

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA
PENYELESAIAN CYBERPUNK 2077 BREACH
PROTOCOL DENGAN ALGORITMA *BRUTE FORCE*



Oleh :
Moh Fairuz Alauddin Yahya
13522057

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

Daftar Isi

BAB I.....	3
Deskripsi Program	3
BAB II.....	4
Algoritma Program	4
BAB III	6
Source Code.....	6
<i>File Input Handling</i>	6
<i>Random</i>	10
<i>Download</i>	12
Server Side.....	14
BAB IV	18
Eksperimen	18
A. Masukan <i>Token</i> Matrix Tidak Valid	18
B. Masukkan <i>Token Target</i> Tidak Valid.....	18
C. Masukan Matrix M x N.....	19
D. Masukan Tidak Punya Solusi.....	19
E. Masukan Kasus Test <i>Case Best Case</i>	20
F. Masukan Kasus Test <i>Case Worst Case</i>	20
G. Masukan dengan <i>Randomize</i> dan Tidak Valid	21
H. Masukan dengan <i>Randomize</i> tetapi Tidak Memiliki Solusi	21
I. Masukan dengan <i>Randomize</i> dan Memiliki Solusi	22
LAMPIRAN.....	23

BAB I

Deskripsi Program

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*

(Paragraf di atas dikutip dari : informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/stima23-24.htm#PRdanTugas).



BAB II

Algoritma Program

Pada program ini digunakan algoritma brute-force untuk menemukan 1 solusi yang mungkin dengan melakukan evaluasi terhadap seluruh kemungkinan *sequence* yang terbentuk.

Hasil Langkah-langkah yang digunakan program dibagi menjadi 2 yaitu:

Client Side:

1. Program terdiri dari 2 halaman, yaitu halaman *randomize* untuk melakukan input dengan fungsi *randomize*, dan halaman *file* untuk mengunggah *file* dalam format txt.
2. Pada halaman *file*, pengguna mengunggah *file* txt yang kemudian di-parse untuk memasukkan nilai-nilainya ke dalam state-state data.
3. Semua validasi terkait kesalahan pada *file* di-handle saat parsing dan hasilnya ditampilkan melalui warning error menggunakan toast untuk antarmuka pengguna (UI).
4. Pada halaman *randomize*, pengguna memasukkan semua field yang diperlukan untuk menghasilkan matriks dan *target*, seperti tinggi, lebar matriks, *buffer* size, dan jumlah *target* yang ingin digenerate.
5. Semua validasi juga dilakukan untuk mencegah kesalahan saat melakukan permintaan API pada sisi klien.
6. Selanjutnya, data state dikirim melalui API yang sudah di-deploy pada <https://fairuzald-tucil-1-stima.hf.space/> saat production, dan pada <http://localhost:8000> saat pengembangan (development).
7. Lalu hasil pengolahan data dari server akan ditangkap lagi dan ditampilkan pada UI.
8. User dapat mendownload solusi dimana data kemudian akan di parse lagi ke dalam *file* txt. User juga dapat memberikan input untuk mengubah default *filename* untuk download hasilnya.

Server Side:

1. Program menerima masukan dari pengguna berupa matriks, *target* string yang memiliki poin, ukuran *buffer* maksimum, lebar dan panjang matriks.
2. Selanjutnya, program mencari semua kandidat *sequence* yang mungkin sepanjang ukuran *buffer* maksimum dengan menggunakan prinsip *stack* dan sesuai ketentuan soal untuk generate bergantian berdasarkan orientasi pergerakan, dimana program akan melakukan push data tupel (x,y) yang menyatakan koordinat dalam matriks ke dalam *stack* sambil decrement ukuran *buffer*, lalu pop dan push lagi hingga ukuran *buffer* habis atau bernilai nol, kemudian menghasilkan hasil berupa semua kemungkinan *sequence* yang memiliki panjang tupel koordinat sebanyak ukuran *buffer*.
3. Setelah itu, program dievaluasi, untuk keperluan optimasi dibagi menjadi dua, yaitu *mini case evaluation* ketika ukuran *buffer* lebih besar dari tujuh atau kolom matriks lebih dari enam atau baris matriks lebih dari enam. Sisanya, kami melakukan evaluasi lebih mendalam.
4. Pada evaluasi *mini case*, semua nilai dari *token sequence* digabungkan menjadi sebuah string, begitu juga semua *target token*. Program akan melakukan *looping* dengan mengenumerasi indeks dan nilai string dari string *sequence* tersebut, kemudian mencocokkannya untuk mencari skor maksimum yang mungkin. Jika skor oleh suatu *sequence* mencapai full point dari *target*, maka program akan langsung return, tetapi jika tidak, program akan mengevaluasi lagi untuk

mendapatkan skor maksimum, lalu akan dilakukan pemotongan *token* yang tidak diperlukan di akhir *sequence* jika ada, sehingga bisa dicapai susunan urutan dengan skor maksimum dan panjang *token* paling minimum, lalu hanya mengembalikan satu nilai saja.

5. Pada evaluasi kasus, prinsipnya hampir mirip dengan evaluasi *mini case*, tetapi terlebih dahulu string *sequence* dibuat menjadi himpunan untuk menghilangkan string yang memiliki nilai *sequence token* yang sama, sehingga untuk pengecekan selanjutnya bisa lebih minimum dengan hanya mengecek nilai *sequence* yang unik.
6. Hasil akhir program akan mengembalikan hasil yang berisi urutan yang berisi susunan koordinat, skor, dan string *token* dari koordinat matrix, beserta waktu eksekusi.

