algoritma selesai menempuh sebanyak panjang buffer atau sudah melalui langkah tersebut, maka algoritma akan mundur hingga jalur yang belum pernah dilalui oleh algoritma atau dikenal dengan istilah *backtrack*. Contohnya, AC BD EF sudah dilalui, maka akan mencari jalur lain seperti AC BD EG.
  4. Algoritma bruteforce tersebut juga akan langsung melakukan pengecekan dengan langkah pertama dari baris pertama akibat bruteforce yang telah dilakukan melakukan perulangan secara horizontal jika gerakan pertama merupakan gerakan vertikal.
  5. Setelah seluruh kemungkinan dilalui oleh algoritma, maka sesuai dengan perbandingan yang terjadi pada *basis rekursi*, maka akan didapatkan indeks terakhir optimal (untuk mengetahui langkah yang paling sedikit) untuk mencapai *reward* yang maksimal pula. Maka, hasil algoritma bruteforce yang sepanjang ukuran *buffer* yang dimasukkan user, akan diambil hanya sesuai dengan indeks terakhir optimal. Dengan kata lain, hasil dari algoritma bruteforce menghasilkan token-token yang dilewati beserta koordinatnya dalam matriks untuk mencapai *reward* yang setinggi-tingginya dengan langkah yang sedikit-dikitnya.

Algoritma bruteforce ini tidak menyimpan seluruh kemungkinan, melainkan melakukan perbandingan pada *basis rekursi* setiap ditemukan langkah baru yang dapat ditempuh algoritma. Dengan mekanisme perbandingan melalui *basis rekursi*, maka kompleksitas dari program akan menjadi lebih efisien.

## BAB III : IMPLEMENTASI PROGRAM DENGAN PYTHON DAN JAVASCRIPT

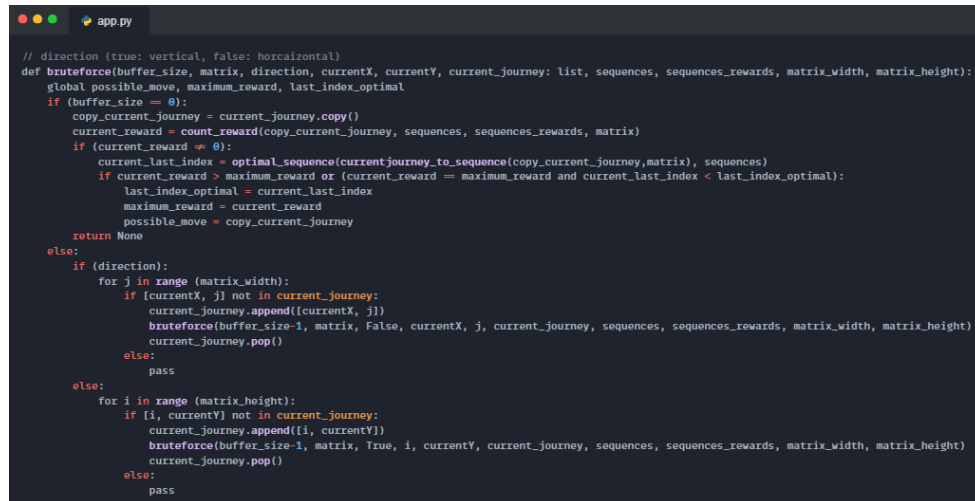
Algoritma pada Bab II diimplementasikan dengan menggunakan program yang ditulis dengan bahasa Python. Namun, beberapa fungsi juga diterapkan dengan menggunakan bahasa Javascript yang diproses di *client-side* dikarenakan GUI program ini menggunakan gabungan teknologi ReactJS sebagai framework *frontend* dan Flask sebagai framework *backend*. Algoritma utama untuk menyelesaikan permasalahan Breach Protocol ditulis dengan menggunakan bahasa pemrograman Python.

Terdapat dua file program yang ditulis dengan Python, yaitu `app.py` dan `functions.py`. Sedangkan, untuk pemrograman tampilan GUI website dibuat dengan ReactJS sehingga fungsi-fungsi pemrosesan masukan acak seperti *randomizer* ditulis dengan bahasa pemrograman Javascript. Fungsi-fungsi yang digunakan dan krusial dalam program ini akan dideskripsikan sebagai berikut.

### 1. Framework Backend Flask ( Bahasa Pemrograman Python )

#### a. `app.py`

`app.py` memuat kode-kode program yang berfungsi sebagai API (Application Programming Interface) yang menjadi media bagi *frontend* untuk berinteraksi dengan *backend* sehingga segala bentuk logika penting seperti perlakuan bruteforce dilakukan pada *backend*. Agar informasi yang dimasukkan user, yaitu masukan user seperti panjang maksimal *buffer*, *matrix*, *sequences*, hadiah *sequences*, dan masukan lainnya dapat disampaikan ke *backend website* sehingga dapat diproses dengan menggunakan Python. Pada file ini juga, terdapat API untuk mengirim kembali informasi yang telah diproses ke *frontend*.



```

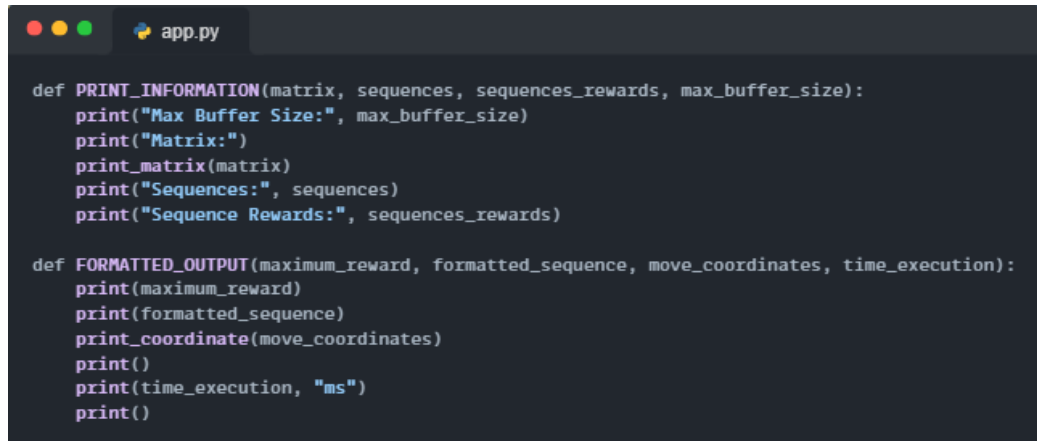
// direction (true: vertical, false: horizontal)
def bruteforce(buffer_size, matrix, direction, currentX, currentY, current_journey: List, sequences, sequences_rewards, matrix_width, matrix_height):
    global possible_move, maximum_reward, last_index_optimal
    if (buffer_size == 0):
        copy_current_journey = current_journey.copy()
        current_reward = count_reward(copy_current_journey, sequences, sequences_rewards, matrix)
        if (current_reward > 0):
            current_last_index = optimal_sequence(current_journey_to_sequence(copy_current_journey, matrix), sequences)
            if current_reward > maximum_reward or (current_reward == maximum_reward and current_last_index < last_index_optimal):
                last_index_optimal = current_last_index
                maximum_reward = current_reward
                possible_move = copy_current_journey
            return None
        else:
            if (direction):
                for j in range (matrix_width):
                    if [currentX, j] not in current_journey:
                        current_journey.append([currentX, j])
                        bruteforce(buffer_size-1, matrix, False, currentX, j, current_journey, sequences, sequences_rewards, matrix_width, matrix_height)
                        current_journey.pop()
                    else:
                        pass
            else:
                for i in range (matrix_height):
                    if [i, currentY] not in current_journey:
                        current_journey.append([i, currentY])
                        bruteforce(buffer_size-1, matrix, True, i, currentY, current_journey, sequences, sequences_rewards, matrix_width, matrix_height)
                        current_journey.pop()
                    else:
                        pass

