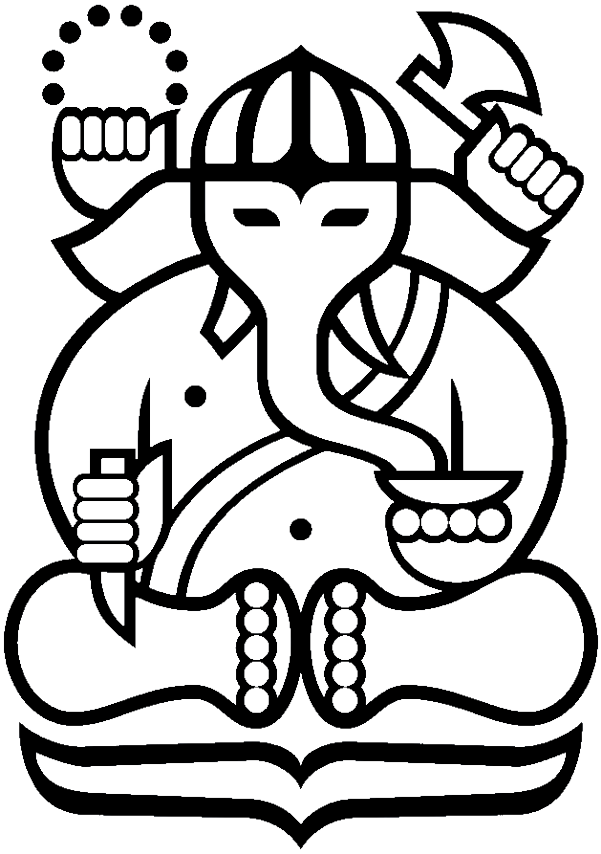
LAPORAN TUGAS KECIL 1

Penyelesaian Pola Paling Optimal pada Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun Oleh:

Bagas Sambega Rosyada – 13522071

Mata Kuliah IF2211 Strategi Algoritma

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

DAFTAR ISI 1
DAFTAR GAMBAR 2
A. Algoritma Program 3
B. Kode dan Logika Program 4
a. Layout 4
b. Logika Program 6
C. Pengujian Program 12
a. File Upload 12
b. Input Acak 15

DAFTAR GAMBAR

Gambar 1. Kode HTML untuk upload file 5
Gambar 2. Kode HTML untuk random input 6
Gambar 3. Variabel global pada Javascript 7
Gambar 4. Fungsi untuk menghasilkan matriks dan sekuens secara acak. 7
Gambar 5. Fungsi untuk mengambil data dari file yang diunggah pengguna. 8
Gambar 6. Fungsi untuk menampilkan data ke pengguna 9
Gambar 7. Fungsi untuk memanggil fungsi rekursif yang mencari solusi optimum dan menunjukkan hasilnya ke pengguna. 10
Gambar 8. Fungsi rekursif untuk mencari seluruh kemungkinan kombinasi token yang ada. 11
Gambar 9. Fungsi untuk mengunduh hasil permainan dalam file ekstensi .txt. 12
Gambar 10. File .txt untuk pengunggahan pada pengujian pertama 12
Gambar 11. Tangkapan layar hasil matriks dan solusi dari file yang diunggah. 13
Gambar 12. Solusi dari permainan yang sudah disimpan dalam file .txt 13
Gambar 13. File .txt untuk pengujian kedua 13
Gambar 14. Tangkapan layar solusi hasil pemrosesan dari file .txt kedua 14
Gambar 15. File .txt untuk pengujian ketiga 14
Gambar 16. Tangkapan layar hasil solusi dari unggahan file ketiga 15
Gambar 17. Tangkapan layar pengujian pertama input acak dan matriks solusi 15
Gambar 18. Tangkapan layar solusi permainan dari input acak kedua 16
Gambar 19. File solusi .txt dari input acak kedua 16
Gambar 20. Hasil pengujian dengan input acak dari pengguna 16

A. Algoritma Program

Algoritma *brute force* adalah algoritma umum yang mencari setiap kemungkinan satu per satu untuk mendapatkan suatu penyelesaian. Algoritma ini juga sering disebut sebagai *exhaustive algorithm*.

Pada permainan Cyberpunk 2077, terdapat sebuah *minigame* bernama Breach Protocol, dan dapat diimplementasikan algoritma *brute force*. Pada program ini, akan terdapat beberapa istilah yang digunakan, yaitu,

1. Matriks, yaitu kumpulan token yang akan dicari *buffer* dengan pola vertikal-horizontal-vertikal-horizontal, dan seterusnya dengan nilai maksimal yang dapat dihasilkan dari sekumpulan sekuens.
2. Sekuens, sekumpulan token dengan pola khusus yang memiliki nilai *reward*.
3. *Buffer* saat ini, yaitu sekumpulan token yang sedang dicari atau ditambahkan oleh program selama proses pencarian rekursif berlangsung.

Algoritma yang digunakan untuk mencari solusi dengan *reward* paling optimal adalah sebagai berikut,

1. Akan ditentukan titik awal pencarian di baris paling atas matriks.
2. Setelah ditentukan suatu titik awal, program akan melakukan pencarian secara rekursif, diawali dengan pencarian pada pola vertikal pada matriks atau pada suatu kolom yang sama.
3. Setelah ditemukan suatu titik lain, maka pola pencarian vertikal akan berganti menjadi horizontal atau pada satu baris yang sama, lalu setelah ditemukan titik lain, pola pencarian akan berganti kembali menjadi vertikal.
4. Setiap ditemukan suatu titik, maka token pada titik tersebut akan ditambahkan ke *string buffer* saat ini sehingga panjang *buffer* saat ini akan bertambah satu. Untuk setiap penambahan panjang *buffer*, akan dicek nilai *reward* yang dihasilkan *buffer* saat itu,

lalu dibandingkan dengan nilai *reward* maksimal saat itu yang dimiliki. Jika *reward* yang ditemukan lebih besar dari *reward* maksimal saat ini, maka *reward* maksimal saat ini akan diganti dengan nilai *reward* saat ini. Hal ini dikarenakan dimungkinkan didapat *reward* maksimal tanpa perlu *string buffer* memiliki panjang yang sama dengan maksimal *buffer*.

5. Program akan mencari seluruh kemungkinan dimulai dari panjang *buffer* saat ini minimal satu, hingga panjang *buffer* saat ini sama dengan maksimal *buffer*.
6. Program akan berhenti mencari saat panjang *buffer* yang saat ini sedang dicari sama dengan panjang maksimal *buffer*, dan seluruh kemungkinan tersebut sudah selesai dicari.

