Tugas Kecil 1

IF2211 Strategi Algoritma

"Penyelesaian Cryberpunk 2077 Breach Protocol dengan Algoritma Brute Force"

Disusun oleh:

Imanuel Sebastian Girsang (13522058)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR HALAMAN

BAB I
DESKRIPSI MASALAH

Cyberpunk 2077 Breach Protocol merupakan suatu fitur yang menghadirkan simulasi peretasan jaringan lokal dalam permainan video Cyberpunk 2077. Minigame ini menawarkan pemainnya tantangan yang rumit yang memerlukan pemahaman mendalam terhadap konsep-konsep seperti token, matriks, sekuens, dan buffer. Pemain dihadapkan pada kompleksitas dalam menyusun kombinasi token alfanumerik dalam matriks untuk membentuk sekuens-sekuens yang diperlukan untuk meretas sistem dengan sukses. Dengan batasan buffer yang harus dipertimbangkan, pemain dihadapkan pada tantangan tambahan untuk mengelola alokasi token yang optimal dalam sekuens yang mereka susun.

Aturan permainan Cyberpunk 2077 Breach Protocol menambahkan lapisan kompleksitas yang signifikan. Pola gerakan pemain harus mengikuti urutan horizontal dan vertikal secara bergantian, mempertegas aspek strategis dalam setiap langkah yang diambil. Selain itu, setiap token yang dipilih harus ditempatkan dengan bijak dalam matriks, dengan pemain diminta untuk memilih token awal dengan cermat dari baris paling atas. Pencocokkan sekuens-sekuens pada token-token dalam buffer juga memerlukan pemikiran strategis, seiring dengan pemilihan sekuens yang menawarkan reward maksimal.

Strategi algoritma menjadi kunci untuk mengatasi tantangan yang ada dalam Cyberpunk 2077 Breach Protocol. Pemain harus menggunakan pendekatan yang sistematis dan efisien dalam memilih langkah-langkah yang akan diambil, dengan mempertimbangkan semua kemungkinan kombinasi token untuk mencapai tujuan akhir. Algoritma bruteforce, meskipun bisa menjadi solusi, seringkali tidak efisien dalam konteks ini, mengingat kompleksitas permainan yang dapat meningkat secara signifikan dengan penambahan token dan panjang sekuens yang harus dipertimbangkan.

BAB II
ALGORITMA BRUTE FORCE

2.1. Pendahuluan singkat

Untuk menyelesaikan permasalahan ini, metode *brute force* diaplikasikan dengan cara menggunakan beberapa variabel global untuk melakukan *tracking* terhadap data-data penting seperti matrix yang digunakan, sekuens yang sedang menjadi sekuens maksimum, serta hadiah maksimum yang sedang didapatkan. Dari situ dilakukan iterasi terhadap semua kemungkinan dari yang panjang lintasan terpendek hingga terpanjang untuk memperoleh solusi paling optimal. Sebuah prosedur bernama generatePath digunakan untuk membuah sebuah sekuens dengan panjang tertentu dan berdasarkan starting point tertentu. Prosedur ini memanfaatkan rekursi untuk melakukan pembuatan jalur terus menerus hingga habis

2.2. Penjelasan algortima

Prosedur menerima parameter-parameter berupa sisa buffer yang harus diisi, sekuens sebelumnya (dalam bentuk struct), arah (boolean) true untuk vertical, false untuk horizontal, token saat ini (struct), serta array of Point (struct).

a. Prosedur memiliki basis yaitu ketika sisa buffer == 0.
b. Jika basis terpenuhi, maka sekuens (didapat dari parameter) yang telah dikumpulkan sebelum-sebelumnya dievaluasi nilainya dengan menggunakan fungsi lain, dan dibandingkan dengan nilai global variabel dari hadiah maksimum saat ini serta sekuens maksimum saat ini. Jika kemudian hadiah maksimum saat ini sudah bernilai hadiah paling maksimal, maka pemanggilan prosedur akan dihentikan.
c. Jika tidak, maka akan ada dua loop untuk membuat lintasan.
    i. Loop pertama akan melakukan iterasi ke semua kemungkinan titik SETELAH titik saat ini yaitu semakin ke kanan bila arah saat ini horizontal, dan semakin ke bawah apabila arah saat ini vertikal.
    ii. Loop kedua akan melakukan iterasi ke semua kemungkinan titik SEBELUM titik saat ini yaitu semakin ke kiri apabila

arah saat ini horizontal dan semakin ke atas apabila arah saat ini vertikal

d. Pada setiap loop, akan dilakukan pengecekan mengenai sudah pernah dikunjungi atau belum titik tersebut. Jika sudah, maka tidak dilakukan apa. Jika belum, maka akan dilakukan pemanggilan kembali prosedur ini dengan sisa buffer yang berkurang satu dan token saat ini ditambahkan ke kumpulan sekuens yang telah dilalui dan sudah terbentuk.
e. Dari sini, semua kemungkinan lintasan sepanjang buffer dari titik awal yang telah ditentukan akan terbentuk

2.3. Pemanggilan prosedur

Pemanggilan prosedur dilakukan dalam sebuah *nested loop* yang mengiterasi semua kemungkinan *buffer* dari 2 hingga *buffer size* (1 hingga *buffer size* - 1 ketika di parameter). Serta didalamnya diiterasi pula semua kemungkinan titik awal (semua kolom pada baris pertama). Dari sini nilai *global variable* akan berubah dan lintasan-lintasan dapat terbentuk.

