LAPORAN TUGAS KECIL
IF2211 STRATEGI ALGORITMA

Oleh:

Julian Caleb Simandjuntak - 13522099

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

I. Persoalan Masalah

Persoalan masalah diambil dari spesifikasi tucil.

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.

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

II. Penjelasan Kode Program

Persoalan masalah diselesaikan dengan menggunakan bahasa JavaScript dan memanfaatkan HTML (serta Tailwind) sebagai GUI.

Disclaimer 1: Sebelumnya, saya merasa yakin bahwa ada algoritma yang lebih optimal dan lebih baik dan masih termasuk dalam jenis algoritma brute force, tetapi saya untuk sekarang mengusahakan program bekerja terlebih dahulu.

Disclaimer 2: Tidak semua code saya masukkan di Bab Penjelasan Kode Program. Untuk keseluruhan code dapat dilihat di repository maupun Bab Program Secara Lengkap.

1. Input

   Input dibagi menjadi 2 jenis input, yaitu input dari file txt dan input dari CLI dengan memasukkan jumlah token unik, token, ukuran buffer, ukuran matriks (panjang dan lebar), jumlah sekuens, dan ukuran maksimal sekuens.

   Untuk input dari file, user akan memasukkan nama file dan kemudian akan dicek oleh program, jika ada akan diekstrak ukuran buffer, ukuran matrix, matrix, ukuran sequence, dan sequencesnya. Asumsi ukuran matrix dan ukuran sequence valid untuk memudahkan pembacaan file, akan divalidasi ukuran tiap sequencenya (harus lebih dari sama dengan 2).

```
// Memproses file
async function loadTextFile() {
```

```
document.getElementById("invalidContainer").classList.add("hidden");

    // Mengambil nama file
    let filetxt = document.getElementById("fileInput").value.trim();

    // Fetch file content
    const response = await fetch(filetxt);
    if (!response.ok) {
        alert('File not found or other network error');
    }
    const fileContent = await response.text();

    // Split isi file berdasarkan enter
    const lines = fileContent.trim().split('\n');

    // Instantiasi iterasi baris
    let line_num = 0;

    // Mengambil jumlah buffer
    const bufferSize = parseInt(lines[line_num++]);

    // Mengambil panjang dan lebar matrix
    const [matrixWidth, matrixHeight] = lines[line_num++].split(' ').map(Number);

    // Instantiasi matrix dan mengisi matrix
    const matrix = new Array(matrixHeight);
    for (let i = 0; i < matrixHeight; i++) {
        matrix[i] = new Array(matrixWidth);
    }
    for (let i = 0; i < matrixHeight; i++) {
        const elements = lines[line_num++].trim().split(' ');
        for (let j = 0; j < matrixWidth; j++) {
            matrix[i][j] = elements[j];
        }
    }

    // Mengambil banyak sequence
    const numberOfSequence = parseInt(lines[line_num++]);
    // Instantiasi array sequence dan sequenceReward dan diisi
    const sequences = [];
    for (let i = 0; i < numberOfSequence; i++) {
        const sequence = lines[line_num++].trim().split(' ');
                                            const     sequenceReward     =
parseInt(lines[line_num++].trim());
        sequences.push({ sequence, sequenceReward });
    }

    // Print untuk debug
    console.log("Buffer Size:", bufferSize);
    console.log("Matrix Width:", matrixWidth);
    console.log("Matrix Height:", matrixHeight);
    console.log(matrix);
    console.log(numberOfSequence);
    console.log(sequences);

    // Asumsi panjang lebar matrix dan banyak sequence valid
    // Validasi panjang tiap sequence harus > 2
    if (isSequenceLengthValid(sequences)) {
          processSequence(bufferSize, matrixHeight, matrixWidth,
matrix, tokens, sequences);
    } else {

document.getElementById("invalidContainer").classList.remove("hi
dden");
    }

}
```

Sedangkan untuk input dari user, akan diminta (terdapat GUI) ukuran buffer, jumlah token unik, token, lebar matrix, tinggi matrix, jumlah sequence, dan ukuran maksimal sequence. Asumsi ukuran matrix dan ukuran sequence valid, dilakukan validasi untuk memastikan jumlah token sesuai.

```
// Memproses input
function submitUser() {


document.getElementById("invalidContainer").classList.add("hidde
n");

    // Mengambil data
                                                bufferSize          =
parseInt(document.getElementById("bufferSize").value);
```

```
                                    numberOfTokens          =
parseInt(document.getElementById("numberOfTokens").value);
                                          tokens            =
document.getElementById("tokens").value.trim().split(" ");
                                      matrixWidth           =
parseInt(document.getElementById("matrixWidth").value);
                                     matrixHeight           =
parseInt(document.getElementById("matrixHeight").value);
                                   sequenceAmount           =
parseInt(document.getElementById("sequenceAmount").value);
                          maximumSequenceContent            =
parseInt(document.getElementById("maximumSequenceContent").value
);

    // Generate matrix
    const matrix = new Array(matrixHeight);
    for (let i = 0; i < matrixHeight; i++) {
        matrix[i] = new Array(matrixWidth);
    }
    for (let i = 0; i < matrixHeight; i++) {
        for (let j = 0; j < matrixWidth; j++) {
                     tokenIndex  =  Math.floor(Math.random()   *
numberOfTokens);
            matrix[i][j] = tokens[tokenIndex];
        }
    }

    // Generate sequences dan sequenceReward
    const sequences = [];
    for (let i = 0; i < sequenceAmount; i++) {
        const sequence = []
             const  sequenceContent  =  Math.floor(Math.random()  *
(maximumSequenceContent - 2 + 1)) + 2;;
        for (let j = 0; j < sequenceContent; j++) {
                     tokenIndex  =  Math.floor(Math.random()   *
numberOfTokens);
            sequence.push(tokens[tokenIndex]);
        }
        const sequenceReward = Math.floor(Math.random() * 100)
        sequences.push({ sequence, sequenceReward });
    }

    // Print untuk debug
    console.log("Buffer Size:", bufferSize);
    console.log("Number of Tokens", numberOfTokens);
    console.log("Matrix Width:", matrixWidth);
    console.log("Matrix Height:", matrixHeight);
    console.log("Tokens:", tokens);
    console.log("Matrix: ", matrix);
    console.log("Sequences: ", sequences);

    if (numberOfTokensValid(tokens, numberOfTokensValid)) {
          processSequence(bufferSize, matrixHeight, matrixWidth,
matrix, tokens, sequences);
    } else {

document.getElementById("invalidContainer").classList.remove("hi
dden");
    }

}
```

2. Inisiasi variable

   Sebelum dimulai pengecekan, akan dibuat beberapa variable untuk memenuhi spesifikasi dari program yang meliputi sebuah matrix berukuran sama dengan matrix yang akan diolah tetapi berisi 0 untuk menandai koordinat mana saja yang sudah dicek mengingat masing-masing sel hanya boleh dicek sekali, array berukuran sama dengan banyak sequence dengan tujuan serupa, inisiasi 2 array kosong (global) untuk menyimpan sequence yang sudah diolah dan koordinatnya, serta 2 buah integer (global) untuk menyimpan reward terbaik dan untuk menyimpan berapa lama diproses. Di saat yang sama, ditampilkan semua variable yang sudah diekstrak sebelumnya.

```
// Variable global maximum
let maxSequence = [];
let maxCoords = [];
let maxReward = 0;
let elapsedTime = 0;
```

```
// Saatnya mengproses
function processSequence(bufferSize, matrixHeight, matrixWidth,
matrix, tokens, sequences) {


document.getElementById("invalidContainer").classList.add("hidde
n");
```

```
document.getElementById("debugContainer").classList.remove("hidd
en");

    maxSequence = [];
    maxCoords = [];
    maxReward = 0;
    elapsedTime = 0;

    // Menampilkan input
            document.getElementById("buffer-size").textContent   =
bufferSize;
                document.getElementById("matrix").textContent    =
JSON.stringify(matrix);
             document.getElementById("sequences").textContent   =
JSON.stringify(sequences);

    // Memulai timer
    const startTime = performance.now();

      // Membuat array sequence untuk mengecek apakah sudah di
visit atau belum
    let sequenceVisited = new Array(sequences.length);
    for (let i = 0; i < sequences.length; i++) {
        sequenceVisited[i] = 0;
    }

     // Membuat matrix untuk mengecek apakah sudah di visit atau
belum
    let matrixVisited = [];
    for (let i = 0; i < matrixHeight; i++) {
        const row = []
        for (let j = 0; j < matrixWidth; j++) {
        row.push(0)
        }
        matrixVisited.push(row)
    }

     // Buat array kosong untuk menyimpan sequence yang terpakai
dan koordinatnya
    let coordSequence = [];
    let savedSequence = [];
```

```
            startCheckSequence(matrix,   sequences,   matrixWidth,
matrixHeight,   sequenceVisited,   matrixVisited,   coordSequence,
savedSequence, bufferSize)


. . .
```