B. Kode dan Logika Program

Bahasa logika program yang digunakan adalah Javascript, dengan tampilan menggunakan HTML dan CSS. Pada bab ini, hanya akan ditunjukkan layout untuk *upload.html* dan *random.html* saja sebagai bagian utama dari program. *Source code* untuk program ini dapat diakses melalui https://github.com/bagassambega/Tucil1_13522071.

a. Layout

GUI pada program ini menggunakan HTML dan CSS. Berikut adalah kode untuk tampilan pada *upload.html* dan *random.html*.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cyberpunk 2077 Breach Protocol Solver</title>
    <link href="upload.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
    <div class="navigation">
        <div class="buttons">
            <a href="main.html">
                <i class="fa fa-home"></i>
            </a>
            <a href="random.html">
                <i class="fa fa-file-text-o"></i>
            </a>
        </div>
    </div>
    <div class="container">
        <div id="first">
            <div class="card instruction">
                <h2>Instruksi</h2>
                <p>Upload file .txt yang berisi kode Breach Protocol dengan format:</p>
                <ul>
                    <li>Ukuran buffer</li>
                    <li>Lebar matriks [spasi] Tinggi matriks</li>
                    <li>Elemen-elemen matriks (setiap row dipisahkan newline, dan setiap elemen kolom dipisahkan spasi)</li>
                    <li>Banyak sekuens</li>
                    <li>Sekuens 1</li>
                    <li>Reward sekuens 1</li>
                    <li>Sekuens 2</li>
                    <li>Reward sekuens 2</li>
                    ...
                    <li>Sekuens <em>n</em></li>
                    <li>Reward sekuens <em>n</em></li>
                </ul>
            </div>
            <div class="card box">
                <label for="fileInput">Choose a file</label><br>
                <input type="file" name="fileInput" id="fileInput" accept=".txt" hidden>
                <p id="fileChosen"></p>
            </div>
        </div>

        <div id="second">
            <div class="card content">
                <h2>Data</h2>
                <h4>Matriks</h4>
                <div id="matriks" class="matriks"></div>
                <h4>Sekuens dan reward</h4>
                <div id="sekuens" class="sekuens"></div>
                <button id="solve" onclick="solve()" hidden>Solve</button>
                <p id="processing">Processing...</p>
            </div>
            <div class="card result">
                <h4>Hasil dan Reward</h4>
                <div id="sequence"></div>
                <button id="download" onclick="download()" hidden>Download</button>
            </div>
        </div>
    </div>
    <script src="upload.js"></script>
</body>
</html>
```

Gambar 1. Kode HTML untuk upload file

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Generate Random</title>
    <link href="random.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
<div class="navigation">
    <div class="buttons">
        <a href="index.html">
            <i class="fa fa-home"></i>
        </a>
        <a href="random.html">
            <i class="fa fa-file-text-o"></i>
        </a>
    </div>
</div>
    <div class="container">
        <div id="first">
            <h1>Input</h1>
            <div class="inputForm">
                <div class="box">
                    <label for="rowMatrix">Tinggi matriks</label>
                    <input type="number" name="rowMatrix" id="rowMatrix" placeholder="0" required   >
                </div>
                <div class="box">
                    <label for="colMatrix">Lebar matriks</label>
                    <input type="number" name="colMatrix" id="colMatrix" placeholder="0">
                </div>
                <div class="box">
                    <label for="bufferSize">Ukuran buffer</label>
                    <input type="number" name="bufferSize" id="bufferSize" placeholder="0">
                </div>
                <div class="box">
                    <label for="token">Token (pisahkan oleh spasi)</label>
                    <input type="text" name="token" id="token" placeholder="7A BD 1C 55">
                </div>
                <div class="box">
                    <label for="numSeq">Banyak sekuens</label>
                    <input type="number" name="numSeq" id="numSeq" placeholder="0">
                </div>
                <div class="box">
                    <label for="sequenceSize">Ukuran maksimal sekuens</label>
                    <input type="number" id="sequenceSize" name="sequenceSize" placeholder="0">
                </div>
                <p id="inputError"></p>
                <button id="generate" onclick="generate()">Generate</button>
            </div>
        </div>

        <div id="second">
            <h1>Output</h1>
            <div class="generated">
                <h3>Matriks</h3>
                <div class="matriks" id="matriks"></div>
                <h3>Sekuens</h3>
                <div class="sekuens" id="sekuens"></div>
                <button id="solve" onclick="solve()"> Solve </button>
                <p id="processing">Processing...</p>
            </div>
            <div class="result" id="result">
                <h3>Hasil dan Reward</h3>
                <div class="hasil" id="hasil"></div>
                <button id="download" onclick="download()">Download solution</button>
            </div>
        </div>
    </div>

    <script src="random.js"></script>
</body>
</html>
```

Gambar 2. Kode HTML untuk random input

b. Logika Program

Garis besar cara kerja program adalah program menerima input/membuat suatu matriks yang berisi kumpulan token, lalu akan dicari sekuens yang berasal dari pencarian secara vertikal-horizontal-vertikal-horizontal dan seterusnya, hingga ditemukan suatu kumpulan token yang memiliki nilai *reward* yang terbesar.

