

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

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



Benardo – 13522055

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

# Daftar Isi

<b>BAB 1.....</b>	<b>3</b>
<b>Deskripsi Masalah.....</b>	<b>3</b>
<b>BAB 2.....</b>	<b>5</b>
<b>Algoritma Brute Force yang Digunakan.....</b>	<b>5</b>
<b>BAB 3.....</b>	<b>9</b>
<b>Implementasi Program dengan Javascript (node js).....</b>	<b>9</b>
1. Backend Express (Bahasa Pemrograman Javascript).....	9
a. index.js.....	9
2. Frontend ReactJs (Bahasa Pemrograman Javascript).....	16
a. file.js dan manual.js.....	16
<b>BAB 4.....</b>	<b>18</b>
<b>Eksperimen.....</b>	<b>18</b>
a. Masukan acak oleh program.....	18
1) Contoh 1.....	18
2) Contoh 2.....	19
3) Contoh 3.....	21
4) Contoh 4.....	22
5) Contoh 5.....	24
6) Contoh 6.....	25
b. Masukan dengan file.txt.....	27
1) Contoh 7.....	27
2) Contoh 8.....	28
3) Contoh 9.....	30
<b>Lampiran.....</b>	<b>32</b>
Tabel Spesifikasi.....	32

# BAB 1

## 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 di cocokan.
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.

Pada program ini akan dilakukan pencarian solusi dari permainan Breach Protocol yang paling optimal (mendapatkan reward terbesar) dengan menggunakan algoritma brute force.



*Gambar 1. Contoh  
Permainan CyberPunk  
Breach Protocol 2077*

## BAB 2

### Algoritma Brute Force yang Digunakan

Algoritma *brute force* adalah metode yang digunakan untuk memecahkan masalah atau mencari solusi dengan cara mencoba semua kemungkinan secara berurutan hingga menemukan solusi yang benar. Metode ini sering digunakan ketika tidak ada cara yang lebih efisien untuk menyelesaikan masalah tersebut. Algoritma *brute force* adalah metode yang sederhana dan kuat, karena dia tidak mengandalkan pengetahuan sebelumnya atau optimasi khusus.

Penerapan algoritma Brute Force untuk menyelesaikan pencarian solusi optimal pada permainan Cyberpunk 2077 Breach Protocol didasarkan pada beberapa pertimbangan yaitu pertama, karena dengan menggunakan algoritma Brute Force dapat dilakukan pengecekan pada semua kombinasi pada matriks dengan ukuran buffer yang telah ditentukan. Ini mengakibatkan tidak akan ada solusi yang terlewatkan, sehingga dengan algoritma Brute Force ini dapat dilakukan pengecekan reward dan kemudian dapat menemukan solusi yang paling optimal (solusi yang mendapatkan reward terbesar) .

Langkah - langkah dalam penerapan Algoritma Brute Force pada Program :

#### 1. Inisialisasi Data

Algoritma Brute Force dimulai dengan mempersiapkan data yang akan digunakan , berupa matriks yang digunakan, ukuran buffer, jumlah token, jumlah sequence dan sequence beserta dengan rewardnya. Setelah memperoleh data-data ini, barulah data ini akan digunakan sebagai parameter dalam penerapan algoritma Brute Force ini.

## 2. Menentukan Posisi Mulai

Posisi awal akan ditentukan dari baris yang paling atas. Algoritma akan melakukan pengecekan dengan menetapkan setiap kolom pada baris pertama sebagai posisi mulia, kemudian dari hasil yang diperoleh, dipilih solusi yang mendapatkan reward yang paling besar.

## 3. Eksplorasi Kemungkinan

Dengan posisi awal yang digunakan, akan dilakukan pencarian semua kombinasi rute atau langkah selanjutnya yaitu dimulai dengan bergerak vertikal , kemudian horizontal secara bergantian. Algoritma Brute Force ini menerapkan pendekatan Rekursif. Berikut adalah penjelasan yang lebih rinci:

### a. Basis

Basis rekursif dari algoritma ini adalah ketika panjang dari lintasan (path) telah sama dengan ukuran buffer yang ditentukan. Pada saat ini, akan dilakukan perhitungan reward yang diperoleh , kemudian program akan menghentikan proses rekursif lebih lanjut. Dengan adanya basis ini, membatasi program untuk mengecek kombinasi yang panjang lintasan (path) nya lebih besar dari ukuran buffer.

### b. Rekurens

Pada proses rekurens ini, akan dilakukan perhitungan reward sementara pada lintasan (path) saat itu, kemudian akan dilanjutkan dengan mencari semua langkah selanjutnya yang mungkin

berdasarkan urutan gerakan vertikal atau horizontal secara bergantian. Setelah mendapatkan semua langkah selanjutnya yang mungkin pada posisi saat itu, akan dilakukan pemanggilan diri sendiri (rekursif) untuk setiap posisi selanjutnya yang mungkin.

#### c. Backtracking

Backtracking terjadi ketika algoritma kembali dari rekursi tanpa menemukan solusi yang lebih optimal dari yang telah ditemukan sebelumnya. Ini memungkinkan algoritma untuk mundur dan mencoba langkah alternatif dari posisi sebelumnya, secara efektif membatalkan pilihan terakhir dan menggantinya dengan pilihan baru yang belum dieksplorasi. Ini krusial untuk memastikan semua kemungkinan jalur diuji.

#### 4. Perhitungan Reward

Program akan melakukan perhitungan reward setiap lintasan (path) ditambahkan token. Perhitungan reward dilakukan dengan melakukan pencocokan terhadap sequences yang telah ditetapkan sebelumnya. Apabila ada sequences yang memenuhi maka akan dilakukan penambahan reward pada skor.

#### 5. Optimalisasi Solusi

Algoritma Brute Force ini melakukan perhitungan reward setiap ada token yang ditambahkan pada lintasan, sehingga semua solusi (bukan hanya solusi yang memiliki panjang lintasan sama dengan buffer) akan dibandingkan dan dipilih solusi dengan reward yang tertinggi. Hal ini memungkinkan solusi

yang dihasilkan bukan hanya optimal , namun jug efektif (karena panjang lintasan yang lebih pendek, namun mempunyai reward yang paling tinggi).

#### 6. Keluaran atau Output

Setelah mengkesplorasi semua kemungkinan dan telah mendapatkan solusi lintasan (path) yang optimal (reward paling tinggi) , maka akan ditampilkan total reward , diikuti dengan lintasan yang optimal dan koordinat dari setiap token dalam lintasan tersebut dan waktu eksekusi kepada user.

Langkah - langkah diatas memastikan tidak ada solusi yang terlewat, dan dilakukan pengecekan pada setiap kemungkinan kombinasi, sehingga bisa mendapatkan solusi yang optimal.



## **BAB 3**

# **Implementasi Program dengan Javascript (node js)**

Program ini menerapkan sistem antar muka berbasis web (GUI) dan logika pemrosesan backend. Untuk bagian Frontend , digunakan framework ReactJs , kemudian untuk bagian Backend digunakan framework Express. Fungsi-fungsi untuk melakukan pemrosesan data dan penerapan algoritma Brute Force terpadu pada file index.js

Berikut fungsi- fungsi yang digunakan dalam program :

### **1. Backend Express (Bahasa Pemrograman Javascript)**

#### **a. index.js**

File ini berfungsi sebagai pengendali yang mengatur interaksi antara frontend dan backend , pada file ini juga algoritma Brute Force diterapkan. Dengan menggunakan framework Express, didefinisikan beberapa API endpoint yang akan digunakan untuk menerima data yang diperoleh dari frontend , kemudian mengembalikan data hasil olahan kepada frontend. Data-data yang diterima antara lain file hasil inputan, ukuran buffer, jumlah sequence, ukuran matriks ,dan lain-lain. Sehingga dapat dikatakan file index.js ini merupakan jembatan penghubung antara frontend dan backend.

```

app.post("/solveFile", upload.single('file'), async (req, res) => {
  if (!req.file) {
    return res.status(400).json({ message: "No file uploaded." });
  }

  try {
    const startTime = Date.now();
    const data = await readFromFile(req.file.path);
    const matrix = data.matrix;
    const sequences = data.sequences.map(seq => new Sequence(seq.tokens, seq.reward));

    let overallBestPath = [];
    let maxOverallReward = 0;

    // Loop through each column in the first row
    for (let col = 0; col < data.matrixWidth; col++) {
      const visited = Array.from({ length: data.matrixHeight }, () => Array(data.matrixWidth).fill(false));
      const path = [];
      const bestPath = [];
      const maxRewardObject = { max: 0 };
      const startPos = new Position(col, 0); // Starting position for this iteration

      explorePaths(matrix, startPos, path, visited, data.bufferSize, true, sequences, maxRewardObject, bestPath);

      // Compare and update overall best path and reward
      if (maxRewardObject.max > maxOverallReward) {
        maxOverallReward = maxRewardObject.max;
        overallBestPath = bestPath.slice(); // Make a copy of the best path
      }
    }

    const endTime = Date.now(); // End timing
    const executionTime = endTime - startTime;

    // Respond with the best results found among all starting positions
    res.json({
      message: "File processed successfully",
      data: data,
      maxReward: maxOverallReward,
      bestPath: overallBestPath.map(t => ({ value: t.value, pos: t.pos })),
      executionTime: executionTime
    });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Error processing file", error: error.message });
  }
});

