

# **Laporan Tugas Kecil 1 IF2211 Strategi Algoritma**



**Disusun oleh:**

Dewantoro Triatmojo (13522011)

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

**2023**

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>BAB 1: Deskripsi Singkat</b>	<b>3</b>
<b>BAB 2: Algoritma Program</b>	<b>4</b>
<b>BAB 3: Source Code</b>	<b>6</b>
1. Program Utama	6
2. Input	7
3. Solve	12
4. Output	16
5. Utils	19
<b>BAB 4: Uji Coba</b>	<b>20</b>
1. Test Case 1	20
2. Test Case 2	21
3. Test Case 3	22
4. Test Case 4	23
5. Test Case 5	24
6. Test Case 6	25
<b>BAB 5: Penutup</b>	<b>27</b>
Kesimpulan	27
Lampiran	27

# BAB 1: Deskripsi Singkat

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. Program ini melakukan algoritma brute force untuk mencari path dengan reward yang optimal.

Beberapa istilah dalam permainan ini:

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.

Beberapa aturan permainan ini:

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.

## BAB 2: Algoritma Program

Program menyelesaikan permainan Cyberpunk 2077 Breach Protocol dengan algoritma brute force. Berikut proses cara kerja program.

1. Inisialisasikan variabel untuk menyimpan nilai maksimum dan path optimum. Untuk memulai brute force, dilakukan looping sepanjang baris pertama matriks untuk memulai brute force pada setiap elemen pada baris pertama. Untuk setiap elemen yang sedang di loop akan mengembalikan reward maksimum path yang dimulai pada sel tersebut. Bandingkan dengan variabel yang diinisialisasikan di awal tadi, simpan nilai yang lebih besar.
2. Pada fungsi rekursi/brute force terdapat parameter untuk menyimpan state gerak (vertikal/horizontal). Kita juga perlu tambahkan parameter untuk menyimpan current path yang sudah ditempuh.
3. Saat fungsi rekursi dipanggil, pertama, inisialisasikan variabel untuk reward optimum dan path optimum. Pertama dapatkan koordinat sekarang pada current path (yaitu koordinat token terakhir pada current path).
  - a. Jika ukuran buffer  $\leq$  width\*height matriks dan panjang current path sudah sama dengan buffer (artinya buffer penuh), kembalikan path kosong. Jika ukuran buffer  $>$  width\*height matriks dan panjang current path sama dengan width\*height matriks (artinya semua elemen matriks sudah terkunjungi), kembalikan path kosong juga.
  - b. Jika state gerakan vertikal, maka lakukan looping sepanjang kolom dengan baris yang sama dengan koordinat yang pertama dicari (koordinat sekarang). Untuk setiap elemen yang sedang diloop, jika sudah ada dalam current path, maka abaikan karena sel matriks hanya boleh digunakan sekali pada current path. Jika tidak ada, maka sambungkan dengan current path. Hitung reward dari path yang sudah disambungkan tadi (reward 1). Lalu kita panggil fungsi rekursif dengan path yang sudah disambungkan tadi dan arah horizontal lalu dari nilai pengembaliannya hitung nilai rewardnya (reward 2). Jika reward 1  $\geq$  reward 2 dan reward 1  $>$  rewardOptimum (variabel yang diinisialisasi di awal) maka simpan

reward1 ke reward optimum. Namun jika reward2 > reward1 dan reward2 > reward optimum, maka simpan reward2 ke reward optimum.

- c. Jika state gerakan horizontal, maka lakukan looping sepanjang baris dengan kolom yang sama dengan koordinat pertama dicari (koordinat sekarang). Untuk setiap elemen yang sedang diloop, jika sudah ada dalam current path, maka abaikan karena sel matriks hanya boleh digunakan sekali pada current path. Jika tidak ada, maka sambungkan dengan current path. Hitung reward dari path yang sudah disambungkan tadi (reward 1). Lalu kita panggil fungsi rekursif dengan path yang sudah disambungkan tadi dan arah vertikal lalu dari nilai pengembaliannya hitung nilai rewardnya (reward 2). Jika reward 1 >= reward 2 dan reward1 > rewardOptimum (variabel yang diinisialisasi di awal) maka simpan reward1 ke reward optimum. Namun jika reward2 > reward1 dan reward2 > reward optimum, maka simpan reward2 ke reward optimum.
- d. Setelah selesai, fungsi kembalikan nilai path optimum