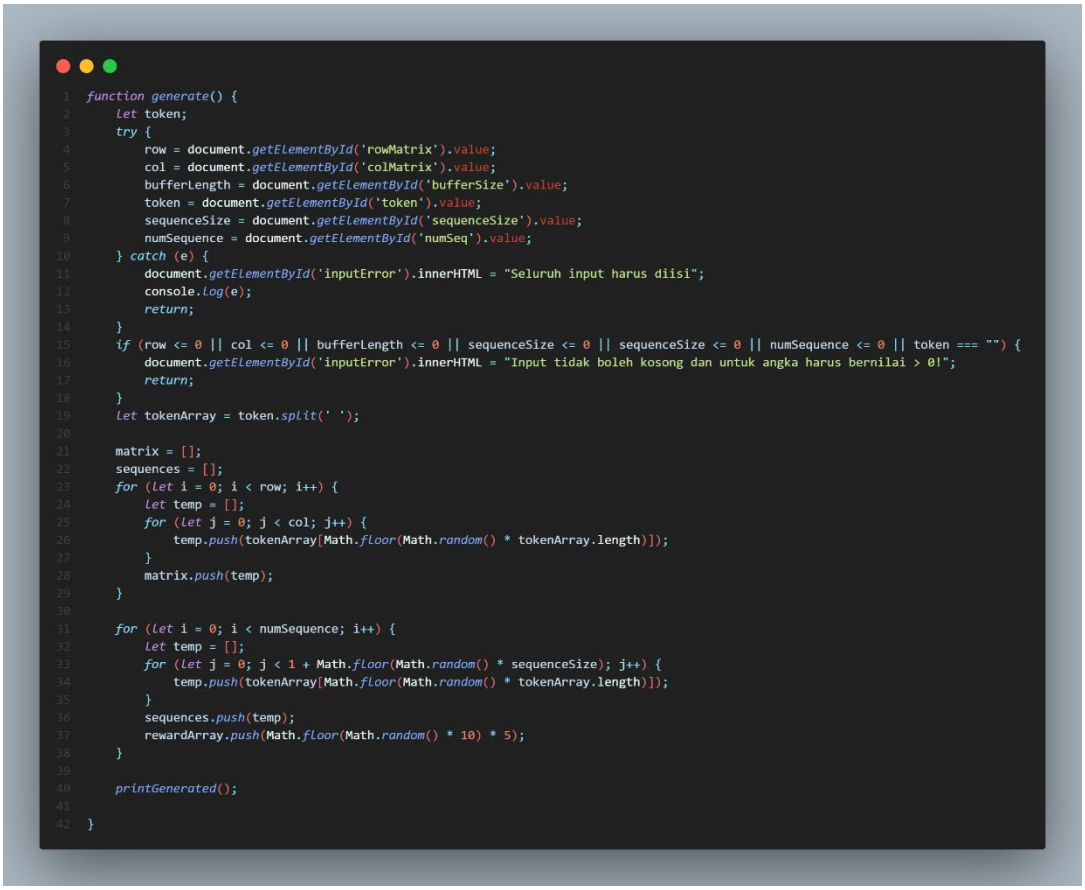
Pada setiap kode program Javascript, terdapat beberapa variabel global yang digunakan untuk menyimpan data yang akan terus menerus dipakai selama program berlangsung.

```javascript
// Global variable

let matrix = [];
let bufferLength;
let sequences = [];
let rewardArray = [];
let maxReward = 0;
let pathResult = [];
let coordinateResult = [];
let sequenceSize;
let numSequence;
let row, col;
let timeElapsed;
```

Gambar 3. Variabel global pada Javascript

Jika pengguna memilih untuk melakukan input acak, maka setelah pengguna mengisi seluruh data yang diperlukan, yaitu panjang kolom dan baris matriks, banyak sekuens, token-token, dan panjang maksimum tiap sekuens, program akan menghasilkan suatu matriks dan sekuens acak beserta *reward*-nya menggunakan fungsi *Math.random()* dari Javascript.

Sementara jika pengguna memilih untuk menjalankan program dengan melakukan unggah file berekstensi “txt”, maka akan dijalankan suatu fungsi *upload()* pada file *upload.js*. Fungsi tersebut akan membaca file yang diunggah oleh pengguna, dan akan dilakukan *parsing* pada setiap barisnya untuk mendapatkan data panjang *buffer*, banyak kolom dan baris matriks, matriks token, banyak sekuens, sekuens-sekuens beserta dengan besaran *reward*-nya. Setelah didapat data tersebut, variabel global akan diisi dengan data-data yang berasal dari file tersebut.



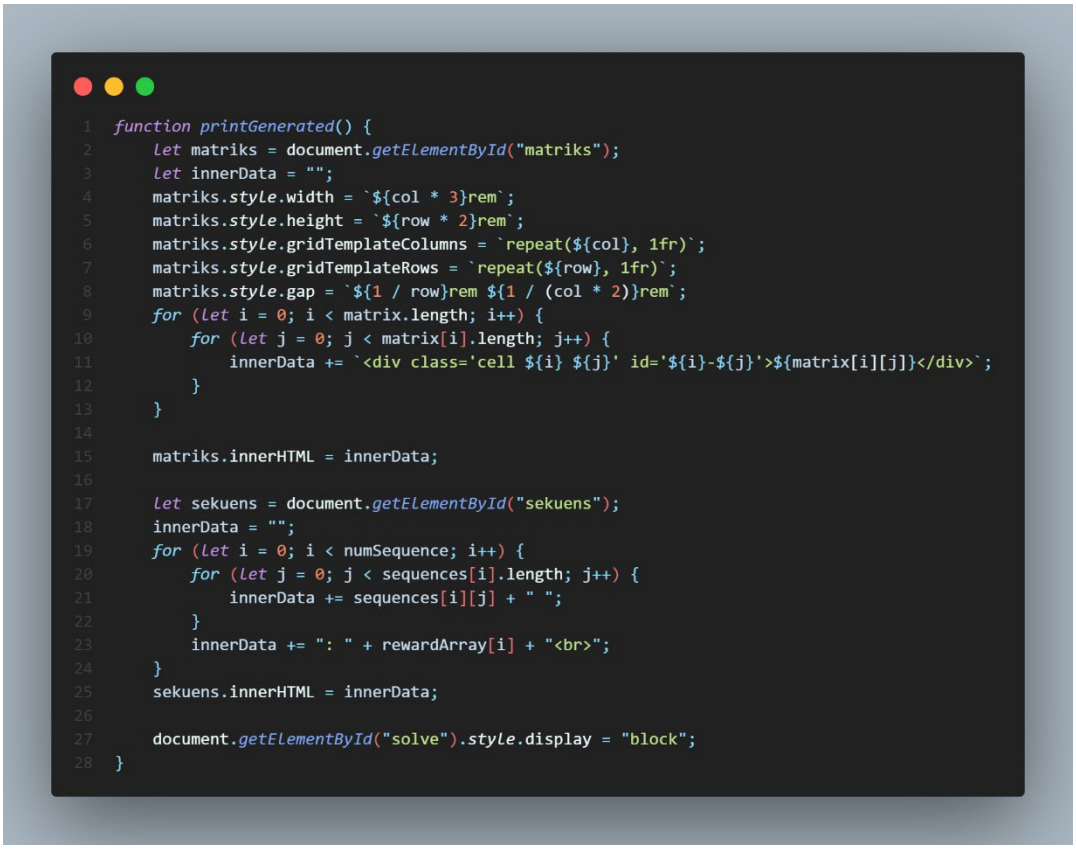
```javascript
function generate() {
    let token;
    try {
        row = document.getElementById('rowMatrix').value;
        col = document.getElementById('colMatrix').value;
        bufferLength = document.getElementById('bufferSize').value;
        token = document.getElementById('token').value;
        sequenceSize = document.getElementById('sequenceSize').value;
        numSequence = document.getElementById('numSeq').value;
    } catch (e) {
        document.getElementById('inputError').innerHTML = "Seluruh input harus diisi";
        console.log(e);
        return;
    }
    if (row <= 0 || col <= 0 || bufferLength <= 0 || sequenceSize <= 0 || sequenceSize <= 0 || numSequence <= 0 || token === "") {
        document.getElementById('inputError').innerHTML = "Input tidak boleh kosong dan untuk angka harus bernilai > 0!";
        return;
    }
    let tokenArray = token.split(' ');

    matrix = [];
    sequences = [];
    for (let i = 0; i < row; i++) {
        let temp = [];
        for (let j = 0; j < col; j++) {
            temp.push(tokenArray[Math.floor(Math.random() * tokenArray.length)]);
        }
        matrix.push(temp);
    }

    for (let i = 0; i < numSequence; i++) {
        let temp = [];
        for (let j = 0; j < 1 + Math.floor(Math.random() * sequenceSize); j++) {
            temp.push(tokenArray[Math.floor(Math.random() * tokenArray.length)]);
        }
        sequences.push(temp);
        rewardArray.push(Math.floor(Math.random() * 10) * 5);
    }

    printGenerated();

}
```

