

**LAPORAN TUGAS KECIL I**  
**IF2211 STRATEGI ALGORITMA**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

Adril Putra Merin 13522068

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2024**

## Daftar Isi

BAB I ALGORITMA BRUTE FORCE.....	3
BAB II SOURCE CODE.....	4
BAB III HASIL PROGRAM.....	15
LAMPIRAN .....	23

## BAB I

### ALGORITMA BRUTE FORCE

Algoritma *brute force* adalah algoritma yang memiliki pendekatan sederhana, tetapi cukup mampu untuk menyelesaikan berbagai persoalan. Sesuai dengan namanya, algoritma *brute force* atau, diterjemahkan secara bebas, *tenaga kasar*, menyelesaikan persoalan dengan mencoba semua kemungkinan solusi hingga ditemukan solusi yang benar atau optimal. Secara umum, algoritma *brute force* tidak selalu menjadi pilihan yang efisien karena cenderung membutuhkan waktu yang sangat besar seiring meningkatnya ukuran ruang solusi. Karena itu, algoritma yang lebih cerdas dan efisien seringkali lebih dipilih untuk menyelesaikan persoalan tertentu.

Dalam penyelesaian Cyberpunk 2077 Breach Protocol, penulis menggunakan algoritma *brute force* dengan metode rekursif untuk mencari setiap kemungkinan buffer hingga ditemukan solusi yang optimal. Adapun langkah-langkah algoritma *brute force* pada program ini adalah sebagai berikut.

1. Buat sebuah matriks *boolean visited* dengan nilai awal false yang menandakan setiap elemen pada matriks belum dikunjungi sebelumnya. Selanjutnya, tentukan setiap token pada baris pertama (secara horizontal), pilih token tersebut sebagai titik awal pencarian dan jalankan fungsi rekursif secara vertikal.
2. Pada fungsi rekursi tersebut, ubah nilai matriks *visited* pada elemen tersebut menjadi *true* yang menandakan bahwa elemen tersebut sudah dikunjungi. Selanjutnya, periksa *reward* saat ini. Jika *reward* saat ini lebih besar daripada *reward* optimal sebelumnya, ubah nilai *reward* optimal menjadi nilai *reward* saat ini dan simpan *buffer* saat ini menjadi *buffer* optimal. Jika *reward* saat ini sama dengan *reward* optimal sebelumnya, tetapi panjang *buffer* saat ini lebih pendek dari pada *buffer* optimal sebelumnya, ubah *buffer* optimal sebelumnya menjadi *buffer* saat ini.
3. Jika buffer belum penuh, lakukan pencarian secara rekursif terhadap arah berbeda dari pencarian sebelumnya. Misalnya, jika pencarian sebelumnya adalah horizontal, maka lakukan pencarian secara vertikal.
4. Pada akhir fungsi rekursi, ubah nilai matriks *visited* pada elemen saat ini menjadi *false* yang menandakan bahwa pencarian melalui titik ini sudah selesai dilakukan.

Kita akan menganalisis kompleksitas waktu dari algoritma ini menggunakan notasi *big-O*. Definisikan  $T(N, M, B, K, S)$  sebagai banyak operasi yang dilakukan oleh algoritma ini dengan  $N$  adalah panjang baris,  $M$  adalah panjang kolom, dan  $B$  adalah panjang *buffer*,  $K$  adalah banyak sekuens, dan  $S$  adalah panjang sekuens maksimum. Misalkan  $F(N, M, B)$  adalah banyak kemungkinan *buffer* yang dibentuk pada matriks dengan panjang baris  $N$ , panjang kolom  $M$ , dan panjang *buffer*  $B$ . Untuk setiap kemungkinan *buffer* yang dibentuk, kita akan melakukan pengecekan string token (*string matching*). Jadi, banyak operasi dapat dituliskan sebagai

$$T(N, M, B, K, S) \leq F(N, M, B) \times BKS$$

Perhatikan bahwa,  $F(N, M, B) < N^{B/2} M^{B/2}$  sehingga kompleksitas waktu dari algoritma ini adalah

$$T(N, M, B, K, S) = O(N^{B/2} M^{B/2} \times BKS) = O(\max(N, M)^B \times BKS)$$

## BAB II

### SOURCE CODE

Program ini diimplementasikan menggunakan bahasa typescript dengan menggunakan framework Next.JS. Karena *source code* pada program ini cukup panjang, penulis hanya akan menyajikan bagian program utama. Adapun fungsi-fungsi yang diimplementasikan pada program utama adalah sebagai berikut.

