

Laporan Tugas Kecil IF2211 Strategi Algoritma

**Penyelesaian Cyberpunk 2077 Breach Protocol
dengan Algoritma Brute Force**



Dibuat Oleh:

Farel Winalda / 13522047

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2023**

DAFTAR ISI

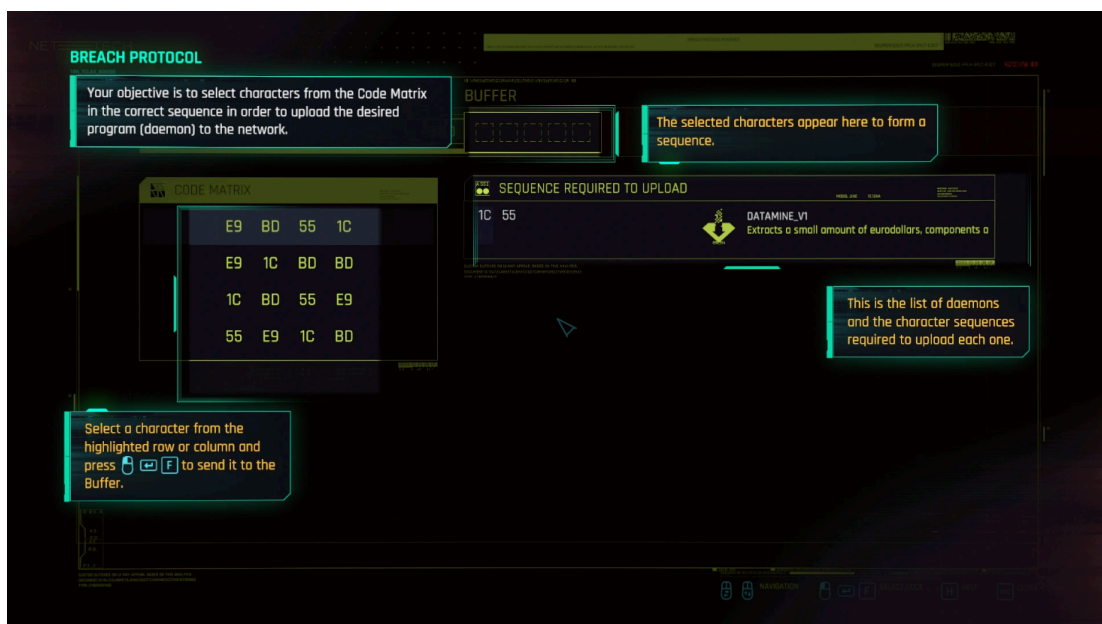
DAFTAR ISI.....	3
BAB I DESKRIPSI MASALAH.....	4
BAB II IMPLEMENTASI PROGRAM.....	6
2.1. Algoritma BruteForce (Bahasa Pemrograman Go).....	6
2.2. Backend (Bahasa Pemrograman Go).....	7
2.3. Frontend (Bahasa Pemrograman Typescript).....	8
BAB III EKSPERIMEN.....	11
3.1. Masukan secara Acak oleh Program.....	11
3.1.1. Studi Kasus 1.....	11
3.1.2. Studi Kasus 2.....	12
3.1.3. Studi Kasus 3.....	13
3.1.4. Studi Kasus 4.....	15
3.2. Masukan file '.txt'.....	17
3.2.1. Studi Kasus 5.....	17
3.2.2. Studi Kasus 6.....	18
LAMPIRAN.....	20

BAB I

DESKRIPSI MASALAH

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.



Gambar 1 Permainan Breach Protocol

(Sumber: <https://cyberpunk.fandom.com/wiki/Quickhacking>)

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.

Pada Laporan ini, akan dibahas implementasi program menggunakan *High-Level Programming Language* dengan tampilan UI dalam memasukkan sebuah masukan / *input*, sebuah endpoint API untuk menghubungkan *FrontEnd* dengan *Backend*, dan juga algoritma penyelesaian dengan menggunakan algoritma *bruteforce*.

