Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2023/2024

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma Brute Force

Julian Chandra Sutadi 13522080

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

DAFTAR ISI

BAB I
DESKRIPSI PERMASALAHAN

Cyberpunk 2077 Breach Protocol adalah *minigame* dalam sebuah permainan video *Cyberpunk* 2077. *Minigame* ini merupakan simulasi peretasan jaringan lokal dari *ICE* (*Intrusion Countermeasures Electronics*) pada permainan *Cyberpunk 2077*. Komponen pada permainan ini antara lain:

1. token yang terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55,
2. matriks yang terdiri atas token-token yang akan dipilih untuk menyusun urutan kode,
3. sekuens, yaitu sebuah rangkaian token (dua atau lebih) yang harus dicocokkan, dan
4. buffer, yaitu jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan *minigame* ini sebagai berikut:

1. Pemain memulai dengan memilih salah satu token pada posisi baris paling atas matriks.
2. Pemain bergerak vertikal, kemudian horizontal-vertikal bergantian hingga semua sekuens berhasil dicocokkan atau buffer penuh.
3. Sekuens dicocokkan dengan token-token yang berada di buffer
4. Satu token pada buffer dapat digunakan pada satu atau lebih sekuens
5. Setiap sekuens memiliki bobot hadiah atau *reward* yang variative
6. Sekuens memiliki panjang minimal dua token.

Tugas kecil ini bertujuan untuk merancang algoritma *brute force* untuk menemukan solusi dengan hadiah terbesar dari permainan *Cyberpunk 2077 Breach Protocol*. Pada Tugas Kecil 1 ini, akan digunakan bahasa pemrograman Go.

BAB II
ALGORITMA BRUTE FORCE

Algoritma brute force adalah pendekatan yang lempang untuk memecahkan suatu persoalan. Algoritma ini didasarkan pada problem statement dan definisi atau konsep yang dilibatkan. Algoritma ini akan memecahkan persoalan dengan langsung, jelas, dan sangat sederhana. Pada *minigame Cyberpunk 2077 Breach Protocol*, algoritma ini akan mencari semua kemungkinan isi buffer (*path*) dengan panjang sebesar panjang maksimal buffer. Kemudian, algoritma akan mencari *path* yang memberikan *reward* maksimal namun dengan indeks akhir paling kecil.

Algoritma *brute force* yang dibuat untuk menyelesaikan *minigame* tersebut memiliki langkah-langkah sebagai berikut:

1. Mencari semua *path* dengan panjang sesuai kapasitas buffer, dimulai dari setiap token teratas pada matriks permainan, memastikan bahwa tidak ada token dengan lokasi yang sama tidak digunakan dua kali. *Path* akan disimpan pada struktur data *listOfPaths*.
2. Pencarian tiap *path* dilakukan secara rekursif dan bergantian vertikal-horizontal, dilakukan sampai iterator mencapa nilai kapasitas buffer dikurangi satu.
3. Inisialisasi *path* dengan *path* pertama.
4. Iterasi semua *path*, kemudian hitung *reward* dan indeks terakhir dari path tersebut. Pada iterasi, jika ada *path* dengan *reward* yang lebih besar, maka *path* tersebut menjadi *path* paling optimal. Jika *reward* bernilai sama namun indeks terakhir lebih kecil, maka *path* tersebut menjadi *path* paling optimal.