#### 1. Fungsi *runBruteForce*

Fungsi ini adalah fungsi utama yang menjalankan algoritma bruteforce. Awalnya, fungsi ini menerima beberapa input seperti ukuran buffer, matriks token, list sekuens, dan list rewards. Fungsi rekursif diimplementasikan pada fungsi *searchOptimalValue* yang terletak di dalam fungsi ini.

```
export default function runBruteForce(data: {
  bufferSize: number;
  matrix: string[][];
  sequences: string[];
  rewards: number[];
}) {
  // variable declaration
  const { bufferSize, matrix, sequences, rewards } = data;
  const vis = Array.from(Array(matrix.length), () =>
    Array(matrix[0].length).fill(false)
  );
  let maxPoint = 0;
  let maxCoordinate: { x: number; y: number }[] = [];
  let isSolutionFound: boolean = false;

  // function declaration
  function searchOptimalValue(
    row: number,
    col: number,
    coordinates: { x: number; y: number }[],
    direction: boolean,
    token: string,
    steps: number
  ) {
    // init
    coordinates.push({ x: row, y: col });
    vis[row][col] = true;
```

```

    // check current point
    let currPoint = getPoint(token, sequences, rewards);
    if (currPoint === maxPoint && maxCoordinate.length >
coordinates.length) {
        maxCoordinate = [...coordinates];
    } else if (currPoint > maxPoint) {
        maxPoint = currPoint;
        maxCoordinate = [...coordinates];
    }

    // check if steps is equal to buffer size
    if (steps < bufferSize) {
        if (direction) {
            for (let i = 0; i < matrix.length; i++) {
                if (!vis[i][col]) {
                    searchOptimalValue(
                        i,
                        col,
                        coordinates,
                        !direction,
                        token + " " + matrix[i][col],
                        steps + 1
                    );
                }
            }
        } else {
            for (let i = 0; i < matrix[0].length; i++) {
                if (!vis[row][i]) {
                    searchOptimalValue(
                        row,
                        i,
                        coordinates,
                        !direction,
                        token + " " + matrix[row][i],
                        steps + 1
                    );
                }
            }
        }
    }
}

```

```

    }
  }

  // termination
  coordinates.pop();
  vis[row][col] = false;
  return;
}

const startTime = performance.now();

if (bufferSize > 0 && sequences.length > 0) {
  for (let i = 0; i < matrix[0].length; i++) {
    searchOptimalValue(0, i, [], true, matrix[0][i], 1);
  }
}

const endTime = performance.now();
const runTime = endTime - startTime;

return { maxPoint, maxCoordinate, runTime };
}

```

## 2. Fungsi *readRawData*

Fungsi ini membaca *raw data* yang diterima dari *front-end* dan mengembalikan data yang kemudian diolah oleh fungsi lainnya seperti *runBruteForce*.

```

export function readRawData(rawData: string) {
  try {
    const arr = rawData.split("\r");
    const filteredArr = arr.map((val) => val.replace(/\n/g, ""));

    const bufferSize: number = parseInt(filteredArr[0]);
    const matrixSize = filteredArr[1].split(/\s+/);
    const col = parseInt(matrixSize[0]);
    const row = parseInt(matrixSize[1]);

    const matrix = [];
    const sequences = [];
  }
}

```

```

const rewards = [];
let currIdx: number = 2;

for (let i = 0; i < row; i++) {
    const currRow = filteredArr[currIdx].trim().split(/\s+/);
    matrix.push(currRow);
    currIdx++;
}

const numberOfSequence = parseInt(filteredArr[currIdx]);
currIdx++;

for (let i = 0; i < numberOfSequence; i++) {
    // make sure that there are no more than one whitespace
    const currSequence = filteredArr[currIdx].trim().split(/\s/g);
    const str = currSequence.join(" ");
    sequences.push(str);

    currIdx++;

    const reward = parseInt(filteredArr[currIdx]);
    rewards.push(reward);
    currIdx++;
}

return {
    bufferSize,
    matrix,
    sequences,
    rewards,
};
} catch (error) {
    return null;
}
}

```

### 3. Fungsi *random*

Fungsi ini adalah fungsi pembantu (*utility function*) yang menerima input berupa *lowerbound* dan *upperbound*. Fungsi ini akan mengembalikan sebuah angka acak yang berada di antara *lowerbound* dan *upperbound* secara inklusif.

```
export function random(lowerbound: number, upperbound: number) {  
  return Math.floor(Math.random() * (upperbound - lowerbound)) +  
  lowerbound;  
}
```

#### 4. Fungsi *getPoint*

Fungsi ini mengembalikan banyak *reward* yang diperoleh berdasarkan *buffer* saat ini dan sekuens yang ada. Fungsi ini melakukan *string matching* untuk mendapatkan *reward* yang diterima.

```
export function getPoint(  
  token: string,  
  sequences: string[],  
  rewards: number[]  
) {  
  let points = 0;  
  for (let i = 0; i < sequences.length; i++) {  
    if (token.includes(sequences[i])) {  
      points += rewards[i];  
    }  
  }  
  
  return points;  
}
```