3. Algoritma (1)

   Program yang dibuat akan mencari solusi optimal (reward terbesar) dari permasalahan yang terdapat pada Bab Persoalan Masalah. Solusi akan dicari dengan menggunakan algoritma Brute Force, yang berarti mencari semua kemungkinan yang ada tanpa terlalu memikirkan optimasi program.

```
// Fungsi untuk memulai iterasi pengecekan sequence dalam matrix
function   startCheckSequence(matrix,   sequences,   matrixWidth,
matrixHeight,   sequenceVisited,   matrixVisited,   coordSequence,
savedSequence, bufferSize) {
    for (let i = 0; i < sequences.length; i++) {
        // Iterasi sebanyak banyak kolom
        for (let j = 0; j < matrixWidth; j++) {
                      cekSatuKolom(matrix,  j,  0,  matrixVisited,
matrixWidth,   matrixHeight,   sequences,   sequences[i].sequence,
sequenceVisited, 0, coordSequence, savedSequence, bufferSize)
        }
    }
}
```

   Berdasarkan spesifikasi, solusi akan dimulai dari baris pertama matrix, lalu horizontal, vertikal, dan selang-seling selanjutnya. Karena pengecekan secara brute force mencari semua kemungkinan, algoritma program diawali dengan untuk iterasi tiap sequence yang ada, dilakukan iterasi tiap elemen pada baris pertama untuk pengecekan secara kolom.

4. Algoritma (2)

   Setiap kolom pada matriks kemudian akan dicek satu per satu elemennya. Di saat yang sama akan dilakukan tracking pada index keberapa sequence tersebut dicek. Untuk setiap elemen yang sama dengan sequence index kesekian, akan dilakukan pengecekan satu per satu secara baris. Selanjutnya akan dilakukan pemanggilan fungsi secara selang-seling antara kolom dan baris hingga sequence selesai atau tidak ditemukan elemen yang sama.

```
// Fungsi untuk iterasi dalam satu kolom
function    cekSatuKolom(matrix,    col,    index,    matrixVisited,
matrixWidth, matrixHeight, sequences, sequence, sequenceVisited,
currReward, coordSequence, savedSequence, bufferSize) {
     // Cek apakah sequence sudah berakhir belum dan cek apakah
buffer sudah penuh
```

```
        if ((!check(index, sequence)) && (savedSequence.length <
bufferSize)) {
          // Looping sebanyak tinggi matrix / banyak baris
          for (let i = 0; i < matrixHeight; i++) {
               // Jika sama dengan yang dicek dengan sequence,
                         if ((matrix[i][col] == sequence[index]) &&
(matrixVisited[i][col] != 1)) {
                         // Simpan koordinatnya, tokennya, dan tandai
kalau sudah dicek
                        // Kalau sequence belum terisi, nambahin dulu
koordinat di baris pertama
                      console.log("Baris", i+1, "kolom", col+1, "=",
matrix[i][col]);

                    let coordSequence_copy = [...coordSequence];
                    let savedSequence_copy = [...savedSequence];
                                           let matrixVisited_copy =
JSON.parse(JSON.stringify(matrixVisited));

                           // Kalau sequence kosong dan awal sequence
ditemukan di row pertama
                    if (i == 0 && savedSequence.length == 0) {
                             coordSequence_copy.push([col+1, i+1]); //
Ingat (col, row)
                         savedSequence_copy.push(matrix[i][col]);
                         matrixVisited_copy[i][col] = 1;

                         // Cek kolom lagi
                                     cekSatuKolom(matrix, col, index+1,
matrixVisited_copy, matrixWidth, matrixHeight, sequences,
sequence, sequenceVisited, currReward, coordSequence_copy,
savedSequence_copy, bufferSize);

                     // Kalau sequence kosong dan awal sequence tidak
ditemukan di row pertama
                      } else if (i != 0 && savedSequence.length == 0)
{
                         // Tambahin yang di row pertama dulu
                               coordSequence_copy.push([col+1, 1]); //
Ingat (col, row)
                         savedSequence_copy.push(matrix[0][col]);
                         matrixVisited_copy[0][col] = 1;
                        coordSequence_copy.push([col+1, i+1]); //
Ingat (col, row)
                    savedSequence_copy.push(matrix[i][col]);
                    matrixVisited_copy[i][col] = 1;

                    // Saatnya ngecek baris
                                cekSatuBaris(matrix, i, index+1,
matrixVisited_copy, matrixWidth, matrixHeight, sequences,
sequence, sequenceVisited, currReward, coordSequence_copy,
savedSequence_copy, bufferSize);

                // Kalau sequence tidak kosong
                } else {
                        coordSequence_copy.push([col+1, i+1]); //
Ingat (col, row)
                    savedSequence_copy.push(matrix[i][col]);
                    matrixVisited_copy[i][col] = 1;

                    // Saatnya ngecek baris
                                cekSatuBaris(matrix, i, index+1,
matrixVisited_copy, matrixWidth, matrixHeight, sequences,
sequence, sequenceVisited, currReward, coordSequence_copy,
savedSequence_copy, bufferSize);
                }
            }
        }
    } else {
        console.log()
        console.log("-- SELESAI! -- ");
        currReward = countReward(savedSequence, sequences);
        console.log("Current reward:", currReward);
        console.log("All coords:", coordSequence);
        console.log("All tokens saved:", savedSequence);
        console.log()

        // Mencatat bahwa sequence ini sudah selesai
               let currSequenceIndex = sequenceIndex(sequence,
sequences);
        let sequenceVisited_copy = [...sequenceVisited];
        sequenceVisited_copy[currSequenceIndex] = 1;

        // Lanjut sequence lain
```

```
                    continueCheckSequence(matrix, matrixVisited,
matrixWidth, matrixHeight, sequences, sequenceVisited_copy,
currReward, coordSequence, savedSequence, "kolom", col,
bufferSize);
    }
}

// Fungsi untuk iterasi dalam satu baris
function cekSatuBaris(matrix, row, index, matrixVisited,
matrixWidth, matrixHeight, sequences, sequence, sequenceVisited,
currReward, coordSequence, savedSequence, bufferSize) {
    // Cek apakah sequence sudah berakhir belum dan cek apakah
buffer sudah penuh
    if ((!check(index, sequence)) && (savedSequence.length <
bufferSize)) {
        // Looping sebanyak panjang matrix / banyak kolom
        for (let i = 0; i < matrixWidth; i++) {
            // Jika sama dengan yang dicek dengan sequence,
                    if ((matrix[row][i] == sequence[index]) &&
(matrixVisited[row][i] != 1)) {
                    // Simpan koordinatnya, tokennya, dan tandai
kalau sudah dicek
                  console.log("Baris", row+1, "kolom", i+1, "=",
matrix[row][i]);

                let coordSequence_copy = [...coordSequence];
                  coordSequence_copy.push([i+1, row+1]); // Ingat
(col, row)

                let savedSequence_copy = [...savedSequence];
                savedSequence_copy.push(matrix[row][i]);

                                    let matrixVisited_copy =
JSON.parse(JSON.stringify(matrixVisited));
                matrixVisited_copy[row][i] = 1;

                // Saatnya ngecek kolom
                                cekSatuKolom(matrix, i, index+1,
matrixVisited_copy, matrixWidth, matrixHeight, sequences,
sequence, sequenceVisited, currReward, coordSequence_copy,
savedSequence_copy, bufferSize)
            }
        }
    } else {
        console.log()
        console.log("-- SELESAI! -- ");
        currReward = countReward(savedSequence, sequences);
        console.log("Current reward:", currReward);
        console.log("All coords:", coordSequence);
        console.log("All tokens saved:", savedSequence);
        console.log()

        // Mencatat bahwa sequence ini sudah selesai
               let  currSequenceIndex  =  sequenceIndex(sequence,
sequences);
        let sequenceVisited_copy = [...sequenceVisited];
        sequenceVisited_copy[currSequenceIndex] = 1;

        // Lanjut sequence lain
                   continueCheckSequence(matrix,  matrixVisited,
matrixWidth,   matrixHeight,   sequences,   sequenceVisited_copy,
currReward,   coordSequence,   savedSequence,   "baris",   row,
bufferSize);
    }
}
```