BAB III

SOURCE CODE

type.go (Deklarasi tipe data)

```
type matrix struct {
	Width  int
	Height int
	Buffer [][]string
}

type sequence struct {
	Tokens []string
	Reward int
}

type listOfSequence struct {
	Buffer []sequence
	Neff   int
}

type point struct {
	Column int
	Row    int
}

type path struct {
	Coordinates []point
	Neff        int
}

type listOfPath struct {
	Buffer []path
	Neff   int
}
```

txtprocessor.go (Menangani i/o dengan file)

```
package main

import (
	"fmt"
	"os"
	"path/filepath"
	"strconv"
	"strings"
	"time"
)

func parseFile(filename string) (int, matrix, listOfSequence) {
	var err error
	data, err := os.ReadFile(filename)
	if err != nil {
		fmt.Println("Error:", err)
```

```
	}

	str := string(data)

	var bufferSize int
	var gameMatrix matrix
	var listOfSeq listOfSequence

	lines := splitLines(str)

	bufferSize, err = strconv.Atoi(lines[0])
	if err != nil {
		fmt.Println("Error:", err)
	}

	temp := strings.Split(lines[1], " ")
	gameMatrix.Width, err = strconv.Atoi(temp[0])
	if err != nil {
		fmt.Println("Error:", err)
	}

	gameMatrix.Height, err = strconv.Atoi(temp[1])
	if err != nil {
		fmt.Println("Error:", err)
	}

	currentLine := 2
	for i := 0; i < gameMatrix.Height; i++ {
		gameMatrix.Buffer = append(gameMatrix.Buffer,
strings.Split(lines[currentLine], " "))
		currentLine += 1
	}

	numOfSequences, err := strconv.Atoi(lines[currentLine])
	if err != nil {
		fmt.Println("Error:", err)
	}
	currentLine += 1

	listOfSeq.Neff = 0
	for i := 0; i < numOfSequences; i++ {
		tempToken := strings.Split(lines[currentLine], " ")
		currentLine += 1
		tempReward, err := strconv.Atoi(lines[currentLine])
		if err != nil {
			fmt.Println("Error:", err)
		}

		newSeq := sequence{
			Tokens: tempToken,
			Reward: tempReward,
		}
		listOfSeq.Buffer = append(listOfSeq.Buffer, newSeq)
		listOfSeq.Neff += 1
```

```
            currentLine += 1
      }
      return bufferSize, gameMatrix, listOfSeq
}

func splitLines(s string) []string {
      return strings.FieldsFunc(s, func(r rune) bool {
            return r == '\n' || r == '\r'
      })
}

func saveResult(tokenMatrix matrix, reward int, optimalPath path, lastIdx
int, runtime time.Duration, filePath string) {
      var newPath string
      var err error
      if filePath != "" {
            extension := filepath.Ext(filePath)
            newPath = filePath[:len(filePath)-len(extension)]
            newPath += "-result" + ".txt"
      } else {
            var filename string
            fmt.Print("Masukkan nama file untuk menyimpan hasil: ")
            fmt.Scanln(&filename)
            exePath, err := os.Executable()
            if err != nil {
                  fmt.Println("Error:", err)
                  return
            }

            exeDir := filepath.Dir(exePath)
            projectRoot := filepath.Dir(exeDir)
            newPath = filepath.Join(projectRoot, "test", filename+".txt")
      }
      file, err := os.Create(newPath)

      if err != nil {
            fmt.Println("Error:", err)
            return
      }

      defer file.Close()

      _, err = file.WriteString(fmt.Sprintf("%d", reward))
      if err != nil {
            fmt.Println("Error:", err)
      }

      if reward != 0 {
            var tempString string = "\n"
            for i := 0; i <= lastIdx; i++ {
                  tempString +=
tokenMatrix.Buffer[optimalPath.Coordinates[i].Row][optimalPath.Coordinates[
i].Column]
                  if i != lastIdx {
```

```
                        tempString += " "
                  }
            }
            tempString += "\n"

            _, err = file.WriteString(tempString)
            if err != nil {
                  fmt.Println("Error:", err)
            }

            for i := 0; i <= lastIdx; i++ {
                  _, err = file.WriteString(fmt.Sprintf("%d, %d\n",
optimalPath.Coordinates[i].Column+1, optimalPath.Coordinates[i].Row+1))
                  if err != nil {
                        fmt.Println("Error:", err)
                  }
            }

            tempString = "\n" + runtime.String()

            _, err = file.WriteString(tempString)
            if err != nil {
                  fmt.Println("Error:", err)
            }
      }
}
```