Untuk cara yang saya lakukan tersebut merupakan cara yang telah saya uji dari beberapa cara lain yang terpikirkan oleh saya, dimana detail lengkap untuk test *case* saya lampirkan di bagian lampiran, diantaranya:

1. Penggunaan rekursi untuk generate possible sequence daripada menggunakan stack.
Saya sudah menguji untuk penggunaan rekursi jauh sangat buruk daripada penggunaan stack, hal ini dikarenakan Setiap kali fungsi rekursif dipanggil, informasi kontekstual harus disimpan di dalam stack call, seperti nilai-nilai parameter dan alamat return. Hal ini dapat mengakibatkan penumpukan memori yang besar, terutama untuk sekuens yang sangat panjang, dan dapat memperlambat kinerja program. Terbukti dengan tes untuk matriks 10x10 sangat berat hingga bisa mencapai 2 menit.
2. Pengecekan dengan generate semua kemungkinan sequence terlebih dahulu lebih baik dari pada generate satu per satu sembari checking
Mungkin benar bahwa untuk kasus uji yang dapat menemukan urutan dengan poin penuh dari target (*best case*), itu mungkin lebih baik. Namun, saya yakin kasus uji seperti itu jarang terjadi dan akan menjadi lebih buruk pada *worst case* di mana pemeriksaan harus dilakukan sampai akhir. Saya juga telah mengujinya, dan waktu runtime pada *best case* untuk kasus 1 tidak jauh berbeda dengan kasus 2. Namun, perbedaan sangat signifikan pada *worst case*, terutama untuk matriks dan buffer berukuran besar.

Pranala Repository: https://github.com/fairuzald/Tucil1_13522057

BAB III

Source Code

Client Menggunakan TypeScript

File Input Handling

```
const [targets, setTargets] = useState<Target[]>([]);
const [matrix, setMatrix] = useState<Matrix>({});
const [buffer, setBuffer] = useState<number>(0);
const [data, setData] = useState<any>({});

// handle file change upload
const handleFileChange = async (event: ChangeEvent<HTMLInputElement>) => {
  // Reset the state
  setBuffer(0);
  setTargets([]);
  setMatrix({});
  setData({});
  const selectedFile = event.target.files?.[0];

  if (selectedFile) {
    const fileContent = await readFileContent(selectedFile);
    const { buffer, parsedTargets, parsedMatrix } =
      parseFileContent(fileContent);

    setBuffer(buffer);
    setTargets(parsedTargets);
    setMatrix(parsedMatrix);
  }
};

// parse file content
const readFileContent = (file: File): Promise<string> => {
  return new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.onload = (event) => {
      if (event.target?.result) {
        resolve(event.target.result as string);
      } else {
        reject(new Error("Error reading file."));
      }
    };
    reader.onerror = (error) => {
      reject(error);
    };
    reader.readAsText(file);
  });
};
```

```

const parseFileContent = (content: string): ParsedFile => {
  const lines = content.split("\n");
  const parsedTargets: Target[] = [];
  let parsedMatrix: Matrix = {};
  let matrixRowIndex = 0;
  let rowCount = 0;
  let colCount = 0;
  let buffer = 0;
  let numberOfTargets = 0;
  let errorToastShown = false;

  // Parse the content line by line
  lines.forEach((line, index) => {
    const trimmedLine = line.trim();
    if (index === 0) {
      // Parse buffer size 1st line
      buffer = parseInt(trimmedLine, 10);
    }
    // Parse row and column count 2nd line
    else if (index === 1) {
      const [cols, rows] = trimmedLine.split(" ").map((value) => parseInt(value, 10));
      if (isNaN(cols) || isNaN(rows) || cols < 2 || rows < 2) {
        toast.error("Invalid matrix size. The number of columns and rows must match the specified width at least 2.");
        errorToastShown = true;
        resetVariables();
        return;
      }
      rowCount = rows;
      colCount = cols;
    }
    // Parse matrix values
    else if (index >= 2 && index < 2 + rowCount) {
      const values = trimmedLine.split(" ");

      // Validate row and column count matrix section line
      if (values.length !== colCount && !errorToastShown) {
        toast.error("Invalid matrix width.");
        errorToastShown = true;
        resetVariables();
        return;
      }
    }
  })
}

```

```

    const matrixRow = values.map((value) => value);
    parsedMatrix[matrixRowIndex++] = matrixRow;
  }
  // Parse the number of targets
  else if (index === 2 + rowCount) {
    if (trimmedLine.split(" ").length !== 1 && !errorToastShown) {
      toast.error("Invalid height matrix format. The number of targets must be a single integer.");
      errorToastShown = true;
      resetVariables();
      return;
    }

    numberOfTargets = parseInt(trimmedLine, 10);
  }
  // Parse and validate target values
  else if (index > 2 + rowCount && index <= 2 + rowCount + numberOfTargets * 2) {
    if (index % 2 === 0 && rowCount % 2 === 1 || (index % 2 === 1 && rowCount % 2 === 0)) {
      // Parse target values (sequence and points)
      const sequenceArray = trimmedLine.split(" ");
      if (sequenceArray.length <= 1) {
        toast.error("Invalid target sequence format. Each sequence must have at least 2 tokens.");
        resetVariables();
        return;
      }

      const isValid = validateSequenceFormat(sequenceArray);

      if (isValid) {
        const sequence = sequenceArray.join("");
        const pointsIndex = index + 1;
        const points = parseInt(lines[pointsIndex], 10);
        parsedTargets.push({ sequence: sequence.split(""), points });
      } else {
        resetVariables();
      }
    }
  }
});

return { buffer, parsedTargets, parsedMatrix };