Khusus untuk kolom, akan dicek untuk 3 kasus, yaitu jika elemen pertama solusi yang ditemukan dalam sequence ada pada baris pertama (karena kalau ada pada baris pertama maka dilakukan searching secara kolom), tidak ada pada baris pertama, dan pengecekan bukan untuk elemen pertama solusi.
Setelah menyelesaikan sebuah sequence, akan dihitung rewardnya (mengecek sequence apa saja yang terdapat dalam solusi) kemudian akan diupdate jika reward lebih besar atau reward sama tetapi panjang solusi lebih pendek.

```
function isMore (sequence, coords, reward) {
      //  Prosedur  untuk  mengecek  apakah  sequence  yang  dimiliki
sekarang merupakan seq
    // dengan reward terbanyak atau tidak
       if  ((reward  >  maxReward)  ||  ((reward  ==  maxReward)  &&
(sequence.length < maxSequence.length))) {
        maxReward = reward;
        maxCoords = coords;
        maxSequence = [...sequence];
    }
    console.log("Reward terbesar:", maxReward);
    console.log("Koordinat: ", maxCoords);
    console.log("Sequence terbaik:", maxSequence);
}
```

5. Algoritma (3)
   Jika satu sequence telah selesai, akan dilakukan pengecekan untuk sequence lainnya. Terdapat 2 kemungkinan, antara sequence yang dicek sudah ada pada sequence yang sudah menjadi solusi (subset) atau belum. Jika merupakan subset, akan dihitung kembali reward dan dicek maksimumnya. Jika bukan, akan dilanjutkan untuk baris atau kolom.

```
// Fungsi untuk melanjutkan cek ke sequence lain
function continueCheckSequence (matrix, matrixVisited,
matrixWidth, matrixHeight, sequences, sequenceVisited,
currReward, coordSequence, savedSequence, cond, startIndex,
bufferSize) {
    for (let i = 0; i < sequences.length; i++) {
        // Untuk sequence yang belum pernah digunakan
        if (sequenceVisited[i] == 0) {
            // Cek apakah sequence tersebut ada pada sequence
yang sudah ada
                                if (isSubstring(savedSequence,
sequences[i].sequence)) {
                console.log()
                console.log("-- SELESAI! -- ");
                        currReward = countReward(savedSequence,
sequences);
                console.log("Current reward:", currReward);
                console.log("All coords:", coordSequence);
                console.log("All tokens saved:", savedSequence);
                console.log()

                // Cek maks
                            isMore(savedSequence, coordSequence,
currReward);

                // Mencatat bahwa sequence ini sudah selesai
                let currSequenceIndex = i;
                let sequenceVisited_copy = [...sequenceVisited];
                sequenceVisited_copy[currSequenceIndex] = 1;

                // Lanjut sequence lain
                     continueCheckSequence(matrix, matrixVisited,
matrixWidth, matrixHeight, sequences, sequenceVisited_copy,
currReward, coordSequence, savedSequence, cond, startIndex,
bufferSize);
```

```
              } else { // Cek apakah sequence tersebut merupakan
subset

                // Cek maks dulu
                            isMore(savedSequence, coordSequence,
currReward);

                // Cari di mana (kalau tidak subset return 0)
                if (savedSequence.length < bufferSize) {
                                    let seqStartIndex =
checkStartIndex(savedSequence, sequences[i].sequence);
                    if (cond === "baris") {
                                cekSatuBaris(matrix, startIndex,
seqStartIndex, matrixVisited, matrixWidth, matrixHeight,
sequences, sequences[i].sequence, sequenceVisited, currReward,
coordSequence, savedSequence, bufferSize);
                    } else {
                                cekSatuKolom(matrix, startIndex,
seqStartIndex, matrixVisited, matrixWidth, matrixHeight,
sequences, sequences[i].sequence, sequenceVisited, currReward,
coordSequence, savedSequence, bufferSize);
                    }
                }
            }
        }
    }
}
```

6. Menampilkan hasil
   Setelah semua kemungkinan diproses, akan didapatkan sebuah solusi dengan reward terbaik dan dengan kemungkinan sequence terpendek, yang berada di variable global. Timer akan dihentikan dan dicatat, kemudian akan ditampilkan.

```
. . .

     // Menghentikan timer dan menghitung lama waktu yang
dibutuhkan
    const endTime = performance.now();
    elapsedTime = endTime - startTime;
       console.log("Lama waktu eksekusi: " + elapsedTime + "
milliseconds");

    displayResult();
}
```

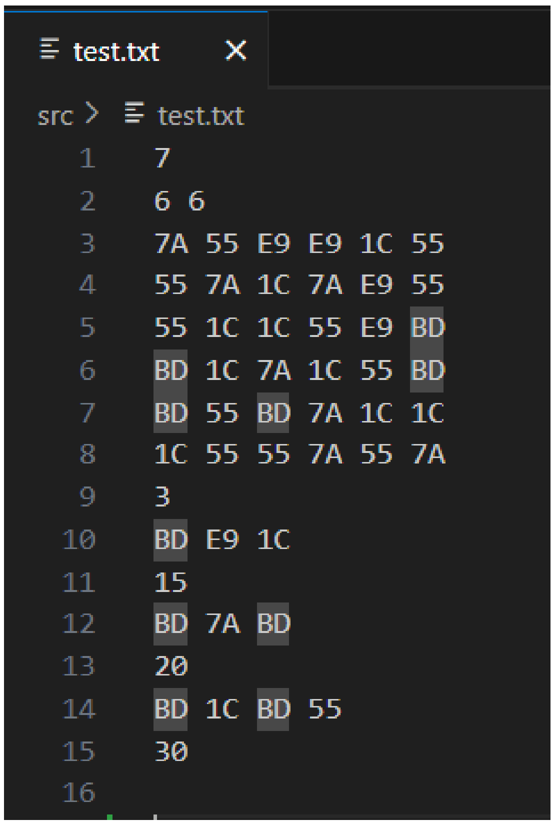
```

```

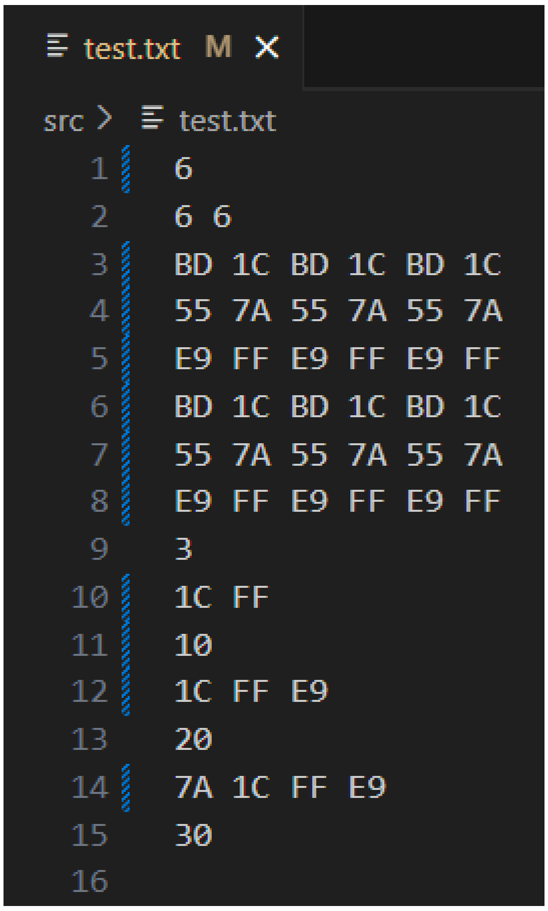
```
// Menampilkan hasil
function displayResult() {

            document.getElementById("max-reward").textContent   =
maxReward;

    var formattedSequence = maxSequence.join(' ');
          document.getElementById("max-sequence").textContent   =
formattedSequence;

    var formattedCoords = maxCoords.map(function(coord) {
        return coord.join(', ');
    }).join('\n');
            document.getElementById("max-coords").textContent   =
formattedCoords;

          document.getElementById("elapsed-time").textContent   =
elapsedTime + " ms";
}
```