Gambar 4. Fungsi untuk menghasilkan matriks dan sekuens secara acak.

```
function upload() {
    const fileInput = document.getElementById("fileInput");
    const files = fileInput.files;

    const file = files[0];
    const reader = new FileReader();

    reader.onload = function(e) {
        try {
            matrix = [];
            sequences = [];
            maxReward = 0;
            pathResult = [];
            coordinateResult = [];
            const content = e.target.result;
            const lines = content.split(/\r\n|\r|\n/).map(line => line.replace(/\r|\n/g, ''))
            bufferLength = parseInt(lines[0]);
            let splitInt = lines[1].split(' ');
            col = parseInt(splitInt[0]);
            row = parseInt(splitInt[1]);
            for (let i = 0; i < row; i++) {
                let splitInt = lines[i + 2].split(' ');
                let temp = [];
                for (let j = 0; j < col; j++) {
                    temp.push(splitInt[j]);
                }
                matrix.push(temp);
            }

            let now = row + 2;
            sequenceNumber = parseInt(lines[now]);
            now += 1;
            for (let i = 0; i < sequenceNumber * 2; i++) {
                if (i % 2 === 0) {
                    sequences.push(lines[now + i].split(' '));
                } else {
                    rewardArray.push(parseInt(lines[now + i]));
                }
            }

            // Show in HTML

            let matriks = document.getElementById("matriks");
            let innerData = "";
            matriks.style.width = `${col * 3}rem`;
            matriks.style.height = `${row * 2}rem`;
            matriks.style.gridTemplateColumns = `repeat(${col}, 1fr)`;
            matriks.style.gridTemplateRows = `repeat(${row}, 1fr)`;
            matriks.style.gap = `${1 / row}rem ${1 / (col * 2)}rem`;
            for (let i = 0; i < matrix.length; i++) {
                for (let j = 0; j < matrix[i].length; j++) {
                    innerData += `<div class='cell ${i} ${j}' id='${i}-${j}'>${matrix[i][j]}</div>`;
                }
            }

            matriks.innerHTML = innerData;

            let sekuens = document.getElementById("sekuens");
            innerData = "";
            for (let i = 0; i < sequenceNumber; i++) {
                for (let j = 0; j < sequences[i].length; j++) {
                    innerData += sequences[i][j] + " ";
                }
                innerData += ": " + rewardArray[i] + "<br>";
            }
            sekuens.innerHTML = innerData;
        } catch (e) {
            document.getElementById("fileChosen").textContent = "Terdapat kesalahan pada pembacaan file!";
            console.log("error" + e);
            return;
        }
    };


    reader.readAsText(file);
}
```

Gambar 5. Fungsi untuk mengambil data dari file yang diunggah pengguna.