#### 5. Fungsi *getResultFromFile* dan *getRandomResult*

Fungsi-fungsi ini adalah fungsi yang bertujuan untuk menangani *request* dari *client* berdasarkan jenis *input* yang diberikan. Fungsi *getResultFromFile* menangani *input* dari file, sedangkan Fungsi *getRandomResult* menangani *input* berupa batasan-batasan dari matriks dan sekuens acak.

```
"use server";  
  
import runBruteForce from "../bruteforce";  
import { readRawData } from "../util";  
import { random } from "../util";
```



```

export async function getResultFromFile(formData: FormData) {
  const file: File = formData.get("file") as File;
  const inputString: string = await file.text();
  const data = readRawData(inputString);

  if (!data) {
    return { errorMsg: "Wrong input format" };
  }

  const result = runBruteForce(data);

  const coordinates = result.maxCoordinate;

  const styleArr = Array.from(Array(data.matrix.length), () =>
    Array(data.matrix[0].length).fill(0)
  );

  const vertical = Array.from(Array(data.matrix.length), () =>
    Array(data.matrix[0].length).fill(false)
  );

  const horizontal = Array.from(Array(data.matrix.length), () =>
    Array(data.matrix[0].length).fill(false)
  );

  for (let i = 0; i < coordinates.length; i++) {
    styleArr[coordinates[i].x][coordinates[i].y] = 1;
    if (!i) continue;

    if (i % 2 !== 0) {
      let maxIndex = Math.max(coordinates[i].x, coordinates[i - 1].x);
      let minIndex = Math.min(coordinates[i].x, coordinates[i - 1].x);

      for (let j = minIndex + 1; j < maxIndex; j++) {
        vertical[j][coordinates[i].y] = true;
      }

      if (maxIndex === minIndex + 1) {
        if (coordinates[i].x > coordinates[i - 1].x) {

```

```

        styleArr[coordinates[i].x][coordinates[i].y] = 2;
    } else {
        styleArr[coordinates[i].x][coordinates[i].y] = 3;
    }
}
} else {
    let maxIndex = Math.max(coordinates[i].y, coordinates[i - 1].y);
    let minIndex = Math.min(coordinates[i].y, coordinates[i - 1].y);

    for (let j = minIndex + 1; j < maxIndex; j++) {
        horizontal[coordinates[i].x][j] = true;
    }

    if (maxIndex === minIndex + 1) {
        if (coordinates[i].y > coordinates[i - 1].y) {
            styleArr[coordinates[i].x][coordinates[i].y] = 4;
        } else {
            styleArr[coordinates[i].x][coordinates[i].y] = 5;
        }
    }
}
}

const sequences: string[][] = [];

for (let i = 0; i < data.sequences.length; i++) {
    let temp: string[] = data.sequences[i].split(/\s+/);
    sequences.push(temp);
}

return {
    maxPoint: result.maxPoint,
    runTime: result.runTime,
    matrix: data.matrix,
    coordinates: coordinates,
    styleArr: styleArr,
    rewards: data.rewards,
    sequences,
    vertical,

```

```

    horizontal,
    errorMsg: "",
  };
}

export async function getRandomResult(formData: FormData) {
  const row = formData.get("row") as unknown as number;
  const col = formData.get("column") as unknown as number;
  const bufferSize = formData.get("buffer") as unknown as number;
  const numberOfSequence = formData.get("sequenceNumber") as unknown as
number;
  const maxSequenceLength = formData.get(
    "maxSequenceLength"
  ) as unknown as number;
  const tokenNumber = formData.get("tokenNumber") as unknown as number;
  const rawTokens = formData.get("tokens") as unknown as string;
  const tokens: string[] = rawTokens.trim().split(/\s+/);

  if (tokens.length !== tokenNumber) {
    return { errorMsg: "Token length is incorrect" };
  }

  // generate random sequences and rewards
  const sequences: string[] = [];
  const rewards: number[] = [];

  for (let i = 0; i < numberOfSequence; i++) {
    let isContinue = false;
    // prevent impossible permutation
    let cnt = 0;

    while (!isContinue && cnt < 1000) {
      // generate random sequence length
      const rdLen = random(2, maxSequenceLength);
      let temp = "";
      for (let j = 0; j < rdLen; j++) {
        // generate random token index
        const rdIndex = random(0, tokenNumber - 1);
        temp += tokens[rdIndex];
      }
    }
  }
}

```

```

        if (j !== rdLen - 1) {
            temp += " ";
        }
    }

    // if temp has been added before
    if (!sequences.includes(temp)) {
        isContinue = true;

        // generate random rewards
        const reward: number = random(0, 1000);

        rewards.push(reward);
        sequences.push(temp);
    }
    cnt++;
}

if (cnt === 1000) {
    // handle impossible permutation
    return { errorMsg: "Fail to randomize sequences" };
}

// generate random matrix content
const matrix: string[][] = [];

for (let i = 0; i < row; i++) {
    const temp = [];
    for (let j = 0; j < col; j++) {
        let rnIndex: number = random(0, tokenNumber - 1);
        temp.push(tokens[rnIndex]);
    }
    matrix.push(temp);
}

const data = {
    bufferSize,
    matrix,