7. Menyimpan
   Terakhir, akan diberikan kesempatan untuk menyimpan ke dalam file (namun dalam kasus ini akan di download filenya, dikarenakan web development tidak memungkinkan save ke local oleh karena masalah keamanan). User dapat memasukkan nama file, dan text yang disimpan sama dengan text yang ditampilkan (beda dengan spesifikasi, dapat dilihat di bab selanjutnya).

```
// Save ke bin/file.txt
function saveContent() {
                               var        debugContainer        =
document.getElementById('container-answer');
    var debugContent = debugContainer.textContent;
    let debugLines = debugContent.split('\n');

             let    debugFormatted    =    debugLines.map(line    =>
line.trim()).join('\n');
     let blob = new Blob([debugFormatted], { type: 'text/plain'
});

    console.log(debugFormatted);

    let fileName = prompt("Enter the filename:", "save.txt");
    if (fileName === null) return;

    let a = document.createElement('a');
    a.href = URL.createObjectURL(blob);
    a.download = fileName;
    a.click();
}
```

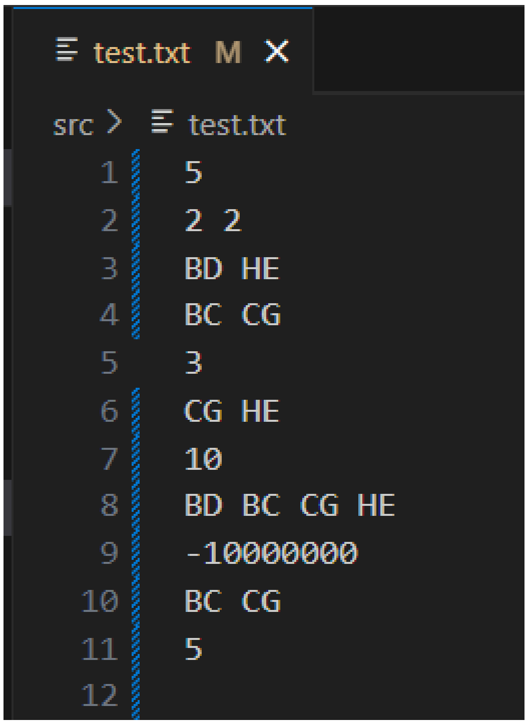
III. Tangkapan Layar Input dan Output

Input	Output
 test.txt src > test.txt 7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	HASIL: 50 7A BD 7A BD 1C BD 55 1, 1 1, 4 3, 4 3, 5 6, 5 6, 3 1, 3 7.199999999254942 ms



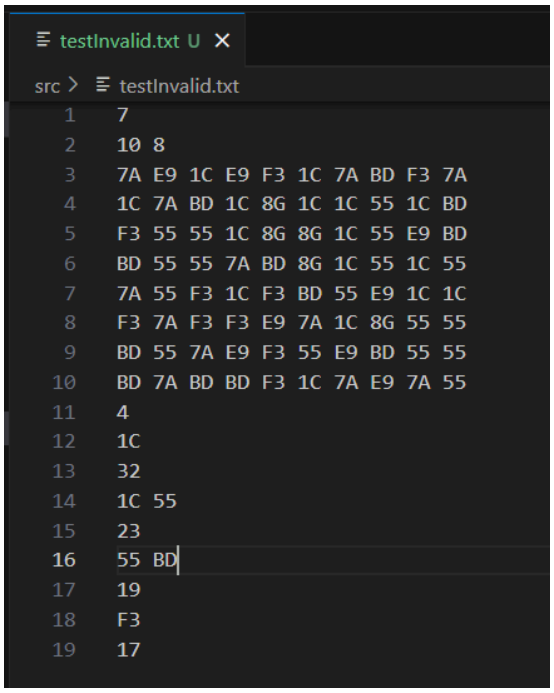
```
6
6 6
BD 1C BD 1C BD 1C
55 7A 55 7A 55 7A
E9 FF E9 FF E9 FF
BD 1C BD 1C BD 1C
55 7A 55 7A 55 7A
E9 FF E9 FF E9 FF
3
1C FF
10
1C FF E9
20
7A 1C FF E9
30

```

```
HASIL:
30
1C FF E9
2, 1
2, 3
1, 3

12.600000001490116 ms
```



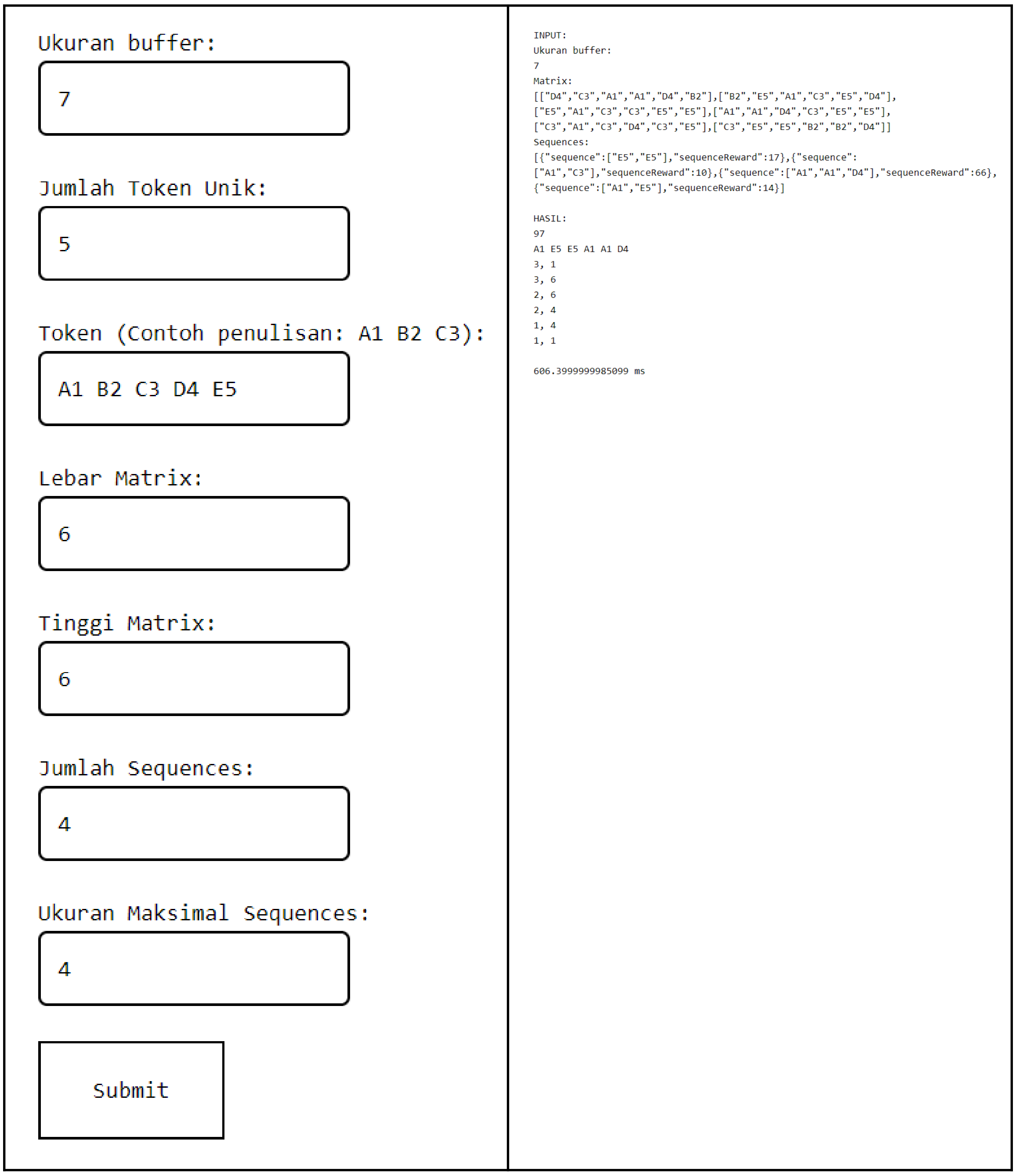
```
5
2 2
BD HE
BC CG
3
CG HE
10
BD BC CG HE
-10000000
BC CG
5