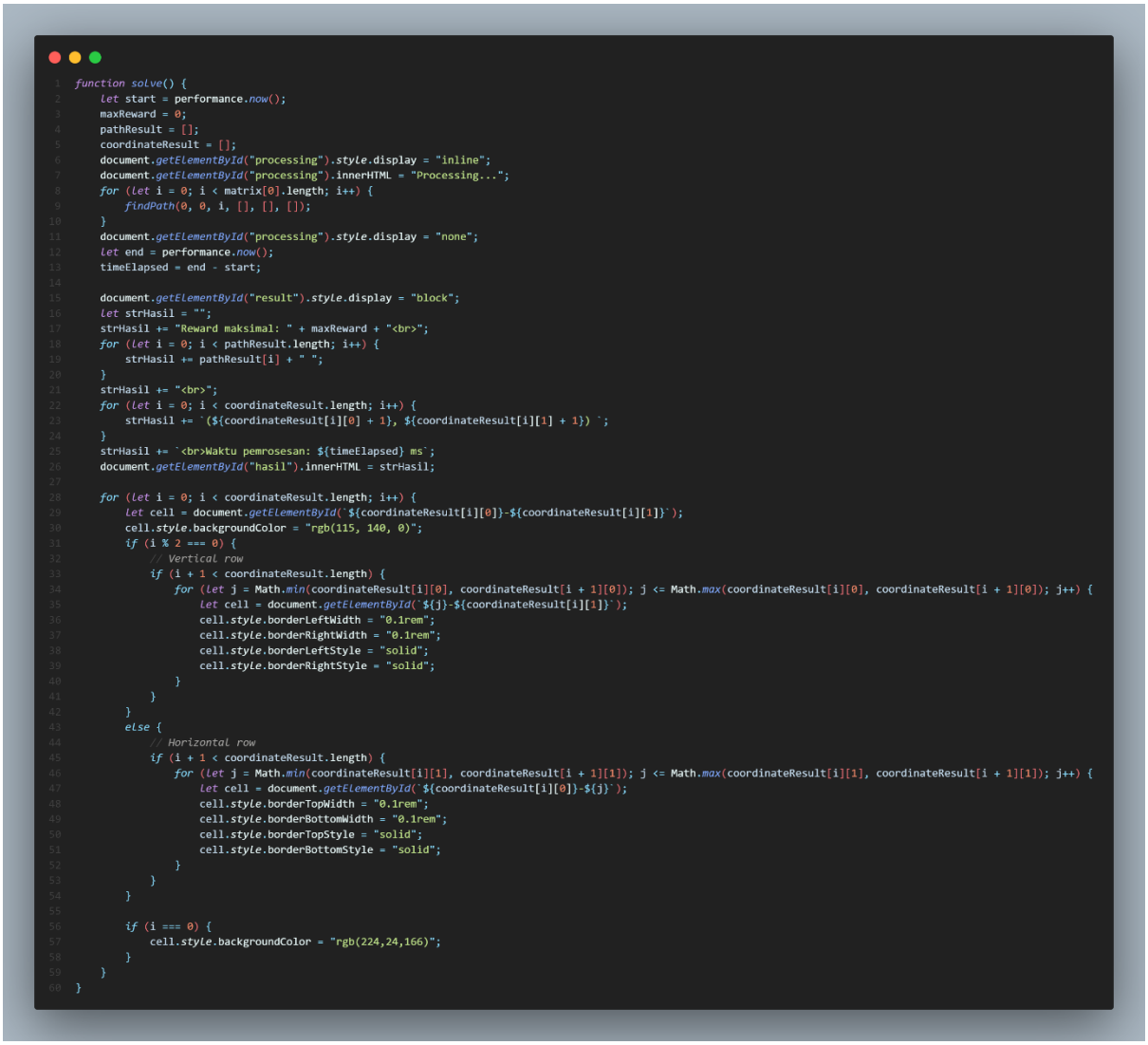
Setelah seluruh data diterima, maka akan dijalankan fungsi untuk menampilkan matriks yang telah disimpan pada Javascript ke pengguna.



```
function printGenerated() {
    let matriks = document.getElementById("matriks");
    let innerData = "";
    matriks.style.width = `${col * 3}rem`;
    matriks.style.height = `${row * 2}rem`;
    matriks.style.gridTemplateColumns = `repeat(${col}, 1fr)`;
    matriks.style.gridTemplateRows = `repeat(${row}, 1fr)`;
    matriks.style.gap = `${1 / row}rem ${1 / (col * 2)}rem`;
    for (let i = 0; i < matrix.length; i++) {
        for (let j = 0; j < matrix[i].length; j++) {
            innerData += `<div class='cell ${i} ${j}' id='${i}-${j}'>${matrix[i][j]}</div>`;
        }
    }

    matriks.innerHTML = innerData;

    let sekuens = document.getElementById("sekuens");
    innerData = "";
    for (let i = 0; i < numSequence; i++) {
        for (let j = 0; j < sequences[i].length; j++) {
            innerData += sequences[i][j] + " ";
        }
        innerData += ": " + rewardArray[i] + "<br>";
    }
    sekuens.innerHTML = innerData;

    document.getElementById("solve").style.display = "block";
}
```

Gambar 6. Fungsi untuk menampilkan data ke pengguna

Pengguna dapat memulai program dengan menekan tombol *Solve* untuk mencari solusi optimum dari matriks, dan setelah tombol tersebut ditekan, maka fungsi utama program akan dimulai. Fungsi akan mencari seluruh kemungkinan secara rekursif, menyimpan dan memperbarui hasil optimal di variabel-variabel global, dan kemudian menunjukkan hasilnya ke pengguna.



```
function solve() {
    let start = performance.now();
    maxReward = 0;
    pathResult = [];
    coordinateResult = [];
    document.getElementById("processing").style.display = "inline";
    document.getElementById("processing").innerHTML = "Processing...";
    for (let i = 0; i < matrix[0].length; i++) {
        findPath(0, 0, i, [], [], []);
    }
    document.getElementById("processing").style.display = "none";
    let end = performance.now();
    timeElapsed = end - start;

    document.getElementById("result").style.display = "block";
    let strHasil = "";
    strHasil += "Reward maksimal: " + maxReward + "<br>";
    for (let i = 0; i < pathResult.length; i++) {
        strHasil += pathResult[i] + " ";
    }
    strHasil += "<br>";
    for (let i = 0; i < coordinateResult.length; i++) {
        strHasil += `(${coordinateResult[i][0] + 1}, ${coordinateResult[i][1] + 1}) `;
    }
    strHasil += `<br>Waktu pemrosesan: ${timeElapsed} ms`;
    document.getElementById("hasil").innerHTML = strHasil;

    for (let i = 0; i < coordinateResult.length; i++) {
        let cell = document.getElementById(`${coordinateResult[i][0]}-${coordinateResult[i][1]}`);
        cell.style.backgroundColor = "rgb(115, 140, 0)";
        if (i % 2 === 0) {
            // Vertical row
            if (i + 1 < coordinateResult.length) {
                for (let j = Math.min(coordinateResult[i][0], coordinateResult[i + 1][0]); j <= Math.max(coordinateResult[i][0], coordinateResult[i + 1][0]); j++) {
                    let cell = document.getElementById(`${j}-${coordinateResult[i][1]}`);
                    cell.style.borderLeftWidth = "0.1rem";
                    cell.style.borderRightWidth = "0.1rem";
                    cell.style.borderLeftStyle = "solid";
                    cell.style.borderRightStyle = "solid";
                }
            }
        }
        else {
            // Horizontal row
            if (i + 1 < coordinateResult.length) {
                for (let j = Math.min(coordinateResult[i][1], coordinateResult[i + 1][1]); j <= Math.max(coordinateResult[i][1], coordinateResult[i + 1][1]); j++) {
                    let cell = document.getElementById(`${coordinateResult[i][0]}-${j}`);
                    cell.style.borderTopWidth = "0.1rem";
                    cell.style.borderBottomWidth = "0.1rem";
                    cell.style.borderTopStyle = "solid";
                    cell.style.borderBottomStyle = "solid";
                }
            }
        }

        if (i === 0) {
            cell.style.backgroundColor = "rgb(224,24,166)";
        }
    }
}
```