BAB II

IMPLEMENTASI PROGRAM

2.1. Algoritma BruteForce (Bahasa Pemrograman Go)

Dalam pembuatan algoritma *bruteforce*, bahasa pemrograman yang digunakan dalam program adalah bahasa pemrograman *Go*. Algoritma dan fungsi *bruteforce* terdapat dalam file *bruteforce.go*.

Dalam file tersebut, terdapat 3 fungsi yang digunakan dan dipecah. Yang pertama adalah fungsi *BruteForce* dengan masukan parameter berupa matriks, matriks boolean, baris, kolom, hasil, array dari hasil, dan juga ukuran yang kemudian akan dicari secara rekursif, dengan inisialisasi token pertama yang merupakan token dari baris pertama, lalu akan dicari semua kemungkinan token yang lain sampai panjang token sama dengan jumlah yang diinginkan, pencarian token dilakukan secara bergantian yaitu dengan kolom, lalu baris, dan seterusnya yang sama dengan token terakhir yang dimasukkan, dan dilakukan secara rekursi untuk mencari semua kemungkinan.

```
func BruteForce(matrix [][]string, visited [][]bool, row, col int, combination []Result, allCombinations *[]Result, size int) {
    if len(combination) == size {
        *allCombinations = append(*allCombinations, combination)
        return
    }

    if len(combination) == 0 {
        for i := 0; i < len(matrix[0]); i++ {
            newComb := []Result{{Col: i, Row: 0, Token: matrix[0][i]}}
            visited[0][i] = true
            BruteForce(matrix, visited, 0, i, newComb, allCombinations, size)
            visited[0][i] = false
        }
    } else {
        if len(combination)%2 == 1 {
            for j := 0; j < len(matrix); j++ {
                if j != row && !visited[j][col] {
                    newComb := append([]Result{}, combination...)
                    newComb = append(newComb, Result{Col: col, Row: j, Token: matrix[j][col]})
                    visited[j][col] = true
                    BruteForce(matrix, visited, j, col, newComb, allCombinations, size)
                    visited[j][col] = false
                }
            }
        } else {
            for j := 0; j < len(matrix[0]); j++ {
                if j != col && !visited[row][j] {
                    newComb := append([]Result{}, combination...)
                    newComb = append(newComb, Result{Col: j, Row: row, Token: matrix[row][j]})
                    visited[row][j] = true
                    BruteForce(matrix, visited, row, j, newComb, allCombinations, size)
                    visited[row][j] = false
                }
            }
        }
    }
}
```

Gambar 2.1 Fungsi *BruteForce*

Lalu, terdapat fungsi *SearchPoint* dengan parameter input berupa array dari hasil (string dan reward integer) yang akan menghasilkan total reward dan hasil akhir

yang paling optimal, fungsi ini digunakan untuk mencari semua kemungkinan reward, lalu dicari reward tertinggi dan juga buffer yang paling optimal.

```
func SearchPoint(combination [][]Result, seq []struct {
    Sequence []string
    Reward    int
}, total int) (int, []Result) {
    mark:=0
    for i := 0; i < len(combination); i++ {
        temp := 0
        for j := 0; j < len(seq); j++ {
            if containsSequence(combination[i], seq[j].Sequence) {
                temp += seq[j].Reward
            }
        }

        if temp > total {
            total = temp
            mark=i
        }
    }

    return total, combination[mark]
}
```

Gambar 2.2 Fungsi *SearchPoint*

Lalu, terdapat fungsi *containsSequence* dengan parameter input berupa array dari hasil dan target yang akan menghasilkan output berupa boolean, fungsi ini digunakan sebagai helper untuk mengecek apakah dalam suatu untaian kode terdapat kesamaan dengan sequence, dan bila ada maka akan menghasilkan boolean.

```
func containsSequence(combination []Result, target []string) bool {
    if len(combination) < len(target) {
        return false
    }

    for i := 0; i <= len(combination)-len(target); i++ {
        found := true
        for j := 0; j < len(target); j++ {
            if combination[i+j].Token != target[j] {
                found = false
                break
            }
        }
        if found {
            return true
        }
    }
    return false
}
```