```

*Gambar 2. API endPoint solveFile*

Endpoint diatas menerima file yang diinput oleh user dari frontend melalui method POST. Pada awalnya dilakukan pengecekan terlebih dahulu apakah data tersebut berhasil diterima atau tidak. Setelah berhasil akan dilakukan pembacaan dari file dengan menggunakan fungsi `readFromFile()`. Hasil dari fungsi ini, mengembalikan sebuah data yang terdiri dari matriks, ukuran matriks, ukuran buffer, sequence beserta dengan reward nya. Setelah berhasil mendapatkan data tersebut akan dilakukan looping untuk menguji setiap kolom pada baris pertama sebagai titik awal dalam explorasi semua kombinasi. Pengeksplorasian semua kombinasi dilakukan

dengan memanggil function `explorePath()`. Hasil dari pemanggilan `explorepath()` akan diperoleh `maxReward` dan juga lintasan optimal. Setelah mendapatkan data-data tersebut, kemudian data -data ini akan dikirimkan kepada frontend untuk ditampilkan.

```
app.post('/solve',upload.none(), async (req, res) => {
  let { token, bufferSize, matrixHeight, matrixWidth, sequenceSize, maxSequence } = req.body;

  // Convert token to array if it's a string
  if (typeof token === 'string') {
    token = token.split(' ');
  }

  // Validate input
  if (!token || !bufferSize || !matrixHeight || !matrixWidth || !sequenceSize || !maxSequence || token.length === 0) {
    return res.status(400).json({ message: "All fields are required and token array cannot be empty" });
  }

  // Parse numeric values from strings
  bufferSize = parseInt(bufferSize, 10);
  matrixWidth = parseInt(matrixWidth, 10);
  matrixHeight = parseInt(matrixHeight, 10);
  sequenceSize = parseInt(sequenceSize, 10);
  maxSequence = parseInt(maxSequence, 10);

  // Initialize data structure
  let data = {
    bufferSize,
    matrixWidth,
    matrixHeight,
    matrix: [],
    sequences: []
  };

  // Generate the matrix
  for (let i = 0; i < matrixHeight; ++i) {
    let row = [];
    for (let j = 0; j < matrixWidth; ++j) {
      let tokenIndex = Math.floor(Math.random() * token.length);
      row.push(token[tokenIndex]);
    }
    data.matrix.push(row);
  }
}
```

*Gambar 3. API EndPoint solve*

```

// generate sequences
for (let i = 0; i < sequenceSize; ++i) {
  let sequenceLength = 2 + Math.floor(Math.random() * (maxSequence - 1));
  let sequenceTokens = [];
  for (let j = 0; j < sequenceLength; ++j) {
    let tokenIndex = Math.floor(Math.random() * token.length);
    sequenceTokens.push(token[tokenIndex]);
  }
  let reward = (Math.floor(Math.random() * 10) + 1) * 5;
  data.sequences.push({ tokens: sequenceTokens, reward });
}

// Find best path starting from each column in the first row
let bestOverallReward = 0;
let bestOverallPath = [];
const startTime = Date.now();

for (let col = 0; col < matrixWidth; ++col) {
  const startPos = new Position(col, 0);
  const visited = Array.from({ length: matrixHeight }, () => Array(matrixWidth).fill(false));
  const path = [];
  const bestPath = [];
  const maxRewardObject = { max: 0 };

  explorePaths(data.matrix, startPos, path, visited, bufferSize, true, data.sequences, maxRewardObject, bestPath);

  if (maxRewardObject.max > bestOverallReward) {
    bestOverallReward = maxRewardObject.max;
    bestOverallPath = bestPath.slice(); // Make a copy of the best path
  }
}

const endTime = Date.now();
const executionTime = endTime - startTime;

// Send response
res.json({
  message: "Matrix and sequences generated successfully",
  data,
  maxReward: bestOverallReward,
  bestPath: bestOverallPath.map(t => ({ value: t.value, pos: t.pos })),
  executionTime
});
});

```

*Gambar 4. Lanjutan API Endpoint solve*

Endpoint ini akan menerima data ukuran matriks, ukuran buffer, jumlah token unik, token unik, jumlah sequence, dan ukuran maksimal sequence dari frontend, kemudian akan dilakukan pembuatan matriks secara random berdasarkan informasi yang diperoleh. Data sequence beserta rewardnya juga dihasilkan dari hasil random. Setelah memperoleh data matriks dan sequence, akan dilakukan pencarian solusi optimal dengan menggunakan function `explorePaths()`, hasil dari fungsi ini akan diperoleh data reward terbesar dan solusi lintasan optimal beserta dengan koordinatnya, kemudian akan dikirimkan ke frontend untuk ditampilkan.

```

async function readFromFile(filePath) {
  const fileStream = fs.createReadStream(filePath);
  const rl = readline.createInterface({
    input: fileStream,
    crlfDelay: Infinity
  });

  const data = {
    bufferSize: 0,
    matrixWidth: 0,
    matrixHeight: 0,
    matrix: [],
    sequences: []
  };

  let lineCount = 0;
  let numberOfSequences = 0;
  let isReadingSequence = true;
  let tokens;
  for await (const line of rl) {
    if (line.trim() === '') continue;
    if (lineCount === 0) {
      data.bufferSize = parseInt(line);
    } else if (lineCount === 1) {
      [data.matrixWidth, data.matrixHeight] = line.split(' ').map(Number);
    } else if (lineCount <= data.matrixHeight + 1) {
      data.matrix.push(line.split(' '));
    } else if (lineCount === data.matrixHeight + 2) {
      numberOfSequences = parseInt(line);
    } else {
      if (isReadingSequence) {
        tokens = line.split(' ');
        isReadingSequence = false;
      } else {
        const reward = parseInt(line, 10);
        data.sequences.push({ tokens, reward });
        isReadingSequence = true;
      }
    }
  }
  lineCount++;
}

return data;
}

```

*Gambar 5. Fungsi readFromFile()*

Fungsi ini membaca file yang diterima , dan kemudian menyimpan data -data yang dibaca ke dalam sebuah objek data yang

terdiri dari ukuran buffer, ukuran matriks, matriks, object sequences (terdiri dari sequence dan reward).

```
function explorePaths(matrix, pos, path, visited, bufferSize, moveVertical, sequences, maxReward, bestPath) {
  if (pos.x < 0 || pos.x >= matrix[0].length || pos.y < 0 || pos.y >= matrix.length || visited[pos.y][pos.x]) {
    return;
  }

  visited[pos.y][pos.x] = true;
  path.push(new TokenInfo(matrix[pos.y][pos.x], pos));

  let currentReward = calculatePathReward(path, sequences);
  if (currentReward > maxReward.max) {
    maxReward.max = currentReward;
    bestPath.length = 0; // Clear existing path
    path.forEach(p => bestPath.push(p)); // Copy current path to bestPath
  }

  if (path.length < bufferSize) {
    let nextPositions = [];
    if (moveVertical) {
      for (let newY = pos.y + 1; newY < matrix.length; newY++) {
        nextPositions.push(new Position(pos.x, newY));
      }
      for (let newY = pos.y - 1; newY >= 0; newY--) {
        nextPositions.push(new Position(pos.x, newY));
      }
    } else {
      for (let newX = pos.x + 1; newX < matrix[0].length; newX++) {
        nextPositions.push(new Position(newX, pos.y));
      }
      for (let newX = pos.x - 1; newX >= 0; newX--) {
        nextPositions.push(new Position(newX, pos.y));
      }
    }

    nextPositions.forEach(nextPos => {
      explorePaths(matrix, nextPos, path, visited, bufferSize, !moveVertical, sequences, maxReward, bestPath);
    });
  }

  path.pop();
  visited[pos.y][pos.x] = false;
}
```