# BAB 3: Source Code

## 1. Program Utama

```
package main

import (
    "Tucil1_13522011/lib/initialize"
    "Tucil1_13522011/lib/input"
    "Tucil1_13522011/lib/output"
    "Tucil1_13522011/lib/solve"
    "time"
)

func main() {
    // Welcome message
    initialize.PrintWelcome()

    // Get Input Data
    var inputData input.InputData
    input.GetInputData(&inputData)

    // Solve
    timeStart := time.Now()
    optimalSolution := solve.GetOptimalSolution(inputData)
    timeEnd := time.Now()
    deltaTime := timeEnd.Sub(timeStart)

    // Print Result
    output.PrintResult(optimalSolution, deltaTime, inputData)
}
```

Gambar 3.1: Fungsi main pada file main.go

## 2. Input

```
package input

import (
    "Tucil1_13522011/lib/utils"
    "bufio"
    "fmt"
    "os"
    "regexp"
    "slices"
    "strings"
)

type Matrix struct {
    Width  int
    Height int
    Buffer  [][]string
}

type Sequence struct {
    Sequence []string
    Reward   int
}

type ArraySequence struct {
    Buffer []Sequence
    Size  int
}

type InputData struct {
    BufferSize int
    Matrix     Matrix
    Sequences  ArraySequence
}
```

Gambar 3.2: Definisi struct InputData

```

func isValidToken(token string) bool {
    // Token terdiri dari dua karakter alfa numerik
    if len(token) != 2 {
        return false
    }

    // Check if token is alphanumeric
    return regexp.MustCompile(`^[a-zA-Z0-9]*$`).MatchString(token)
}

```

Gambar 3.3: Fungsi isValidToken memvalidasi apakah input token valid

```

func printInputData(inputData InputData) {
    // Function to print input data
    fmt.Println("=====")
    fmt.Println("Hasil input data yang dilakukan:")
    fmt.Println("Buffer size:", inputData.BufferSize)
    fmt.Println("Matrix width:", inputData.Matrix.Width)
    fmt.Println("Matrix height:", inputData.Matrix.Height)
    fmt.Println("Matrix data:")
    for i := 0; i < inputData.Matrix.Height; i++ {
        for j := 0; j < inputData.Matrix.Width; j++ {
            fmt.Print(inputData.Matrix.Buffer[i][j])
            if j != inputData.Matrix.Width-1 {
                fmt.Print(" ")
            } else {
                fmt.Println()
            }
        }
    }

    fmt.Println("Sequence size:", inputData.Sequences.Size)
    fmt.Println("Sequence data:")
    for i := 0; i < inputData.Sequences.Size; i++ {
        for j := 0; j < len(inputData.Sequences.Buffer[i].Sequence); j++ {
            fmt.Print(inputData.Sequences.Buffer[i].Sequence[j])
            if j != len(inputData.Sequences.Buffer[i].Sequence)-1 {
                fmt.Print(" ")
            } else {
                fmt.Println()
            }
        }
        fmt.Println(inputData.Sequences.Buffer[i].Reward)
    }
    fmt.Println("=====")
}

```

Gambar 3.4: Fungsi printInputData mencetak data input (bertipe InputData)



```
func getInputMethod(inputMethod *string) {  
    fmt.Println("=====")  
    fmt.Println("Pilih metode input:")  
    fmt.Println("1. File")  
    fmt.Println("2. Keyboard")  
  
    fmt.Scan(inputMethod)  
    for *inputMethod != "1" && *inputMethod != "2" {  
        fmt.Println("Pilihan tidak valid, silakan masukkan pilihan yang valid")  
        fmt.Scan(inputMethod)  
    }  
    fmt.Println("=====")  
}
```