2.4. Klarifikasi algoritma

Algoritma *brute force* ini bergerak dari panjang buffer paling kecil hingga paling besar. Maka dari itu, solusi yang diberikan akan selalu merupakan yang paling pendek dan menghasilkan hadiah terbanyak (sebab sekuens maksimal hanya akan berubah ketika ada sekuens lain yang hadiahnya lebih besar. Definisi optimal yang dicari dengan algoritma ini adalah hadiah terbesar dengan sekuens terpendek.

Ketika ada sekuens yang bernilai 0, maka tidak akan dihiraukan sebab tidak membuat hadiah menjadi semakin besar. Begitu pula dengan negatif, tidak akan terevaluasi menjadi sekuens paling optimal kecuali memang lintasan tersebut diperlukan untuk mencapai solusi paling optimal.

BAB III
IMPLEMENTASI

3.1. Algoritma Brute Force utama dan beberapa function bantuan (CPP)

```cpp
#ifndef HELPER_H
#define HELPER_H

#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <math.h>
#include <random>
#include <chrono>
#include <thread>
#include "nlohmann/json.hpp"

using namespace std;
using json = nlohmann::json;

struct Point
{
    int baris;
    int kolom;

    bool operator==(const Point &rhs) const
    {
        return this->baris == rhs.baris && this->kolom == rhs.kolom;
    }
};

struct Token
{
    string identifier;
    Point position;
};

struct Sequence
{
    vector<Token> tokens;
    int length;
    int prize;
};

struct Matrix
{
    vector<vector<Token>> element;
    int row;
    int col;
};

struct Info
{
    int buffer_size;
    Matrix matrix;
    vector<Sequence> sequences;
};

extern vector<Sequence> prizeSequences;
extern Sequence maxSequence;
extern Matrix evaluateMatrix;
extern vector<Sequence> testSequences;
extern int actualPrizeMax;
extern int currPrizeMax;
```

Gambar 1. Header file yang digunakan untuk file cpp

```
bool isSubsetOf(Sequence prize, Sequence actual)
{
    if (prize.length > actual.length)
    {
        return false;
    }
    else
    {
        int deficit = actual.length - prize.length;
        for (int i = 0; i <= deficit; i++)
        {
            int k = 0;
            for (int j = i; j < prize.length + i; j++)
            {
                if (actual.tokens[j].identifier != prize.tokens[k].identifier)
                {
                    break;
                }
                if (k == prize.length - 1)
                {
                    return true;
                }
                k++;
            }
        }
        return false;
    }
}

int evaluateSequence(Sequence sequence, vector<Sequence> sequences)
{
    int prize = 0;
    for (int i = 0; i < sequences.size(); i++)
    {
        if (isSubsetOf(sequences[i], sequence))
        {
            prize += sequences[i].prize;
        }
    }
    return prize;
}

bool hasVisited(vector<Point> visited, Point p)
{
    for (int i = 0; i < visited.size(); i++)
    {
        if (visited[i] == p)
        {
            return true;
        }
    }
    return false;
}
```

Gambar 2. Pembantu algoritma *brute force* utama

```
void generatePath(int remainingPath, Sequence prevSequence, bool Vertical, Token current, vector<Point> visited)
{

    if (remainingPath == 0)
    {
        int prize = evaluateSequence(prevSequence, prizeSequences);
        if (prize > currPrizeMax)
        {
            currPrizeMax = prize;
            maxSequence = prevSequence;
        }
    }
    else
    {
        for (int i = Vertical ? current.position.baris + 1 : current.position.kolom + 1; Vertical ? i < evaluateMatrix.row : i < evaluateMatrix.col; i++)
        {
            Token nextToken = Vertical ? getToken(evaluateMatrix, i, current.position.kolom) : getToken(evaluateMatrix, current.position.baris, i);
            if (!hasVisited(visited, nextToken.position))
            {
                visited.push_back(nextToken.position);
                generatePath(remainingPath - 1, appendToken(prevSequence, nextToken), !Vertical, nextToken, visited);
            }
        }
        for (int i = Vertical ? current.position.baris - 1 : current.position.kolom - 1; Vertical ? i >= 0 : i >= 0; i--)
        {
            Token nextToken = Vertical ? getToken(evaluateMatrix, i, current.position.kolom) : getToken(evaluateMatrix, current.position.baris, i);
            if (!hasVisited(visited, nextToken.position))
            {
                visited.push_back(nextToken.position);
                generatePath(remainingPath - 1, appendToken(prevSequence, nextToken), !Vertical, nextToken, visited);
            }
        }
    }
}
```