```

```

    sequences,
    rewards,
  };

  const result = runBruteForce(data);

  const coordinates = result.maxCoordinate;
  const styleArr = Array.from(Array(data.matrix.length), () =>
    Array(data.matrix[0].length).fill(0)
  );

  const vertical = Array.from(Array(data.matrix.length), () =>
    Array(data.matrix[0].length).fill(false)
  );

  const horizontal = Array.from(Array(data.matrix.length), () =>
    Array(data.matrix[0].length).fill(false)
  );

  for (let i = 0; i < coordinates.length; i++) {
    styleArr[coordinates[i].x][coordinates[i].y] = 1;
    if (!i) continue;

    if (i % 2 !== 0) {
      let maxIndex = Math.max(coordinates[i].x, coordinates[i - 1].x);
      let minIndex = Math.min(coordinates[i].x, coordinates[i - 1].x);

      for (let j = minIndex + 1; j < maxIndex; j++) {
        vertical[j][coordinates[i].y] = true;
      }

      if (maxIndex === minIndex + 1) {
        if (coordinates[i].x > coordinates[i - 1].x) {
          styleArr[coordinates[i].x][coordinates[i].y] = 2;
        } else {
          styleArr[coordinates[i].x][coordinates[i].y] = 3;
        }
      }
    }
  }
} else {

```

```

    let maxIndex = Math.max(coordinates[i].y, coordinates[i - 1].y);
    let minIndex = Math.min(coordinates[i].y, coordinates[i - 1].y);

    for (let j = minIndex + 1; j < maxIndex; j++) {
        horizontal[coordinates[i].x][j] = true;
    }

    if (maxIndex === minIndex + 1) {
        if (coordinates[i].y > coordinates[i - 1].y) {
            styleArr[coordinates[i].x][coordinates[i].y] = 4;
        } else {
            styleArr[coordinates[i].x][coordinates[i].y] = 5;
        }
    }
}

const resultSequence: string[][] = [];
for (let i = 0; i < sequences.length; i++) {
    let temp: string[] = sequences[i].split(/\s+/);
    resultSequence.push(temp);
}

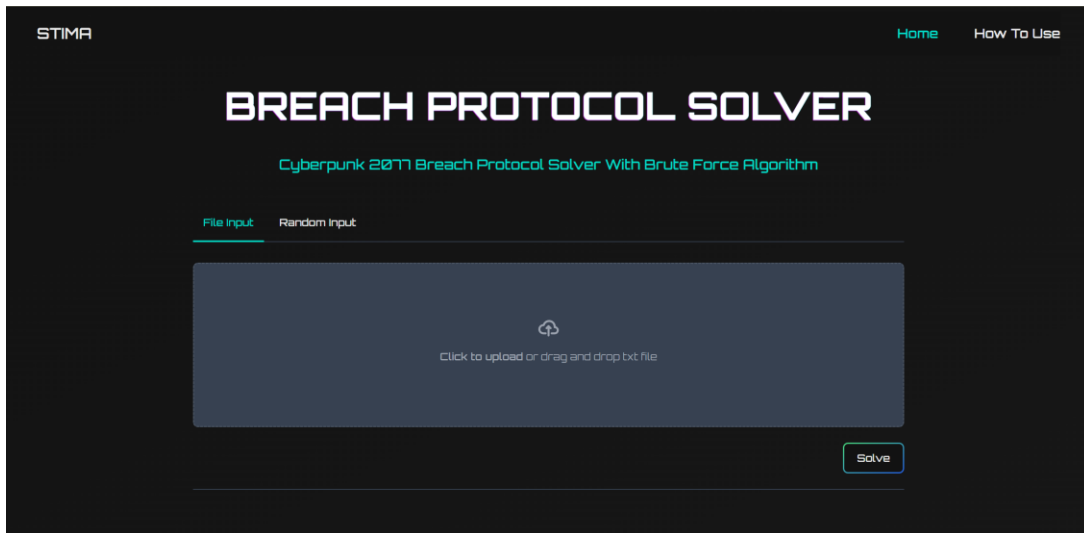
return {
    maxPoint: result.maxPoint,
    runTime: result.runTime,
    matrix: data.matrix,
    coordinates: coordinates,
    styleArr: styleArr,
    sequences: resultSequence,
    rewards: data.rewards,
    horizontal,
    vertical,
    errorMsg: "",
};
}