Gambar 2.3 Fungsi *containsSequence*

2.2. Backend (Bahasa Pemrograman Go)

Bahasa pemrograman yang digunakan dalam pemrosesan input yang diberikan oleh user adalah bahasa pemrograman *Go* dengan menggunakan *Framework Backend Fiber-Go*. Dalam program ini fungsi untuk pembuatan API terdapat dalam file *main.go*. Dengan menggunakan *framework fiber* dimulai dengan mengatur CORS yang diterima dari semua endpoint dengan Method berupa GET, POST, OPTIONS. Lalu membuat konfigurasi method POST untuk endpoint *‘/fileinput’* dan juga *‘/manualinput’*. Endpoint *fileinput* digunakan untuk menerima file yang berisi

masukan yang diperlukan dalam penyelesaian *Breach Protocol* yang akan mengembalikan respon Json String yang berisi solusi optimal (jalan dan reward), matriks token, dan juga waktu pemrosesan. Hal serupa juga digunakan di Endpoint manualinput, akan tetapi dalam endpoint manualinput dipanggil fungsi randomize yang menghasilkan sebuah masukan acak dan kemudian dihasilkan solusi penyelesaian yang sama seperti pada endpoint fileinput

```
package main

import (
    "fmt"
    "io"
    "src/handler"
    "strings"
    "time"

    "github.com/gofiber/fiber/v2"
)

func main() {
    app := fiber.New()

    app.Use(func(c *fiber.Ctx) error {
        c.Set("Access-Control-Allow-Origin", "*")
        c.Set("Access-Control-Allow-Methods", "GET, POST, OPTIONS")
        c.Set("Access-Control-Allow-Headers", "Content-Type")
        if c.Method() == "OPTIONS" {
            return c.SendStatus(fiber.StatusOK)
        }
        return c.Next()
    })

    app.Post("/fileinput", func(c *fiber.Ctx) error { ...
    app.Post("/manualinput", func(c *fiber.Ctx) error { ...

    fmt.Println("Server listening on port 8080")
    app.Listen(":8080")
}
```

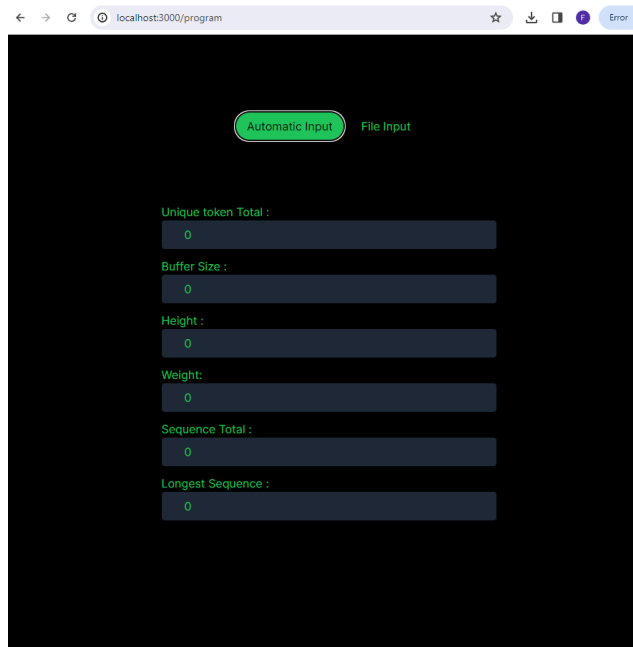
Gambar 2.4 API beserta Configuration Endpoint