*Gambar 6. Fungsi explorePaths()*

Fungsi ini merupakan inti dari program ini, yaitu untuk mencari solusi optimal. Fungsi ini menerapkan pendekatan algoritma Brute Force. Fungsi ini menerima masukan matriks, posisi, lintasan, matriks boolean yang merepresentasikan apakah posisi tersebut telah dikunjungi atau belum, ukuran buffer, arah gerakan (dalam boolean), sekuens, object maxReward, dan lintasan optimal. Fungsi ini akan menentukan terlebih dahulu semua kemungkinan langkah selanjutnya dengan arah berdasarkan kondisi moveVertical, lalu setelah diperoleh semua kemungkinan langkah selanjutnya, akan dilakukan looping untuk melakukan rekursif pada setiap posisi selanjutnya yang

mungkin. Dalam algoritma ini, juga dilakukan perhitungan reward dan menandakan matriks visited menjadi true pada posisi saat itu setiap ada token yang ditambahkan pada lintasan, sehingga semua kemungkinan solusi yang mungkin tidak akan terlewatkan. Jika proses rekursif tidak melanjutkan pada tahap berikutnya akan dilakukan backtracking pada posisi dan keadaan sebelumnya.

```
function calculatePathReward(path, sequences) {  
  let totalReward = 0;  
  sequences.forEach(sequence => {  
    if (sequence.tokens.length > path.length) return;  
    for (let i = 0; i <= path.length - sequence.tokens.length; i++) {  
      let match = true;  
      for (let j = 0; j < sequence.tokens.length; j++) {  
        if (path[i + j].value !== sequence.tokens[j]) {  
          match = false;  
          break;  
        }  
      }  
      if (match) {  
        totalReward += sequence.reward;  
        break; // Assuming a path can match a sequence only once  
      }  
    }  
  });  
  return totalReward;  
}
```

*Gambar 7. Fungsi calculateReward()*

Fungsi ini melakukan perhitungan reward dengan mengecek apakah sequence terdapat pada lintasan yang dihitung rewardnya, jika ternyata ada yang cocok, maka nilai dari reward akan ditambahkan pada total skor.

## 2. Frontend ReactJs (Bahasa Pemrograman Javascript)

### a. file.js dan manual.js

```
const handleDownloadResult = () => {
  if (!results || !results.bestPath) {
    alert('No results to download');
    return;
  }

  let content = `${results.maxReward}\n`;
  content += results.bestPath.map(tokenInfo => tokenInfo.value).join(" ") + "\n";
  results.bestPath.forEach((tokenInfo, index) => {
    content += `${tokenInfo.pos.x + 1}, ${tokenInfo.pos.y + 1}\n`;
  });
  content += `\n${results.executionTime} ms\n`;

  const blob = new Blob([content], { type: 'text/plain' });

  const fileURL = URL.createObjectURL(blob);

  const tempLink = document.createElement('a');
  tempLink.href = fileURL;
  tempLink.setAttribute('download', 'results.txt');
  document.body.appendChild(tempLink);

  tempLink.click();

  document.body.removeChild(tempLink);
  URL.revokeObjectURL(fileURL);
};
```

*Gambar 8. Fungsi handleDownloadResult()*

Fungsi ini bertugas untuk memungkinkan pengguna mengunduh hasil solusi Breach Protocol ke dalam sebuah file teks. Awalnya, fungsi ini memverifikasi keberadaan hasil dan jalur terbaik yang telah dihitung. Jika hasil tersedia, fungsi ini selanjutnya mengkompilasi isi file, termasuk total reward, urutan token dalam jalur terbaik, koordinat setiap langkah dalam format (x, y), dan waktu eksekusi algoritma. Semua informasi ini dikemas dalam format teks sederhana. Kemudian, dengan menggunakan Blob, sebuah objek URL



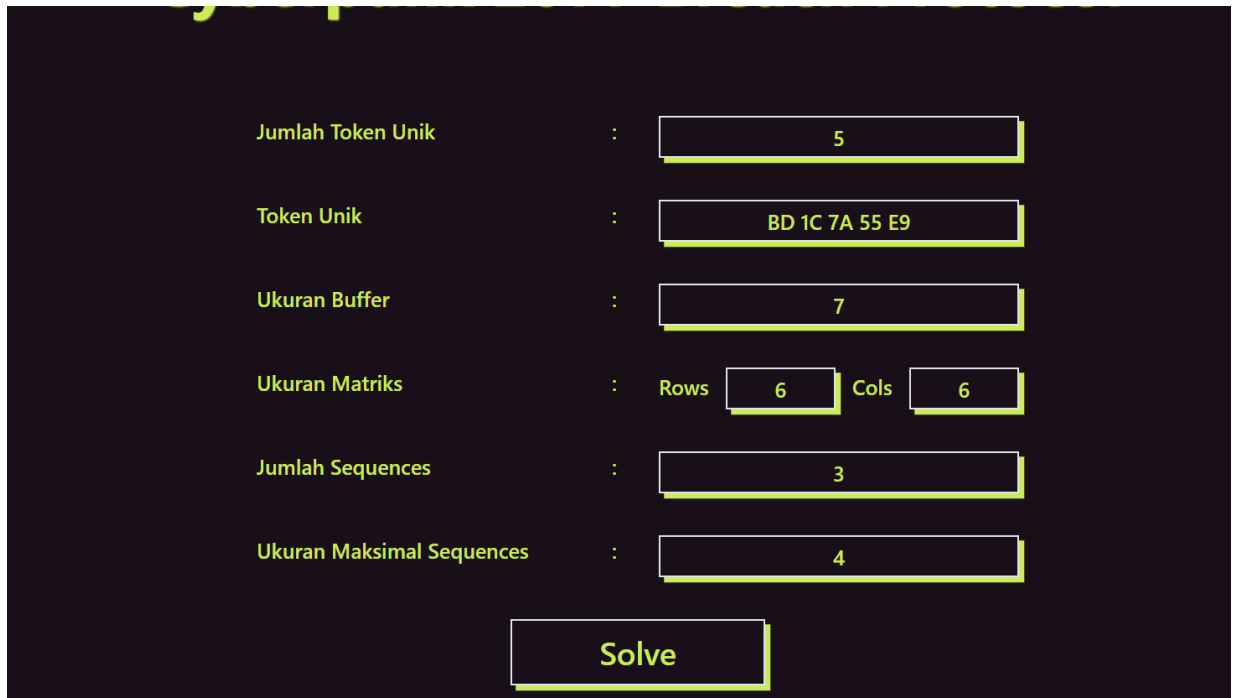
sementara dibuat yang mengarah ke data ini, memungkinkan file tersebut diunduh. Untuk memicu unduhan, fungsi ini secara dinamis membuat sebuah elemen tautan (anchor) dengan atribut download, menetapkan URL Blob ke atribut href, dan memicu klik pada tautan tersebut. Setelah proses unduhan dimulai, tautan sementara dihapus dan URL Blob dicabut untuk mengoptimalkan penggunaan sumber daya. Proses ini menyediakan metode yang efisien dan intuitif bagi pengguna untuk menyimpan hasil solusi dalam bentuk file teks yang dapat diakses dan dianalisis kapan saja.