```

## BAB III

### HASIL PROGRAM

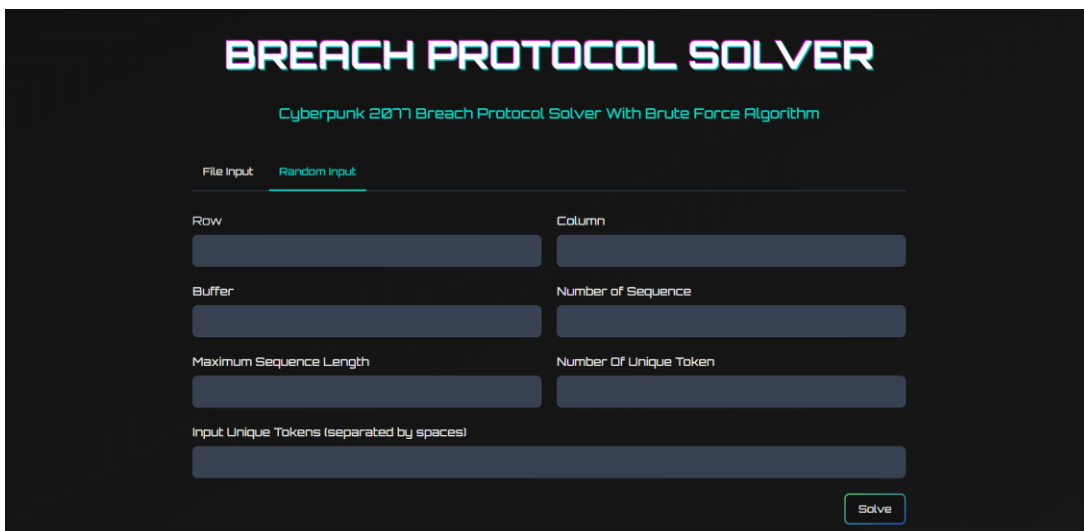
#### 1. Tampilan Masukan File



The screenshot shows the 'STIMA' logo in the top left corner and 'Home' and 'How To Use' links in the top right corner. The main title is 'BREACH PROTOCOL SOLVER' in large, bold, white letters. Below it, a subtitle reads 'Cyberpunk 2077 Breach Protocol Solver With Brute Force Algorithm'. There are two tabs: 'File Input' (selected) and 'Random Input'. The 'File Input' tab contains a large dark gray box with a cloud upload icon and the text 'Click to upload or drag and drop txt file'. A 'Solve' button is located at the bottom right of the interface.

**Gambar 3.1** Tampilan Masukan dari *File*

#### 2. Tampilan Masukan Acak



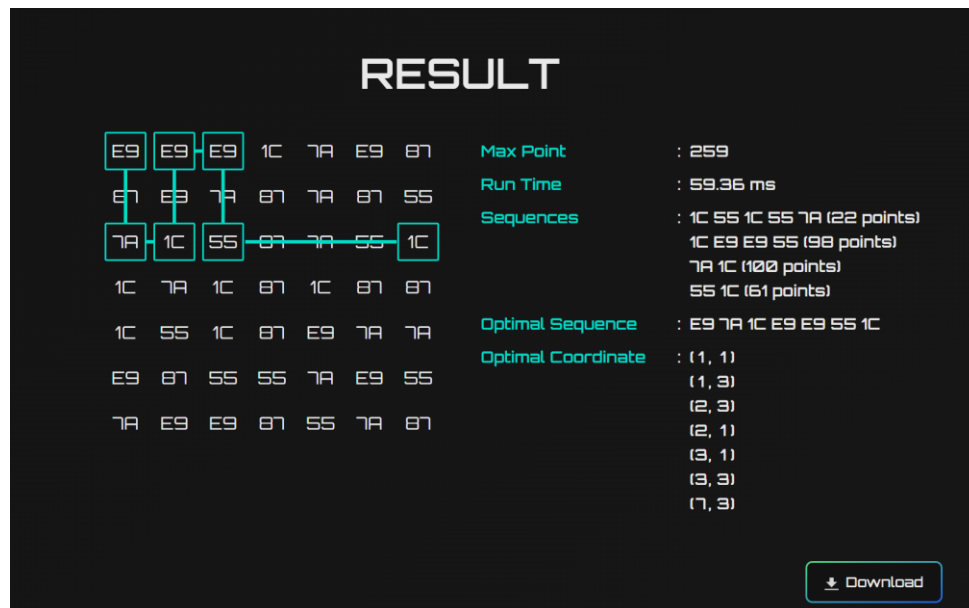
The screenshot shows the same header and title as Gambar 3.1. The 'Random Input' tab is selected. The interface contains several input fields: 'Row' and 'Column' (top row), 'Buffer' and 'Number of Sequence' (second row), 'Maximum Sequence Length' and 'Number Of Unique Token' (third row), and a single wide field for 'Input Unique Tokens (separated by spaces)' (bottom row). A 'Solve' button is located at the bottom right of the interface.

**Gambar 3.2** Tampilan Masukan Acak

### 3. Tes Pertama (File)

```
test > input > 1.txt
1 7
2 7 7
3 E9 E9 E9 1C 7A E9 87
4 87 E9 7A 87 7A 87 55
5 7A 1C 55 87 7A 55 1C
6 1C 7A 1C 87 1C 87 87
7 1C 55 1C 87 E9 7A 7A
8 E9 87 55 55 7A E9 55
9 7A E9 E9 87 55 7A 87
10 4
11 1C 55 1C 55 7A
12 22
13 1C E9 E9 55
14 98
15 7A 1C
16 100
17 55 1C
18 61
```