```

```
HASIL:
5
BD BC CG
1, 1
1, 2
2, 2

0.6999999992549419 ms
```



```
7
10 8
7A E9 1C E9 F3 1C 7A BD F3 7A
1C 7A BD 1C 8G 1C 1C 55 1C BD
F3 55 55 1C 8G 8G 1C 55 E9 BD
BD 55 55 7A BD 8G 1C 55 1C 55
7A 55 F3 1C F3 BD 55 E9 1C 1C
F3 7A F3 F3 E9 7A 1C 8G 55 55
BD 55 7A E9 F3 55 E9 BD 55 55
BD 7A BD BD F3 1C 7A E9 7A 55
4
1C
32
1C 55
23
55 BD
19
F3
17
```

```
Input invalid!
```



Ukuran buffer:

7

Jumlah Token Unik:

5

Token (Contoh penulisan: A1 B2 C3):

A1 B2 C3 D4 E5

Lebar Matrix:

6

Tinggi Matrix:

6

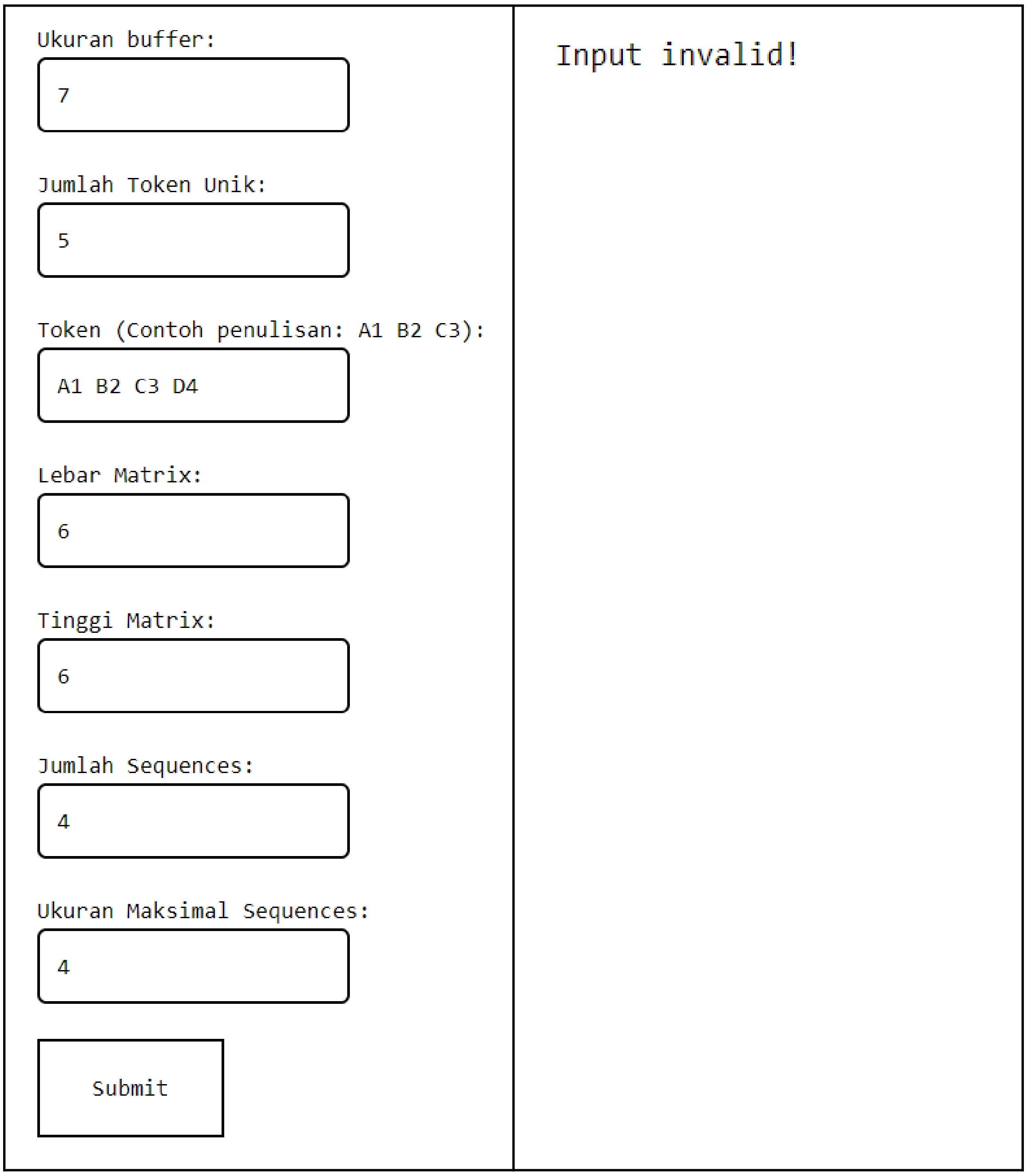
Jumlah Sequences:

4

Ukuran Maksimal Sequences:

4

```
INPUT:
Ukuran buffer:
7
Matrix:
[["D4","C3","A1","A1","D4","B2"],["B2","E5","A1","C3","E5","D4"],
["E5","A1","C3","C3","E5","E5"],["A1","A1","D4","C3","E5","E5"],
["C3","A1","C3","D4","C3","E5"],["C3","E5","E5","B2","B2","D4"]]
Sequences:
[{"sequence":["E5","E5"],"sequenceReward":17},{"sequence":
["A1","C3"],"sequenceReward":10},{"sequence":["A1","A1","D4"],"sequenceReward":66},
{"sequence":["A1","E5"],"sequenceReward":14}]

HASIL:
97
A1 E5 E5 A1 A1 D4
3, 1
3, 6
2, 6
2, 4
1, 4
1, 1

606.3999999985099 ms
```


Ukuran buffer:

7

Jumlah Token Unik:

5

Token (Contoh penulisan: A1 B2 C3):

A1 B2 C3 D4

Lebar Matrix:

6

Tinggi Matrix:

6

Jumlah Sequences:

4

Ukuran Maksimal Sequences:

4

```
Input invalid!
```

IV. Link Repository

Link Github: https://github.com/Julian-Caleb/Tucil1_13522099

V. Tabel Checklist

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	V	
Program berhasil dijalankan	V	
Program dapat membaca masukan berkas .txt	V	

Program dapat menghasilkan masukan secara acak	V	
Solusi yang diberikan program optimal	V	
Program dapat menyimpan solusi dalam berkas .txt	V	
Program memiliki GUI	V	

VI. Program Secara Lengkap

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
         <meta name="viewport" content="width=device-width,
initial-scale=1.0">
                                                              <link
href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css"
rel="stylesheet">
    <title>Document</title>
</head>
<body class="w-full h-min-screen flex flex-col items-center
justify-center font-mono">
    <!-- Pembukaan -->
    <h1>Selamat datang di Punk Dunia Maya!</h1>
    <p>Mau input dari mana?</p>
    <br />