```



```

// Helper function to reset variables
function resetVariables() {
  buffer = 0;
  rowCount = 0;
  colCount = 0;
  parsedTargets.length = 0;
  parsedMatrix = {};
}

// Helper function to validate sequence format
function validateSequenceFormat(sequenceArray: string[]): boolean {
  let isValid = true;

  sequenceArray.forEach((value) => {
    if (value.length !== 2 && !errorToastShown) {
      toast.error("Invalid target sequence format. Each tokens must have exactly 2 characters.");
      errorToastShown = true;
      isValid = false;
    }
  });

  return isValid;
}
};

```

```

const handleClick = async () => {
  if (Object.keys(matrix).length < 1 || targets.length < 1) {
    toast.error("Matrix or targets must be initialized first");
    return;
  }
  if (buffer < 1) {
    toast.error("Buffer must be initialized first");
    return;
  }
  if (getRowCount(matrix) < 1 || getColumnCount(matrix) < 1) {
    toast.error("Matrix must be initialized first");
    return;
  }
  try {
    const requestBody = {
      matrix: Object.values(matrix),
      targets: targets,
      buffer: buffer,
      rowCount: getRowCount(matrix),
      colCount: getColumnCount(matrix),
    };
    makeApiRequest({
      body: JSON.stringify(requestBody),
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      loadingMessage: "Loading....",
      successMessage: "Successful!",
      endpoint: "/api/breach_protocol_solve",
      onSuccess: (data) => {
        setData(data);
      },
    });
  } catch (error) {
    console.error("Fetch error:", error);
  }
};

```

Random

```
// Function to randomize a matrix with the given row and column count
✓ const randomizeMatrix = (rowCount: number, colCount: number): Matrix => {
  // Check if rowCount is less than 2, show error and return empty matrix
  ✓ if (rowCount < 2) {
    toast.error("Matrix height must be at least 2.");
    return [];
  }
  // Check if colCount is less than 2, show error and return empty matrix
  ✓ if (colCount < 2) {
    toast.error("Matrix width must be at least 2.");
    return [];
  }
  // Define characters to be used in the matrix
  const characters = "ABCDEFGHGIJKLMNOPQRSTUVWXYZ0123456789";
  const matrix: Matrix = {};

  // Loop through rows
  ✓ for (let i = 0; i < rowCount; i++) {
    const row: string[] = [];
    // Loop through columns
    ✓ for (let j = 0; j < colCount; j++) {
      // Generate random characters for each cell
      const randomCharIndex1 = Math.floor(Math.random() * characters.length);
      const randomCharIndex2 = Math.floor(Math.random() * characters.length);
      const randomChar1 = characters.charAt(randomCharIndex1);
      const randomChar2 = characters.charAt(randomCharIndex2);
      row.push(randomChar1 + randomChar2);
    }
    matrix[i] = row;
  }
  // Return the generated matrix
  return matrix;
};
```

```

// Function to randomize targets based on a matrix, count, and buffer
const randomizeTarget = (
  matrix: Matrix,
  count: number,
  buffer: number
  🐛 Target[] => {
    // Check if count is less than 1, show error and return empty array    You, 5
    if (count < 1) {
      toast.error("Target count must be more than 0");
      return [];
    }
    // Check if buffer is less than 1, show error and return empty array
    if (buffer < 1) {
      toast.error("Buffer must be more than 0");
      return [];
    }
    // Check if the matrix is not initialized, show error and return empty array
    if (Object.keys(matrix).length < 1) {
      toast.error("Matrix must be initialized first");
      return [];
    }
    let randomTargets: Target[] = [];
    // Loop to generate random targets
    for (let i = 0; i < count; i++) {
      let random = [];
      // Generate random count for the sequence (between 2 and 6)
      let randomCountSeq = Math.floor(Math.random() * 6) + 1;
      const minimumValue = 2;
      // Ensure minimum count for the sequence
      if (randomCountSeq < minimumValue) {
        randomCountSeq = minimumValue;
      }

      // Loop to generate random sequence based on matrix
      for (let j = 0; j < randomCountSeq; j++) {
        // Generate random row and column indices
        const randomRowIndex = Math.floor(
          Math.random() * Object.keys(matrix).length
        );
        const randomColIndex = Math.floor(
          Math.random() * matrix[randomRowIndex].length
        );
        random.push(matrix[randomRowIndex][randomColIndex]);
      }
    }
  }

```

```

    // If the sequence is longer than the buffer, trim it
    if (random.length > buffer) {
      random = random.slice(0, buffer);
    }
    // Create a random target with the generated sequence and points
    const randomTarget: Target = {
      sequence: random,
      points: Math.floor(Math.random() * 100),
    };
    randomTargets.push(randomTarget);
  }

  return randomTargets;
};

export { randomizeMatrix, randomizeTarget };