functions.go (Algoritma utama dan inisiasi, termasuk i/o dengan command line)

```
package main

import (
      "bufio"
      "fmt"
      "math"
      "math/rand"
      "os"
      "path/filepath"
      "regexp"
      "strings"
      "time"
)

func initiate() (int, matrix, listOfSequence, string) {
      var input string

      fmt.Println("Pilih cara memberi masukkan\n1. Melalui file
\".txt\"\n2. Melalui Command Line (Matriks dan sekuens akan dihasilkan
secara acak)")
      fmt.Print("\nPilihan (1/2): ")
      fmt.Scanln(&input)
      fmt.Println()

      for input != "1" && input != "2" {
            fmt.Println("Pilihan tidak tersedia!")
            fmt.Println("Pilih cara memberi masukkan\n1. Melalui file
```

```
\".txt\"\n2. Melalui Command Line (Matriks dan sekuens akan dihasilkan
secara acak)")
            fmt.Print("\nPilihan (1/2): ")
            fmt.Scanln(&input)
            fmt.Println()
      }

      if input == "1" {
            var fileFound bool = false
            var relativePath string
            for !fileFound {
                  fmt.Print("Masukkan nama file (.txt): ")
                  fmt.Scanln(&input)

                  exePath, err := os.Executable()
                  if err != nil {
                        fmt.Println("Error:", err)
                        return 0, matrix{}, listOfSequence{}, ""
                  }

                  exeDir := filepath.Dir(exePath)
                  projectRoot := filepath.Dir(exeDir)
                  relativePath = filepath.Join(projectRoot, "test", input)

                  _, err = os.Stat(relativePath)

                  if err == nil {
                        fileFound = true
                  } else if os.IsNotExist(err) {
                        fmt.Println("\nFile tidak tersedia!")
                  }
            }
            bufferSize, tokenMatrix, sequences := parseFile(relativePath)
            return bufferSize, tokenMatrix, sequences, relativePath
      } else {
            bufferSize, tokenMatrix, sequences := initiateRandomly()

            fmt.Println("\nMatriks:")
            for i := 0; i < tokenMatrix.Height; i++ {
                  for j := 0; j < tokenMatrix.Width; j++ {
                        fmt.Print(tokenMatrix.Buffer[i][j])
                        if j != tokenMatrix.Width-1 {
                              fmt.Print(" ")
                        }
                  }
                  fmt.Println()
            }

            fmt.Println("\nSekuens:")
            for i := 0; i < sequences.Neff; i++ {
                  for j := 0; j < len(sequences.Buffer[i].Tokens); j++ {
                        fmt.Printf("%s ", sequences.Buffer[i].Tokens[j])
                  }
                  fmt.Printf("Reward: %d\n", sequences.Buffer[i].Reward)
```

```
            }
            return bufferSize, tokenMatrix, sequences, ""
      }
}

func initiateRandomly() (int, matrix, listOfSequence) {
      var numOfTokens int
      var inputString string
      var bufferSize int
      var width, height int
      var tokenMatrix matrix
      var numOfSequences int
      var maxSequenceLength int
      var sequences listOfSequence
      var tokens []string

      fmt.Print("Masukkan jumlah token: ")
      fmt.Scanln(&numOfTokens)
      for numOfTokens <= 0 {
            fmt.Print("Jumlah token harus positif! Masukkan jumlah token:
")
            fmt.Scanln(&numOfTokens)
      }

      fmt.Print("Masukkan token: ")
      inputScanner := bufio.NewScanner(os.Stdin)
      if inputScanner.Scan() {
            inputString = inputScanner.Text()
      }
      tokens = strings.Split(inputString, " ")

      for len(tokens) != numOfTokens || !checkToken(tokens) {
            fmt.Printf("Token tidak sesuai! Masukkan token (Sebanyak %d dan
alfanumerik dua karakter): ", numOfTokens)
            inputScanner := bufio.NewScanner(os.Stdin)
            if inputScanner.Scan() {
                  inputString = inputScanner.Text()
            }
            tokens = strings.Split(inputString, " ")
      }

      fmt.Print("Masukkan ukuran buffer: ")
      fmt.Scanln(&bufferSize)

      for bufferSize <= 0 {
            fmt.Print("Ukuran buffer harus positif! Masukkan ukuran buffer:
")
            fmt.Scanln(&bufferSize)
      }

      fmt.Print("Masukkan lebar dan tinggi matriks: ")