Gambar 3. Algoritma *Brute Force utama*

```cpp
#include <iostream>
#include "helpers.hpp"

void printJSON(int maximumReward, const std::string &sequence, const std::vector<Point> &paths, const std::string &time)
{
    std::cout << "{\n";
    std::cout << "  \"maximum_reward\": " << maximumReward << ",\n";
    std::cout << "  \"sequence\": \"" << sequence << "\",\n";
    std::cout << "  \"paths\": [\n";
    for (size_t i = 0; i < paths.size(); ++i)
    {
        const auto &path = paths[i];
        std::cout << "    { \"baris\": " << path.baris << ", \"kolom\": " << path.kolom << " }";
        if (i != paths.size() - 1)
        {
            std::cout << ",";
        }
        std::cout << "\n";
    }
    std::cout << "  ],\n";
    std::cout << "  \"time\": \"" << time << "\"\n";
    std::cout << "}\n";
}
void processLoop(Info &info, int start_col, int end_col, int size)
{
    while (start_col < end_col && currPrizeMax != actualPrizeMax)
    {
        Sequence Start = generateSequence(1, vector<Token>{info.matrix.element[0][start_col]}, 0);
        generatePath(size, Start, true, Start.tokens[0], {info.matrix.element[0][start_col].position});
        start_col++;
    }
}

int main(int argc, char const *argv[])
{
    srand(time(NULL));
    Info info;
    string JSONString = string(argv[1]);
    json j = json::parse(JSONString);

    info = ParseJSON(j);

    prizeSequences = info.sequences;
    evaluateMatrix = info.matrix;
    for (int i = 0; i < info.sequences.size(); i++)
    {

        actualPrizeMax += info.sequences[i].prize;
    }

    const int num_threads = thread::hardware_concurrency() > evaluateMatrix.col ? evaluateMatrix.col : thread::hardware_concurrency();

    int cols_per_thread = evaluateMatrix.col / num_threads;

    vector<thread> threads;
    // Start threads
    auto start = chrono::high_resolution_clock::now();
    for (int j = 1; j < info.buffer_size; j++)
    {
        for (int i = 0; i < num_threads; ++i)
        {
            int start_col = i * cols_per_thread;
            int end_col = (i == num_threads - 1) ? evaluateMatrix.col : start_col + cols_per_thread;
            threads.push_back(thread(processLoop, ref(info), start_col, end_col, j));
        }
    }
    for (auto &t : threads)
    {
        t.join();
    }
    auto end = chrono::high_resolution_clock::now();
    auto diff = chrono::duration_cast<std::chrono::milliseconds>(end - start);
    string sequence = "";
    for (int i = 0; i < maxSequence.tokens.size(); i++)
    {
        sequence += maxSequence.tokens[i].identifier + " ";
    }
    vector<Point> paths;
    for (int i = 0; i < maxSequence.tokens.size(); i++)
    {
        paths.push_back(maxSequence.tokens[i].position);
    }

    printJSON(currPrizeMax, sequence, paths, to_string(diff.count()) + " ms");

    return 0;
}
```

Gambar 4. program utama yang akan dipanggil melalui API Route dengan passing arguments berupa JSON

3.2. TypeScript

```typescript
import type { matrix, sequence, Tokens } from "../../app/types/main";
type Result = {
  sequence: sequence[];
  matrix: matrix;
};
export default function randomizeInput(
  lengthsequence: number,
  sequence: string[],
  row: number,
  col: number,
  numsequence: number
): Result {
  const randomMatrix = generateRandomMatrix(row, col, sequence);
  const randomSequences = generateRandomSequence(
    sequence,
    lengthsequence,
    numsequence
  );
  return {
    sequence: randomSequences,
    matrix: randomMatrix,
  };
}

function generateRandomMatrix(
  row: number,
  col: number,
  tokens: string[]
): matrix {
  const matrix: matrix = {
    row,
    col,
    element: [],
  };

  for (let i = 0; i < row; ++i) {
    matrix.element[i] = [];
    for (let j = 0; j < col; ++j) {
      const index = Math.floor(Math.random() * tokens.length);
      matrix.element[i][j] = generateToken(tokens[index], i, j);
    }
  }

  return matrix;
}

function generateToken(
  identifier: string,
  baris: number,
  kolom: number
): Tokens {
  return {
    identifier: identifier,
    position: {
      baris: baris,
      kolom: kolom,
    },
  };
}
function generateSequence(
  randomLength: number,
  randomTokens: Tokens[],
  randomPrize: number
): sequence {
  return {
    tokens: randomTokens,
    length: randomLength,
    prize: randomPrize,
  };
}

function generateRandomSequence(
  tokens: string[],
  maxLength: number,
  num: number
): sequence[] {
  const sequences: sequence[] = [];
  for (let i = 0; i < num; i++) {
    const randomLength = Math.floor(Math.random() * (maxLength - 1)) + 2;

    const randomTokens: Tokens[] = [];
    for (let j = 0; j < randomLength; j++) {
      const randomIndex = Math.floor(Math.random() * tokens.length);
      randomTokens.push(generateToken(tokens[randomIndex], 0, 0));
    }

    const randomPrize = Math.floor(Math.random() * (50 - 10 + 1)) + 10;
    sequences.push(generateSequence(randomLength, randomTokens, randomPrize));
  }
  return sequences;
}