```

Download

```

// Function to format the solution data for display
const formatSolution = (data: Solution): string => {
  const formattedData: string[] = [];

  // Baris 1: Buffer size
  if (Boolean(data.result.seq && data.result.string)) {
    formattedData.push(`${data.result.score}`);
  }

  // Baris 2: String token
  if (Boolean(data.result.seq && data.result.string)) {
    let formattedString: string = "";

    // Iterate through the string and group every two characters with a space
    for (let i = 0; i < data.result.string.length; i += 2) {
      const twoChars = data.result.string.slice(i, i + 2);
      formattedString += twoChars + " ";
    }
    formattedString = formattedString.trim();
    formattedData.push(formattedString);
  } else {
    // If no answer sequence is present, add a default message
    formattedData.push("No answer sequence to get the prize");
  }

  // Baris 3 dan seterusnya: Koordinat data seq index
  data.result.seq.forEach((seq) => {
    formattedData.push(`${seq}`);
  });

  // Baris terakhir: Hasil dalam milisekon
  formattedData.push(
    `${parseFloat(String(data.runtime * 1000)).toFixed(0)} ms`
  );
}

```

```

// Function to save and download the formatted solution data
✓ const saveAndDownloadSolution = (data: any): void => {
  // Check if there is no result to save and download, show error and return
✓ if (!data.result) {
    toast.error("No result to save and download");
    return;
  }

  // Format the solution data
  const formattedData = formatSolution(data);

  // Prompt the user to enter a filename, default to "solution" if not provided
✓ const fileName = prompt(
    "Enter a filename:",
    data.result.string + " solution" || "solution"
  );

  // Check if the user canceled the prompt, return in that case
✓ if (!fileName) {
    return;
  }

  // Create a Blob with the formatted data and set the MIME type to text/plain
  const blob = new Blob([formattedData], { type: "text/plain" });

  // Create a download link
  const url = URL.createObjectURL(blob);
  const a = document.createElement("a");

  a.href = url;
  a.download = `${fileName}.txt`; // Use the entered filename with a .txt extension

  // Append the link to the document body, trigger a click, and remove the link
  document.body.appendChild(a);
  a.click();
  document.body.removeChild(a);

  // Revoke the object URL to free up resources
  URL.revokeObjectURL(url);
}

export { saveAndDownloadSolution };

```

Server Side

```
...
from fastapi import FastAPI, HTTPException
from api.main import BreachProtocolSolver
from typing import List
from pydantic import BaseModel
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()
solver = BreachProtocolSolver()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"], # Replace with your frontend URL in production
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/api/python")
def hello_world():
    return {"message": "Hello World"}

...

class Target(BaseModel):
    sequence: List[str]
    points: int

...

class BreachProtocolInput(BaseModel):
    matrix: List[List[str]]
    targets: List[Target]
    buffer: int
    rowCount: int
    colCount: int

@app.post("/api/breach_protocol_solve")
def solve(data: BreachProtocolInput):
    try:
        result = solver.breach_protocol_solve(data.matrix, data.targets, data.buffer, data.rowCount, data.colCount)
        return {"result": result["results"], "runtime": result["runtime"]}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

```

...
import time
...
class BreachProtocolSolver:
    # The working principle involves BFS using a stack, where registers are pushed at the same buffer level and popped upon decrementing the buffer.
    # The avoidance of recursion is due to concerns about the potential heaviness for large buffer and matrix sizes.
    def get_sequences_candidate(self, buffer, axis, current_position, col_matrix, row_matrix):
        # Initialize push stack and result candidate
        stack = [(buffer, axis, current_position, ())]
        result = []
        while stack:
            # Pop to get the data
            buffer, axis, current_position, sequence = stack.pop()
            # Generate possible next moves
            next_moves = [(i + 1, current_position) if axis == 'x' else (current_position, i + 1) for i in range(col_matrix if axis == 'x' else row_matrix)]
            # Last buffer attempt
            if buffer == 1:
                result.extend([sequence + ((x, y),) for x, y in next_moves if (x, y) not in sequence])
            else:
                for x, y in next_moves:
                    # Prevent push itself
                    if (x, y) not in sequence:
                        stack.append((buffer - 1, ('y' if axis == 'x' else 'x'), (x if axis == 'x' else y), sequence + ((x, y),)))

        return result

    def evaluate(self, seqs, matrix, targets):
        # Initialize object to minimize calculation
        strings = [{'index': i, 'string': ''.join(matrix[y-1][x-1] for x, y in seq)} for i, seq in enumerate(seqs)]

        # Create a set of unique strings
        ustrings = set(entry['string'] for entry in strings)

        # Create a mapping between unique strings and their original indices
        ustrings_indices = {entry['string']: entry['index'] for entry in strings}

        # Join the sequence become a string
        target_strings = [''.join(target.sequence) for target in targets]

        # Initialize max_score, results, and full_score
        max_score = float('-inf')
        result = []
        full_score = sum(targets[i].points for i in range(len(targets)))
        found = False

```

```

def evaluate(self, seqs, matrix, targets):
    # Initialize object to minimize calculation
    strings = [{'index': i, 'string': ''.join(matrix[y-1][x-1] for x, y in seq)} for i, seq in enumerate(seqs)]