Gambar 3.5: Fungsi getInputData mendapatkan metode input

```

func getFromFile(inputData *InputData) {
    fmt.Println("=====")
    fmt.Println("Masukkan nama file (di folder /test/input beserta ekstensi txt):")
    var filename string
    // Input filename
    fmt.Scan(&filename)

    // Check if file doesnt exists
    for {
        if _, err := os.Stat("../test/input/" + filename); os.IsNotExist(err) {
            fmt.Println("File tidak ditemukan, silakan masukkan nama file yang valid (di folder /test/input beserta ekstensi txt):")
            fmt.Scan(&filename)
        } else {
            break
        }
    }
    fmt.Println("=====")

    // Open file
    file, err := os.Open("../test/input/" + filename)
    if err != nil {
        fmt.Println(err)
    }
    defer file.Close()

    // Read file & parse input
    scanner := bufio.NewScanner(file)
    for i := 0; scanner.Scan(); i++ {
        if i == 0 {
            // Buffer size
            _, err := fmt.Sscanf(scanner.Text(), "%d", &inputData.BufferSize)
            if err != nil {
                fmt.Println("Error: buffer size bukan angka yang valid")
                os.Exit(1)
            }
        } else if i == 1 {
            // Matrix width and height
            _, err := fmt.Sscanf(scanner.Text(), "%d %d", &inputData.Matrix.Width, &inputData.Matrix.Height)
            if err != nil {
                fmt.Println("Error: width atau height matriks bukan angka yang valid")
                os.Exit(1)
            }
        } else if i >= 2 && i < 2+inputData.Matrix.Height {
            // Matrix data
            rowString := scanner.Text()

            // Trim beginning and trailing spaces
            rowString = strings.TrimSpace(rowString)

            // Check if row length is not equal to matrix width
            rows := strings.Split(rowString, " ")
            if len(rows) != inputData.Matrix.Width {
                fmt.Println("Error: input matriks tidak sesuai dengan input panjang width matriks")
                os.Exit(1)
            }
            inputData.Matrix.Buffer = append(inputData.Matrix.Buffer, rows)
        } else if i == 2+inputData.Matrix.Height {
            // Sequence size
            _, err := fmt.Sscanf(scanner.Text(), "%d", &inputData.Sequences.Size)
            if err != nil {
                fmt.Println("Error: sequence size bukan angka yang valid")
                os.Exit(1)
            }
        } else if i > 2+inputData.Matrix.Height && i <= 2+inputData.Matrix.Height+inputData.Sequences.Size*2 {
            if (i-2-inputData.Matrix.Height)%2 == 1 {
                // Sequence data
                rowString := scanner.Text()

                // Trim beginning and trailing spaces
                rowString = strings.TrimSpace(rowString)

                sequence := strings.Split(rowString, " ")
                inputData.Sequences.Buffer = append(inputData.Sequences.Buffer, Sequence{sequence, 0})
            } else {
                // Reward data
                _, err := fmt.Sscanf(scanner.Text(), "%d", &inputData.Sequences.Buffer[(i-3-inputData.Matrix.Height)/2].Reward)
                if err != nil {
                    fmt.Println("Error: reward bukan angka yang valid")
                    os.Exit(1)
                }
            }
        }
    }
}