```

Gambar 5. Kode Typescript untuk membuat sekuens random pada website

```typescript
export default function readTxt(content) {
  const lines = content.split("\n");

  // Validate input content
  if (lines.length < 5) {
    throw new Error("Invalid input format. Not enough lines.");
  }

  // Parse buffer size
  const bufferSize = parseInt(lines[0]);
  if (isNaN(bufferSize)) {
    throw new Error("Invalid buffer size.");
  }

  // Parse matrix size
  const matrixSize = lines[1].split(" ");
  if (matrixSize.length !== 2) {
    throw new Error("Invalid matrix size format.");
  }
  const matrixRows = parseInt(matrixSize[0]);
  const matrixCols = parseInt(matrixSize[1]);
  if (isNaN(matrixRows) || isNaN(matrixCols)) {
    throw new Error("Invalid matrix size.");
  }

  // Parse matrix elements
  const matrixElements = [];
  for (let i = 0; i < matrixRows; i++) {
    const rowLine = lines[2 + i];
    if (rowLine.length !== matrixCols * 3) {
      throw new Error("Invalid matrix element format.");
    }
    const rowTokens = [];
    for (let j = 0; j < matrixCols; j++) {
      const token = rowLine.substr(j * 3, 2);
      if (!/^[a-zA-Z0-9]{2}$/.test(token)) {
        throw new Error("Invalid token format in matrix.");
      }
      rowTokens.push({
        identifier: token,
        position: { baris: i, kolom: j },
      });
    }
    matrixElements.push(rowTokens);
  }

  // Parse sequences
  const sequences = [];
  let index = 2 + matrixRows;
  const numSequences = parseInt(lines[index]);
  if (isNaN(numSequences)) {
    throw new Error("Invalid number of sequences.");
  }
  for (let i = 0; i < numSequences; i++) {
    index++;
    const sequenceLine = lines[index];
    if (sequenceLine.length % 3 !== 0) {
      throw new Error("Invalid sequence format.");
    }
    const sequenceTokens = sequenceLine
      .split(" ")
      .filter((token) => token.trim());
    const lastTokenIndex = sequenceTokens.length - 1;
    sequenceTokens[lastTokenIndex] = sequenceTokens[lastTokenIndex].replace(
      /\r/g,
      ""
    );
    console.log(sequenceTokens);
    const tokens = sequenceTokens.map((token) => {
      if (!/^[a-zA-Z0-9]{2}$/.test(token)) {
        throw new Error("Invalid token format in sequence.");
      }
      return {
        identifier: token,
        position: { baris: 0, kolom: 0 },
      };
    });
    const prizeLine = lines[index + 1];
    const prize = parseInt(prizeLine);
    if (isNaN(prize)) {
      throw new Error("Invalid prize value.");
    }
    sequences.push({
      tokens: tokens,
      length: tokens.length,
      prize: prize,
    });
    index++;
  }

  return {
    buffer_size: bufferSize,
    matrix: {
      element: matrixElements,
      row: matrixRows,
      col: matrixCols,
    },
    sequences: sequences,
  };
}
```

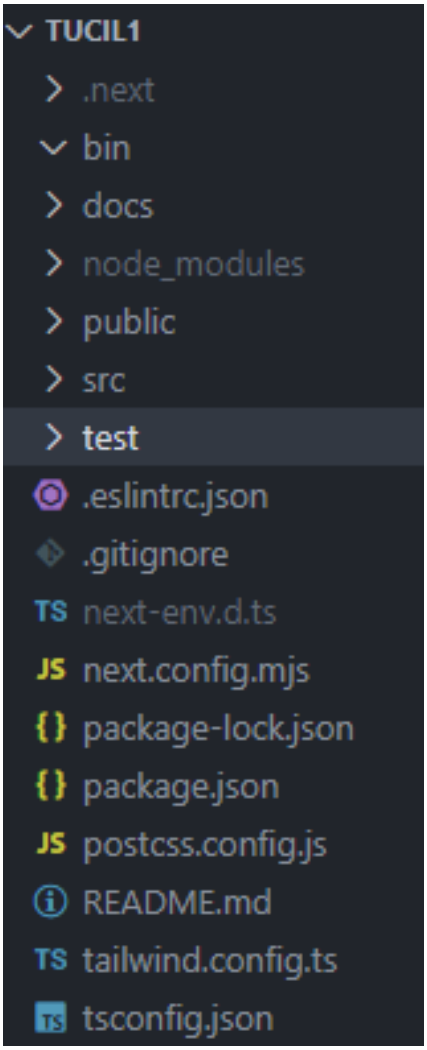
Gambar 6. Kode Typescript untuk membaca file txt, beserta berbagai validasi errornya

```
import { NextRequest, NextResponse } from "next/server";
import { spawn } from "child_process";

export const POST = async (req: NextRequest) => {
  const data = JSON.stringify(await req.json()); // Convert data to JSON string
  return new Promise((resolve, reject) => {
    const cppProcess = spawn("src/app/api/main/tes.exe", [data]);
    let output = "";
    let error = "";

    cppProcess.stdout.on("data", (data) => {
      output += data.toString();
      console.log(output);
    });

    cppProcess.stderr.on("data", (data) => {
      error += data.toString();
    });

    cppProcess.on("close", (code) => {
      if (code !== 0) {
        output = error;
      }

      const response = NextResponse.json({ output }, { status: 200 });
      resolve(response);
    });

    cppProcess.on("error", (err) => {
      reject(err); // Reject the promise if there's an error with the child process
    });
  });
};

```

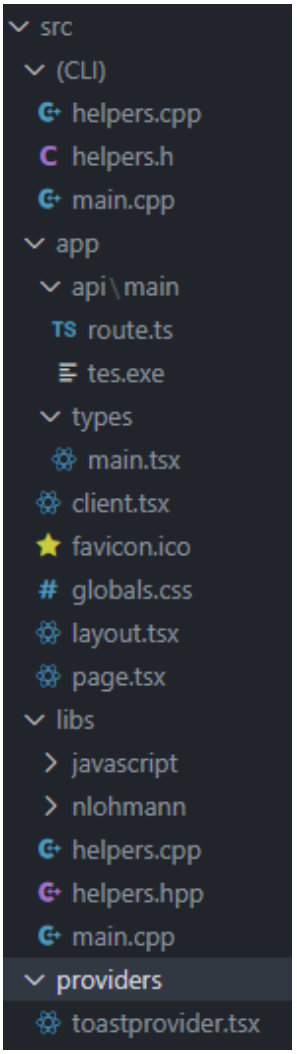
Gambar 7. Kode API Route untuk mengeksekusi program cpp yang dibuat dengan mempassing arguments berupa JSON