    # Create a set of unique strings
    ustrings = set(entry['string'] for entry in strings)

    # Create a mapping between unique strings and their original indices
    ustrings_indices = {entry['string']: entry['index'] for entry in strings}

    # Join the sequence become a string
    target_strings = [''.join(target.sequence) for target in targets]

    # Initialize max_score, results, and full_score
    max_score = float('-inf')
    result = []
    full_score = sum(targets[i].points for i in range(len(targets)))
    found = False

    # Evaluate only set string
    for ustring in ustrings:
        string_index = ustrings_indices[ustring]
        score = 0
        seq_length = 0
        for i, target_string in enumerate(target_strings):
            # If found
            location = ustring.find(target_string)
            if location > 0 or ustring == target_string:
                score += targets[i].points
                end_location = location + len(target_string)
                seq_length = max(seq_length, end_location)
                if not found:
                    found = True
                max_score = max(score, max_score)
                result.append({'stringIndex': string_index, 'seqLength': seq_length, 'score': score, 'string': ustring})

        if score == full_score and full_score > 0:
            return [{'seq': seqs[string_index], 'score': score, 'string': ustring}]

    if not found:
        return [{'seq': (), 'score': -1, "string": ""}]

    # Evaluate in an array the sequence with max score and min sequence length
    with_max_scores = [entry for entry in result if entry['score'] == max_score]
    min_seq_length = min(entry['seqLength'] for entry in with_max_scores)
    finals = [entry for entry in with_max_scores if entry['seqLength'] == min_seq_length]

    if min_seq_length > 1:
        min_seq_length = int(min_seq_length / 2)

        seen_seqs = set()
        unique_pre_chosen = []

        for entry in (
            {'seq': seqs[entry['stringIndex']][min_seq_length:], 'score': entry["score"], "string": entry["string"]}
            for entry in finals
        ):
            current_seq = tuple(entry['seq'])
            if current_seq not in seen_seqs:
                seen_seqs.add(current_seq)
                unique_pre_chosen.append(entry)

        return unique_pre_chosen
    return finals

```



```

def mini_case_evaluate(self, seqs, matrix, targets):
    # Extract strings from the matrix based on provided sequences
    strings = [''.join(matrix[y-1][x-1] for x, y in seq) for seq in seqs]

    # Extract target strings from the targets
    target_strings = [''.join(target.sequence) for target in targets]

    # Initialize variables to track maximum score and result
    max_score = float("-inf")
    result = []

    # Calculate the full score based on the points of each target
    full_score = sum(targets[i].points for i in range(len(targets)))

    # Flag to check if any valid sequence is found
    found = False

    # Loop through each string extracted from the matrix
    for string_index, string_value in enumerate(strings):
        score = 0
        seq_length = 0

        # Iterate through target strings and check for matches
        for i, ts in enumerate(target_strings):
            location = string_value.find(ts)
            if location > 0 or string_value == ts:
                # Update score and sequence length
                score += targets[i].points
                end_location = location + len(ts)
                seq_length = max(seq_length, end_location)

            if not found:
                found = True

        # Update maximum score and append result
        max_score = max(score, max_score)
        result.append({'score': score, 'stringIndex': string_index, 'seqLength': seq_length, 'score': score, 'string': string_value})

    # If the score matches the full score and full score is greater than 0, return the result
    if score == full_score and full_score > 0:
        return [{'seq': seqs[string_index], 'score': score, 'string': string_value}]

    # If no valid sequence is found, return a default result
    if not found:
        return [{'seq': (), 'score': -1, "string": ""}]

# Filter entries with maximum scores
with_max_scores = [entry for entry in result if entry['score'] == max_score]

# Find the minimum sequence length among entries with maximum scores
min_seq_length = min(entry['seqLength'] for entry in with_max_scores)

# Filter entries with minimum sequence length
finals = [entry for entry in with_max_scores if entry['seqLength'] == min_seq_length]

# If minimum sequence length is greater than 1, reduce it to half
if min_seq_length > 1:
    min_seq_length = int(min_seq_length / 2)
    seen_seqs = set()
    unique_pre_chosen = []

    # Iterate through final entries and filter unique pre-chosen entries
    for entry in (
        {'seq': seqs[entry['stringIndex']][:min_seq_length], 'score': entry["score"], "string": entry["string"]}
        for entry in finals
    ):
        current_seq = tuple(entry['seq'])
        if current_seq not in seen_seqs:
            seen_seqs.add(current_seq)
            unique_pre_chosen.append(entry)

    # Return unique pre-chosen entries
    return unique_pre_chosen

# Return final entries
return finals

def breach_protocol_solve(self, matrix, targets, total_buffer_size, row_matrix, col_matrix):
    try:
        start_time = time.time() # Record the start time
        sequences = self.get_sequences_candidate(total_buffer_size, 'x', 1, col_matrix, row_matrix)
        results = self.evaluate(sequences, matrix, targets) if total_buffer_size >= 8 or row_matrix >= 7 or col_matrix >= 7 else self.mini_case_evaluate(sequences, matrix, targets)
        end_time = time.time() # Record the end time
        runtime = end_time - start_time # Calculate the runtime
        return ('results': results[0] if (len(results) > 0) else [], 'runtime': runtime)
    except Exception as e:
        print(f"An error occurred: {e}")
        return {'error': str(e)}