2.3. Frontend (Bahasa Pemrograman Typescript)

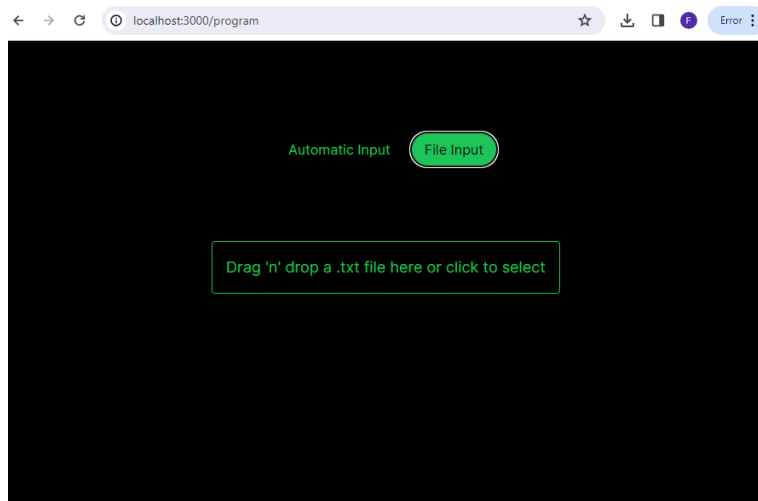
Bahasa pemrograman yang digunakan dalam pembuatan tampilan pengguna atau *User Interface (UI)* adalah bahasa pemrograman *TypeScript* dengan menggunakan *Framework NextJS*. Page dibagi menjadi 2 *LandingPage* dan page program dengan endpoint *'/program'*. Pada page program dibagi menjadi 2 yaitu *Automatic Input* dan *File Input* yang baru bisa submit setelah mengisi semua kebutuhan form dan kemudian akan dikirim dengan Method POST menuju ke backend dengan endpoint sesuai dengan masing-masing opsi yang dipilih



***Gambar 2.4** Tampilan LandingPage*



***Gambar 2.4** Tampilan Program Utama Auto Input*



Gambar 2.4 Tampilan Program Utama File Input

BAB III

EKSPERIMEN

3.1. Masukan secara Acak oleh Program

3.1.1. Studi Kasus 1

The screenshot shows a web interface with a dark background. At the top, there are two buttons: 'Automatic Input' (highlighted in green) and 'File Input'. Below these are several input fields with labels and values:

- Unique token Total : 7
- Unique Token Strings : 7A 55 1C E9 BD 1D FF
- Buffer Size : 7
- Height : 6
- Weight: 6
- Sequence Total : 4
- Longest Sequence : 4

At the bottom, there is a green 'Submit' button.

Gambar 3.1 Masukan Contoh 1

The screenshot shows the 'Result' page. It includes a 'Matrix' table, a list of 'Steps', and a list of 'Sequences'.

Result

Matrix:

7A	BD	55	1C	7A	55
1C	BD	BD	FF	FF	1D
BD	BD	FF	1C	FF	55
55	BD	1C	E9	7A	E9
BD	BD	1D	1C	FF	FF
1D	1C	1D	E9	1D	E9

Steps :

1. Row: 0, Col: 3, Token: 1C
2. Row: 3, Col: 3, Token: E9
3. Row: 3, Col: 5, Token: E9
4. Row: 2, Col: 5, Token: 55
5. Row: 2, Col: 3, Token: 1C
6. Row: 1, Col: 3, Token: FF
7. Row: 1, Col: 5, Token: 1D

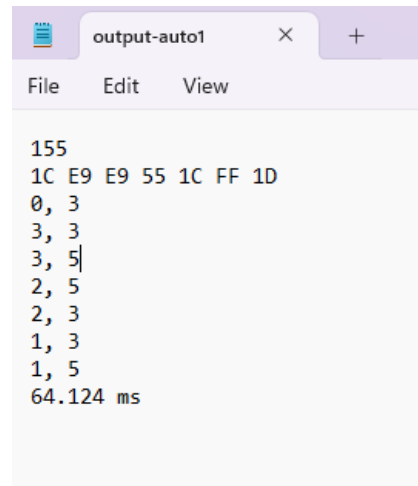
Time: 64.124 ms
Total point: 155

Sequences :

- Sequence: 1C, FF, 1D, Reward: 5
- Sequence: E9, E9, 55, Reward: 70
- Sequence: 1C, E9, Reward: 80
- Sequence: 1C, 7A, 7A, 1D, Reward: 25

At the bottom right, there is a blue button labeled 'Download Result as TXT'.