```

Gambar 3.6: Fungsi getFromFile untuk mendapatkan data input melalui pembacaan file

```

func getFromKeyboard(inputData *InputData) {
    fmt.Println("=====")
    fmt.Println("Masukkan jumlah token unik:")
    var tokenUnikCount int
    fmt.Scan(&tokenUnikCount)
    // Validasi jumlah token unik > 0
    for tokenUnikCount ≤ 0 {
        fmt.Println("Jumlah token unik harus lebih dari 0, silakan masukkan jumlah token unik yang valid:")
        fmt.Scan(&tokenUnikCount)
    }

    fmt.Println("Masukkan token unik:")
    tokenArray := make([]string, tokenUnikCount)
    for i := 0; i < tokenUnikCount; i++ {
        var newToken string
        fmt.Scan(&newToken)
        // Validasi token
        isTokenExist := slices.Contains(tokenArray, newToken)
        for !isTokenValid(newToken) || isTokenExist {
            if !isTokenValid(newToken) {
                fmt.Println("Token harus memiliki panjang 2 dan berupa karakter alfanumerik, silakan masukkan token yang valid:")
            } else {
                fmt.Println("Token sudah Anda masukkan, silakan masukkan token yang belum ada:")
            }
            fmt.Scan(&newToken)
            isTokenExist = slices.Contains(tokenArray, newToken)
        }
        // Add token to array
        tokenArray[i] = newToken
    }

    fmt.Println("Masukkan buffer size:")
    fmt.Scan(&inputData.BufferSize)
    // Validasi buffer size > 0
    for inputData.BufferSize ≤ 0 {
        fmt.Println("Buffer size harus lebih dari 0, silakan masukkan buffer size yang valid:")
        fmt.Scan(&inputData.BufferSize)
    }

    fmt.Println("Masukkan matriks width:")
    fmt.Scan(&inputData.Matrix.Width)
    // Validasi matriks width > 0
    for inputData.Matrix.Width ≤ 0 {
        fmt.Println("Matriks width harus lebih dari 0, silakan masukkan matriks width yang valid:")
        fmt.Scan(&inputData.Matrix.Width)
    }

    fmt.Println("Masukkan matriks height:")
    fmt.Scan(&inputData.Matrix.Height)
    // Validasi matriks height > 0
    for inputData.Matrix.Height ≤ 0 {
        fmt.Println("Matriks height harus lebih dari 0, silakan masukkan matriks height yang valid:")
        fmt.Scan(&inputData.Matrix.Height)
    }

    fmt.Println("Masukkan jumlah sequence:")
    fmt.Scan(&inputData.Sequences.Size)
    // Validasi jumlah sequence > 0
    for inputData.Sequences.Size ≤ 0 {
        fmt.Println("Jumlah sequence harus lebih dari 0, silakan masukkan jumlah sequence yang valid:")
        fmt.Scan(&inputData.Sequences.Size)
    }

    var ukuranMaxSequence int
    fmt.Println("Masukkan ukuran max sequence:")
    fmt.Scan(&ukuranMaxSequence)
    // Validasi ukuran max sequence ≥ 2 (peraturan nomor 6 sekuens minimal berupa 2 token)
    for ukuranMaxSequence < 2 {
        fmt.Println("Ukuran sequence minimal 2, silakan masukkan ukuran max sequence yang valid:")
        fmt.Scan(&ukuranMaxSequence)
    }

    // Generate random matrix
    for i := 0; i < inputData.Matrix.Height; i++ {
        // Generate row
        row := make([]string, inputData.Matrix.Width)
        for j := 0; j < inputData.Matrix.Width; j++ {
            randomTokenIndex := utils.GetRandom(0, tokenUnikCount-1)
            row[j] = tokenArray[randomTokenIndex]
        }
        inputData.Matrix.Buffer = append(inputData.Matrix.Buffer, row)
    }

    // Generate random sequence
    for i := 0; i < inputData.Sequences.Size; i++ {
        // Generate sequence
        // Sequence minimal 2 token (dari spec)
        sequenceLength := utils.GetRandom(2, ukuranMaxSequence)
        sequence := make([]string, sequenceLength)
        for j := 0; j < sequenceLength; j++ {
            randomTokenIndex := utils.GetRandom(0, tokenUnikCount-1)
            sequence[j] = tokenArray[randomTokenIndex]
        }
        // Generate sequence reward (distur min = 0 dan max = 50)
        randomReward := utils.GetRandom(0, 50)
        inputData.Sequences.Buffer = append(inputData.Sequences.Buffer, Sequence{sequence, randomReward})
    }
    fmt.Println("=====")
}

```

Gambar 3.7: Fungsi getFromKeyboard mendapatkan data input melalui keyboard

```

func GetInputData(inputData *InputData) {
    var inputMethod string
    getInputMethod(&inputMethod)

    if inputMethod == "1" {
        getFromFile(inputData)
    } else if inputMethod == "2" {
        getFromKeyboard(inputData)
    }

    printInputData(*inputData)
}

```

Gambar 3.8: Fungsi GetInputData fungsi yang dipanggil di program utama untuk mendapatkan input data

### 3. Solve

```

package solve

import (
    "Tucil1_13522011/lib/input"
)

type Coordinate struct {
    X int
    Y int
}

type Path []Coordinate

type SolutionData struct {
    Reward int
    Path    Path
}