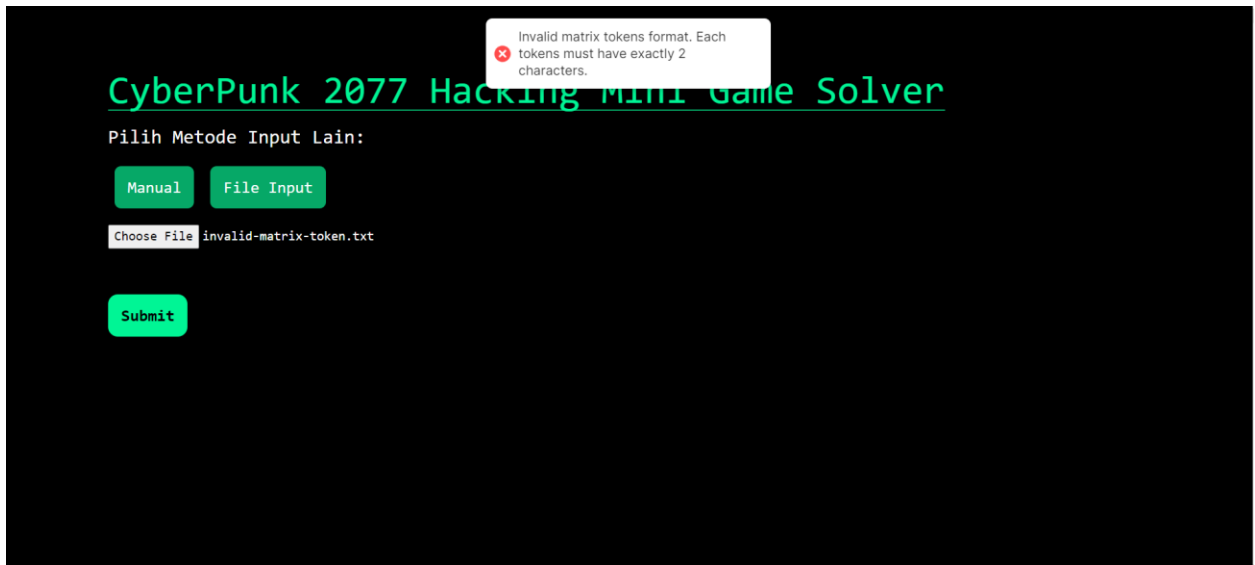
```

BAB IV

Eksperimen

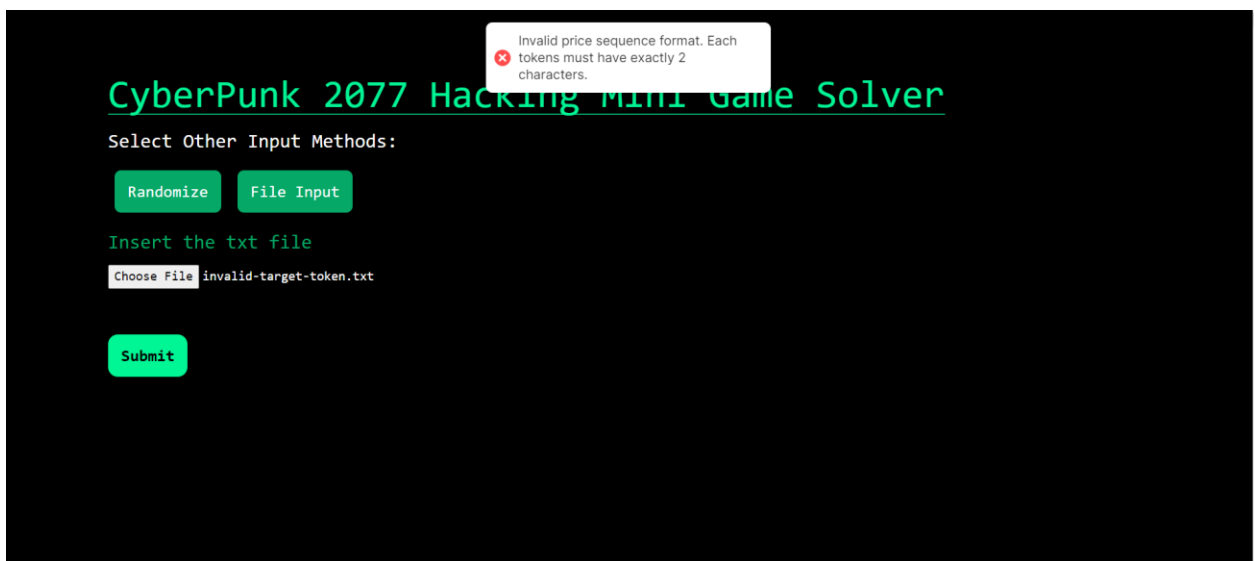
Berikut adalah beberapa contoh output program:

A. Masukan *Token Matrix* Tidak Valid



The screenshot shows the 'CyberPunk 2077 Hacking Mini Game Solver' interface. At the top, there is a red error message box that reads: 'Invalid matrix tokens format. Each tokens must have exactly 2 characters.' Below this, the text 'Pilih Metode Input Lain:' is displayed. There are two buttons: 'Manual' and 'File Input'. Under 'File Input', there is a 'Choose File' button and a text input field containing 'invalid-matrix-token.txt'. At the bottom, there is a 'Submit' button.

B. Masukkan *Token Target* Tidak Valid



The screenshot shows the 'CyberPunk 2077 Hacking Mini Game Solver' interface. At the top, there is a red error message box that reads: 'Invalid price sequence format. Each tokens must have exactly 2 characters.' Below this, the text 'Select Other Input Methods:' is displayed. There are two buttons: 'Randomize' and 'File Input'. Under 'File Input', there is a 'Choose File' button and a text input field containing 'invalid-target-token.txt'. At the bottom, there is a 'Submit' button.

C. Masukan Matrix M x N

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

RandomizeFile Input

Insert the txt file

Choose File 6x5-mxn.txt

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

How the step to get the optimal answer?

Step 1: Start on (6, 1)
Step 2: Move to (6, 4)
Step 3: Move to (3, 4)
Step 4: Move to (3, 5)
Step 5: Move to (6, 5)
Step 6: Move to (6, 3)
Step 7: Move to (1, 3)

Points: 50
Runtime: 88.6 ms

Buffer Size: 7
Target 1: BD 7A BD - 20 points
Target 2: BD 1C BD 55 - 30 points

SubmitSimpan solusi dan Download

D. Masukan Tidak Punya Solusi

✔ Successfull

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

RandomizeFile Input

Insert the txt file

Choose File no-answer.txt

F1	P0	KK	I9
VA	AG	R7	TI
HX	H7	BQ	63

No answer sequence to get the prize
Runtime: 0.2 ms

Buffer Size: 4
Target 1: F1 KK - 20 points
Target 2: I9 TI P0 TI AG I9 - 30 points

SubmitSimpan solusi dan Download

E. Masukan Kasus Test Case Best Case

Maksud *best case* disini adalah *case* ketika *sequence* mencapai full score dari *target*, sehingga tidak perlu cek *case* hingga akhir.

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

Randomize File Input

Insert the txt file

Choose File tc-bestcase.txt

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

How the step to get the optimal answer?

Step 1: Start on (6, 1)
Step 2: Move to (6, 4)
Step 3: Move to (3, 4)
Step 4: Move to (3, 5)
Step 5: Move to (6, 5)
Step 6: Move to (6, 3)
Step 7: Move to (1, 3)

Points: 50
Runtime: 157.9 ms

Buffer Size: 7
Target 1: BD 7A BD - 20 points
Target 2: BD 1C BD 55 - 30 points