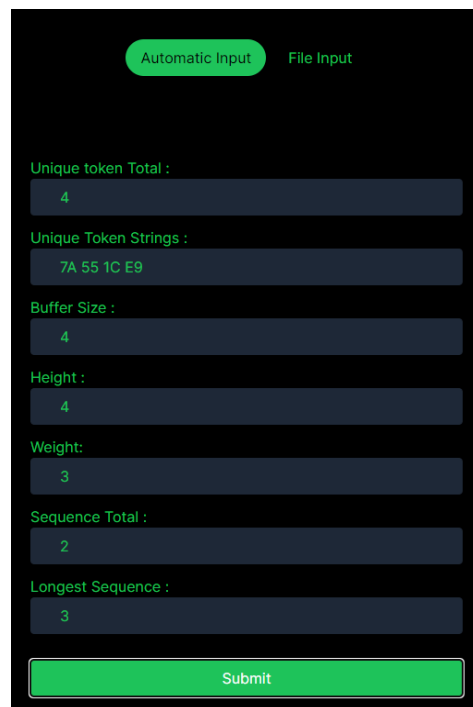
Gambar 3.2 Hasil Masukan Contoh 1



```
155
1C E9 E9 55 1C FF 1D
0, 3
3, 3
3, 5
2, 5
2, 3
1, 3
1, 5
64.124 ms
```

Gambar 3.3 Hasil Keluaran File Contoh 1

3.1.2. Studi Kasus 2



Automatic Input File Input

Unique token Total :
4

Unique Token Strings :
7A 55 1C E9

Buffer Size :
4

Height :
4

Weight:
3

Sequence Total :
2

Longest Sequence :
3

Submit

Gambar 3.4 Masukan Contoh 2

Result

Matrix:

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

Sequences :

Sequence: 1C, 7A, Reward: 15

Sequence: 1C, 55, Reward: 25

Steps :

1. Row: 0, Col: 0, Token: 1C
2. Row: 3, Col: 0, Token: 55
3. Row: 3, Col: 1, Token: 1C
4. Row: 0, Col: 1, Token: 7A

Time: 0.141 ms
Total point: 40

[Download Result as TXT](#)

Gambar 3.5 Hasil Masukan Contoh 2

```

40
1C 55 1C 7A
0, 0
3, 0
3, 1
0, 1
0.141 ms
  
```

Gambar 3.6 Hasil Keluaran File Contoh 2

3.1.3. Studi Kasus 3

☒ Automatic Input
 ☐ File Input

Unique token Total :

Unique Token Strings :

Buffer Size :

Height :

Weight:

Sequence Total :

Longest Sequence :

Gambar 3.7 *Masukan Contoh 3*

Result

Matrix:

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

Sequences :

Sequence: BD, 7A, BD, 1C, BD,
Reward: 45

Sequence: 55, E9, 55, Reward: 75

Sequence: 1C, 7A, E9, 7A, 1C,
Reward: 75

Sequence: BD, BD, 55, E9, 1C,
Reward: 60

Sequence: BD, 7A, Reward: 25

Steps :

1. Row: 0, Col: 0, Token: 55

2. Row: 1, Col: 0, Token: 55

3. Row: 1, Col: 1, Token: E9

4. Row: 0, Col: 1, Token: 55

5. Row: 0, Col: 2, Token: E9

Time: 1 ms

Total point: 75

Download Result as TXT

Gambar 3.8 *Hasil Masukan Contoh 3*

output-auto3

×

+

File Edit View

```

75
55 55 E9 55 E9
0, 0
1, 0
1, 1
0, 1
0, 2
1 ms

```

Gambar 3.9 *Hasil Keluaran File Contoh 3*

3.1.4. Studi Kasus 4

Automatic Input File Input

Unique token Total :
3

Unique Token Strings :
7A 55 1C

Buffer Size :
7

Height :
3

Weight:
3

Sequence Total :
2

Longest Sequence :
4

Submit

Gambar 3.10 Masukan Contoh 4

Result

Matrix:

55	55	1C
1C	1C	55
55	55	1C

Steps :