Gambar 3.3.1 Masukan tes pertama



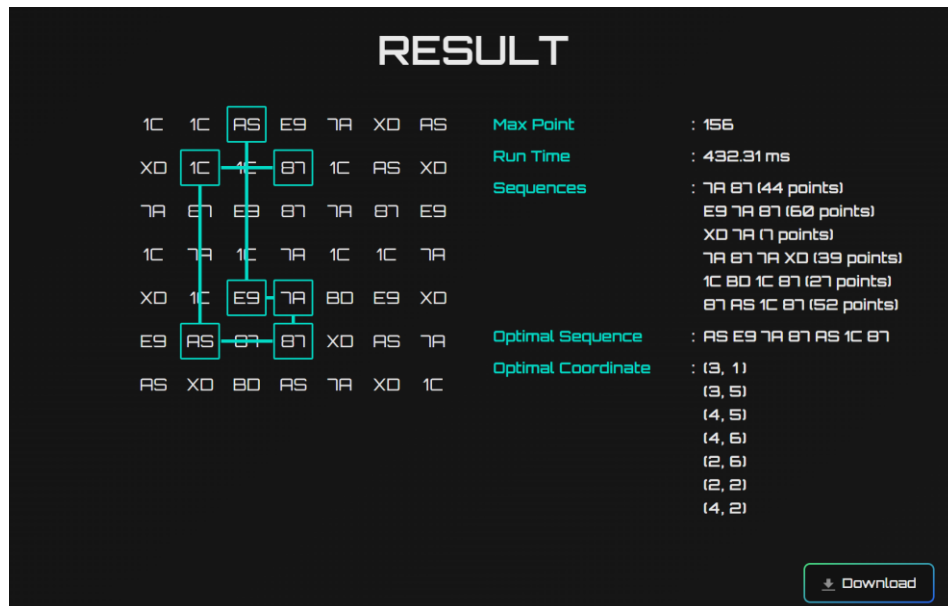
Gambar 3.3.2 Keluaran tes pertama



#### 4. Tes Kedua (File)

```
test > input > 6.txt
1 8
2 7 7
3 1C 1C AS E9 7A XD AS
4 XD 1C 1C 87 1C AS XD
5 7A 87 E9 87 7A 87 E9
6 1C 7A 1C 7A 1C 1C 7A
7 XD 1C E9 7A BD E9 XD
8 E9 AS 87 87 XD AS 7A
9 AS XD BD AS 7A XD 1C
10 6
11 7A 87
12 44
13 E9 7A 87
14 60
15 XD 7A
16 7
17 7A 87 7A XD
18 39
19 1C BD 1C 87
20 27
21 87 AS 1C 87
22 52
```

Gambar 3.4.1 Masukan tes kedua



Gambar 3.4.2 Keluaran tes kedua

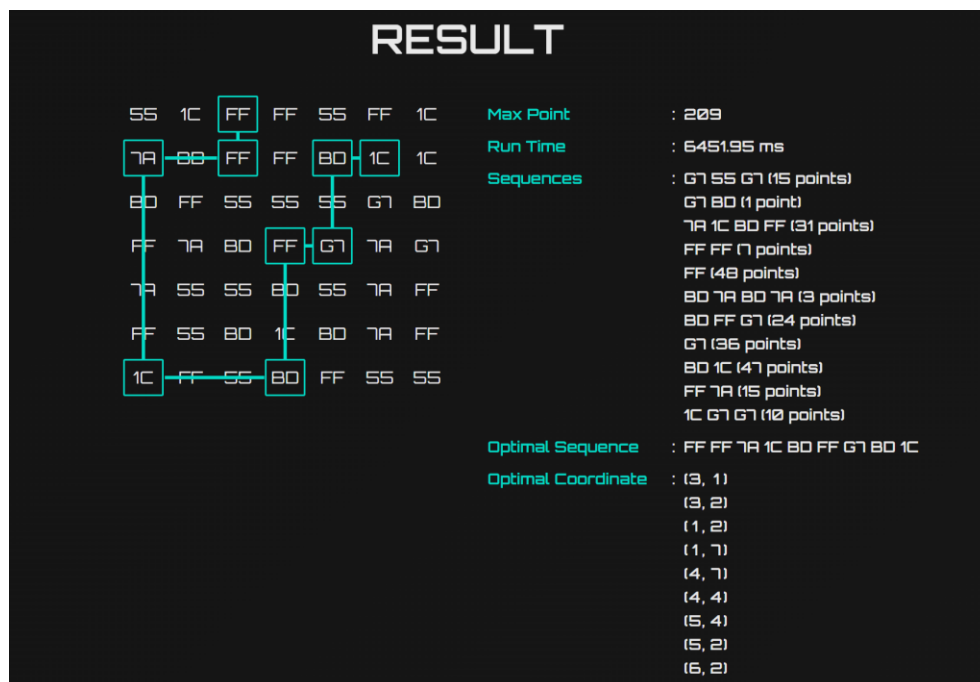
## 5. Tes Ketiga (File)

```

1  9
2  7 7
3  55 1C FF FF 55 FF 1C
4  7A BD FF FF BD 1C 1C
5  BD FF 55 55 55 G7 BD
6  FF 7A BD FF G7 7A G7
7  7A 55 55 BD 55 7A FF
8  FF 55 BD 1C BD 7A FF
9  1C FF 55 BD FF 55 55
10 11
11 G7 55 G7
12 15
13 G7 BD
14 1
15 7A 1C BD FF
16 31
17 FF FF
18 7
19 FF
20 48
21 BD 7A BD 7A
22 3
23 BD FF G7
24 24
25 G7
26 36
27 BD 1C
28 47
29 FF 7A
30 15
31 1C G7 G7
32 10