```

Gambar 3.9: Definisi struct SolutionData

```

func getReward(path Path, inputData input.InputData) int {
    // Function to calculate the reward from a pathforo

    // Initialize reward
    reward := 0

    // Iterate through all sequences
    for _, seq := range inputData.Sequences.Buffer {
        // Iterate through all possible position
        for i := 0; i < len(path)-len(seq.Sequence)+1; i++ {
            allMatch := true

            for j := 0; j < len(seq.Sequence); j++ {
                x := path[i+j].X
                y := path[i+j].Y
                // Check if the sequence match
                if inputData.Matrix.Buffer[y][x] != seq.Sequence[j] {
                    allMatch = false
                    break
                }
            }

            // If all match, add the reward
            if allMatch {
                reward += seq.Reward

                // Each sequence can only be used once
                break
            }
        }
    }

    return reward
}

```

Gambar 3.10: Fungsi getReward menghitung reward dari suatu path yang ditempuh

```
func isCoordinateInPath(coordinate Coordinate, path Path) bool {
    // Function to check if a coordinate is in a path
    for _, p := range path {
        if p.X == coordinate.X && p.Y == coordinate.Y {
            return true
        }
    }

    return false
}
```

Gambar 3.11: Fungsi isCoordinateInPath mengembalikan apakah suatu koordinat ada pada

```
func GetOptimalSolution(inputData input.InputData) SolutionData {
    // Initialize solution
    solution := SolutionData{0, nil}

    // Iterate through all possible starting point
    for i := 0; i < inputData.Matrix.Width; i++ {
        newPath := getOptimalPathRecursive(Path{{i, 0}}, true, inputData)
        newReward := getReward(newPath, inputData)

        // Update solution if new reward is better
        if newReward > solution.Reward {
            solution.Reward = newReward
            solution.Path = newPath
        }
    }

    return solution
}
```

Gambar 3.12: Fungsi GetOptimalSolution dipanggil di program utama untuk mengembalikan data solusi optimal

```

func getOptimalPathRecursive(currentPath Path, isCurrentVertical bool, inputData input.InputData) Path {
    // Base case
    if inputData.BufferSize ≤ inputData.Matrix.Width*inputData.Matrix.Height {
        if len(currentPath) == inputData.BufferSize {
            return nil
        }
    } else {
        if len(currentPath) == inputData.Matrix.Width*inputData.Matrix.Height {
            return nil
        }
    }

    var mostOptimalPath Path
    mostReward := 0

    // Generate path using recursive
    if isCurrentVertical {
        for i := 0; i < inputData.Matrix.Height; i++ {
            initialX := currentPath[len(currentPath)-1].X
            targetCoordinate := Coordinate{initialX, i}
            if !isCoordinateInPath(targetCoordinate, currentPath) {
                // Get new path
                newPath1 := make([]Coordinate, len(currentPath))
                copy(newPath1, currentPath)
                newPath1 = append(newPath1, targetCoordinate)

                // Recurrence
                newPath2 := getOptimalPathRecursive(newPath1, false, inputData)

                reward1 := getReward(newPath1, inputData)
                reward2 := getReward(newPath2, inputData)

                if reward1 ≥ reward2 && reward1 > mostReward {
                    mostReward = reward1
                    mostOptimalPath = newPath1
                }

                if reward2 > reward1 && reward2 > mostReward {
                    mostReward = reward2
                    mostOptimalPath = newPath2
                }
            }
        }
    } else {
        for i := 0; i < inputData.Matrix.Width; i++ {
            initialY := currentPath[len(currentPath)-1].Y
            targetCoordinate := Coordinate{i, initialY}
            if !isCoordinateInPath(targetCoordinate, currentPath) {
                // Get new path
                newPath1 := make([]Coordinate, len(currentPath))
                copy(newPath1, currentPath)
                newPath1 = append(newPath1, targetCoordinate)

                // Recurrence
                newPath2 := getOptimalPathRecursive(newPath1, true, inputData)

                reward1 := getReward(newPath1, inputData)
                reward2 := getReward(newPath2, inputData)

                if reward1 ≥ reward2 && reward1 > mostReward {
                    mostReward = reward1
                    mostOptimalPath = newPath1
                }

                if reward2 > reward1 && reward2 > mostReward {
                    mostReward = reward2
                    mostOptimalPath = newPath2
                }
            }
        }
    }

    return mostOptimalPath
}

```

Gambar 3.13: Fungsi getOptimalPathRecursive mengembalikan path optimal yang dimulai dengan currentPath