# BAB 4

## Eksperimen

### a. Masukan acak oleh program

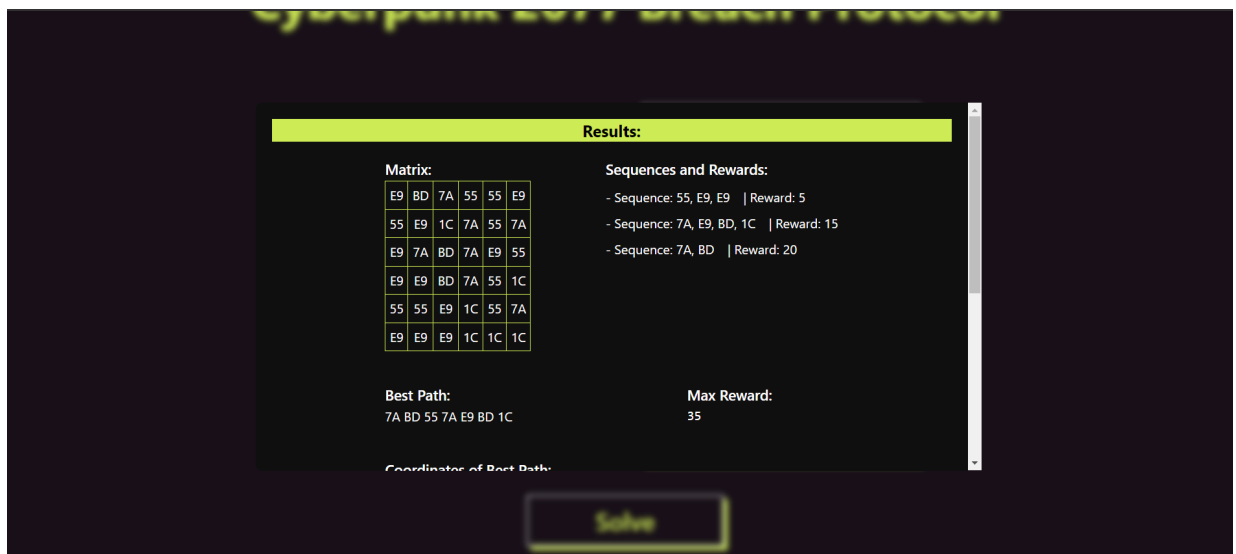
#### 1) Contoh 1



The screenshot shows a form with the following fields and values:

- Jumlah Token Unik: 5
- Token Unik: BD 1C 7A 5S E9
- Ukuran Buffer: 7
- Ukuran Matriks: Rows 6, Cols 6
- Jumlah Sequences: 3
- Ukuran Maksimal Sequences: 4
- Solve button

Gambar 9. Tampilan masukan pada contoh 1



The screenshot shows the results of the experiment, including a matrix, sequences and rewards, and the best path.

**Results:**

**Matrix:**

E9	BD	7A	5S	5S	E9
5S	E9	1C	7A	5S	7A
E9	7A	BD	7A	E9	5S
E9	E9	BD	7A	5S	1C
5S	5S	E9	1C	5S	7A
E9	E9	E9	1C	1C	1C

**Sequences and Rewards:**

- Sequence: 5S, E9, E9 | Reward: 5
- Sequence: 7A, E9, BD, 1C | Reward: 15
- Sequence: 7A, BD | Reward: 20

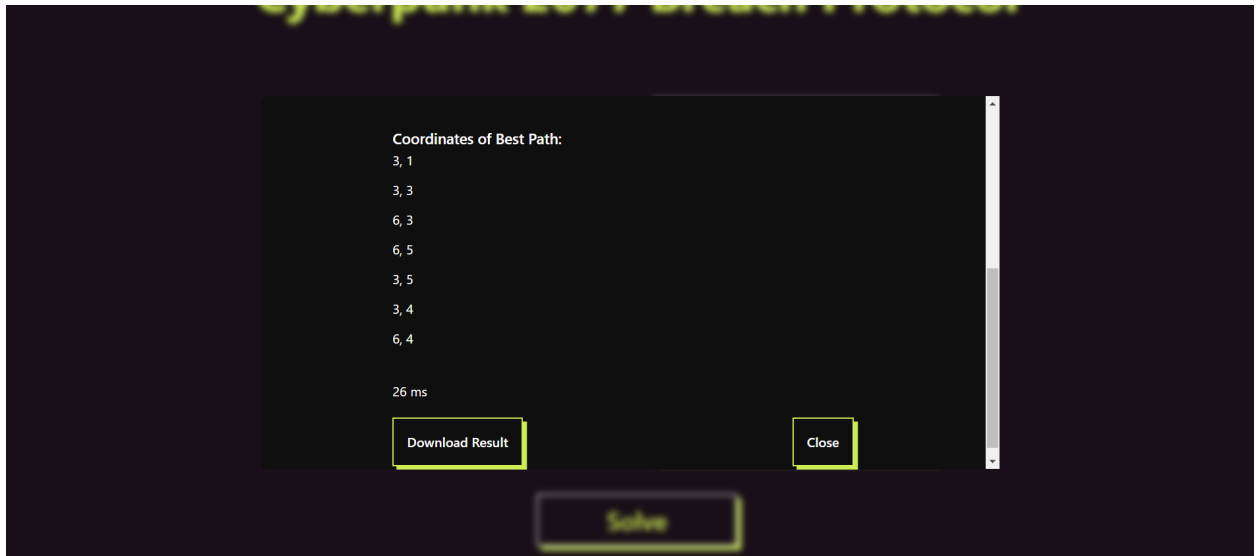
**Best Path:**  
7A BD 5S 7A E9 BD 1C

**Max Reward:**  
35

**Coordinates of Best Path:**

Solve button

Gambar 10. Tampilan keluar pada contoh 1 (part 1)

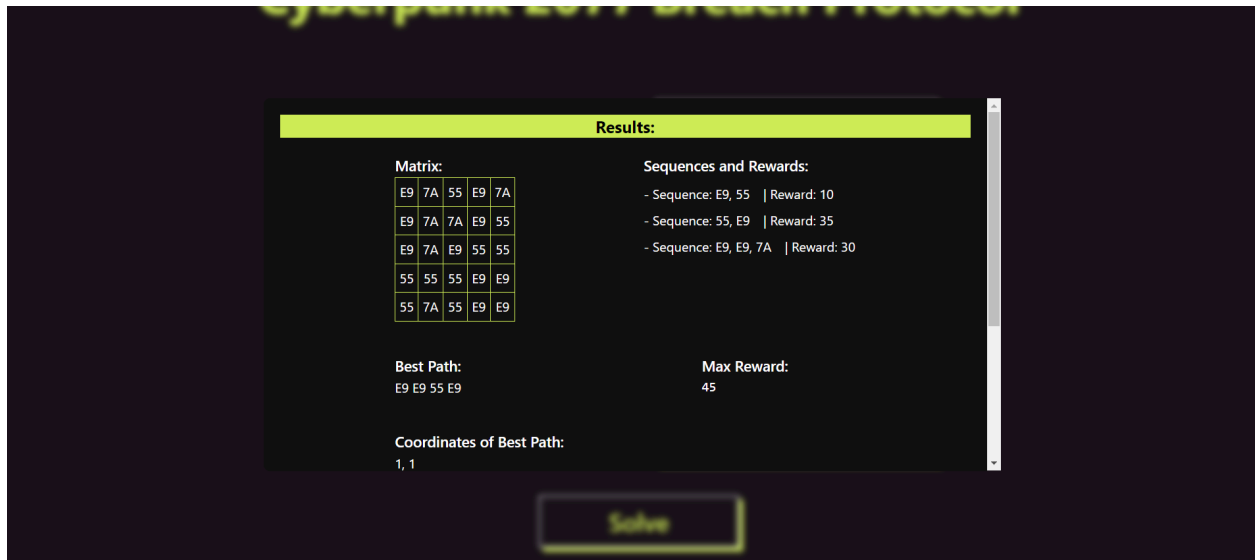


Gambar 11. Tampilan keluar pada contoh 1 (part 2)