3.3. Website

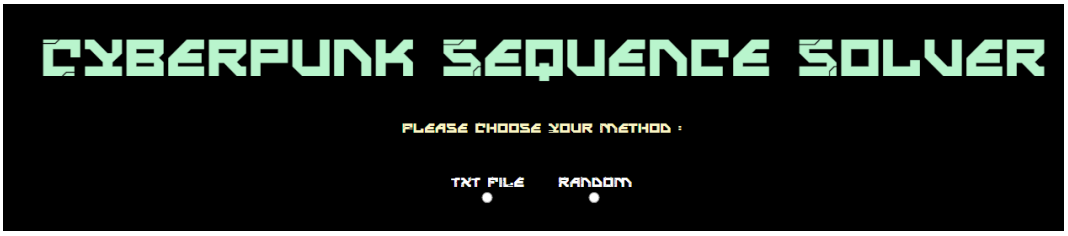
Website solver dibuat dengan menggunakan NextJs. Untuk mengeksekusi kode cpp, digunakan command spawn yang akan ter*invoke* ketika ada API Call ke rute tersebut. Untuk *source code* dari website tidak akan ditampilkan pada laporan ini, tetapi akan dibahas garis besar pengstrukturan serta tampilan dari website.



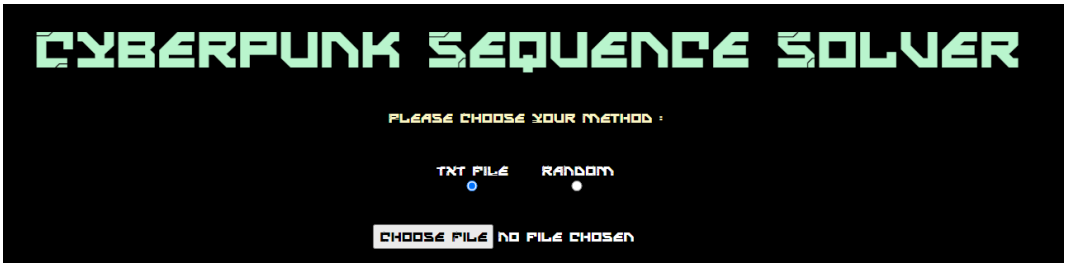
Gambar 8. Struktur Proyek Website



Gambar 9. Isi Folder SRC

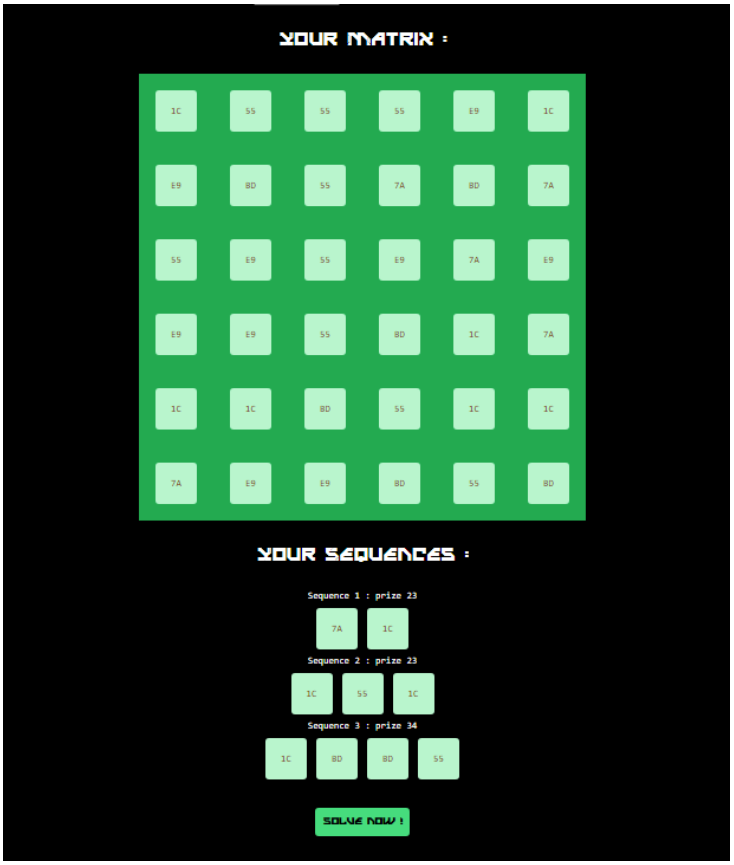


Gambar 10. Tampilan awal ketika masuk website



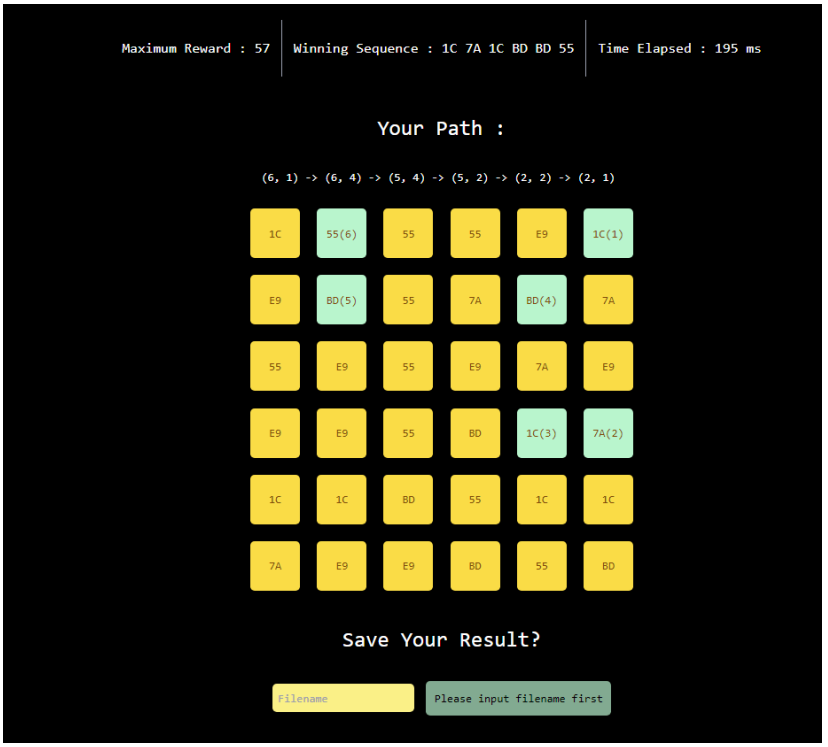
Gambar 11. Tampilan ketika memilih input file