```

Gambar 3.1. Cuplikan kode fungsi bruteforce

Prosedur bruteforce yang terdapat pada **Gambar 3.1.** dibuat dengan tujuan untuk mencari langkah yang paling optimal untuk mendapatkan skor paling maksimum. Pada prosedur tersebut, terdapat variabel *global*, seperti *maximum\_reward* untuk menyimpan skor maksimum sehingga dapat dilakukan perbandingan antara kemungkinan langkah yang satu dengan yang lainnya dan juga sebagai perbandingan antara langkah yang mana lebih optimal, *possible\_move* untuk menyimpan kemungkinan gerakan yang dapat ditempuh dengan mematuhi aturan Breach Protocol yaitu gerakan harus dimulai dengan gerakan vertikal lalu horizontal kemudian selang-seling (*possible\_move* ini bersifat *temporary* sehingga akan selalu berubah-ubah setelah dilakukan pengolahan informasi atas gerakan yang satu dan kemudian digantikan dengan kemungkinan gerakan yang lain), dan *last\_index\_optimal* untuk menyimpan index terakhir pada *possible\_move* yang mengandung *sequence* yang memberikan *reward* sehingga output akhir dari bruteforce adalah gerakan mungkin yang menaati aturan Breach Protocol dengan nilai *reward* yang maksimum dan juga dengan langkah yang paling optimal (langkah yang paling sedikit).





```

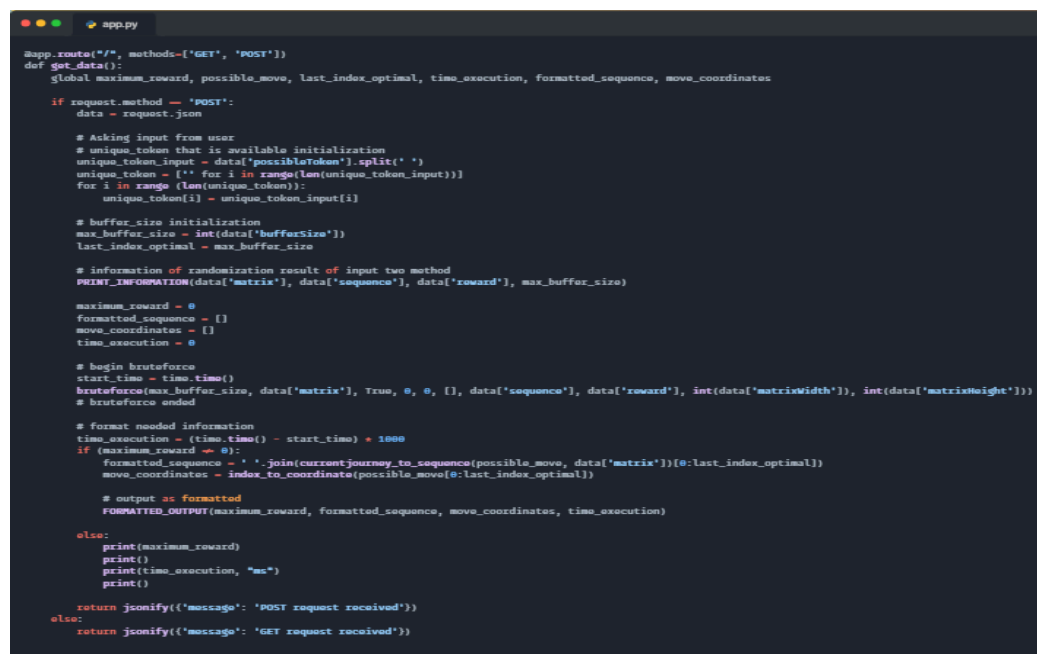
def PRINT_INFORMATION(matrix, sequences, sequences_rewards, max_buffer_size):
    print("Max Buffer Size:", max_buffer_size)
    print("Matrix:")
    print_matrix(matrix)
    print("Sequences:", sequences)
    print("Sequence Rewards:", sequences_rewards)

def FORMATTED_OUTPUT(maximum_reward, formatted_sequence, move_coordinates, time_execution):
    print(maximum_reward)
    print(formatted_sequence)
    print_coordinate(move_coordinates)
    print()
    print(time_execution, "ms")
    print()

```

*Gambar 3.2. Cuplikan kode prosedur untuk mengeluarkan output*

Prosedur `PRINT_INFORMATION` dan `FORMATTED_OUTPUT` yang terdapat pada **Gambar 3.2.** merupakan prosedur yang statis dan hanya menerima parameter yang sudah ada. Prosedur ini bertujuan untuk menampilkan hasil dari pembacaan dan menampilkan hasil pengolahan atau penyelesaian masalah yang telah diformat seperti yang diberikan pada spesifikasi tugas. Hasil yang didapatkan atau dikeluarkan oleh program akan ditampilkan pada CLI (Command Line Interface) yang digunakan untuk menjalankan backend server dengan Flask.



```

@app.route('/', methods=['GET', 'POST'])
def get_data():
    global maximum_reward, possible_move, last_index_optimal, time_execution, formatted_sequence, move_coordinates

    if request.method == 'POST':
        data = request.json

        # Asking input from user
        # unique_token that is available initialization
        unique_token_input = data['possibleToken'].split(' ')
        unique_token = ['' for i in range(len(unique_token_input))]
        for i in range(len(unique_token)):
            unique_token[i] = unique_token_input[i]

        # buffer_size initialization
        max_buffer_size = int(data['bufferSize'])
        last_index_optimal = max_buffer_size

        # information of randomization result of input two method
        PRINT_INFORMATION(data['matrix'], data['sequence'], data['reward'], max_buffer_size)

        maximum_reward = 0
        formatted_sequence = []
        move_coordinates = []
        time_execution = 0

        # begin bruteForce
        start_time = time.time()
        bruteForce(max_buffer_size, data['matrix'], True, 0, 0, [], data['sequence'], data['reward'], int(data['matrixWidth']), int(data['matrixHeight']))
        # bruteForce ended

        # format needed information
        time_execution = (time.time() - start_time) * 1000
        if (maximum_reward == 0):
            formatted_sequence = ' '.join(currentJourney_to_sequence(possible_move, data['matrix'])[0:last_index_optimal])
            move_coordinates = index_to_coordinate(possible_move[0:last_index_optimal])

        # output as formatted
        FORMATTED_OUTPUT(maximum_reward, formatted_sequence, move_coordinates, time_execution)

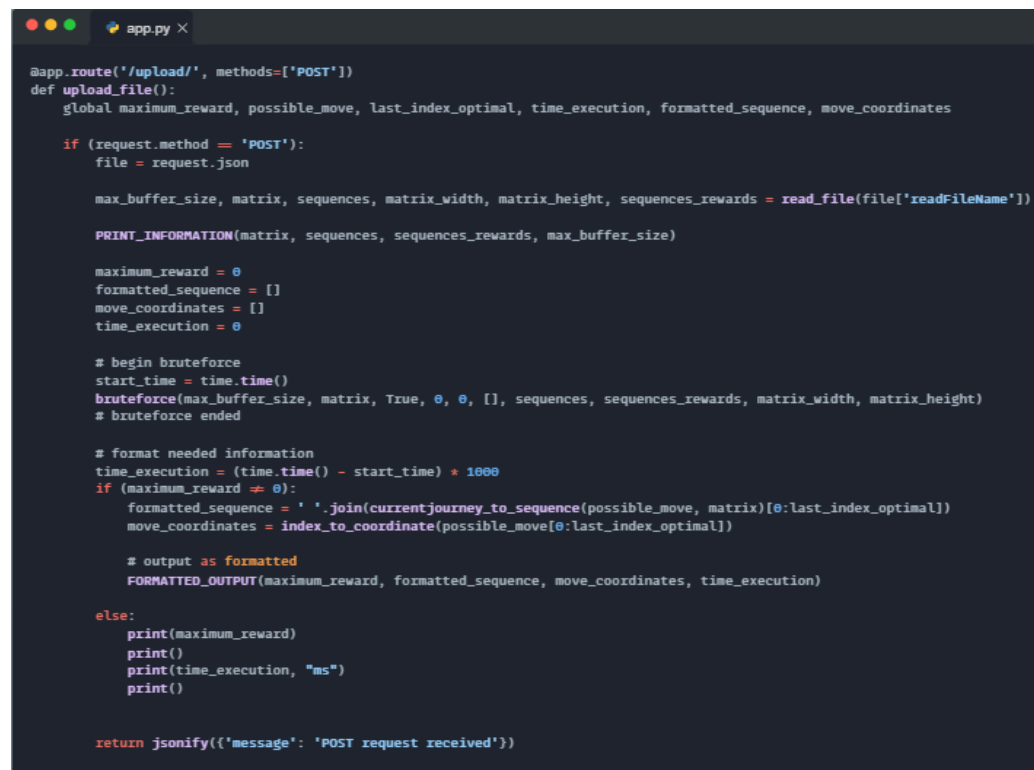
    else:
        print(maximum_reward)
        print()
        print(time_execution, "ms")
        print()

    return jsonify({'message': 'POST request received'})
else:
    return jsonify({'message': 'GET request received'})