Gambar 7. Fungsi untuk memanggil fungsi rekursif yang mencari solusi optimum dan menunjukkan hasilnya ke pengguna.

```
function findPath(currentBuffer, currRow, currCol, currentPath, seenCoordinates, coordinateNow) {
    if (currentBuffer <= bufferLength) {
        let reward = 0;
        let m = currentPath.length;
        for (let i = 0; i < sequences.length; i++) {
            let n = sequences[i].length;
            for (let j = 0; j <= m - n; j++) {
                let found = false;
                for (let k = 0; k < n; k++) {
                    if (currentPath[j + k] !== sequences[i][k]) {
                        found = false;
                        break;
                    }
                    found = true;
                }
                if (found) {
                    reward += rewardArray[i];
                }
            }
            if (reward > maxReward) {
                maxReward = reward;
                pathResult = [...currentPath];
                coordinateResult = [...coordinateNow];
            }
        }
    }

    if (currentBuffer === bufferLength) {
        return;
    }

    if (currentBuffer === 0) {
        for (let i = 0; i < matrix[0].length; i++) {
            findPath(1, currRow, i, [matrix[currRow][i]], [[currRow, i]], [[currRow, i]]);
        }
    }
    // Move horizontally
    else if (currentBuffer % 2 === 0 && currentBuffer !== 0) {
        for (let i = 0; i < matrix[currRow].length; i++) {
            let seen = false;
            for (let j = 0; j < seenCoordinates.length; j++) {
                if (seenCoordinates[j][0] === currRow && seenCoordinates[j][1] === i) {
                    seen = true;
                    break;
                }
            }
            if (seen) {
                continue;
            }
            seenCoordinates.push([currRow, i]);
            currentPath.push(matrix[currRow][i]);
            coordinateNow.push([currRow, i]);
            findPath(currentBuffer + 1, currRow, i, currentPath, seenCoordinates, coordinateNow);
            seenCoordinates.pop();
            currentPath.pop();
            coordinateNow.pop();
        }
    }
    // Move vertically
    else {
        for (let i = 0; i < matrix.length; i++) {
            let seen = false;
            for (let j = 0; j < seenCoordinates.length; j++) {
                if (seenCoordinates[j][0] === i && seenCoordinates[j][1] === currCol) {
                    seen = true;
                    break;
                }
            }
            if (seen) {
                continue;
            }
            seenCoordinates.push([i, currCol]);
            currentPath.push(matrix[i][currCol]);
            coordinateNow.push([i, currCol]);
            findPath(currentBuffer + 1, i, currCol, currentPath, seenCoordinates, coordinateNow);
            seenCoordinates.pop();
            currentPath.pop();
            coordinateNow.pop();
        }

    }
}
```

Gambar 8. Fungsi rekursif untuk mencari seluruh kemungkinan kombinasi token yang ada.

Jika hasil dari matriks dan sekuens telah ditemukan, maka pengguna dapat memilih untuk mengunduh hasil dan solusi permainan *Breach Protocol* ini.

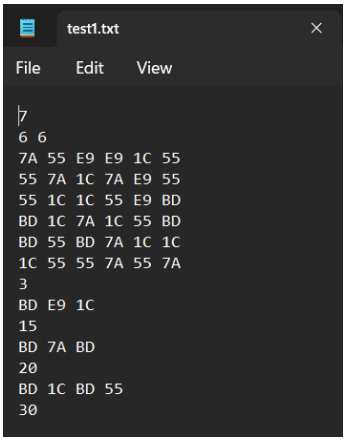
```
function download() {
    let data = "";
    if (pathResult.length === 0) {
        data += "Tidak ada solusi yang optimal\n";
    }
    else {
        data += maxReward + "\n";
        for (let i = 0; i < pathResult.length; i++) {
            data += pathResult[i] + " ";
        }
        data += "\n";
        for (let i = 0; i < coordinateResult.length; i++) {
            data += (coordinateResult[i][1] + 1) + ", " + (coordinateResult[i][0] + 1) + "\n";
        }
    }
    data += "\n" + timeElapsed + " ms";
    let blob = new Blob([data], {type: 'text/plain'});
    let url = window.URL.createObjectURL(blob);
    let a = document.createElement('a');
    a.href = url;
    a.download = 'output.txt';
    a.click();
    window.URL.revokeObjectURL(url);
}
```

Gambar 9. Fungsi untuk mengunduh hasil permainan dalam file ekstensi .txt.