Submit Simpan solusi dan Download

F. Masukan Kasus Test Case Worst Case

Maksud *worst case* disini adalah *case* ketika perlu mengecek dan compare semua nilai kandidat *sequence* karena full score tidak bisa dicapai.

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

Randomize File Input

Insert the txt file

Choose File tc-worst-case.txt

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

How the step to get the optimal answer?

Step 1: Start on (6, 1)
Step 2: Move to (6, 4)
Step 3: Move to (3, 4)
Step 4: Move to (3, 5)
Step 5: Move to (6, 5)
Step 6: Move to (6, 3)
Step 7: Move to (1, 3)

Points: 50
Runtime: 261.3 ms

Buffer Size: 7
Target 1: BD E9 1C - 15 points
Target 2: BD 7A BD - 20 points
Target 3: BD 1C BD 55 - 30 points

Submit Simpan solusi dan Download

G. Masukan dengan *Randomize* dan Tidak Valid

Matrix height must be at least 2.

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

Randomize File Input

Insert height matrix Insert width matrix Insert buffer size

-8 0 0

Randomize Matrix

H. Masukan dengan *Randomize* tetapi Tidak Memiliki Solusi

Success

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

Randomize File Input

Insert height matrix Insert width matrix Insert buffer size

7 7 6

Randomize Matrix

Insert target reward count

2 Randomize Target

8R	BR	PM	TH	SR	JS
AP	KQ	QS	LR	PA	LN
A7	5X	O9	D3	VI	75
LT	65	X7	7A	OY	9E

No answer sequence to get the prize
Runtime: 8.9 ms

Buffer Size: 6
Target 1: SR O9 5X - 81 points
Target 2: D3 8R 75 LR 7A PM - 47 points

Submit Simpan solusi dan Download

I. Masukan dengan *Randomize* dan Memiliki Solusi

CyberPunk 2077 Hacking Mini Game Solver

Select Other Input Methods:

Randomize File Input

Insert height matrix 7 Insert width matrix 7 Insert buffer size 6

Randomize Matrix

Insert target reward count 2 Randomize Target

8R	BR	PM	TH	SR	JS
AP	KQ	QS	LR	PA	LN
A7	5X	09	D3	VI	75
LT	65	X7	7A	OY	9E

How the step to get the optimal answer?

Step 1: Start on (3, 1)

Step 2: Move to (3, 4)

Step 3: Move to (5, 4)

Points: 21

Runtime: 12.0 ms

Buffer Size: 6

Target 1: LN TH - 8 points

Target 2: X7 OY - 21 points

Submit Simpan solusi dan Download

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membacamasukan 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	✓	

LAMPIRAN

A. *Worst case*

1. Optimize Version

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python optimize.py
{'seq': ((1, 1), (1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (4, 3)), 'matchedIndices': [1, 2], 'score': 50}
Sequence: ((1, 1), (1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (4, 3)), Points: 50
Step 1: Matrix[0][0] = 7A
Step 2: Matrix[3][0] = BD
Step 3: Matrix[3][2] = 7A
Step 4: Matrix[4][2] = BD
Step 5: Matrix[4][5] = 1C
Step 6: Matrix[2][5] = BD
Step 7: Matrix[2][3] = 55
Runtime: 0.1769099235534668 seconds
```