```

*Gambar 3.3. Cuplikan kode API get\_data()*

API `get_data()` yang terdapat pada **Gambar 3.3**. Merupakan API yang bertujuan untuk mengolah data-data yang diberikan oleh user melalui UI *frontend* kemudian diolah dan mengubah variabel global yang telah diinisialisasi sebelumnya dengan menggunakan variabel global yang terdapat pada fungsi `bruteforce` yang telah dijelaskan sebelumnya. Perhitungan akan data-data yang diterima dan pembacaan data-data yang diberikan terkait ukuran *buffer*, token-token unik, lebar matriks, tinggi matriks, banyaknya *sequence*, ukuran maksimal setiap *sequence*, isi dari matriks, isi dari *sequence*, dan hadiah untuk tiap *sequence*.

The image shows a code editor window titled 'app.py X' with a dark background. It contains Python code for a Flask application. The code defines a route for '/upload/' using the POST method. Inside this route, there is a function 'upload\_file()' which uses global variables. It checks if the request method is 'POST' and then reads a JSON file. The code then calls 'read\_file()' to process the data, followed by 'PRINT\_INFORMATION()' to log the data. A 'bruteforce' function is called with various parameters including buffer size, matrix dimensions, and sequence information. After the bruteforce calculation, the code formats the results and prints them. Finally, it returns a JSON response indicating the request was received.

```
app.route('/upload/', methods=['POST'])
def upload_file():
    global maximum_reward, possible_move, last_index_optimal, time_execution, formatted_sequence, move_coordinates

    if (request.method == 'POST'):
        file = request.json

        max_buffer_size, matrix, sequences, matrix_width, matrix_height, sequences_rewards = read_file(file['readFileName'])

        PRINT_INFORMATION(matrix, sequences, sequences_rewards, max_buffer_size)

        maximum_reward = 0
        formatted_sequence = []
        move_coordinates = []
        time_execution = 0

        # begin bruteforce
        start_time = time.time()
        bruteforce(max_buffer_size, matrix, True, 0, 0, [], sequences, sequences_rewards, matrix_width, matrix_height)
        # bruteforce ended

        # format needed information
        time_execution = (time.time() - start_time) * 1000
        if (maximum_reward != 0):
            formatted_sequence = ' '.join(current_journey_to_sequence(possible_move, matrix)[0:last_index_optimal])
            move_coordinates = index_to_coordinate(possible_move[0:last_index_optimal])

            # output as formatted
            FORMATTED_OUTPUT(maximum_reward, formatted_sequence, move_coordinates, time_execution)

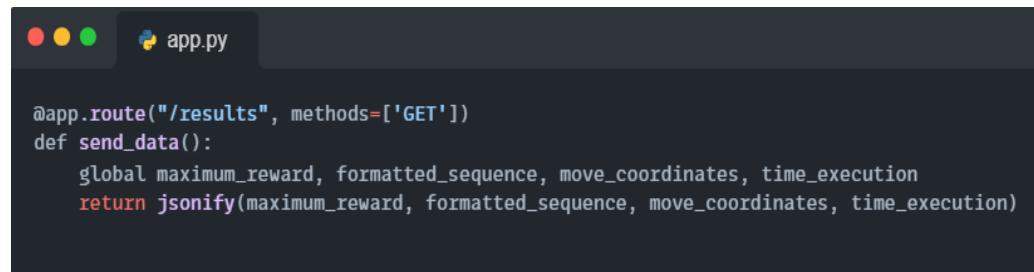
        else:
            print(maximum_reward)
            print()
            print(time_execution, "ms")
            print()

    return jsonify({'message': 'POST request received'})
```

*Gambar 3.4. Cuplikan kode API `upload_file()`*

API `upload_file()` yang terdapat pada **Gambar 3.4**. memiliki fungsionalitas yang sama dengan API `get_data()` yaitu melakukan pengolahan dengan melakukan algoritma bruteforce pada data yang diterima. Namun, perbedaannya hanya terdapat pada informasi yang diterima dari *client*, yaitu berupa filename yang hendak dibaca pada direktori lokal komputer. Pengolahan informasi filename yang diterima dari *client* dipecah dengan membaca file .txt yang terdapat pada lokal dengan menggunakan fungsi `read_file(filename)` sehingga