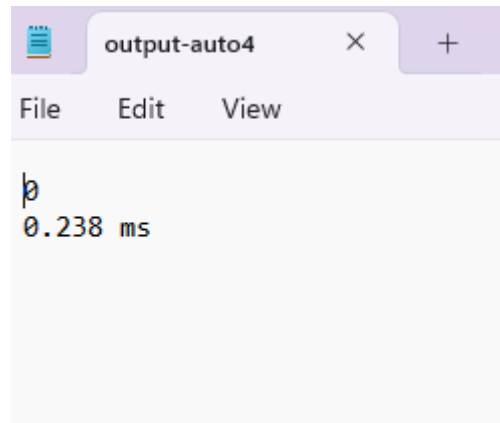
Time: 0.238 ms
Total point: 0

[Download Result as TXT](#)

Sequences :

Sequence: 7A, 1C, 7A, Reward: 65
Sequence: 7A, 55, Reward: 40

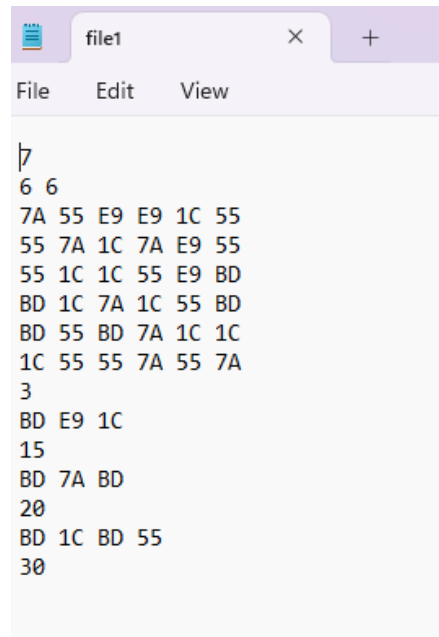
Gambar 3.11 Hasil Masukan Contoh 4



Gambar 3.12 Hasil Keluaran File Contoh 4

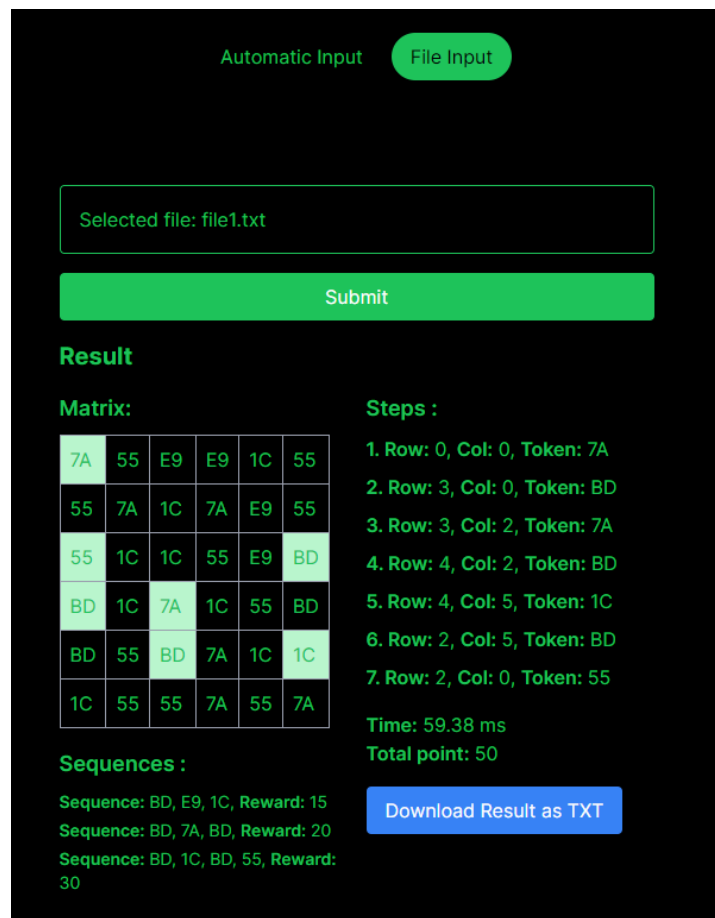
3.2. Masukan file '.txt'

3.2.1. Studi Kasus 5



```
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
```

Gambar 3.13 Masukan Contoh 5



Automatic Input **File Input**

Selected file: file1.txt

Submit

Result

Matrix:

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

Steps :