      fmt.Scanf("%d %d\n", &width, &height)

      for width <= 0 || height <= 0 {
```

```
            fmt.Print("Lebar dan tinggi matriks harus positif! Masukkan
lebar dan tinggi matriks: ")
            fmt.Scanf("%d %d\n", &width, &height)
      }

      fmt.Print("Masukkan jumlah sekuens: ")
      fmt.Scanln(&numOfSequences)

      for numOfSequences <= 0 {
            fmt.Print("Jumlah sekuens harus positif! Masukkan jumlah
sequence: ")
            fmt.Scanln(&numOfSequences)
      }

      minSequenceLength := 2
      totalCombination := int(math.Pow(float64(numOfTokens),
float64(minSequenceLength)))
      for totalCombination < numOfSequences {
            minSequenceLength++
            totalCombination += int(math.Pow(float64(numOfTokens),
float64(minSequenceLength)))
      }

      fmt.Printf("Masukkan panjang sekuens maksimal (minimal %d): ",
minSequenceLength)
      fmt.Scanln(&maxSequenceLength)

      for maxSequenceLength < minSequenceLength {
            fmt.Printf("Panjang maksimal kurang! Masukkan panjang sekuens
maksimal (minimal %d): ", minSequenceLength)
            fmt.Scanln(&maxSequenceLength)
      }

      source := rand.NewSource(time.Now().UnixNano())
      random := rand.New(source)

      tokenMatrix.Buffer = make([][]string, height)
      for i := 0; i < height; i++ {
            tokenMatrix.Buffer[i] = make([]string, width)
            for j := 0; j < width; j++ {
                  tokenMatrix.Buffer[i][j] =
tokens[random.Intn(numOfTokens)]
            }
      }
      tokenMatrix.Width, tokenMatrix.Height = width, height

      for sequences.Neff < numOfSequences {
            var seq sequence
            currentSeqLength := random.Intn(maxSequenceLength-1) + 2
            for j := 0; j < currentSeqLength; j++ {
                  seq.Tokens = append(seq.Tokens,
tokens[random.Intn(numOfTokens)])
            }
            seq.Reward = (random.Intn(10) + 1) * 5 // kelipatan 10, 0..50
```

```
            if !isInSequences(sequences, seq) {
                  sequences.Buffer = append(sequences.Buffer, seq)
                  sequences.Neff++
            }
      }

      return bufferSize, tokenMatrix, sequences
}

func isInSequences(sequences listOfSequence, seq sequence) bool {
      found := false
      i := 0
      for i < sequences.Neff && !found {
            currentSequence := sequences.Buffer[i]
            if len(currentSequence.Tokens) == len(seq.Tokens) {
                  same := true
                  j := 0
                  for j < len(currentSequence.Tokens) && same {
                        if currentSequence.Tokens[j] != seq.Tokens[j] {
                              same = false
                        } else {
                              j++
                        }
                  }
                  if same {
                        found = true
                  } else {
                        i++
                  }
            } else {
                  i++
            }
      }
      return found
}

func checkToken(tokens []string) bool {
      pattern := "^[a-zA-Z0-9][a-zA-Z0-9]$"
      regex := regexp.MustCompile(pattern)

      for i := 0; i < len(tokens); i++ {
            if !regex.MatchString(tokens[i]) {
                  return false
            }
      }
      return true
}

func findAllPaths(tokenMatrix matrix, bufferSize int, paths *listOfPath) {
      if bufferSize > 0 {
            width := tokenMatrix.Width
            for i := 0; i < width; i++ {
                  initialPath := path{[]point{{i, 0}}, 1}
                  findPath(tokenMatrix, bufferSize, 0, &initialPath, paths)
```

```
            }
      }
}

func findPath(tokenMatrix matrix, bufferSize int, iterator int, tempPath
*path, paths *listOfPath) {
      if iterator == bufferSize-1 {
            newPath := path{
                  Coordinates: make([]point, len(tempPath.Coordinates)),
                  Neff:        tempPath.Neff,
            }
            copy(newPath.Coordinates, tempPath.Coordinates)
            paths.Buffer = append(paths.Buffer, newPath)
            paths.Neff++
            return
      } else {
            width := tokenMatrix.Width
            height := tokenMatrix.Height

            var currentPoint point = tempPath.Coordinates[tempPath.Neff-1]
            // gerak vertikal
            if iterator%2 == 0 {