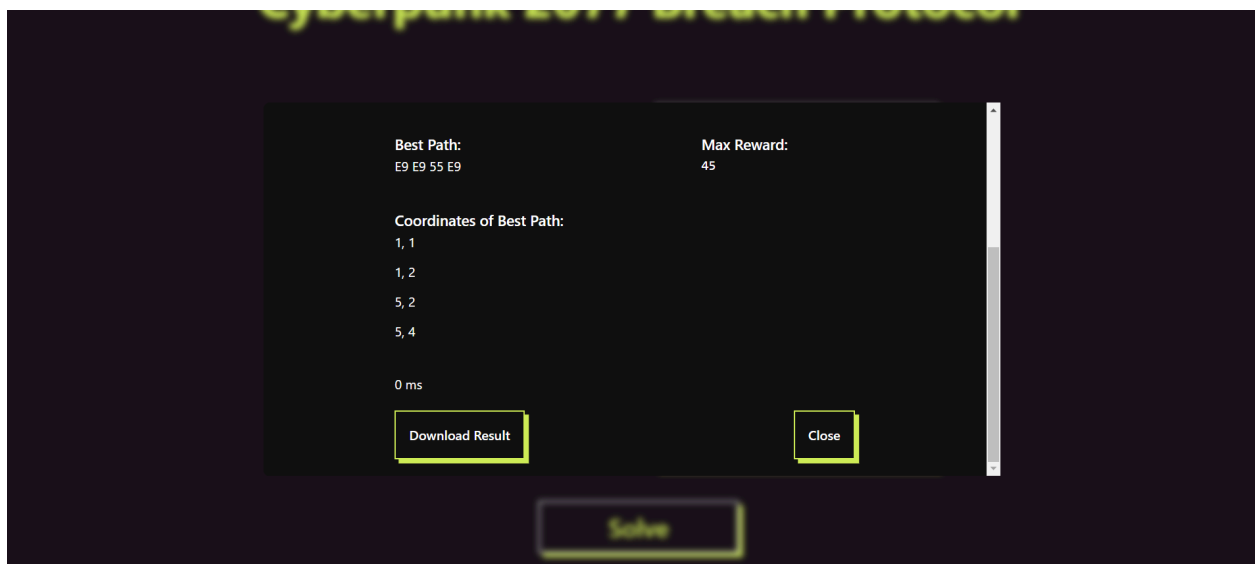
## 2) Contoh 2

A screenshot of a web application interface. At the top, there is a header with the text "Cyberpunk 2077: Breakthrough". Below the header, there is a dark blue box containing several input fields. The fields are labeled: "Jumlah Token Unik" with a value of 3; "Token Unik" with a value of 7A 55 E9; "Ukuran Buffer" with a value of 4; "Ukuran Matriks" with "Rows" and "Cols" both set to 5; "Jumlah Sequences" with a value of 3; and "Ukuran Maksimal Sequences" with a value of 3. At the bottom of the dark blue box, there is a large blue button labeled "Solve". In the top left corner of the dark blue box, there is a button labeled "First Page". In the top right corner, there is a button labeled "Input From File".

Gambar 12. Tampilan masukan pada contoh 2



Gambar 13. Tampilan keluar pada contoh 2 (part 1)



Gambar 14. Tampilan keluar pada contoh 2 (part 2)

### 3) Contoh 3

First Page Input From File

Jumlah Token Unik :

Token Unik :

Ukuran Buffer :

Ukuran Matriks : Rows  Cols

Jumlah Sequences :

Ukuran Maksimal Sequences :

Gambar 15. Tampilan masukan pada contoh 3

Results:

Matrix:

BD	E9	7A
E9	7A	BD
BD	BD	E9

Sequences and Rewards:

- Sequence: 7A, E9 | Reward: 40
- Sequence: BD, 7A | Reward: 20
- Sequence: 55, 7A | Reward: 35

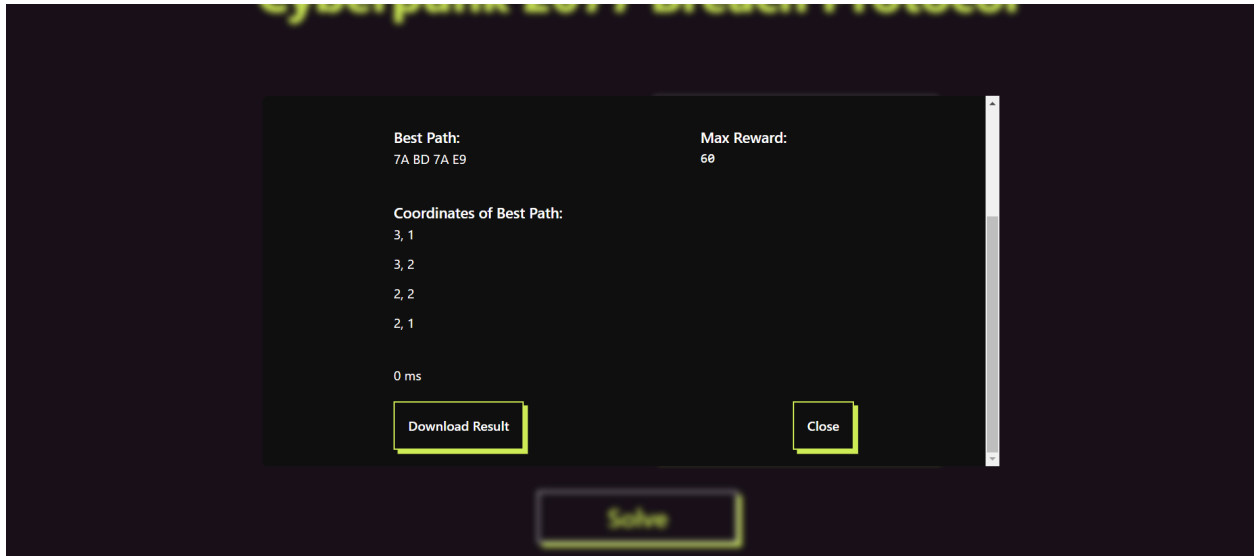
Best Path:  
7A BD 7A E9

Max Reward:  
60

Coordinates of Best Path:

- 3, 1
- 3, 2
- 2, 2

Gambar 16. Tampilan keluar pada contoh 3 (part 1)



Gambar 17. Tampilan keluar pada contoh 3 (part 2)

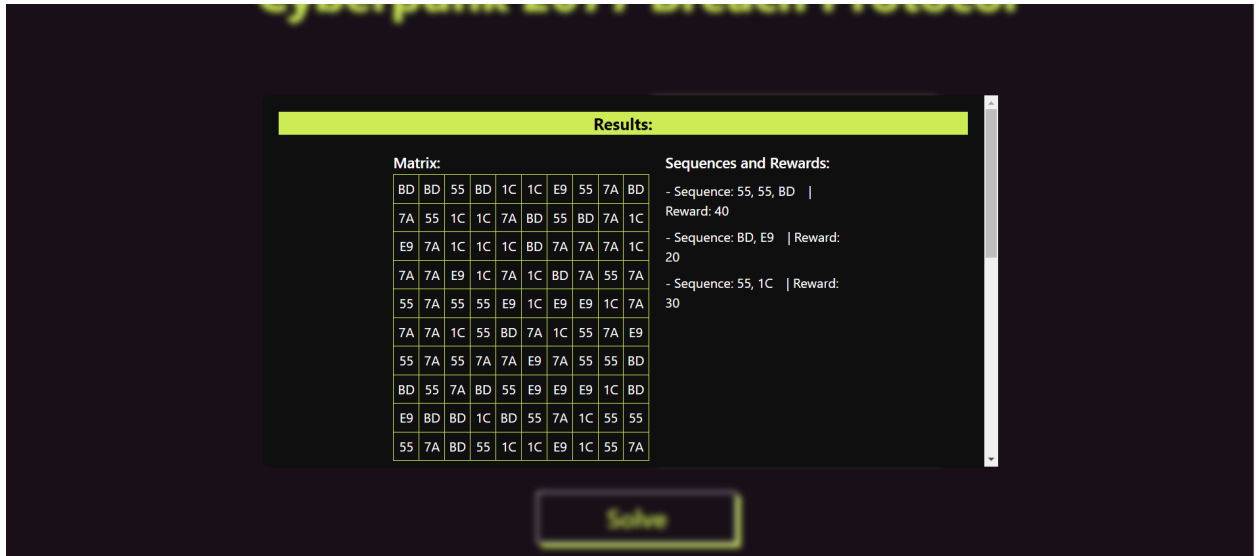
#### 4) Contoh 4

The screenshot shows the 'Cyberpunk 2077 Breach Protocol' interface with input fields for various parameters. The interface includes a 'First Page' button in the top left and an 'Input From File' button in the top right. The input fields are as follows:

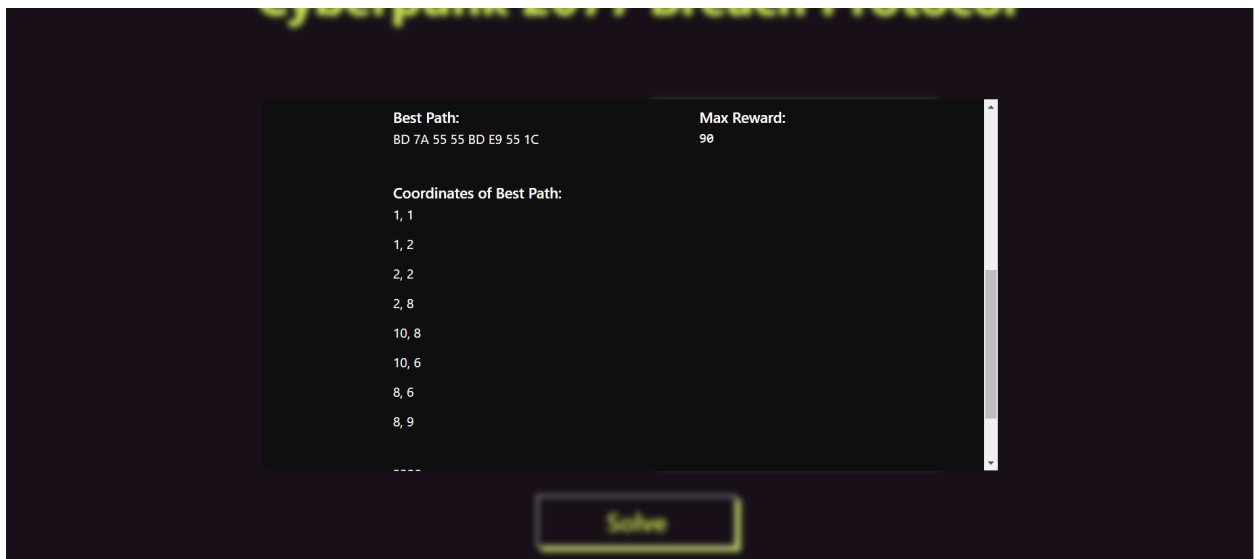
- Jumlah Token Unik :** 5
- Token Unik :** 7A 55 E9 BD 1C
- Ukuran Buffer :** 8
- Ukuran Matriks :** Rows 10 Cols 10
- Jumlah Sequences :** 3
- Ukuran Maksimal Sequences :** 3

A large 'Solve' button is located at the bottom center of the interface.