```

**Gambar 3.5.1** Masukan tes ketiga



**Gambar 3.5.2** Keluaran tes ketiga

## 6. Tes Keempat (Random)

**BREACH PROTOCOL SOLVER**

Cyberpunk 2077 Breach Protocol Solver With Brute Force Algorithm

File Input Random Input

Row: 6

Column: 6

Buffer: 7

Number of Sequence: 3

Maximum Sequence Length: 4

Number Of Unique Token: 5

Input Unique Tokens (separated by spaces): EE 73 H6 G6 CV

Solve

**Gambar 3.6.1** Masukan tes keempat

**RESULT**

Max Point : 1455

Run Time : 24.75 ms

Sequences : H6 H6 EE (465 points)  
G6 G6 73 (651 points)  
H6 73 G6 (804 points)

Optimal Sequence : G6 H6 73 G6 G6 73

Optimal Coordinate : (1, 1)  
(1, 2)  
(3, 2)  
(3, 4)  
(2, 4)  
(2, 1)

Download

**Gambar 3.6.2** Keluaran tes keempat

## 7. Tes Kelima (Random)

**BREACH PROTOCOL SOLVER**

Cyberpunk 2077 Breach Protocol Solver With Brute Force Algorithm

File Input Random Input

Row: 8

Column: 8

Buffer: 7

Number of Sequence: 4

Maximum Sequence Length: 6

Number Of Unique Token: 5

Input Unique Tokens (separated by spaces): EE 73 H6 G6 CV

Solve

Gambar 3.7.1 Masukan tes kelima

**RESULT**

73 H6 EE EE EE G6 EE 73

H6 H6 G6 G6 EE G6 G6 73

EE H6 H6 EE G6 G6 EE EE

73 H6 EE EE 73 EE 73 EE

73 H6 G6 73 G6 73 EE H6

G6 G6 G6 H6 73 H6 H6 73

73 73 73 73 73 73 73

G6 73 73 EE G6 H6 73 G6

Max Point : 1507

Run Time : 200.87 ms

Sequences : G6 G6 G6 G6 (690 points)  
H6 73 EE (251 points)  
73 EE G6 H6 (458 points)  
G6 EE (798 points)

Optimal Sequence : G6 EE H6 73 EE G6 H6

Optimal Coordinate : (6, 1)  
(6, 4)  
(2, 4)  
(2, 8)  
(4, 8)  
(4, 2)  
(1, 2)