Gambar 12. Tampilan ketika memilih input random

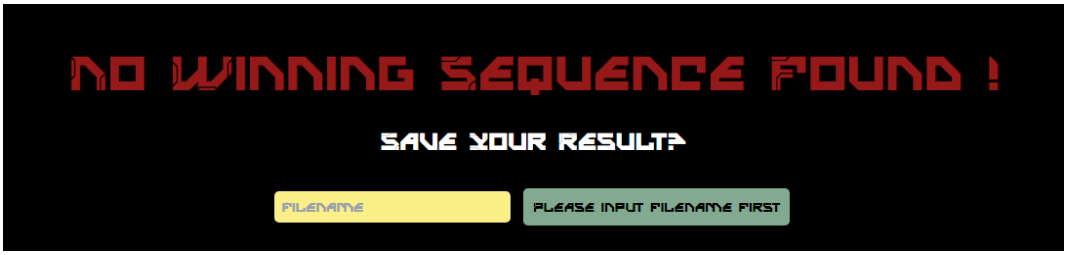


Gambar 13. Tampilan ketika txt file berhasil di parse

Gambar 14. Tampilan ketika txt file gagal di parse



Gambar 15. Tampilan ketika terdapat jawaban lintasan optimal



Gambar 16. Tampilan ketika tidak terdapat jawaban lintasan optimal (prize == 0)

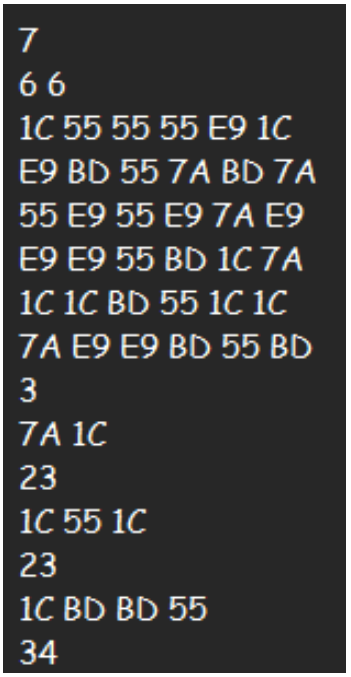
BAB IV
PENGUJIAN

4.1. Cara menggunakan program

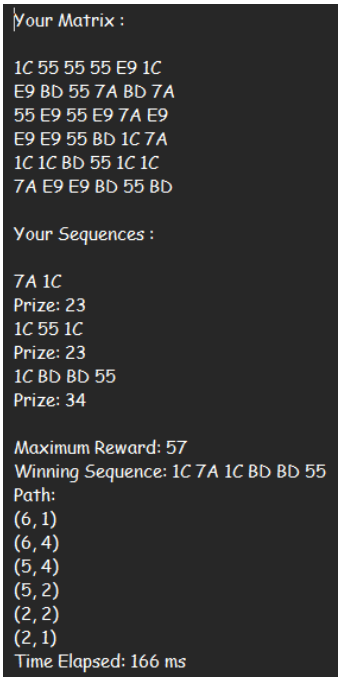
Untuk menggunakan CLI, pergi ke folder bin dan jalankan file executeable disana dengan menuliskan ./cli.exe di CLI. Pengguna akan diarahkan untuk menggunakan file txt maupun random lalu program dapat dijalankan. Untuk menggunakan website, silahkan terlebih dahulu lakukan:

- Npm i
- Npm run dev
- Pergi ke localhost:3000 di browser

4.2. Uji Coba



Gambar 17. Input TC 1



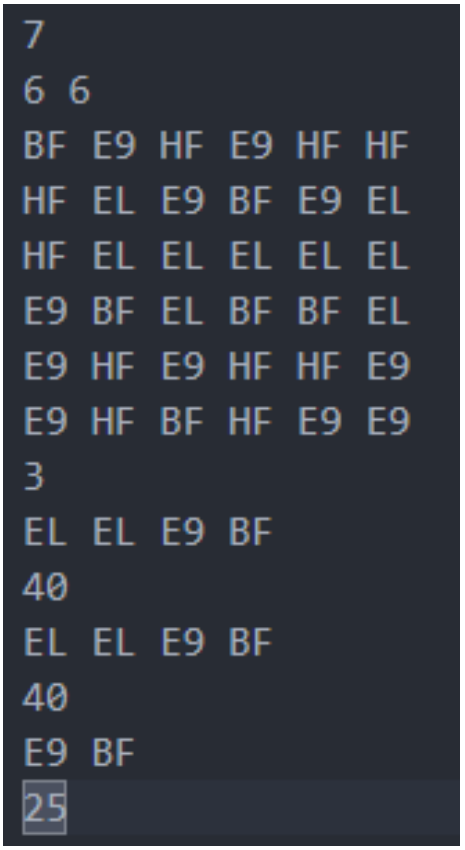
Gambar 18. Output TC 1

```
7
5 6
1C 55 55 55 E9
E9 BD 55 7A BD
55 E9 55 E9 7A
E9 E9 55 BD 1C
1C 1C BD 55 1C
7A E9 E9 BD 55
3
EF GH
23
IJ KL MN
23
OP QR ST TU
34
```