1. Row: 0, Col: 0, Token: 7A
2. Row: 3, Col: 0, Token: BD
3. Row: 3, Col: 2, Token: 7A
4. Row: 4, Col: 2, Token: BD
5. Row: 4, Col: 5, Token: 1C
6. Row: 2, Col: 5, Token: BD
7. Row: 2, Col: 0, Token: 55

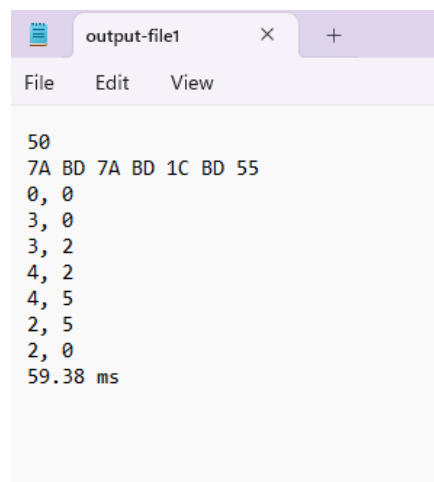
Time: 59.38 ms
Total point: 50

Sequences :

Sequence: BD, E9, 1C, Reward: 15
Sequence: BD, 7A, BD, Reward: 20
Sequence: BD, 1C, BD, 55, Reward: 30

Download Result as TXT

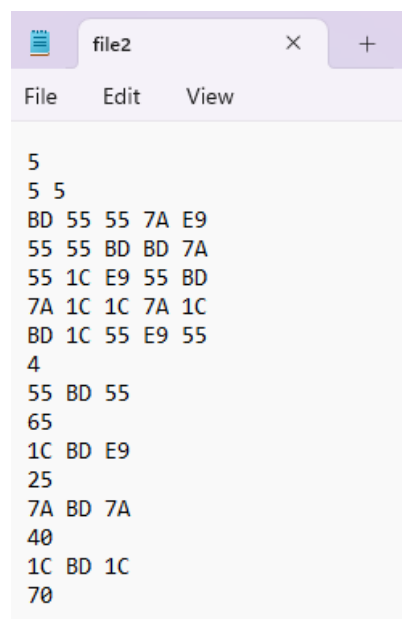
Gambar 3.14 Hasil Masukan Contoh 5



```
50
7A BD 7A BD 1C BD 55
0, 0
3, 0
3, 2
4, 2
4, 5
2, 5
2, 0
59.38 ms
```

Gambar 3.15 Hasil Keluaran File Contoh 5

3.2.2. Studi Kasus 6



```
5
5 5
BD 55 55 7A E9
55 55 BD BD 7A
55 1C E9 55 BD
7A 1C 1C 7A 1C
BD 1C 55 E9 55
4
55 BD 55
65
1C BD E9
25
7A BD 7A
40
1C BD 1C
70
```

Gambar 3.16 Masukan Contoh 6

Automatic Input
File Input

Selected file: file2.txt

Submit

Result

Matrix:

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

Steps :

1. Row: 0, Col: 0, Token: BD
2. Row: 3, Col: 0, Token: 7A
3. Row: 3, Col: 4, Token: 1C
4. Row: 2, Col: 4, Token: BD
5. Row: 2, Col: 1, Token: 1C

Time: 1.018 ms
Total point: 70

Sequences :

Sequence: 55, BD, 55, Reward: 65

Sequence: 1C, BD, E9, Reward: 25

Sequence: 7A, BD, 7A, Reward: 40

Sequence: 1C, BD, 1C, Reward: 70

Download Result as TXT

Gambar 3.17 Hasil Masukan Contoh 6

output-file2

File Edit View

```

|70
BD 7A 1C BD 1C
0, 0
3, 0
3, 4
2, 4
2, 1
1.018 ms

```

Gambar 3.18 Hasil Keluaran File Contoh 6

LAMPIRAN

Repository Github:

<https://github.com/FarelW/Tucil1-13522047>

Tabel Spesifikasi:

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dikompilasi tanpa kesalahan	✓	
3. Program berhasil dikompilasi tanpa kesalahan	✓	
4. Program berhasil dikompilasi tanpa kesalahan	✓	
5. Solusi yang diberikan program optimal	✓	
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI	✓	