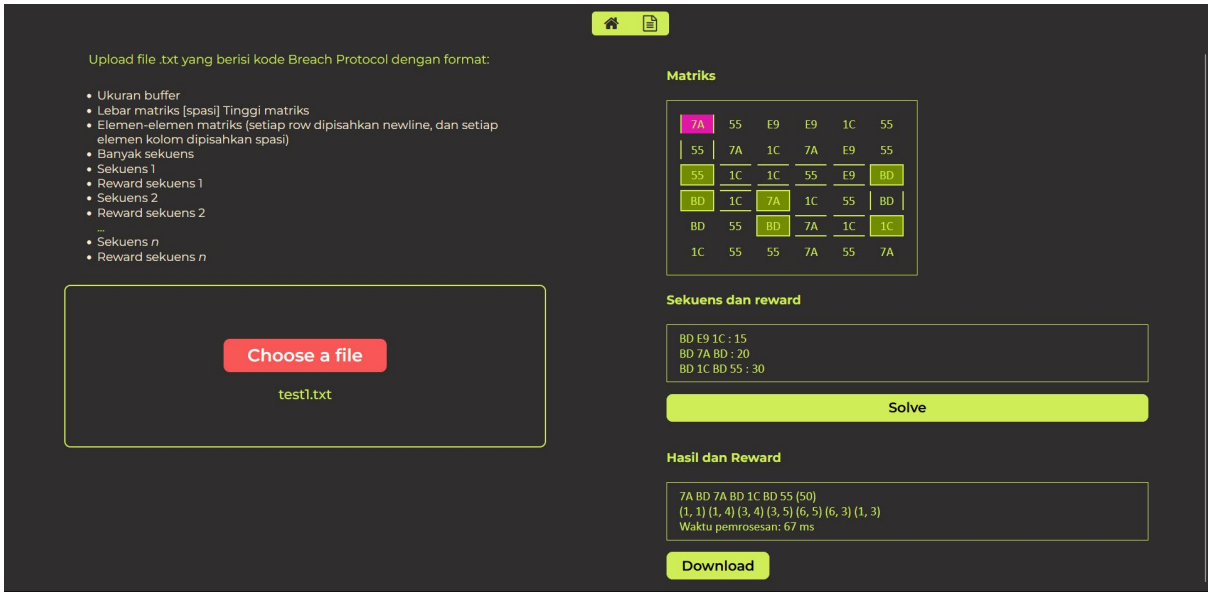
C. Pengujian Program

a. File Upload

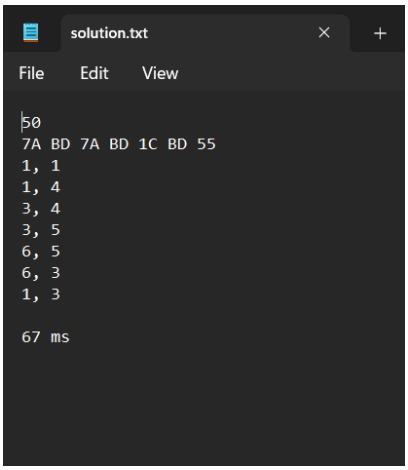
1. Tes 1



Gambar 10. File .txt untuk pengunggahan pada pengujian pertama

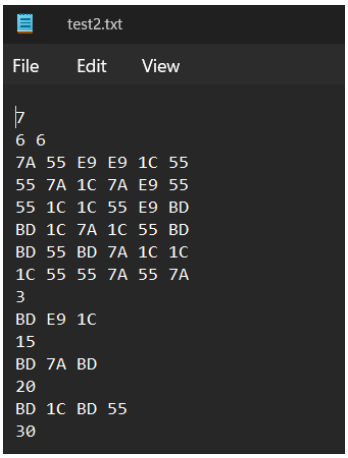


Gambar 11. Tangkapan layar hasil matriks dan solusi dari file yang diunggah.