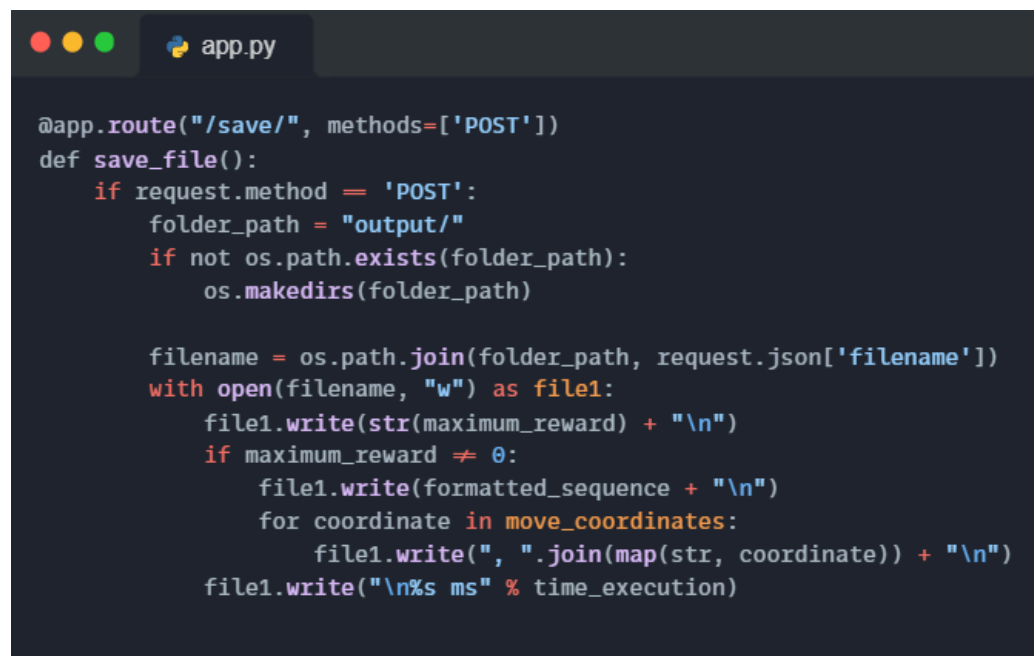
informasi-informasi yang diperlukan untuk melakukan pengolahan dapat terpenuhi.



```
@app.route("/results", methods=['GET'])
def send_data():
    global maximum_reward, formatted_sequence, move_coordinates, time_execution
    return jsonify(maximum_reward, formatted_sequence, move_coordinates, time_execution)
```

Gambar 3.5. Cuplikan kode API `send_data()`

API `send_data()` yang terdapat pada **Gambar 3.5.** hanya bertujuan untuk memberikan media bagi *frontend* untuk mengambil informasi hasil bruteforce yang telah dilakukan. Jadi, API ini hanya sebagai tempat bagi *frontend* untuk mengambil data yang akan ditampilkan kepada user.



```
@app.route("/save/", methods=['POST'])
def save_file():
    if request.method == 'POST':
        folder_path = "output/"
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)

        filename = os.path.join(folder_path, request.json['filename'])
        with open(filename, "w") as file1:
            file1.write(str(maximum_reward) + "\n")
            if maximum_reward != 0:
                file1.write(formatted_sequence + "\n")
                for coordinate in move_coordinates:
                    file1.write(", ".join(map(str, coordinate)) + "\n")
            file1.write("\n%s ms" % time_execution)
```

Gambar 3.6. Cuplikan kode API `save_file()`

API `save_file()` yang terdapat pada **Gambar 3.6.** bertujuan untuk menjalankan fitur *save solution* yang memberikan kesempatan bagi pengguna untuk melakukan penyimpanan ke direktori lokal pada folder `output/` dengan hanya memasukkan input nama file yang hendak disimpan. Dengan API ini, maka

solusi dari hasil bruteforce yang ditampilkan pada user, dapat disimpan dalam bentuk file dengan format yang sesuai dengan spesifikasi.

## b. functions.py

functions.py memuat kode-kode fungsi yang bertujuan untuk membantu berjalannya program utama yang terdapat pada app.py. File program ini hanya berisi gabungan fungsi-fungsi dengan tujuan agar pemrograman menjadi lebih modular sehingga jika terjadi bug pada fungsi, hanya perlu dilakukan fiksasi pada file ini. File ini terdapat fungsi-fungsi yang akan dijelaskan sebagai berikut.

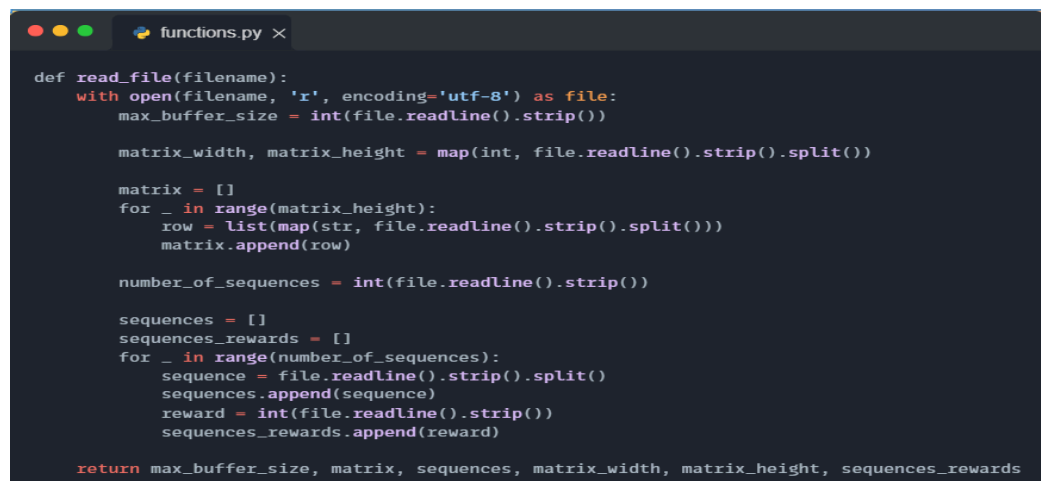


```
def print_matrix(matrix):
    for row in matrix:
        print(" ".join(map(str, row)))

def print_coordinate(matrix):
    for row in matrix:
        print(", ".join(map(str, row)))
```

*Gambar 3.7. Cuplikan fungsi print\_matrix dan print\_coordinate*

Prosedur print\_matrix dengan tujuan untuk melakukan formatting pada matrix sehingga memiliki tampilan yang lebih enak dilihat dan print\_coordinate untuk menampilkan koordinat-koordinat agar sesuai dengan tampilan yang diberikan oleh spesifikasi.



```
def read_file(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        max_buffer_size = int(file.readline().strip())

        matrix_width, matrix_height = map(int, file.readline().strip().split())

        matrix = []
        for _ in range(matrix_height):
            row = list(map(str, file.readline().strip().split()))
            matrix.append(row)

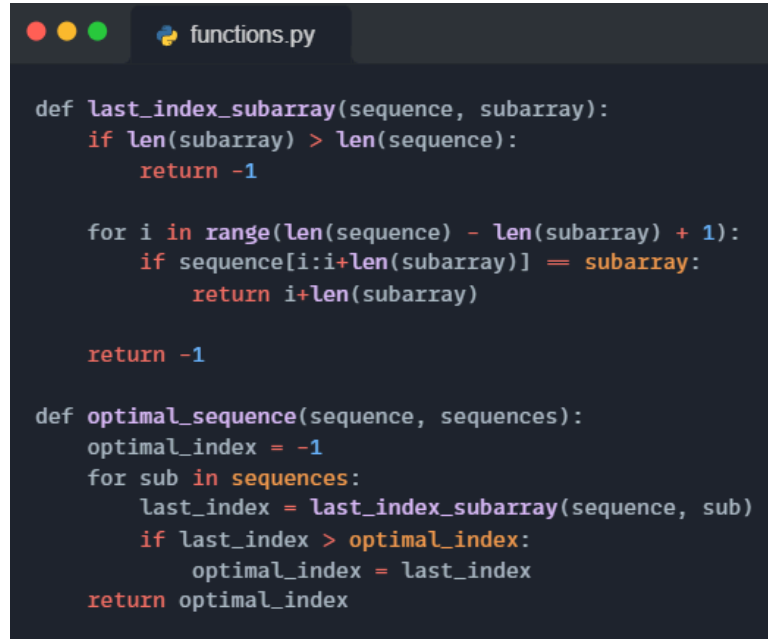
        number_of_sequences = int(file.readline().strip())

        sequences = []
        sequences_rewards = []
        for _ in range(number_of_sequences):
            sequence = file.readline().strip().split()
            sequences.append(sequence)
            reward = int(file.readline().strip())
            sequences_rewards.append(reward)

    return max_buffer_size, matrix, sequences, matrix_width, matrix_height, sequences_rewards
```