## 4. Output

```
func getPathCode(path solve.Path, inputData input.InputData) string {  
    // Function to get the path code  
    var pathCode string  
    for i, p := range path {  
        pathCode += string(inputData.Matrix.Buffer[p.Y][p.X])  
        if i != len(path)-1 {  
            pathCode += " "  
        }  
    }  
    return pathCode  
}
```

Gambar 3.14: Fungsi getPathCode mengkonversikan Path (dalam bentuk array of coordinate) ke token dalam bentuk string



```

func saveToFile(solution solve.SolutionData, deltaTime time.Duration, inputData input.InputData) {
    // Function to save solution to file
    // Get file name
    fmt.Println("Masukkan nama file untuk menyimpan solusi (di folder /test/output beserta ekstensi txt):")
    var fileName string
    fmt.Scan(&fileName)

    // Check if file already exists
    for {
        _, err := os.Stat("../test/output/" + fileName)
        if os.IsNotExist(err) {
            break
        }
        fmt.Println("File sudah ada, apakah ingin menyimpannya? (y/n)")

        var overwrite string
        fmt.Scan(&overwrite)
        for overwrite != "y" && overwrite != "n" {
            fmt.Println("Input tidak valid")
            fmt.Scan(&overwrite)
        }

        if overwrite == "y" {
            break
        }

        fmt.Println("Masukkan nama file untuk menyimpan solusi (di folder /test/output beserta ekstensi txt):")
        fmt.Scan(&fileName)
    }

    // Create file
    file, err := os.Create("../test/output/" + fileName)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer file.Close()

    // Write to file
    // Reward
    file.WriteString(fmt.Sprintf("Optimal Reward: %d\n", solution.Reward))
    // Path
    pathCode := getPathCode(solution.Path, inputData)
    file.WriteString(fmt.Sprintf("Path: %s\n", pathCode))
    // Coordinate
    file.WriteString("Coordinate:\n")
    for _, p := range solution.Path {
        file.WriteString(fmt.Sprintf("%d %d\n", p.X+1, p.Y+1))
    }
    // Delta Time
    file.WriteString("\n")
    file.WriteString(fmt.Sprintf("Time: %d ms\n", deltaTime.Milliseconds()))
}

```

Gambar 3.15: Fungsi saveToFile menyimpan hasil solusi ke sebuah file

```

func PrintResult(solution solve.SolutionData, deltaTime time.Duration, inputData input.InputData) {
    fmt.Println("=====")
    if solution.Reward == 0 {
        // Reward
        fmt.Println("Optimal Reward: 0")
        fmt.Println("No solution found")
    } else {
        // Reward
        fmt.Println("Optimal Reward:", solution.Reward)
        // Path
        pathCode := getPathCode(solution.Path, inputData)
        fmt.Printf("Path: %s\n", pathCode)
        // Coordinate
        fmt.Println("Coordinate:")
        for _, p := range solution.Path {
            fmt.Printf("%d %d\n", p.X+1, p.Y+1)
        }
    }

    // Delta Time
    fmt.Println()
    fmt.Println("Time:", deltaTime.Milliseconds(), "ms")
    fmt.Println()

    // Ingin menyimpan solusi
    fmt.Println("Apakah ingin menyimpan solusi? (y/n)")
    var saveSolution string
    fmt.Scan(&saveSolution)
    for saveSolution != "y" && saveSolution != "n" {
        fmt.Println("Input simpan solusi tidak valid")
        fmt.Scan(&saveSolution)
    }

    if saveSolution == "y" {
        saveToFile(solution, deltaTime, inputData)
    }

    fmt.Println("=====")
    fmt.Println("=====")
}

```

Gambar 3.16: Fungsi PrintResult mencetak hasil solusi perhitungan dan juga opsi untuk menyimpan file.