Gambar 18. Tampilan masukan pada contoh 4



Gambar 19. Tampilan keluar pada contoh 4 (part 1)



Gambar 20. Tampilan keluar pada contoh 4 (part 2)

## 5) Contoh 5

First Page Input From File

Jumlah Token Unik : 5

Token Unik : 7A 55 E9 BD 1C

Ukuran Buffer : 9

Ukuran Matriks : Rows 6 Cols 6

Jumlah Sequences : 5

Ukuran Maksimal Sequences : 3

Solve

Gambar 21. Tampilan masukan pada contoh 5

Results:

Matrix:

1C	7A	55	55	E9	E9
55	7A	55	7A	1C	55
1C	BD	E9	7A	BD	7A
BD	E9	E9	55	7A	7A
7A	E9	55	E9	BD	1C
55	7A	1C	55	BD	E9

Sequences and Rewards:

- Sequence: BD, 55 | Reward: 45
- Sequence: E9, 1C | Reward: 5
- Sequence: BD, BD, 1C | Reward: 45
- Sequence: BD, 55 | Reward: 20
- Sequence: 1C, 1C, 7A | Reward: 30

Best Path:  
1C 1C 7A 55 BD 55 BD BD 1C

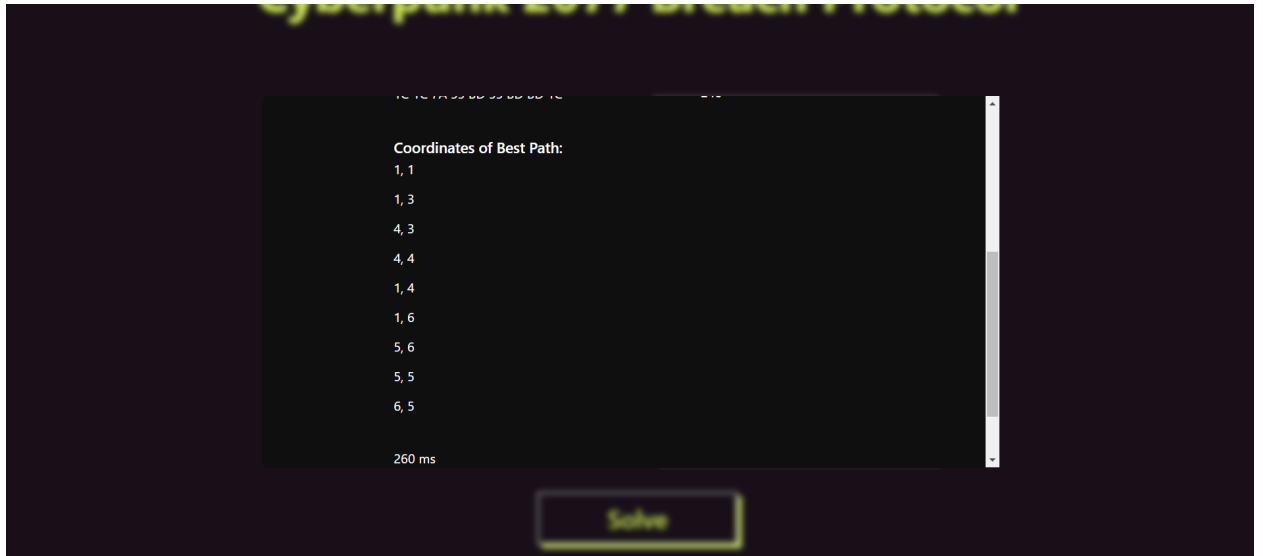
Max Reward:  
140

Coordinator of Best Path:

Solve

Gambar 22. Tampilan keluar pada contoh 5 (part 1)





Gambar 23. Tampilan keluar pada contoh 5 (part 2)