    <!-- Minta input dong -->
     <div class="w-1/5 flex flex-col lg:flex-row gap-4 items-start
justify-between">
         <button class="w-32 h-14 p-5 border-2 border-black"
onclick="inputTextFile()">Text File</button>
         <button class="w-32 h-14 p-5 border-2 border-black"
onclick="inputUser()">User</button>
    </div>
    <br />

    <!-- Kalau minta input file -->
          <div class="w-auto h-full flex flex-col items-start
justify-between hidden" id="textFileInput">
        <label for="fileInput">Masukkan nama file teks:</label>
           <input class="w-80 p-3 rounded-md border-2 border-black"
type="text" id="fileInput">
```

```
        <br />
              <button  class="w-32  h-14  p-5  border-2  border-black"
onclick="loadTextFile()">Submit</button>
    </div>
    <br />

    <!-- Kalau input random -->
          <div   class="w-auto   h-full   flex   flex-col   items-start
justify-between hidden" id="userInput">
        <label for="bufferSize">Ukuran buffer:</label>
           <input  class="w-80  p-3  rounded-md  border-2  border-black"
type="number" id="bufferSize" min="1" required>
        <br />

        <label for="numberOfTokens">Jumlah Token Unik:</label>
           <input  class="w-80  p-3  rounded-md  border-2  border-black"
type="number" id="numberOfTokens" min="1" required>
        <br />

               <label  for="tokens">Token  (Contoh  penulisan:  A1  B2
C3):</label>
           <input  class="w-80  p-3  rounded-md  border-2  border-black"
type="text" id="tokens" pattern=".{2}\s.{2}" required>
        <br />

        <label for="matrixWidth">Lebar Matrix:</label>
           <input  class="w-80  p-3  rounded-md  border-2  border-black"
type="number" id="matrixWidth" min="1" required>
        <br />

        <label for="matrixHeight">Tinggi Matrix:</label>
           <input  class="w-80  p-3  rounded-md  border-2  border-black"
type="number" id="matrixHeight" min="1" required>
        <br />

        <label for="sequenceAmount">Jumlah Sequences:</label>
           <input  class="w-80  p-3  rounded-md  border-2  border-black"
type="number" id="sequenceAmount" min="1" required>
        <br />

                <label  for="maximumSequenceContent">Ukuran  Maksimal
Sequences:</label>
```

```
          <input class="w-80 p-3 rounded-md border-2 border-black"
type="number" id="maximumSequenceContent" min="1" required>
        <br />

             <button class="w-32 h-14 p-5 border-2 border-black"
onclick="submitUser()">Submit</button>
    </div>
    <br />

    <!-- Kalau input invalid -->
    <div class="hidden" id="invalidContainer">
        <p>Input invalid!</p>
    </div>

    <!-- Container input dan hasil -->
    <div class="hidden px-5" id="debugContainer">
        <p>Pemrosesan input berhasil</p>
        <br />
        <div id="container">
            <div id="conainer-input">
                <h2>INPUT:</h2>
                <div>
                    <h3>Ukuran buffer:</h3>
                    <p id="buffer-size"></p>
                </div>
                <div>
                    <h3>Matrix:</h3>
                    <div id="matrix"></div>
                </div>
                <div>
                    <h3>Sequences:</h3>
                    <p id="sequences"></p>
                </div>
            </div>
            <br />
            <div id="container-answer">
                <h2>HASIL:</h2>
                <div>
                    <!-- <h3>Reward maksimal:</h3> -->
                    <p id="max-reward"></p>
                </div>
                <div>
```

```
                    <!-- <h3>Sequence dengan reward maksimal:</h3>
-->
                    <pre id="max-sequence"></pre>
                </div>
                <div>
                    <!-- <h3>Koordinat sequence (col, row):</h3> -->
                    <pre id="max-coords"></pre>
                </div>
                <br />
                <div>
                    <!-- <h3>Waktu yang dibutuhkan:</h3> -->
                    <p id="elapsed-time"></p>
                </div>
            </div>
        </div>
        <br />
            <button class="w-32 h-14 p-5 border-2 border-black"
id="saveDebugBtn" onclick="saveContent()">Save Debug
Content</button>
    </div>


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

script.js

```
// Variable global maximum
let maxSequence = [];
let maxCoords = [];
let maxReward = 0;
let elapsedTime = 0;

// Kalau input dari file
function inputTextFile() {
    // Menampilkan opsi input dari file

document.getElementById("textFileInput").classList.remove("hidden");
    document.getElementById("userInput").classList.add("hidden");

document.getElementById("debugContainer").classList.add("hidden");
}