## 5. Utils

```
package utils

import (
    "math/rand"
)

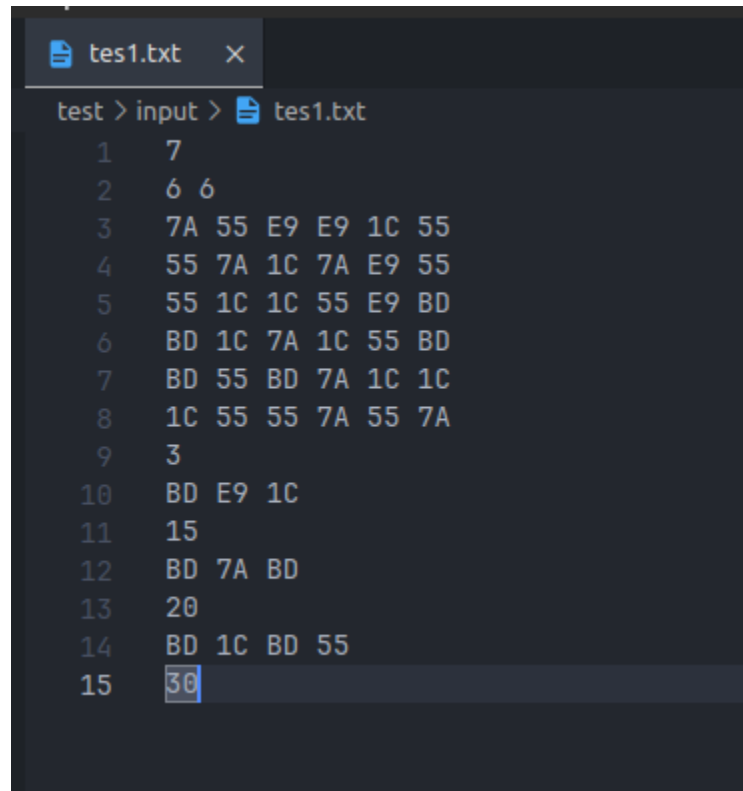
func GetRandom(min int, max int) int {
    return min + rand.Intn(max-min+1)
}
```

Gambar 3.17: fungsi GetRandom menghasilkan bilangan random di interval [min, max]

## BAB 4: Uji Coba

### 1. Test Case 1

Input:



A screenshot of a text editor window titled 'tes1.txt'. The editor shows a list of 15 test cases, each with a line number on the left and a sequence of numbers and hexadecimal values on the right. The text is as follows:

```
test > input > tes1.txt
1      7
2      6 6
3      7A 55 E9 E9 1C 55
4      55 7A 1C 7A E9 55
5      55 1C 1C 55 E9 BD
6      BD 1C 7A 1C 55 BD
7      BD 55 BD 7A 1C 1C
8      1C 55 55 7A 55 7A
9      3
10     BD E9 1C
11     15
12     BD 7A BD
13     20
14     BD 1C BD 55
15     36
```

Gambar 4.1: Input test case 1

Output:

```
=====
=====
Optimal Reward: 50
Path: 7A BD 7A BD 1C BD 55
Coordinate:
1 1
1 4
3 4
3 5
6 5
6 3
1 3

Time: 31 ms

Apakah ingin menyimpan solusi? (y/n)

```

Gambar 4.2: Output test case 1

## 2. Test Case 2

Input:

```
tes2.txt x
test > input > tes2.txt
1 7
2 7 7
3 IJ CD EF EF GH CD IJ
4 AB EF GH CD CD IJ EF
5 GH AB AB AB IJ CD GH
6 EF GH IJ CD CD IJ EF
7 EF AB GH CD AB CD AB
8 CD CD EF GH CD AB GH
9 EF GH GH AB EF AB GH
10 4
11 EF CD IJ
12 48
13 CD CD CD
14 34
15 AB IJ
16 31
17 IJ AB
18 0
```

Gambar 4.3: Input test case 2

Output:

```
=====
=====
Optimal Reward: 82
Path: IJ EF CD IJ CD CD CD
Coordinate:
1 1
1 4
5 4
5 3
6 3
6 1
2 1

Time: 131 ms

Apakah ingin menyimpan solusi? (y/n)
█
```

Gambar 4.4: Output test case 2

### 3. Test Case 3

Input

```
tes3.txt x
test > input > tes3.txt
1 6
2 5 8
3 OP QW ER TY OP
4 OP ER UI AS UI
5 OP OP TY UI TY
6 OP QW TY ER AS
7 UI TY TY AS AS
8 OP UI QW QW TY
9 QW ER QW UI AS
10 TY QW UI ER OP
11 4
12 ER UI
13 19
14 OP OP UI
15 48
16 OP QW
17 25
18 OP UI
19 16
```