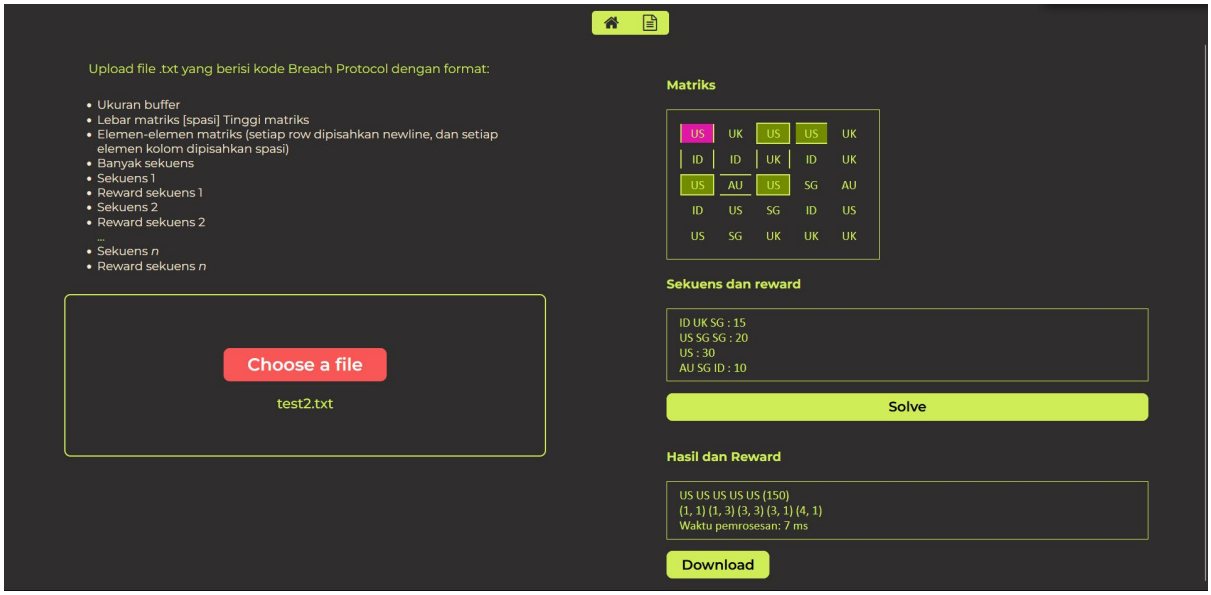


Gambar 12. Solusi dari permainan yang sudah disimpan dalam file .txt

2. Tes 2

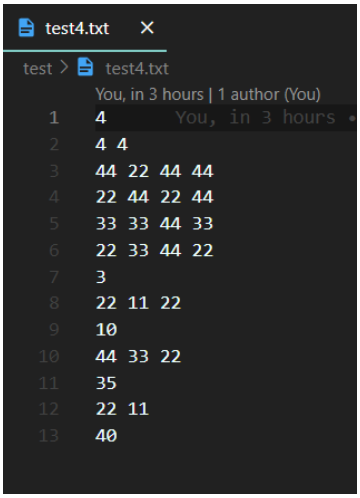


Gambar 13. File .txt untuk pengujian kedua

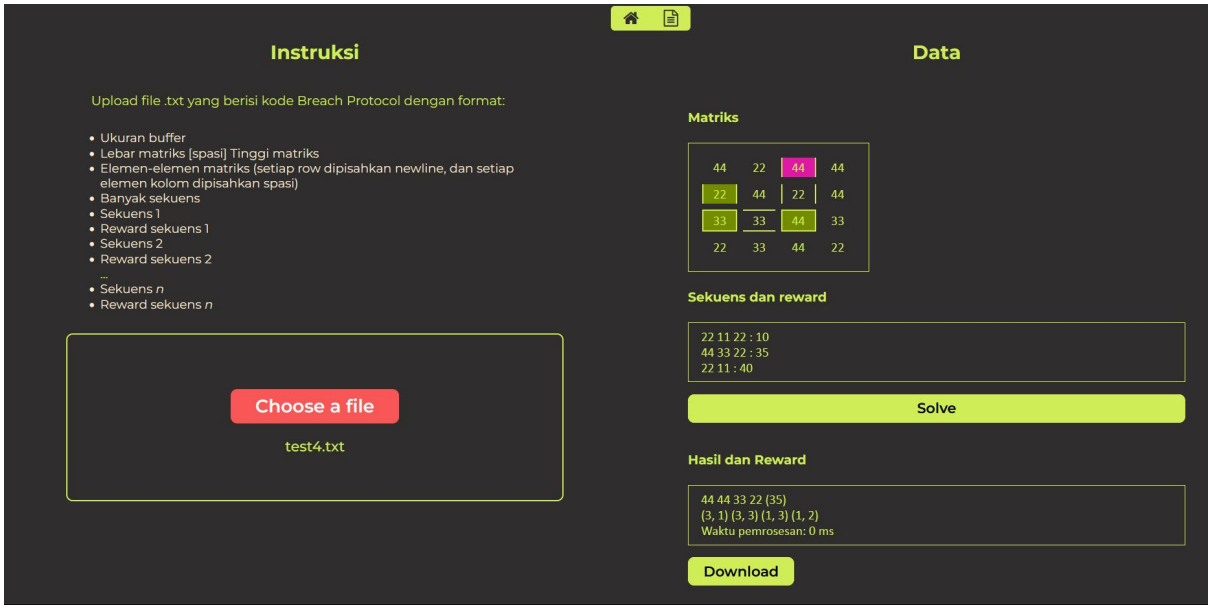


Gambar 14. Tangkapan layar solusi hasil pemrosesan dari file .txt kedua

3. Tes 3



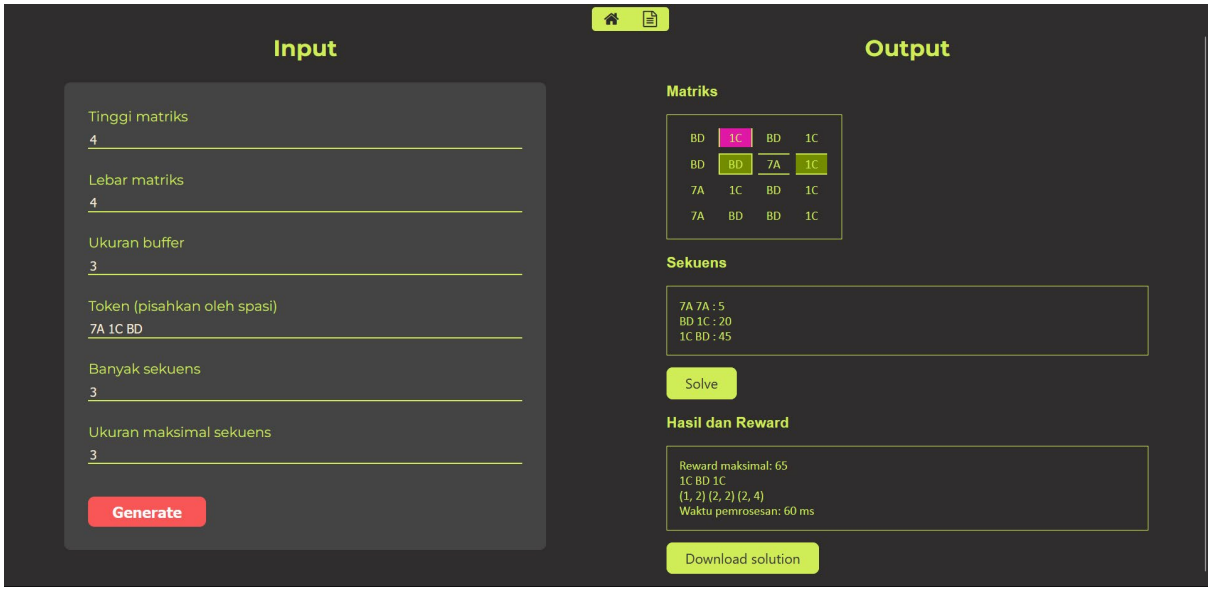
Gambar 15. File .txt untuk pengujian ketiga



Gambar 16. Tangkapan layar hasil solusi dari unggahan file ketiga

b. Input Acak

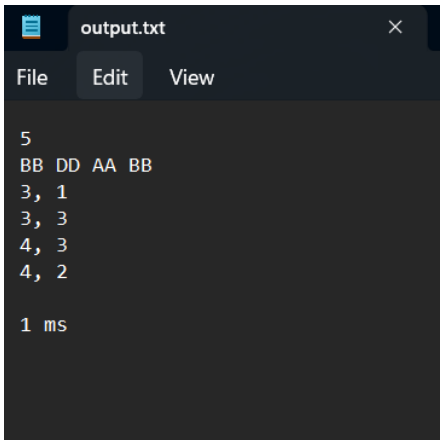
1. Tes 1



Gambar 17. Tangkapan layar pengujian pertama input acak dan matriks solusi

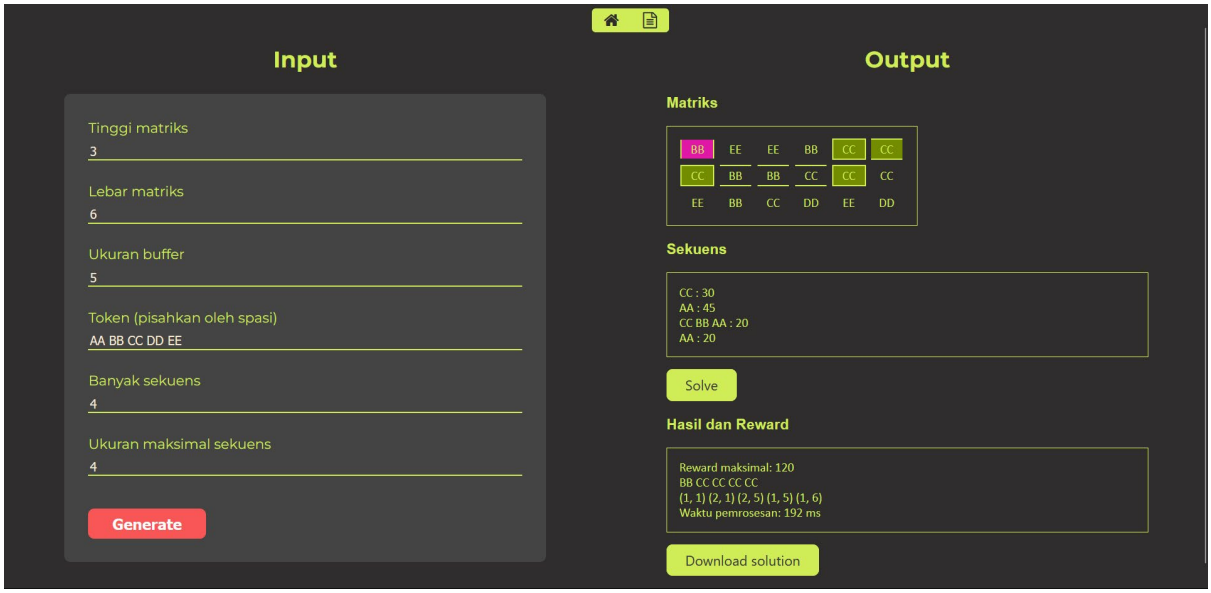
2. Tes 2

Gambar 18. Tangkapan layar solusi permainan dari input acak kedua



Gambar 19. File solusi .txt dari input acak kedua

3. Tes 3



Gambar 20. Hasil pengujian dengan input acak dari pengguna