```

```
// Kalau input dari user
function inputUser() {
    // Menampilkan opsi input dari user
    document.getElementById("userInput").classList.remove("hidden");

document.getElementById("textFileInput").classList.add("hidden");

document.getElementById("debugContainer").classList.add("hidden");
}

// Memproses file
async function loadTextFile() {


document.getElementById("invalidContainer").classList.add("hidden");

    // Mengambil nama file
    let filetxt = document.getElementById("fileInput").value.trim();

    // Fetch file content
    const response = await fetch(filetxt);
    if (!response.ok) {
        alert('File not found or other network error');
    }
    const fileContent = await response.text();

    // Split isi file berdasarkan enter
    const lines = fileContent.trim().split('\n');

    // Instantiasi iterasi baris
    let line_num = 0;

    // Mengambil jumlah buffer
    const bufferSize = parseInt(lines[line_num++]);

    // Mengambil panjang dan lebar matrix
      const [matrixWidth, matrixHeight] = lines[line_num++].split('
').map(Number);

    // Instantiasi matrix dan mengisi matrix
    const matrix = new Array(matrixHeight);
    for (let i = 0; i < matrixHeight; i++) {
        matrix[i] = new Array(matrixWidth);
    }
    for (let i = 0; i < matrixHeight; i++) {
        const elements = lines[line_num++].trim().split(' ');
        for (let j = 0; j < matrixWidth; j++) {
            matrix[i][j] = elements[j];
        }
    }

    // Mengambil banyak sequence
    const numberOfSequence = parseInt(lines[line_num++]);

    // Instantiasi array sequence dan sequenceReward dan diisi
    const sequences = [];
    for (let i = 0; i < numberOfSequence; i++) {
        const sequence = lines[line_num++].trim().split(' ');
        const sequenceReward = parseInt(lines[line_num++].trim());
        sequences.push({ sequence, sequenceReward });
    }

    // Print untuk debug
    console.log("Buffer Size:", bufferSize);
    console.log("Matrix Width:", matrixWidth);
    console.log("Matrix Height:", matrixHeight);
    console.log(matrix);
    console.log(numberOfSequence);
    console.log(sequences);

    // Asumsi panjang lebar matrix dan banyak sequence valid
    // Validasi panjang tiap sequence harus > 2
    if (isSequenceLengthValid(sequences)) {
            processSequence(bufferSize,  matrixHeight,  matrixWidth,
matrix, tokens, sequences);
    } else {

document.getElementById("invalidContainer").classList.remove("hidden
");
    }

}

// Fungsi untuk validasi panjang tiap sequence
function isSequenceLengthValid(sequences) {
    // Validasi panjang tiap sequence
    for (let i = 0; i < sequences.length; i++) {
        if (sequences[i].sequence.length < 2) {
            return false;
        }
    }
    return true;
}

// Memproses input
function submitUser() {


document.getElementById("invalidContainer").classList.add("hidden");

    // Mengambil data
                                              bufferSize           =
parseInt(document.getElementById("bufferSize").value);
                                           numberOfTokens          =
parseInt(document.getElementById("numberOfTokens").value);
     tokens = document.getElementById("tokens").value.trim().split("
");
                                             matrixWidth           =
parseInt(document.getElementById("matrixWidth").value);
                                            matrixHeight           =
parseInt(document.getElementById("matrixHeight").value);
                                           sequenceAmount          =
parseInt(document.getElementById("sequenceAmount").value);
                                     maximumSequenceContent        =
parseInt(document.getElementById("maximumSequenceContent").value);

    // Generate matrix
    const matrix = new Array(matrixHeight);
    for (let i = 0; i < matrixHeight; i++) {
        matrix[i] = new Array(matrixWidth);
    }
    for (let i = 0; i < matrixHeight; i++) {
        for (let j = 0; j < matrixWidth; j++) {
            tokenIndex = Math.floor(Math.random() * numberOfTokens);
            matrix[i][j] = tokens[tokenIndex];
        }
    }

    // Generate sequences dan sequenceReward
    const sequences = [];
    for (let i = 0; i < sequenceAmount; i++) {
        const sequence = []
               const  sequenceContent  =  Math.floor(Math.random()  *
(maximumSequenceContent - 2 + 1)) + 2;;
        for (let j = 0; j < sequenceContent; j++) {
            tokenIndex = Math.floor(Math.random() * numberOfTokens);
            sequence.push(tokens[tokenIndex]);
        }
        const sequenceReward = Math.floor(Math.random() * 100)
        sequences.push({ sequence, sequenceReward });
    }

    // Print untuk debug
    console.log("Buffer Size:", bufferSize);
    console.log("Number of Tokens", numberOfTokens);
    console.log("Matrix Width:", matrixWidth);
    console.log("Matrix Height:", matrixHeight);
    console.log("Tokens:", tokens);
    console.log("Matrix: ", matrix);
    console.log("Sequences: ", sequences);

    if (numberOfTokensValid(tokens, numberOfTokensValid)) {
             processSequence(bufferSize,  matrixHeight,  matrixWidth,
matrix, tokens, sequences);
    } else {

document.getElementById("invalidContainer").classList.remove("hidden
");
    }


}

// Fungsi untuk mengecek apakah jumlah token sesuai
function numberOfTokensValid(tokens, numberOfTokens) {
    return tokens.length == numberOfTokens;
}



// Saatnya mengproses
function   processSequence(bufferSize,   matrixHeight,   matrixWidth,
matrix, tokens, sequences) {
```

```
document.getElementById("invalidContainer").classList.add("hidden");

document.getElementById("debugContainer").classList.remove("hidden")
;

    maxSequence = [];
    maxCoords = [];
    maxReward = 0;
    elapsedTime = 0;

    // Menampilkan input
    document.getElementById("buffer-size").textContent = bufferSize;
                  document.getElementById("matrix").textContent     =
JSON.stringify(matrix);
               document.getElementById("sequences").textContent     =
JSON.stringify(sequences);

    // Memulai timer
    const startTime = performance.now();

     // Membuat array sequence untuk mengecek apakah sudah di visit
atau belum
    let sequenceVisited = new Array(sequences.length);
    for (let i = 0; i < sequences.length; i++) {
        sequenceVisited[i] = 0;
    }

      // Membuat matrix untuk mengecek apakah sudah di visit atau
belum
    let matrixVisited = [];
    for (let i = 0; i < matrixHeight; i++) {
        const row = []
        for (let j = 0; j < matrixWidth; j++) {
        row.push(0)
        }
        matrixVisited.push(row)
    }

     // Buat array kosong untuk menyimpan sequence yang terpakai dan
koordinatnya
    let coordSequence = [];
    let savedSequence = [];
    startCheckSequence(matrix, sequences, matrixWidth, matrixHeight,
sequenceVisited, matrixVisited, coordSequence, savedSequence,
bufferSize)

    // Menghentikan timer dan menghitung lama waktu yang dibutuhkan
    const endTime = performance.now();
    elapsedTime = endTime - startTime;
        console.log("Lama waktu eksekusi: " + elapsedTime + "
milliseconds");

    displayResult();
}

// Fungsi untuk memulai iterasi pengecekan sequence dalam matrix
function startCheckSequence(matrix, sequences, matrixWidth,
matrixHeight, sequenceVisited, matrixVisited, coordSequence,
savedSequence, bufferSize) {
    for (let i = 0; i < sequences.length; i++) {
        // Iterasi sebanyak banyak kolom
        for (let j = 0; j < matrixWidth; j++) {
              cekSatuKolom(matrix, j, 0, matrixVisited, matrixWidth,
matrixHeight, sequences, sequences[i].sequence, sequenceVisited, 0,
coordSequence, savedSequence, bufferSize)
        }
    }
}

// Fungsi untuk melanjutkan cek ke sequence lain
function continueCheckSequence (matrix, matrixVisited, matrixWidth,
matrixHeight, sequences, sequenceVisited, currReward, coordSequence,
savedSequence, cond, startIndex, bufferSize) {
    for (let i = 0; i < sequences.length; i++) {
        // Untuk sequence yang belum pernah digunakan
        if (sequenceVisited[i] == 0) {
             // Cek apakah sequence tersebut ada pada sequence yang
sudah ada
            if (isSubstring(savedSequence, sequences[i].sequence)) {
                console.log()
                console.log("-- SELESAI! -- ");
                currReward = countReward(savedSequence, sequences);
                console.log("Current reward:", currReward);
                console.log("All coords:", coordSequence);
                console.log("All tokens saved:", savedSequence);
                console.log()

                // Cek maks
                isMore(savedSequence, coordSequence, currReward);

                // Mencatat bahwa sequence ini sudah selesai
                let currSequenceIndex = i;
                let sequenceVisited_copy = [...sequenceVisited];
                sequenceVisited_copy[currSequenceIndex] = 1;

                // Lanjut sequence lain
                        continueCheckSequence(matrix, matrixVisited,
matrixWidth, matrixHeight, sequences, sequenceVisited_copy,
currReward, coordSequence, savedSequence, cond, startIndex,
bufferSize);
                } else { // Cek apakah sequence tersebut merupakan
subset

                // Cek maks dulu
                isMore(savedSequence, coordSequence, currReward);

                // Cari di mana (kalau tidak subset return 0)
                if (savedSequence.length < bufferSize) {
                                                let seqStartIndex =
checkStartIndex(savedSequence, sequences[i].sequence);
                    if (cond === "baris") {
                                    cekSatuBaris(matrix, startIndex,
seqStartIndex, matrixVisited, matrixWidth, matrixHeight, sequences,
sequences[i].sequence, sequenceVisited, currReward, coordSequence,
savedSequence, bufferSize);
                    } else {
                                    cekSatuKolom(matrix, startIndex,
seqStartIndex, matrixVisited, matrixWidth, matrixHeight, sequences,