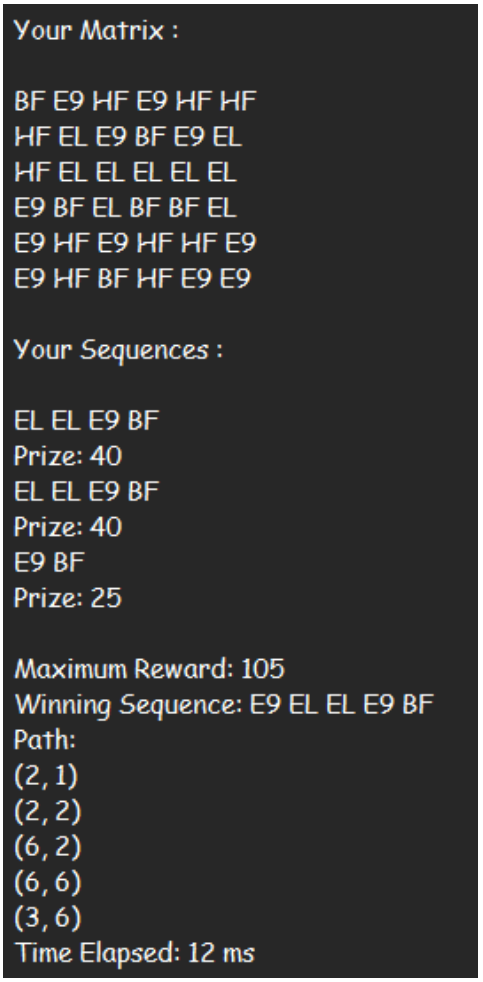
Gambar 19. Input TC 2
(case tidak ada jawaban)

```
Your Matrix :

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

Your Sequences :

EF GH
Prize: 23
IJ KL MN
Prize: 23
OP QR ST TU
Prize: 34

Maximum Reward: 0
Winning Sequence:
Path:
Time Elapsed: 108 ms
```

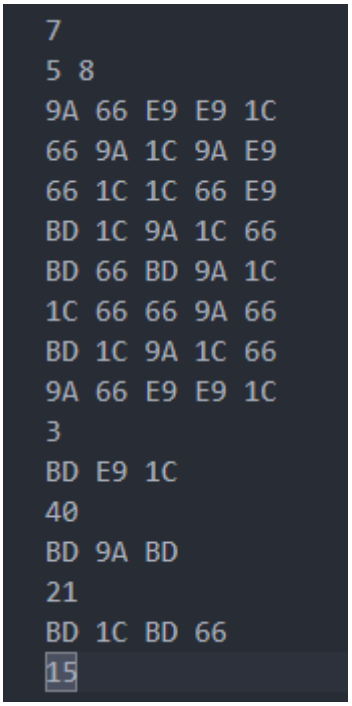
Gambar 20. Output TC 2
(case tidak ada jawaban)



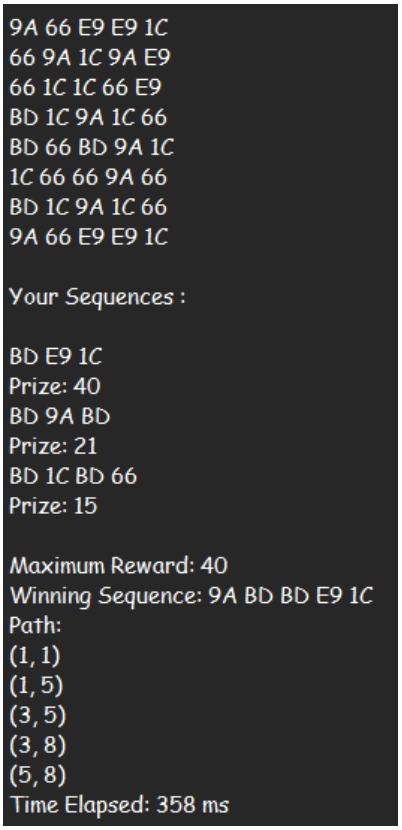
Gambar 21. Input TC 3
(*best case scenario* yaitu hadiah ditemukan di buffer yang masih relatif kecil*)*



Gambar 22. Output TC 3
(*best case scenario* yaitu hadiah ditemukan di buffer yang masih relatif kecil*)*



Gambar 23. Input TC 4
(*Ukuran M x N)*



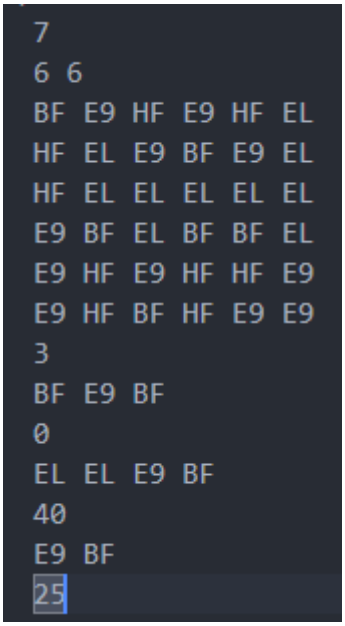
Gambar 24. Input TC 4
(*Ukuran M x N)*

```
7
5 5
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
3
A1 A1 A1 A1 A1
50
A1 A1 A1
-200
A1 A1
30
```