## 6) Contoh 6

First Page

Input From File

Jumlah Token Unik : 3

Token Unik : 1C BD 7A

Ukuran Buffer : 4

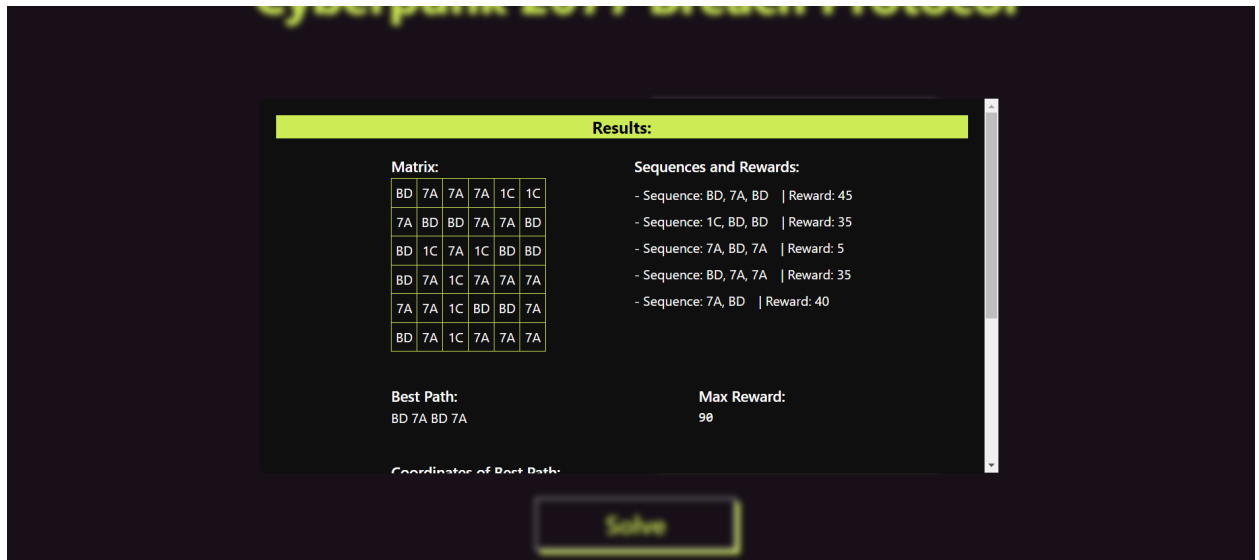
Ukuran Matriks : Rows 6 Cols 6

Jumlah Sequences : 5

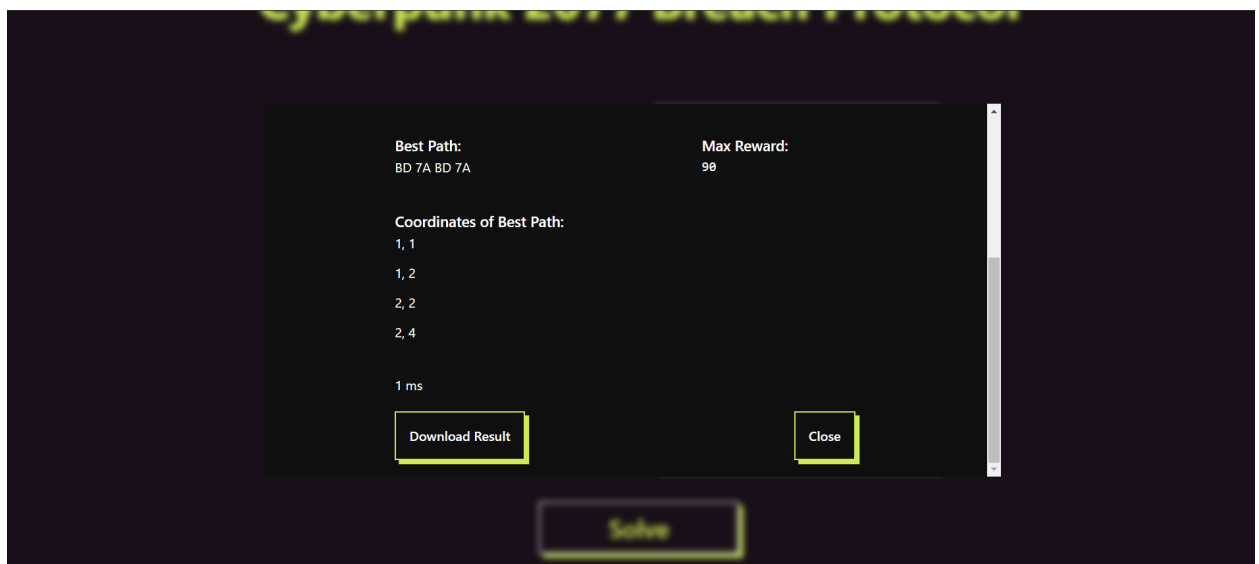
Ukuran Maksimal Sequences : 3

Solve

Gambar 24. Tampilan masukan pada contoh 6



Gambar 25. Tampilan keluar pada contoh 6 (part 1)



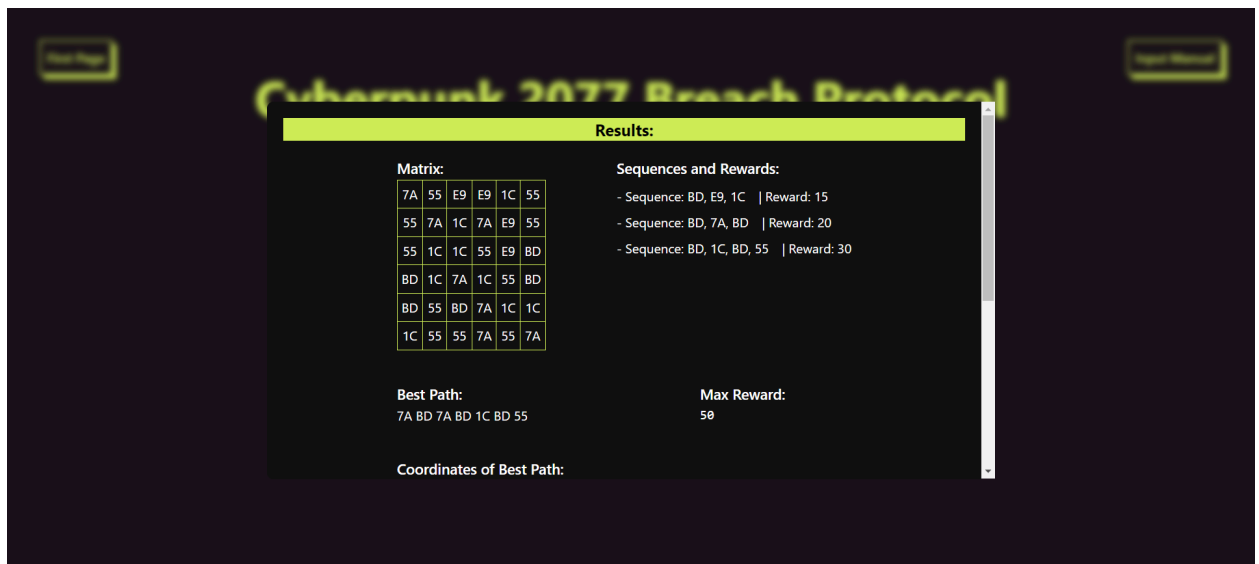
Gambar 26. Tampilan keluar pada contoh 6 (part 2)

b. Masukan dengan file.txt

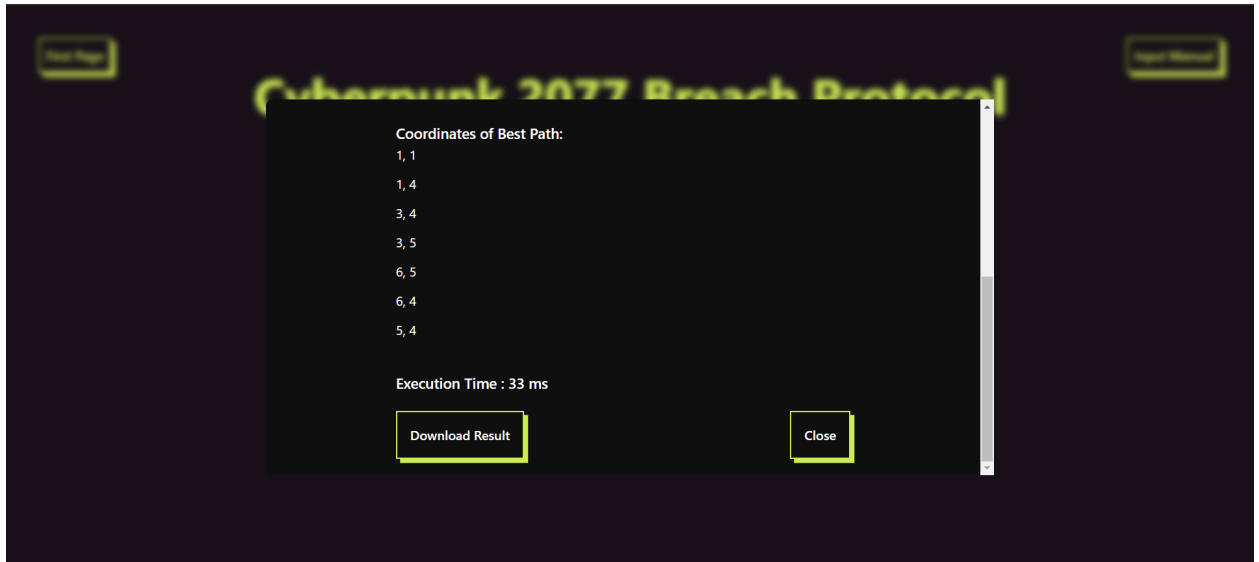
1) Contoh 7



Gambar 27. Tampilan masukan pada contoh 7



Gambar 28. Tampilan keluar pada contoh 7 (part 1)