Gambar 4.5: Input test case 3

## Output

```
=====
=====
Optimal Reward: 89
Path: OP OP UI TY OP QW
Coordinate:
1 1
1 2
3 2
3 3
1 3
1 7
Time: 39 ms
```

Gambar 4.6: Output test case 3

## 4. Test Case 4

### Input

```
tes4.txt x
test > input > tes4.txt
1 10
2 8 8
3 CD EF CD AB EF CD EF AB
4 AB EF AB EF CD AB KL IJ
5 KL EF GH AB EF AB IJ KL
6 GH GH GH IJ IJ EF KL CD
7 IJ AB IJ KL IJ KL EF IJ
8 IJ GH CD EF EF KL CD CD
9 AB IJ EF GH EF KL EF CD
10 CD GH AB CD GH GH IJ IJ
11 5
12 KL CD CD
13 45
14 EF CD IJ
15 18
16 GH AB
17 40
18 GH AB EF KL
19 37
20 EF EF EF
21 22
```

Gambar 4.7: Input test case 4

## Output

```
=====
=====
Optimal Reward: 144
Path: EF EF EF IJ GH AB EF KL CD CD
Coordinate:
2 1
2 2
4 2
4 4
1 4
1 7
7 7
7 4
8 4
8 6

Time: 116087 ms

Apakah ingin menyimpan solusi? (y/n)

```

Gambar 4.8: Output test case 4

## 5. Test Case 5

### Input

```
tes5.txt x
test > input > tes5.txt
1 5
2 6 6
3 GH EF AB AB EF IJ
4 CD GH EF KL AB EF
5 AB IJ CD EF IJ IJ
6 AB CD IJ EF IJ KL
7 IJ KL CD KL EF AB
8 EF AB AB KL KL AB
9 4
10 AB KL
11 17
12 AB KL KL
13 18
14 EF CD GH
15 -44
16 IJ IJ KL
17 -15
```

Gambar 4.9: Input test case 5



## Output

```
=====
=====
Optimal Reward: 35
Path: GH CD AB KL KL
Coordinate:
1 1
1 2
5 2
5 6
4 6

Time: 5 ms

Apakah ingin menyimpan solusi? (y/n)

```

Gambar 4.10: Output test case 5

## 6. Test Case 6

### Input

```
tes6.txt x
test > input > tes6.txt
1 7
2 7 7
3 KL IJ GH IJ AB KL IJ
4 KL CD EF CD EF KL EF
5 GH AB CD EF KL GH KL
6 CD IJ CD AB EF KL EF
7 GH EF CD EF GH GH AB
8 CD EF GH EF KL IJ EF
9 IJ AB GH CD KL IJ IJ
10 5
11 EF EF
12 -49
13 GH GH AB
14 -16
15 CD AB
16 -6
17 KL EF CD
18 -27
19 CD AB
20 -41
```

Gambar 4.11: Input test case 6

## Output

```
=====
=====
Optimal Reward: 0
No solution found

Time: 152 ms

Apakah ingin menyimpan solusi? (y/n)
█
```

Gambar 4.12: Input test case 7

# BAB 5: Penutup

## Kesimpulan

Permainan Cyberpunk 2077 Breach Protocol dapat diselesaikan dengan algoritma brute force. Untuk test case dengan ukuran buffer atau matriks yang kecil, algoritma brute force lumayan cepat. Namun untuk test case dengan ukuran buffer atau matriks yang besar, algoritma brute force lumayan lama.

## Lampiran

1. Link Repository Github

[https://github.com/dewodt/Tucil1\\_13522011](https://github.com/dewodt/Tucil1_13522011)

2. Tabel Poin

No	Poin	Ya	Tidak
1.	Program berhasil dikompilasi tanpa kesalahan	<input checked="" type="checkbox"/>	
2.	Program berhasil dijalankan	<input checked="" type="checkbox"/>	
3.	Program dapat membaca masukan berkas .txt	<input checked="" type="checkbox"/>	
4.	Program dapat menghasilkan masukan secara acak	<input checked="" type="checkbox"/>	
5.	Solusi yang diberikan program optimal	<input checked="" type="checkbox"/>	
6.	Program dapat menyimpan solusi dalam berkas .txt	<input checked="" type="checkbox"/>	
7.	Program memiliki GUI		<input checked="" type="checkbox"/>