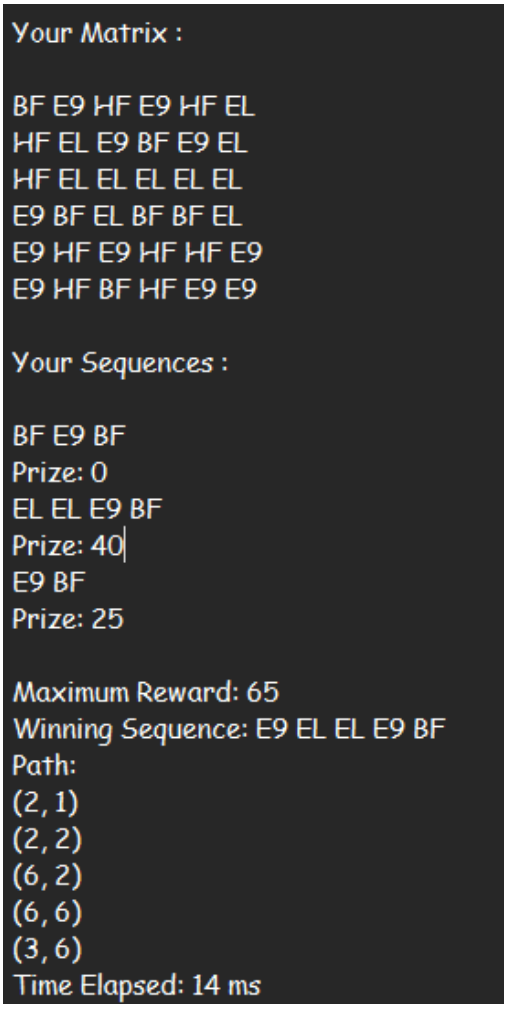
Gambar 25. Input TC 5
(Sekuens negatif menjadi penghambat untuk mengambil lebih banyak*)*

```
Your Matrix :

A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1
A1 A1 A1 A1 A1

Your Sequences :

A1 A1 A1 A1 A1
Prize: 50
A1 A1 A1
Prize: -200
A1 A1
Prize: 30

Maximum Reward: 30
Winning Sequence: A1 A1
Path:
(2, 1)
(2, 2)
Time Elapsed: 42 ms
```

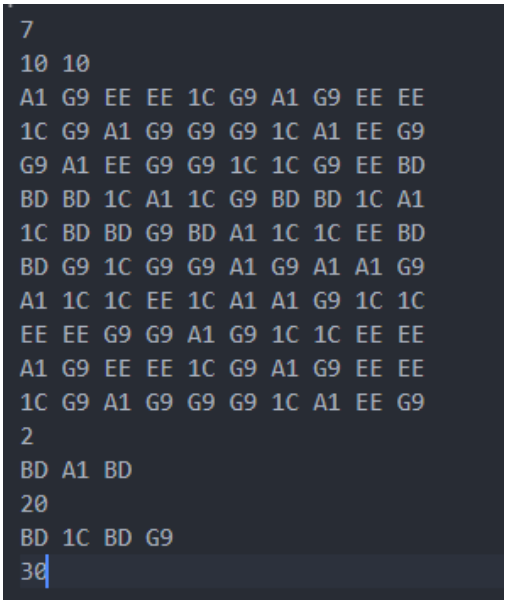
Gambar 26. OutputTC 5
(Sekuens negatif menjadi penghambat untuk mengambil lebih banyak*)*



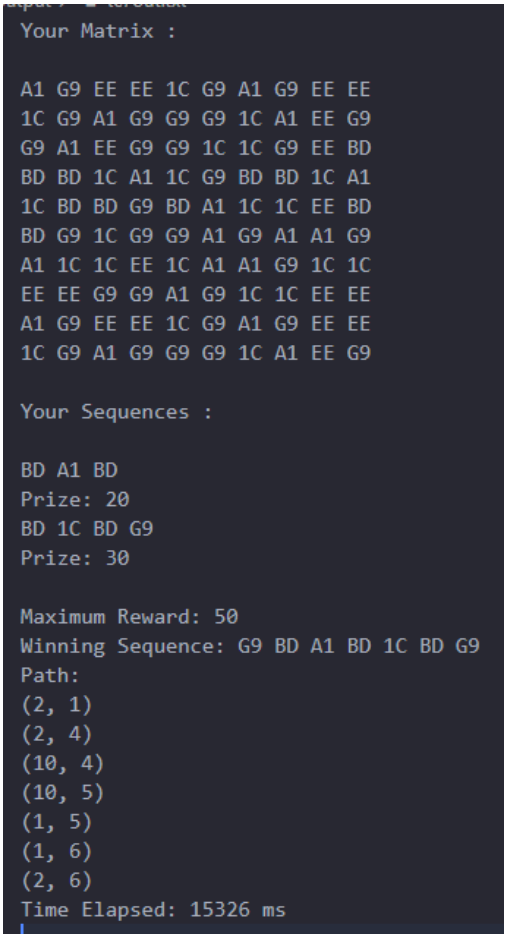
Gambar 27. Input TC 6
(Sekuens 0 menjadi trigger untuk mengambil lebih banyak*)*



Gambar 28. Output TC 6
(Sekuens 0 menjadi trigger untuk mengambil lebih banyak*)*



Gambar 29. Input TC 7
(10 x 10 Matrix)



Gambar 30. Output TC 7
(10 x 10 Matrix)

BAB V
LAMPIRAN

Link repository GitHub : https://github.com/ImanuelSG/Tucil1_13522058

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	✓	