2. *Stack* dan Generate Semua *Sequence* Sebelum Checking

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python stack_all.py
Sequence: ((6, 1), (6, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3)), Points: 50
Step 1: Matrix[0][5] = 55
Step 2: Matrix[3][5] = BD
Step 3: Matrix[3][2] = 7A
Step 4: Matrix[4][2] = BD
Step 5: Matrix[4][5] = 1C
Step 6: Matrix[2][5] = BD
Step 7: Matrix[2][0] = 55
Runtime: 0.18889975547790527 seconds
```

3. *Stack* dan Checking Sembari Generate *Sequence* Satu per Satu

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python stack_single.py
Sequence: ((6, 1), (6, 4), (3, 4), (3, 5), (6, 5), (6, 3), (4, 3)), Points: 50
Step 1: Matrix[0][5] = 55
Step 2: Matrix[3][5] = BD
Step 3: Matrix[3][2] = 7A
Step 4: Matrix[4][2] = BD
Step 5: Matrix[4][5] = 1C
Step 6: Matrix[2][5] = BD
Step 7: Matrix[2][3] = 55
Runtime: 1.2576823234558105 seconds
```

4. Rekursi dan Generate Semua *Sequence* Sebelum Checking

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python recursive_all.py
Sequence: [(1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3)], Points: 50
Step 1: Matrix[3][0] = BD
Step 2: Matrix[3][2] = 7A
Step 3: Matrix[4][2] = BD
Step 4: Matrix[4][5] = 1C
Step 5: Matrix[2][5] = BD
Step 6: Matrix[2][0] = 55
Runtime: 0.6498432159423828 seconds
```

5. Rekursi dan Checking Sembari Generate *Sequence* Satu per Satu

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python recursive_single.py
Sequence: [(1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3), (1, 1)], Points: 50
Step 1: Matrix[3][0] = BD
Step 2: Matrix[3][2] = 7A
Step 3: Matrix[4][2] = BD
Step 4: Matrix[4][5] = 1C
Step 5: Matrix[2][5] = BD
Step 6: Matrix[2][0] = 55
Step 7: Matrix[0][0] = 7A
Runtime: 0.6449310779571533 seconds
```

B. *Best case*:

1. Optimize Version

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python optimize.py
Sequence: ((6, 1), (6, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3)), Points: 50
Step 1: Matrix[0][5] = 55
Step 2: Matrix[3][5] = BD
Step 3: Matrix[3][2] = 7A
Step 4: Matrix[4][2] = BD
Step 5: Matrix[4][5] = 1C
Step 6: Matrix[2][5] = BD
Step 7: Matrix[2][0] = 55
Runtime: 0.11004281044006348 seconds
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python stack_single.py
```


2. *Stack* dan Generate Semua *Sequence* Sebelum Checking

```
Runtime: 0.11369967460632324 seconds
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python stack_all.py
Sequence: ((6, 1), (6, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3)), Points: 50
Step 1: Matrix[0][5] = 55
Step 2: Matrix[3][5] = BD
Step 3: Matrix[3][2] = 7A
Step 4: Matrix[4][2] = BD
Step 5: Matrix[4][5] = 1C
Step 6: Matrix[2][5] = BD
Step 7: Matrix[2][0] = 55
Runtime: 0.11369967460632324 seconds
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python optimize.py
```

3. *Stack* dan Checking Sembari Generate *Sequence* Satu per Satu

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python stack_single.py
Sequence: ((6, 1), (6, 4), (3, 4), (3, 5), (6, 5), (6, 3), (4, 3)), Points: 50
Step 1: Matrix[0][5] = 55
Step 2: Matrix[3][5] = BD
Step 3: Matrix[3][2] = 7A
Step 4: Matrix[4][2] = BD
Step 5: Matrix[4][5] = 1C
Step 6: Matrix[2][5] = BD
Step 7: Matrix[2][3] = 55
Runtime: 0.08188819885253906 seconds
```

4. Rekursi dan Generate Semua *Sequence* Sebelum Checking

```
Runtime: 0.5799953937530518 seconds
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python recursive_all.py
Sequence: [(1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3), (1, 1)], Points: 50
Step 1: Matrix[3][0] = BD
Step 2: Matrix[3][2] = 7A
Step 3: Matrix[4][2] = BD
Step 4: Matrix[4][5] = 1C
Step 5: Matrix[2][5] = BD
Step 6: Matrix[2][0] = 55
Step 7: Matrix[0][0] = 7A
Runtime: 0.5799953937530518 seconds
```

5. Rekursi dan Checking Sembari Generate *Sequence* Satu per Satu

```
PS D:\Kuliah\Tubes-1-Stima\Tubes-1-Stima\bin> python recursive_single.py
Sequence: [(1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3), (1, 1)], Points: 50
Step 1: Matrix[3][0] = BD
Step 2: Matrix[3][2] = 7A
Step 3: Matrix[4][2] = BD
Step 4: Matrix[4][5] = 1C
Step 5: Matrix[2][5] = BD
Step 6: Matrix[2][0] = 55
Step 7: Matrix[0][0] = 7A
Runtime: 0.5506348609924316 seconds
```