                  for j := 1; j < height; j++ {
                        tempPoint := point{currentPoint.Column,
(currentPoint.Row + j) % height}
                        if !isInPath(tempPoint, *tempPath) {
                              tempPath.Coordinates =
append(tempPath.Coordinates, tempPoint)
                              tempPath.Neff++
                              findPath(tokenMatrix, bufferSize, iterator+1,
tempPath, paths)
                              tempPath.Coordinates =
tempPath.Coordinates[:tempPath.Neff-1]
                              tempPath.Neff--
                        }
                  }
            } else { // horizontal
                  for j := 1; j < width; j++ {
                        tempPoint := point{(currentPoint.Column + j) %
width, currentPoint.Row}
                        if !isInPath(tempPoint, *tempPath) {
                              tempPath.Coordinates =
append(tempPath.Coordinates, tempPoint)
                              tempPath.Neff++
                              findPath(tokenMatrix, bufferSize, iterator+1,
tempPath, paths)
                              tempPath.Coordinates =
tempPath.Coordinates[:tempPath.Neff-1]
                              tempPath.Neff--
                        }
                  }
            }
      }
}
```

```
func isInPath(coordinate point, p path) bool {
	i := 0
	found := false
	for i < p.Neff && !found {
		if p.Coordinates[i] == coordinate {
			found = true
		} else {
			i++
		}
	}
	return found
}

func findOptimalPath(bufferSize int, tokenMatrix matrix, paths listOfPath,
sequences listOfSequence) (path, int, int) {
	if paths.Neff != 0 {
		maxReward := 0
		minIdx := 0
		idxPath := 0
		for idx, path := range paths.Buffer {
			reward, lastIdx := findReward(tokenMatrix, path,
sequences)
			if reward > maxReward {
				maxReward = reward
				minIdx = lastIdx
				idxPath = idx
			} else if reward == maxReward {
				if lastIdx < minIdx {
					minIdx = lastIdx
					idxPath = idx
				}
			}
		}
		return paths.Buffer[idxPath], minIdx, maxReward
	} else {
		return path{}, 0, 0
	}
}

func findReward(tokenMatrix matrix, path path, sequences listOfSequence)
(int, int) {
	lastIdx := 0
	reward := 0
outerLoop:
	for _, val := range sequences.Buffer {
		// mencoba semua sequence
		maxIterationIdx := path.Neff - len(val.Tokens)
		for i := 0; i <= maxIterationIdx; i++ {
			// i menunjukkan indeks path, titik awal menyocokkan
sequence
			j := 0
			stillTrue := true
			for j < len(val.Tokens) && stillTrue {
```

```
                        if val.Tokens[j] !=
tokenMatrix.Buffer[path.Coordinates[i+j].Row][path.Coordinates[i+j].Column]
{
                              stillTrue = false
                        } else {
                              j++
                        }
                  }
                  if stillTrue {
                        tempLastIdx := j + i - 1
                        if tempLastIdx > lastIdx {
                              lastIdx = tempLastIdx
                        }
                        reward += val.Reward
                        continue outerLoop
                  }
            }
      }
      return reward, lastIdx
}

func outputResult(tokenMatrix matrix, reward int, optimalPath path, lastIdx
int, startTime time.Time) (bool, time.Duration) {
      var runtime time.Duration

      fmt.Printf("\n%d\n", reward)
      if optimalPath.Neff != 0 {
            for i := 0; i <= lastIdx; i++ {
                  fmt.Printf("%s",
tokenMatrix.Buffer[optimalPath.Coordinates[i].Row][optimalPath.Coordinates[
i].Column])
                  if i != lastIdx {
                        fmt.Print(" ")
                  } else {
                        fmt.Println()
                  }
            }
            for i := 0; i <= lastIdx; i++ {
                  fmt.Printf("%d, %d\n",
optimalPath.Coordinates[i].Column+1, optimalPath.Coordinates[i].Row+1)
            }
            endTime := time.Now()
            runtime = endTime.Sub(startTime)
            fmt.Println()
            fmt.Println(runtime)
            fmt.Println()
      }
      var save string

      for save != "y" && save != "n" {
            fmt.Print("Apakah ingin menyimpan hasil? (y/n) ")
            fmt.Scanln(&save)
      }
```

```
      doSave := (save == "y")
      return doSave, runtime
}
```