*Gambar 3.8. Cuplikan fungsi read\_file*

Prosedur `read_file` yang terdapat pada **Gambar 3.8**. Bertujuan untuk melakukan pengolahan informasi serta *parsing* isian file dari filename yang telah dimasukkan. Kemudian, isi dari filename yang telah dimasukkan akan diproses dan kemudian dikembalikan dalam beberapa nilai, yaitu `max_buffer_size`, `matrix`, `sequences`, `matrix_width`, `matrix_height`, dan `sequences_rewards`.

A screenshot of a code editor window titled 'functions.py'. The window has a dark background with light-colored text. It contains two Python functions. The first function, `last_index_subarray`, takes `sequence` and `subarray` as arguments. It first checks if the length of `subarray` is greater than the length of `sequence`; if so, it returns -1. Then, it iterates over the range from 0 to `len(sequence) - len(subarray) + 1`. For each `i`, it checks if `sequence[i:i+len(subarray)]` equals `subarray`. If it does, it returns `i+len(subarray)`. If no match is found, it returns -1. The second function, `optimal_sequence`, takes `sequence` and `sequences` as arguments. It initializes `optimal_index` to -1. It then iterates over each `sub` in `sequences`. For each `sub`, it calls `last_index_subarray` with `sequence` and `sub` as arguments. If the returned `last_index` is greater than `optimal_index`, it updates `optimal_index` to `last_index`. Finally, it returns `optimal_index`.

```
def last_index_subarray(sequence, subarray):
    if len(subarray) > len(sequence):
        return -1

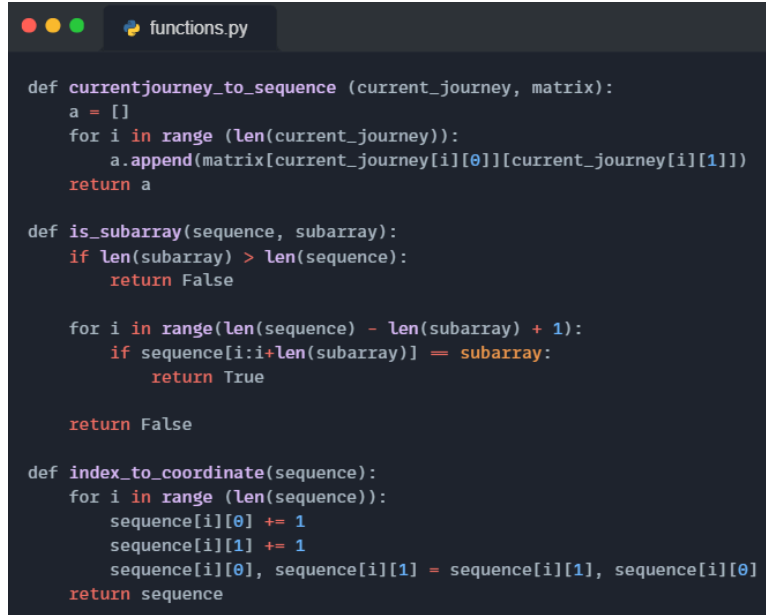
    for i in range(len(sequence) - len(subarray) + 1):
        if sequence[i:i+len(subarray)] == subarray:
            return i+len(subarray)

    return -1

def optimal_sequence(sequence, sequences):
    optimal_index = -1
    for sub in sequences:
        last_index = last_index_subarray(sequence, sub)
        if last_index > optimal_index:
            optimal_index = last_index
    return optimal_index
```

Gambar 3.9. Cuplikan fungsi `last_index_subarray` dan `optimal_sequence`

Kedua fungsi yang terdapat pada **Gambar 3.9**. Bertujuan untuk mencari *sequence* yang paling optimal dengan mencari *sequence* yang memiliki subarray dengan indeks yang paling rendah, maka merupakan *sequence* yang paling optimal. Dengan kata lain, *sequence* dengan jumlah langkah paling sedikit merupakan *sequence* yang paling optimal.



```

def currentjourney_to_sequence (current_journey, matrix):
    a = []
    for i in range (len(current_journey)):
        a.append(matrix[current_journey[i][0]][current_journey[i][1]])
    return a

def is_subarray(sequence, subarray):
    if len(subarray) > len(sequence):
        return False

    for i in range(len(sequence) - len(subarray) + 1):
        if sequence[i:i+len(subarray)] == subarray:
            return True

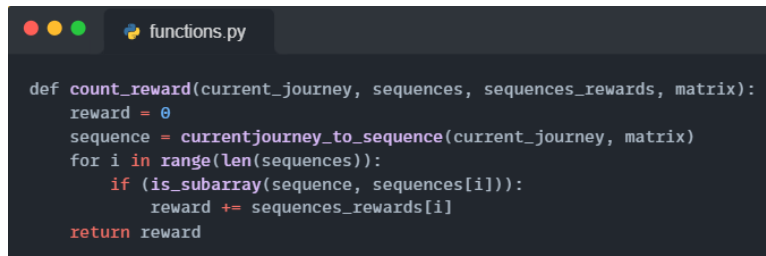
    return False

def index_to_coordinate(sequence):
    for i in range (len(sequence)):
        sequence[i][0] += 1
        sequence[i][1] += 1
        sequence[i][0], sequence[i][1] = sequence[i][1], sequence[i][0]
    return sequence

```

*Gambar 3.10. Cuplikan fungsi `currentjourney_to_sequence`, `is_subarray`, `index_to_coordinate`*

Pada **Gambar 3.10.** , terdapat tiga buah fungsi yang masing-masing memiliki kegunaannya sendiri untuk membantu perhitungan pada proses algoritma bruteforce. Fungsi `currentjourney_to_sequence` bertujuan untuk mengubah array dari koordinat menjadi array yang terdiri dari token-token yang disesuaikan dengan matriks yang diterima. Fungsi `is_subarray` bertujuan untuk membantu pengekan apakah sebuah *reward* terdapat pada sebuah *sequence* atau tidak. Fungsi `index_to_coordinate` hanya membantu mengubah index menjadi koordinat dengan melakukan penambahan nilai satu pada indeks X dan indeks Y pada matriks.



```

def count_reward(current_journey, sequences, sequences_rewards, matrix):
    reward = 0
    sequence = currentjourney_to_sequence(current_journey, matrix)
    for i in range(len(sequences)):
        if (is_subarray(sequence, sequences[i])):
            reward += sequences_rewards[i]
    return reward

```

*Gambar 3.11. Cuplikan fungsi `count_reward`*

Fungsi `count_reward` yang terdapat pada **Gambar 3.11.** bertujuan untuk melakukan perhitungan banyaknya *reward* yang diterima dari sebuah *sequence*

yang ditempuh oleh algoritma dengan memanfaatkan fungsi `is_subarray`. Setiap *sequence* yang terdapat pada tempuhan algoritma akan meningkatkan *reward* yang diperoleh sesuai dengan *reward* masing-masing *sequence*.