Download

Gambar 3.7.2 Keluaran tes kelima

## 8. Tes Keenam (Random)

**BREACH PROTOCOL SOLVER**

Cyberpunk 2077 Breach Protocol Solver With Brute Force Algorithm

File Input Random Input

Row: 10

Column: 10

Buffer: 8

Number of Sequence: 7

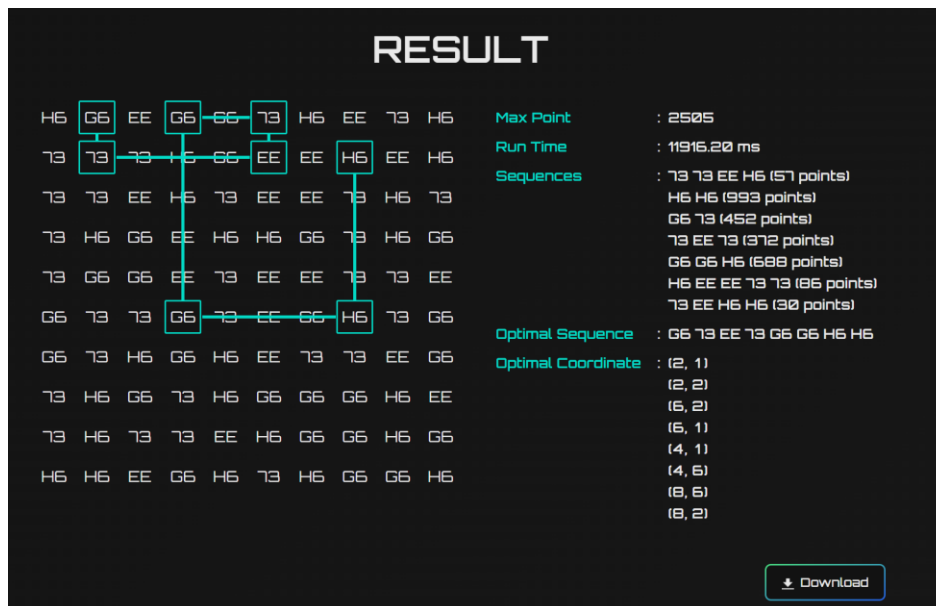
Maximum Sequence Length: 6

Number Of Unique Token: 5

Input Unique Tokens (separated by spaces): EE 73 H6 G6 CV

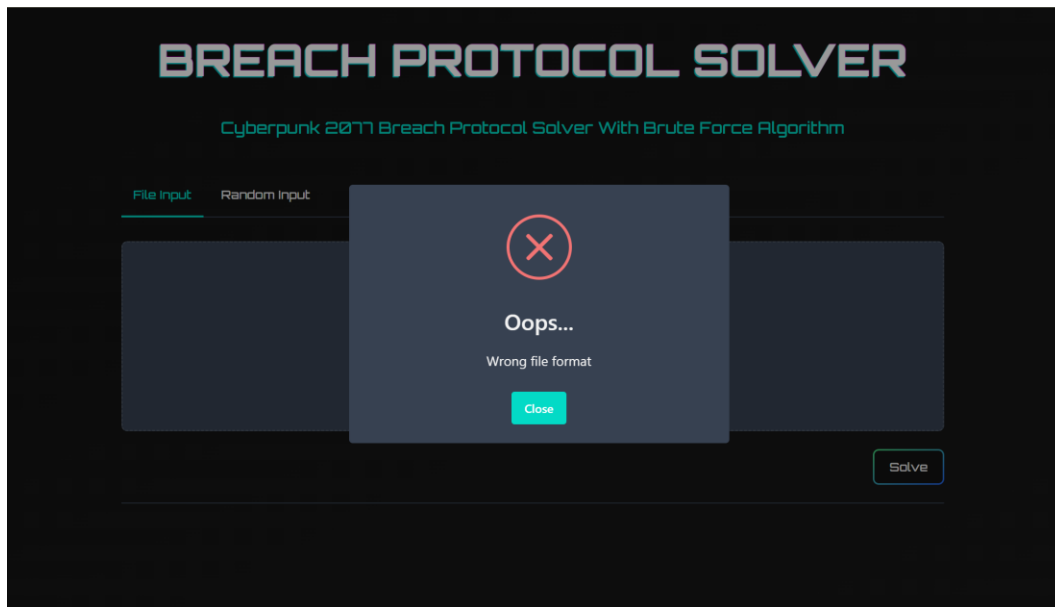
Solve

Gambar 3.8.1 Masukan tes keenam



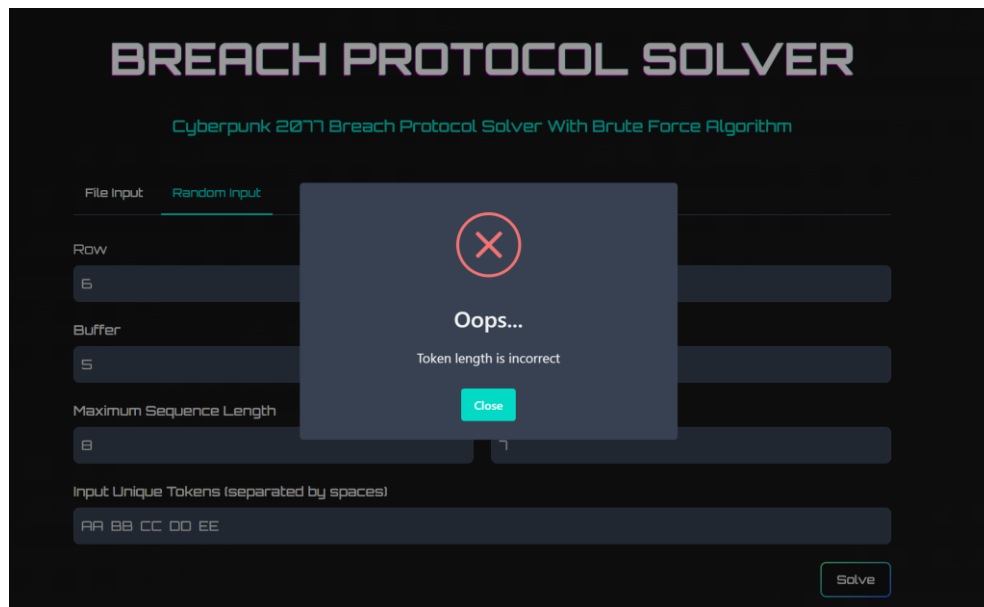
Gambar 3.8.2 Keluaran tes keenam

9. Pesan error untuk kesalahan format file



**Gambar 3.9** Pesan error untuk kesalahan format

10. Pesan error untuk kesalahan banyak token unik dan token masukan



**Gambar 3.10** Pesan error untuk kesalahan format

## LAMPIRAN

### 1. Checklist

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	✓	

### 2. Pranala Repository

[https://github.com/ninoaddict/Tucil1\\_13522068](https://github.com/ninoaddict/Tucil1_13522068)