sequences[i].sequence, sequenceVisited, currReward, coordSequence,
savedSequence, bufferSize);
                    }
                }
            }
        }
    }
}
```

```
// Fungsi untuk iterasi dalam satu kolom
function cekSatuKolom(matrix, col, index, matrixVisited,
matrixWidth, matrixHeight, sequences, sequence, sequenceVisited,
currReward, coordSequence, savedSequence, bufferSize) {
        // Cek apakah sequence sudah berakhir belum dan cek apakah
buffer sudah penuh
          if ((!check(index, sequence)) && (savedSequence.length <
bufferSize)) {
          // Looping sebanyak tinggi matrix / banyak baris
          for (let i = 0; i < matrixHeight; i++) {
               // Jika sama dengan yang dicek dengan sequence,
                             if ((matrix[i][col] == sequence[index]) &&
(matrixVisited[i][col] != 1)) {
                      // Simpan koordinatnya, tokennya, dan tandai kalau
sudah dicek
                           // Kalau sequence belum terisi, nambahin dulu
koordinat di baris pertama
                           console.log("Baris", i+1, "kolom", col+1, "=",
matrix[i][col]);

                    let coordSequence_copy = [...coordSequence];
                    let savedSequence_copy = [...savedSequence];
                                                      let matrixVisited_copy =
JSON.parse(JSON.stringify(matrixVisited));

                    // Kalau sequence kosong dan awal sequence ditemukan
di row pertama
                    if (i == 0 && savedSequence.length == 0) {
                          coordSequence_copy.push([col+1, i+1]); // Ingat
(col, row)
                         savedSequence_copy.push(matrix[i][col]);
                         matrixVisited_copy[i][col] = 1;

                         // Cek kolom lagi
                                         cekSatuKolom(matrix, col, index+1,
matrixVisited_copy, matrixWidth, matrixHeight, sequences, sequence,
sequenceVisited, currReward, coordSequence_copy, savedSequence_copy,
bufferSize);

                        // Kalau sequence kosong dan awal sequence tidak
ditemukan di row pertama
                    } else if (i != 0 && savedSequence.length == 0) {
                         // Tambahin yang di row pertama dulu
                        coordSequence_copy.push([col+1, 1]); // Ingat
(col, row)
                     savedSequence_copy.push(matrix[0][col]);
                     matrixVisited_copy[0][col] = 1;

                      coordSequence_copy.push([col+1, i+1]); // Ingat
(col, row)
                     savedSequence_copy.push(matrix[i][col]);
                     matrixVisited_copy[i][col] = 1;

                     // Saatnya ngecek baris
                                     cekSatuBaris(matrix,  i,  index+1,
matrixVisited_copy,  matrixWidth,  matrixHeight,  sequences,  sequence,
sequenceVisited,  currReward,  coordSequence_copy,  savedSequence_copy,
bufferSize);

                 // Kalau sequence tidak kosong
                 } else {
                      coordSequence_copy.push([col+1, i+1]); // Ingat
(col, row)
                     savedSequence_copy.push(matrix[i][col]);
                     matrixVisited_copy[i][col] = 1;

                     // Saatnya ngecek baris
                                     cekSatuBaris(matrix,  i,  index+1,
matrixVisited_copy,  matrixWidth,  matrixHeight,  sequences,  sequence,
sequenceVisited,  currReward,  coordSequence_copy,  savedSequence_copy,
bufferSize);
                 }
             }
         }
     } else {
         console.log()
         console.log("-- SELESAI! -- ");
         currReward = countReward(savedSequence, sequences);
         console.log("Current reward:", currReward);
         console.log("All coords:", coordSequence);
         console.log("All tokens saved:", savedSequence);
         console.log()

         // Mencatat bahwa sequence ini sudah selesai
         let currSequenceIndex = sequenceIndex(sequence, sequences);
         let sequenceVisited_copy = [...sequenceVisited];
```

```
        sequenceVisited_copy[currSequenceIndex] = 1;

        // Lanjut sequence lain
           continueCheckSequence(matrix, matrixVisited, matrixWidth,
matrixHeight,     sequences,     sequenceVisited_copy,     currReward,
coordSequence, savedSequence, "kolom", col, bufferSize);
    }
}

// Fungsi untuk iterasi dalam satu baris
function    cekSatuBaris(matrix,    row,    index,    matrixVisited,
matrixWidth,  matrixHeight,  sequences,  sequence,  sequenceVisited,
currReward, coordSequence, savedSequence, bufferSize) {
      //  Cek  apakah  sequence  sudah  berakhir  belum  dan  cek  apakah
buffer sudah penuh
        if  ((!check(index,  sequence))  &&  (savedSequence.length  <
bufferSize)) {
        // Looping sebanyak panjang matrix / banyak kolom
        for (let i = 0; i < matrixWidth; i++) {
            // Jika sama dengan yang dicek dengan sequence,
                        if  ((matrix[row][i]  ==  sequence[index])  &&
(matrixVisited[row][i] != 1)) {
                  // Simpan koordinatnya, tokennya, dan tandai kalau
sudah dicek
                      console.log("Baris", row+1, "kolom", i+1, "=",
matrix[row][i]);

                let coordSequence_copy = [...coordSequence];
                     coordSequence_copy.push([i+1, row+1]); // Ingat
(col, row)

                let savedSequence_copy = [...savedSequence];
                savedSequence_copy.push(matrix[row][i]);

                                         let  matrixVisited_copy  =
JSON.parse(JSON.stringify(matrixVisited));
                matrixVisited_copy[row][i] = 1;

                // Saatnya ngecek kolom
                cekSatuKolom(matrix, i, index+1, matrixVisited_copy,
matrixWidth,  matrixHeight,  sequences,  sequence,  sequenceVisited,
currReward, coordSequence_copy, savedSequence_copy, bufferSize)
            }
        }
    } else {
        console.log()
        console.log("-- SELESAI! -- ");
        currReward = countReward(savedSequence, sequences);
        console.log("Current reward:", currReward);
        console.log("All coords:", coordSequence);
        console.log("All tokens saved:", savedSequence);
        console.log()

        // Mencatat bahwa sequence ini sudah selesai
        let currSequenceIndex = sequenceIndex(sequence, sequences);
        let sequenceVisited_copy = [...sequenceVisited];
        sequenceVisited_copy[currSequenceIndex] = 1;

        // Lanjut sequence lain
           continueCheckSequence(matrix, matrixVisited, matrixWidth,
matrixHeight,    sequences,    sequenceVisited_copy,    currReward,
coordSequence, savedSequence, "baris", row, bufferSize);
    }
}

// Menampilkan hasil
function displayResult() {

    document.getElementById("max-reward").textContent = maxReward;

    var formattedSequence = maxSequence.join(' ');
             document.getElementById("max-sequence").textContent    =
formattedSequence;

    var formattedCoords = maxCoords.map(function(coord) {
        return coord.join(', ');
    }).join('\n');
               document.getElementById("max-coords").textContent    =
formattedCoords;

             document.getElementById("elapsed-time").textContent    =
elapsedTime + " ms";
}

// Save ke bin/file.txt
function saveContent() {
```

```
                                  var         debugContainer         =
document.getElementById('container-answer');
    var debugContent = debugContainer.textContent;
    let debugLines = debugContent.split('\n');

               let    debugFormatted    =    debugLines.map(line    =>
line.trim()).join('\n');
    let blob = new Blob([debugFormatted], { type: 'text/plain' });

    console.log(debugFormatted);

    let fileName = prompt("Enter the filename:", "save.txt");
    if (fileName === null) return;

    let a = document.createElement('a');
    a.href = URL.createObjectURL(blob);
    a.download = fileName;
    a.click();
}

// Fungsi-fungsi tambahan
function check(index, sequence) {
    // Fungsi untuk mengecek apakah index merupakan index maks dari
       //  sequence  +  1  atau  bisa  dibilang  sama  dengan  panjang
sequencenya
    return index == sequence.length;
}

function sequenceIndex(sequence, sequences) {
     //  Fungsi  untuk  mencari  index  sequence  terkini  pada  database
sequences
    // Pasti ditemukan
    for (let i = 0; i < sequences.length; i++) {
                                if     (JSON.stringify(sequence)     ==
JSON.stringify(sequences[i].sequence)) {
            return i;
        }
    }
}

function isSubstring (savedSequence, sequence) {
    // Fungsi untuk mencari apakah sebuah sequence sudah terdapat
    // pada sequence lain yang sudah diproses
    return savedSequence.join(' ').includes(sequence.join(' '));
}

function checkStartIndex (savedSequence, sequence) {
    // Akan ada kasus di mana awal sebuah sequence berada pada akhir
sequence lain
    // Fungsi untuk mengecek sampai index berapa sequence yang telah
diproses
    // beririsan dengan sequence lain
    // Mengembalikan 0 jika tidak beririsan
    // Asumsi not isSubstring
    for (let i = 0; i < savedSequence.length; i++) {
          if (savedSequence.slice(-i).join('') === sequence.slice(0,
i).join('')) {
            return i;
        }
    }
    return 0;
}

function isMore (sequence, coords, reward) {
       // Prosedur untuk mengecek apakah sequence yang dimiliki
sekarang merupakan seq
    // dengan reward terbanyak atau tidak
        if ((reward > maxReward) || ((reward == maxReward) &&
(sequence.length < maxSequence.length))) {
        maxReward = reward;
        maxCoords = coords;
        maxSequence = [...sequence];
    }
    console.log("Reward terbesar:", maxReward);
    console.log("Koordinat: ", maxCoords);
    console.log("Sequence terbaik:", maxSequence);
}

function countReward(savedSequence, sequences) {
    // Fungsi untuk menghitung reward berdasarkan sequences yang ada
    let currReward = 0;
    for (let i = 0; i < sequences.length; i++) {
        if (isSubstring(savedSequence, sequences[i].sequence)) {
            currReward += sequences[i].sequenceReward;
        }
    }
    return(currReward);
}
```