## 2. Framework Frontend ReactJS ( Bahasa Pemrograman Javascript )

### a. App.jsx

App.jsx merupakan file yang berisi kode pemrograman untuk membuat tampilan front-end serta terdapat beberapa fungsi-fungsi logik yang dilakukan untuk membuat input acak untuk user. Alasan input acak dilakukan di *frontend* adalah agar hasil acakan yang dibuat oleh user dapat ditampilkan di layar secara langsung tanpa melalui komunikasi API sehingga lebih efisien.



```
function matrixRandomize(matrixWidth, matrixHeight, uniqueToken) {
  let initialMatrix = [];
  for (let i = 0; i < matrixHeight; i++) {
    let row = [];
    for (let j = 0; j < matrixWidth; j++) {
      let randomIndex = Math.floor(Math.random() * uniqueToken.length);
      row.push(uniqueToken[randomIndex]);
    }
    initialMatrix.push(row);
  }
  return initialMatrix;
}

function sequenceRandomize(sequenceAmount, maximalSequenceSize, uniqueToken) {
  let resultSequences = [];
  for (let i = 0; i < sequenceAmount; i++) {
    let sequenceSize =
      Math.floor(Math.random() * (maximalSequenceSize - 2)) + 2;
    let initialSequence = [];
    for (let j = 0; j < sequenceSize + 1; j++) {
      let randomIndex = Math.floor(Math.random() * uniqueToken.length);
      initialSequence.push(uniqueToken[randomIndex]);
    }
    resultSequences.push(initialSequence);
  }
  return resultSequences;
}

function rewardRandomize(sequenceAmount) {
  let resultRewards = [];
  for (let i = 0; i < sequenceAmount; i++) {
    resultRewards.push(Math.floor(Math.random() * 50));
  }
  return resultRewards;
}
```

Gambar 3.12. Cuplikan fungsi-fungsi pengacakan input pada frontend

Fungsi-fungsi yang terdapat pada **Gambar 3.12**. Terdiri dari fungsi `matrixRandomize`, `sequenceRandomize`, dan `rewardRandomize`. Fungsi `matrixRandomize` merupakan fungsi untuk menghasilkan inputan matriks yang acak berdasarkan panjang matriks, tinggi matriks, dan token-token yang tersedia untuk dilakukan pengacakan. Fungsi `sequenceRandomize` merupakan fungsi untuk menghasilkan inputan acak terhadap *sequence* dengan menerima inputan `sequenceAmount` yang menentukan banyaknya *sequence* yang akan dibuat, `maximalSequenceSize` yang membatasi panjang ukuran dari setiap *sequence* yang dibuat, dan `uniqueToken` yang menjadi token-token yang tersedia untuk dilakukan pengacakan. Fungsi `rewardRandomize` bertujuan untuk melakukan pengacakan *reward* untuk setiap *sequence*.

Pada file `App.jsx`, tidak hanya mengandung ketiga fungsi tersebut, melainkan terdapat kode-kode lain yang bertujuan untuk membuat tampilan website, yang selengkapnya dapat dilihat pada pranala *github repository* yang terdapat pada bagian lampiran laporan ini. Alasan hanya dijelaskan ketiga fungsi di atas pada file `App.jsx`, karena hanya fungsi-fungsi tersebut yang berhubungan terhadap masukan user, yaitu masukan metode dua dengan metode acak yang mana berhubungan dengan spesifikasi.



# BAB IV : EKSPERIMEN

## a. Masukan acak oleh program

### i. Contoh 1

**Cyberpunk 2077 Hacking Minigame Solver**  
INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE | RANDOM INPUT

**1. SPECIFY BUFFER SIZE**  
4  
[ ][ ][ ][ ]

**2. ENTER UNIQUE TOKEN**  
7A BB 6C 4E KC

**3. ENTER MATRIX SIZE**  
5 X 5  
RANDOMIZE  
Matrix:  
BB BB 6C 7A 6C  
BB BB 6C 6C 7A  
4E BB 7A BB BB  
BB BB 4E BB 4E  
7A 6C 7A 7A KC

**4. ENTER SEQUENCES**  
Enter sequence amount: 3  
Enter maximal size for sequence: 5  
RANDOMIZE  
Sequence 1 (47) : BB 6C 7A 6C  
Sequence 2 (6) : KC 4E 4E 7A  
Sequence 3 (42) : 4E BB KC KC KC  
SOLVE

Gambar 4.1. Masukan acak contoh 1

**Cyberpunk 2077 Hacking Minigame Solver**  
INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE | RANDOM INPUT

**1. SPECIFY BUFFER SIZE**  
4  
[ ][ ][ ][ ]

**2. ENTER UNIQUE TOKEN**  
7A BB 6C 4E KC

**3. ENTER MATRIX SIZE**  
5 X 5  
RANDOMIZE  
Matrix:  
BB BB 6C 7A 6C  
BB BB 6C 6C 7A  
4E BB 7A BB BB  
BB BB 4E BB 4E  
7A 6C 7A 7A KC

**4. ENTER SEQUENCES**  
Enter sequence amount: 3  
Enter maximal size for sequence: 5  
RANDOMIZE  
Sequence 1 (47) : BB 6C 7A 6C  
Sequence 2 (6) : KC 4E 4E 7A  
Sequence 3 (42) : 4E BB KC KC KC  
SOLVE

**RESULT**  
Maximum Reward: 47  
Optimal Sequence: BB 6C 7A 6C  
Coordinate Movement:  
2,1  
2,5  
3,5  
3,1  
Time Execution: 0.52 ms  
Save Solution | Close

Gambar 4.2. Keluaran acak contoh 1

ii. Contoh 2

Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

5

000000

2. ENTER UNIQUE TOKEN

7A BB 5C 2K 3E

3. ENTER MATRIX SIZE

5

x

5

RANDOMIZE

3E	2K	BB	7A	3E	BB
3E	5C	5C	7A	BB	2K
3E	2K	7A	7A	2K	2K
BB	3E	7A	7A	3E	5C
5C	3E	5C	BB	BB	2K
5C	3E	3E	7A	7A	7A

4. ENTER SEQUENCES

Enter sequence amount:

4

Enter maximal size for sequence:

5

RANDOMIZE

Sequence 1 (0) : 5C 7A 3E

Sequence 2 (2) : 5C 5C BB 5C

Sequence 3 (3) : 5C BB BB BB

Sequence 4 (4) : BB BB 3E

SOLVE

Gambar 4.3. Masukan acak contoh 2

Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

5

000000

2. ENTER UNIQUE TOKEN

7A BB 5C 2K 3E

3. ENTER MATRIX SIZE

5

x

5

RANDOMIZE

3E	2K	BB	7A	3E	BB
3E	5C	5C	7A	BB	2K
3E	2K	7A	7A	2K	2K
BB	3E	7A	7A	3E	5C
5C	3E	5C	BB	BB	2K
5C	3E	3E	7A	7A	7A

4. ENTER SEQUENCES

Enter sequence amount:

4

Enter maximal size for sequence:

5

RANDOMIZE

Sequence 1 (0) : 5C 7A 3E

Sequence 2 (2) : 5C 5C BB 5C

Sequence 3 (3) : 5C BB BB BB

Sequence 4 (4) : BB BB 3E

SOLVE

RESULT

Maximum Reward: 43

Optimal Sequence: 7A BB BB 3E

Coordinate Movement:

4,1

4,5

5,5

5,1

Time Execution: 75.08 ms

Save Solution

Close

Gambar 4.4. Keluaran acak contoh 2

iii.      Contoh 3

Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

7

2. ENTER UNIQUE TOKEN

7A BB 5C 2K

3. ENTER MATRIX SIZE

6

X

6

RANDOMIZE

7A	5C	BB	BB	5C	5C
7A	BB	2K	BB	7A	2K
7A	2K	BB	5C	7A	7A
7A	2K	2K	7A	BB	5C
5C	BB	5C	5C	7A	2K
BB	2K	7A	5C	2K	2K

4. ENTER SEQUENCES

Enter sequence amount:

3

Enter required size for sequence:

4

RANDOMIZE

Sequence 1 (P1): BB 7A BB

Sequence 2 (P2): 7A 7A 5C 5C

Sequence 3 (P3): 5C 5C 2K

SOLVE

Gambar 4.5. Masukan acak contoh 3

Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

7

2. ENTER UNIQUE TOKEN

7A BB 5C 2K

3. ENTER MATRIX SIZE

6

X

6

RANDOMIZE

7A	5C	BB	BB	5C	5C
7A	BB	2K	BB	7A	2K
7A	2K	BB	5C	7A	7A
7A	2K	2K	7A	BB	5C
5C	BB	5C	5C	7A	2K
BB	2K	7A	5C	2K	2K

4. ENTER SEQUENCES

Enter sequence amount:

3

Enter required size for sequence:

4

RANDOMIZE

Sequence 1 (P1): BB 7A BB

Sequence 2 (P2): 7A 7A 5C 5C

Sequence 3 (P3): 5C 5C 2K

SOLVE

RESULT

Maximum Reward: 37

Optimal Sequence: 7A 7A 5C 5C BB 7A BB

Coordinate Movement:

1. 1

1. 3

4. 3

4. 6

1. 6

1. 2

2. 2

Time Execution: 387.08 ms

Save Solution

Close

Gambar 4.6. Keluaran acak contoh 3

iv. Contoh 4

Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

8

2. ENTER UNIQUE TOKEN

7A BB 5C 2K 4J

3. ENTER MATRIX SIZE

7

X

7

RANDOMIZE

2K	2K	5C	4J	2K	5C	7A
BB	2K	BB	2K	7A	2K	5C
2K	4J	7A	7A	4J	7A	BB
4J	7A	2K	BB	BB	7A	7A
BB	BB	BB	BB	4J	4J	2K
4J	2K	BB	7A	4J	7A	5C
2K	4J	BB	7A	4J	4J	7A

4. ENTER SEQUENCES

Enter sequence amount:

3

Enter maximal size for sequence:

4

RANDOMIZE

Sequence 1 (00) : BB BB 7A

Sequence 2 (20) : 5C BB 5C

Sequence 3 (40) : 4J 4J 5C BB

SOLVE

Gambar 4.7. Masukan acak contoh 4

Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

8

2. ENTER UNIQUE TOKEN

7A BB 5C 2K 4J

3. ENTER MATRIX SIZE

7

X

7

RANDOMIZE

2K	2K	5C	4J	2K	5C	7A
BB	2K	BB	2K	7A	2K	5C
2K	4J	7A	7A	4J	7A	BB
4J	7A	2K	BB	BB	7A	7A
BB	BB	BB	BB	4J	4J	2K
4J	2K	BB	7A	4J	7A	5C
2K	4J	BB	7A	4J	4J	7A

4. ENTER SEQUENCES

Enter sequence amount:

3

Enter maximal size for sequence:

4

RANDOMIZE

Sequence 1 (00) : BB BB 7A

Sequence 2 (20) : 5C BB 5C

Sequence 3 (40) : 4J 4J 5C BB

SOLVE

RESULT

Maximum Reward: 61

Optimal Sequence: BB 5C 2K 7A 2K BB 4J

Coordinate Movement:

4.1

4.5

5.5

5.4

1.4

1.5

3.5

Time Execution: 7711.94 ms

Save Solution

Close

Gambar 4.8. Keluaran acak contoh 4

v.      **Contoh 5**

# Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

8

2. ENTER UNIQUE TOKEN

BK 4D 31 4F J8 SD

3. ENTER MATRIX SIZE

7

X

7

RANDOMIZE

31	31	31	BK	SD	4D	SD
SD	31	31	31	SD	SD	SD
BK	BK	4F	31	31	SD	SD
31	BK	31	J8	4D	SD	4D
4F	BK	BK	31	SD	J8	31
BK	4D	31	4D	4D	J8	J8
4F	4D	4D	BK	31	SD	J8

4. ENTER SEQUENCES

Enter sequence amount:

4

Enter minimal size for sequence:

4

RANDOMIZE

Sequence 1 (23): J8 BK J8

Sequence 2 (16): SD 4D 4F

Sequence 3 (19): 31 4D 4D

Sequence 4 (44): J8 4D SD J8

SOLVE

Gambar 4.9. Masukan acak contoh 5

# Cyberpunk 2077 Hacking Minigame Solver

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

READ FROM FILE

RANDOM INPUT

1. SPECIFY BUFFER SIZE

8

2. ENTER UNIQUE TOKEN

BK 4D 31 4F J8 SD

3. ENTER MATRIX SIZE

7

X

7

RANDOMIZE

31	31	31	BK	SD	4D	SD
SD	31	31	31	SD	SD	SD
BK	BK	4F	31	31	SD	SD
31	BK	31	J8	4D	SD	4D
4F	BK	BK	31	SD	J8	31
BK	4D	31	4D	4D	J8	J8
4F	4D	4D	BK	31	SD	J8

4. ENTER SEQUENCES

Enter sequence amount:

4

Enter minimal size for sequence:

4

RANDOMIZE

Sequence 1 (23): J8 BK J8

Sequence 2 (16): SD 4D 4F

Sequence 3 (19): 31 4D 4D

Sequence 4 (44): J8 4D SD J8

SOLVE

RESULT

Maximum Reward: 75

Optimal Sequence: 31 4D 4D J8 4D SD J8

Coordinate Movement:

2.1

2.6

4.6

4.4

5.4

5.5

6.5

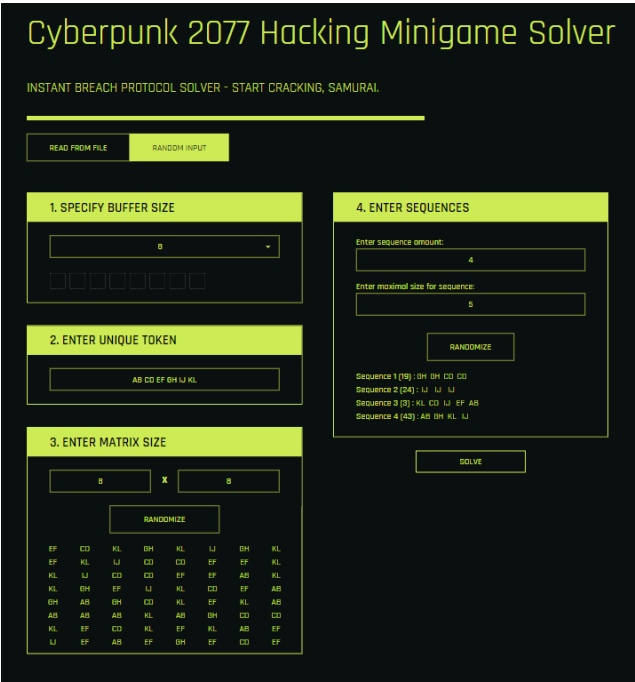
Time Execution: 9664.91 ms

Save Solution

Close

Gambar 4.10. Keluaran acak contoh 5

vi. Contoh 6



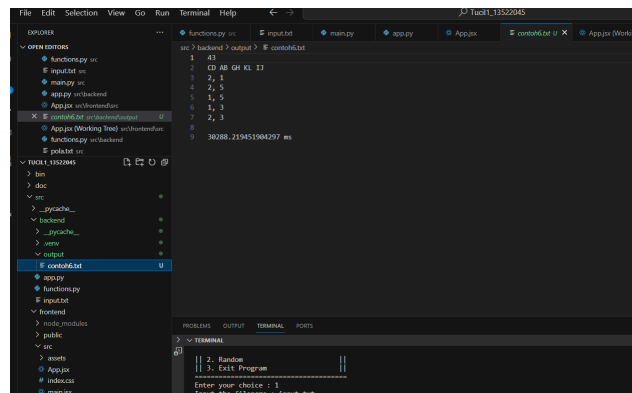
Gambar 4.11. Masukan acak contoh 6



Gambar 4.12. Keluaran acak contoh 6



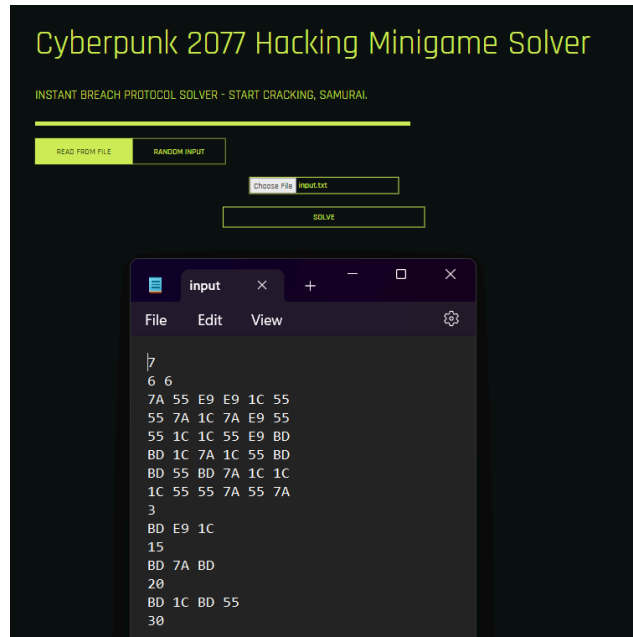
Gambar 4.13. Tampilan konfirmasi untuk menyimpan solusi



Gambar 4.14. Tampilan file tersimpan dengan nama file yang berikan user

**b. Masukkan dengan file .txt**

**i. Contoh 7**



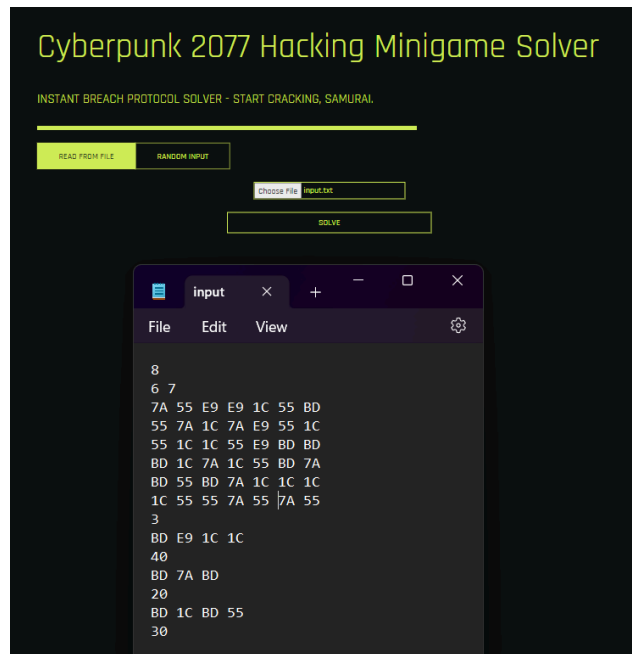
*Gambar 4.15. Tampilan masukan dengan upload file contoh 7*



*Gambar 4.16. Tampilan keluaran dengan upload file contoh 7*



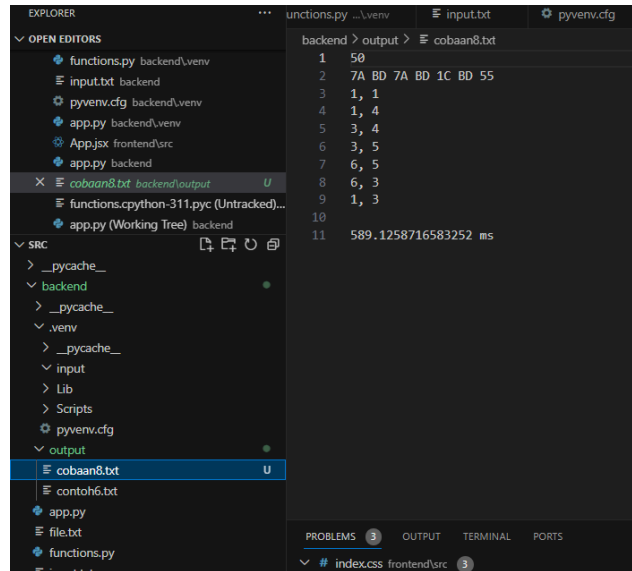
ii. Contoh 8



Gambar 4.17. Tampilan masukan dengan file contoh 8



Gambar 4.18. Tampilan keluaran dengan file contoh 8



Gambar 4.19. Tampilan hasil penyimpanan file contoh 8

Dengan adanya bab eksperimen, dapat terlihat bahwa program dengan GUI website sudah berhasil menjalankan aktivitas-aktivitas program sesuai dengan spesifikasi. Program terbukti dapat melakukan pembacaan file dengan metode masukan dengan upload file (seperti pada contoh 7 dan contoh 8) dan juga dengan metode masukan acak (seperti pada contoh 1, contoh 2, contoh 3, contoh 4, contoh 5, dan contoh 6). Program juga dapat melakukan algoritma bruteforce dan menampilkan solusi setelah pengguna melakukan *solve* atas masukan yang diberikan. Program juga dapat melakukan penyimpanan solusi jika pengguna menggunakan fitur *Save Solution*, yang mana akan dikirimkan ke direktori lokal output/ yang terdapat pada folder backend seperti yang terlihat pada eksperimen di atas.

# LAMPIRAN

## Github Repository

[https://github.com/ChaiGans/Tucil1\\_13522045](https://github.com/ChaiGans/Tucil1_13522045)

## Tabel Spesifikasi

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	✓	