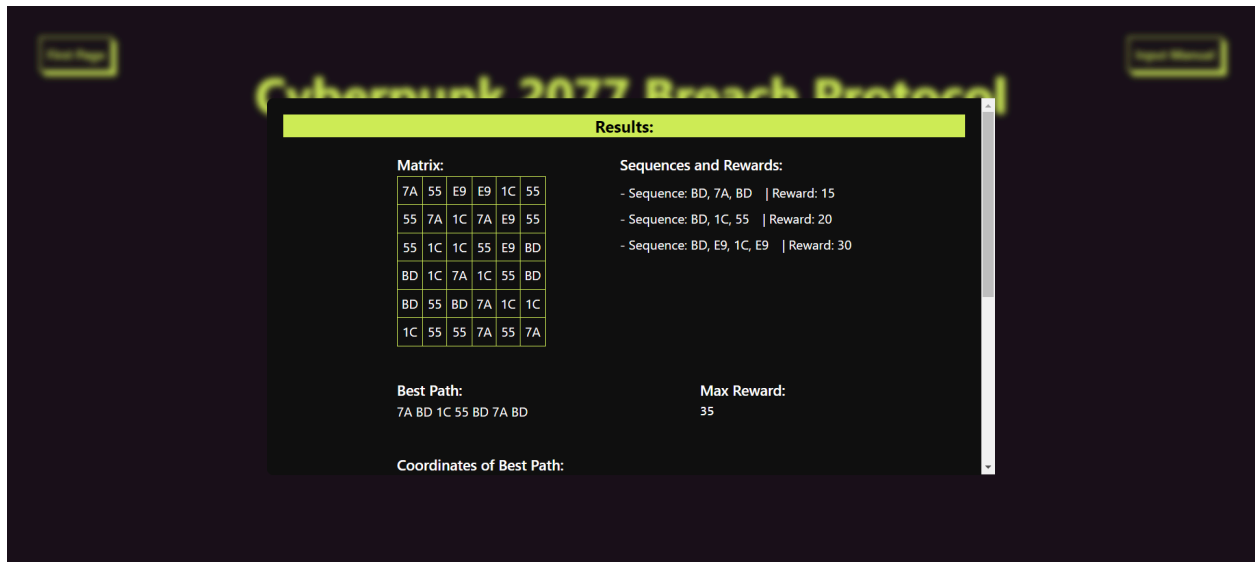


*Gambar 29. Tampilan keluar pada contoh 7 (part 2)*

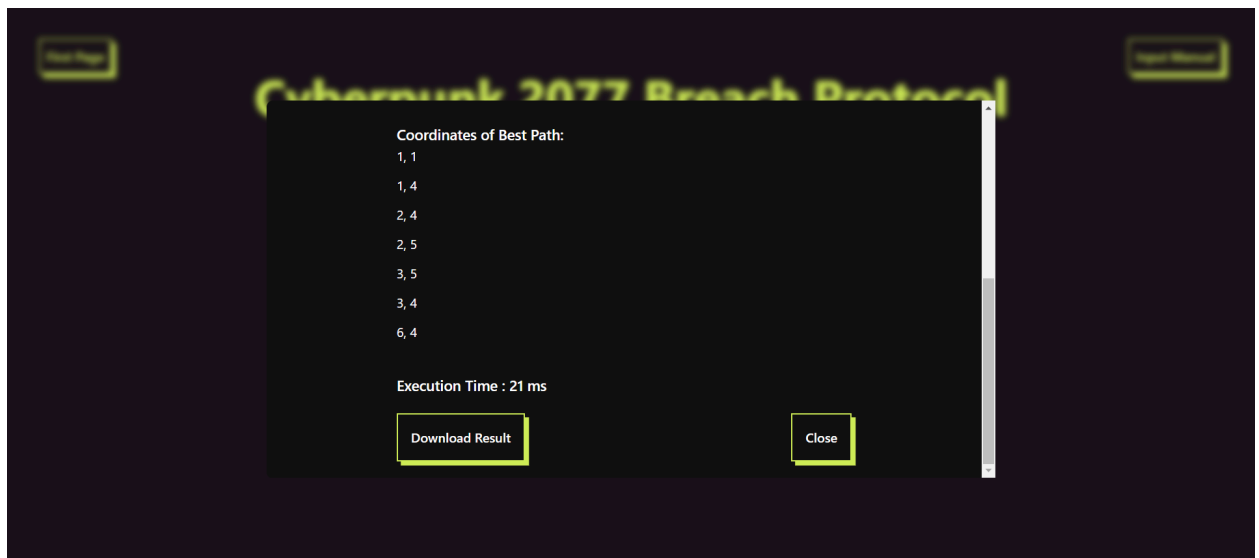
## 2) Contoh 8



*Gambar 29. Tampilan masukan pada contoh 8*



Gambar 30. Tampilan keluar pada contoh 8 (part 1)

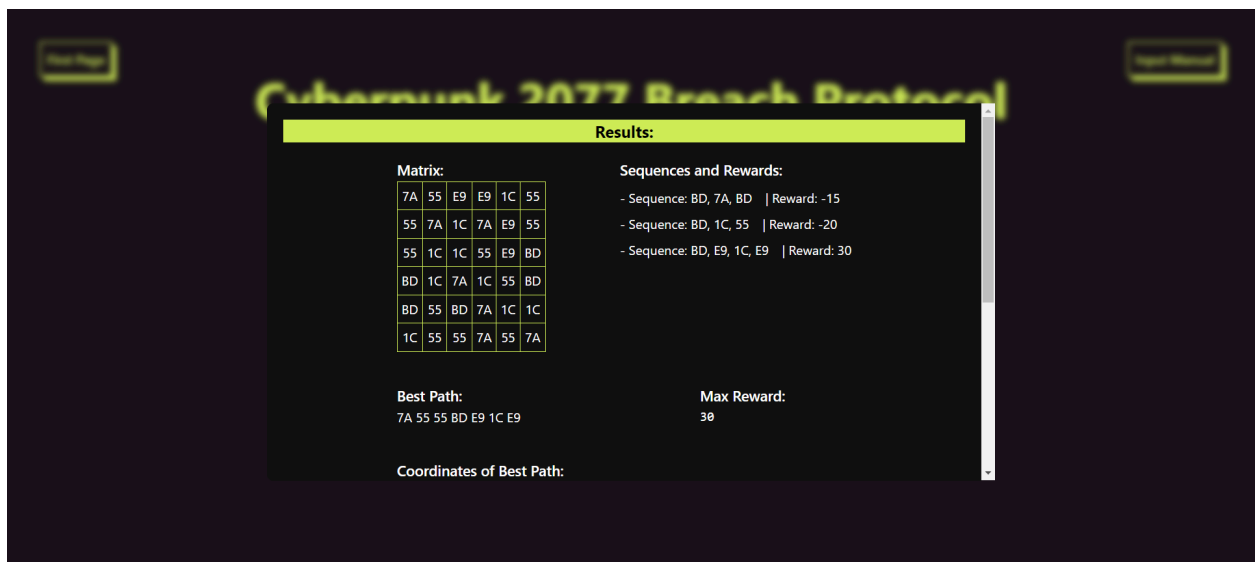


Gambar 31. Tampilan keluar pada contoh 8 (part 2)

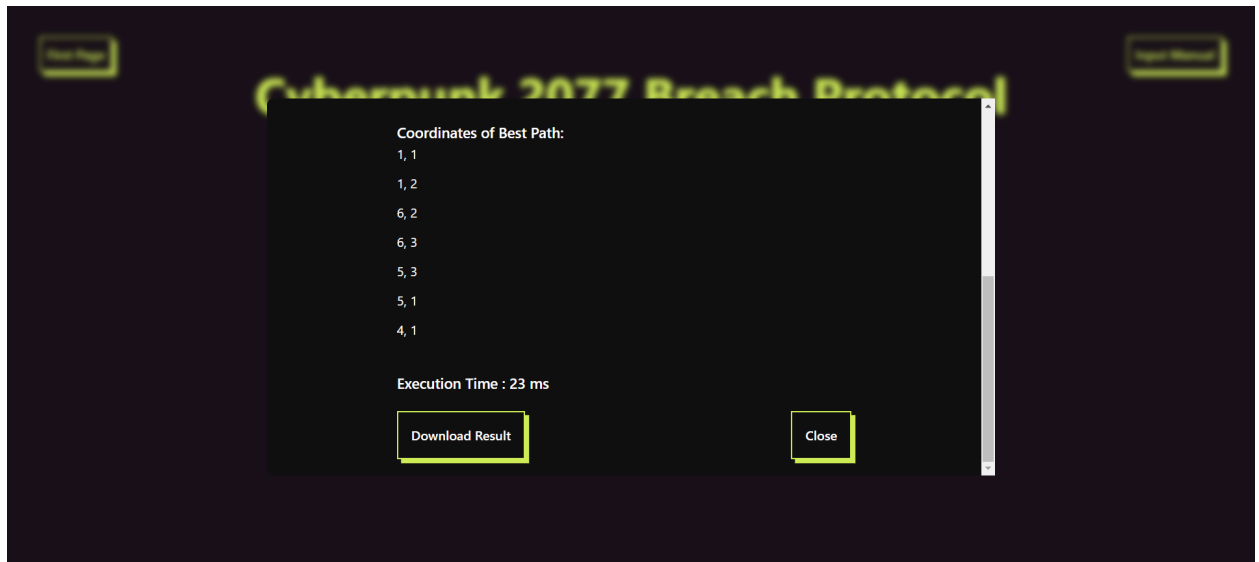
### 3) Contoh 9



Gambar 32. Tampilan masukan pada contoh 9



Gambar 33. Tampilan keluar pada contoh 9 (part 1)



*Gambar 34. Tampilan keluar pada contoh 9 (part 2)*

Dari bab eksperimen, terbukti bahwa program berbasis GUI website berhasil menjalankan fungsi-fungsinya sesuai dengan yang telah ditentukan. Dapat dilihat dari program yang mampu membaca file melalui upload file (seperti ditunjukkan dalam contoh 7,8,9) serta melalui pemilihan secara acak (sepaimana dijelaskan dalam contoh 1 hingga 6). Selain itu, program ini efektif dalam menerapkan algoritma brute force untuk menemukan solusi, yang kemudian dapat ditampilkan kepada pengguna setelah mereka memilih untuk menyelesaikan masukan yang ada. Fitur Download Results juga memungkinkan pengguna untuk menyimpan solusi tersebut ke direktori lokal download/ mereka.

# Lampiran

## Github Repository:

[https://github.com/Benardo07/Tucil1\\_13522055](https://github.com/Benardo07/Tucil1_13522055)

**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	✓	