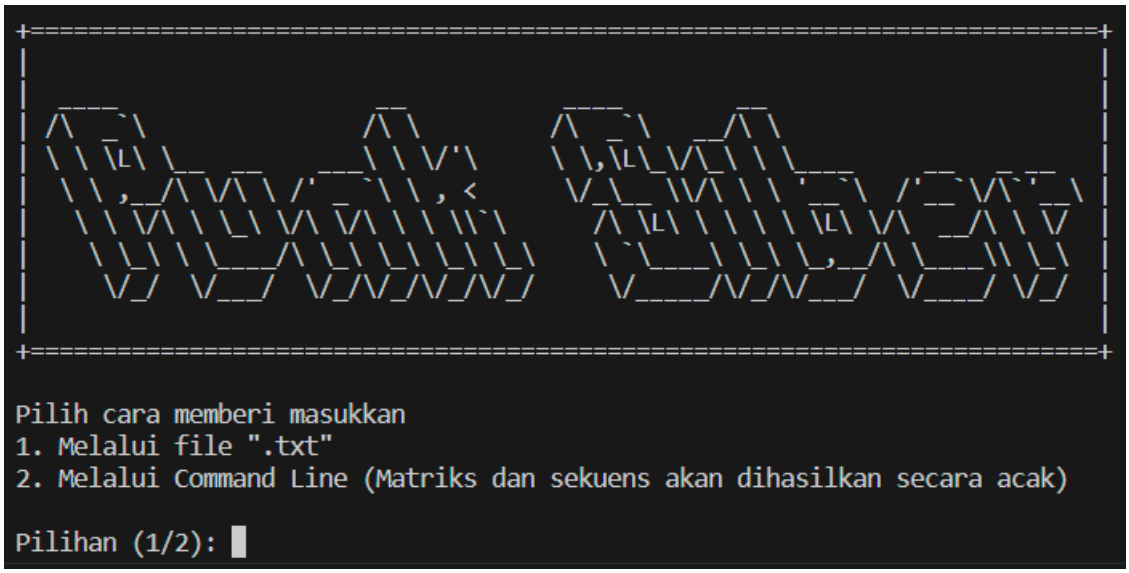
main.go (Program utama)

```
package main

import (
      "fmt"
      "time"
)

func main() {
      fmt.Println()
      fmt.Println("+=========================================================
==================+")
      fmt.Println("|
|")
      fmt.Println("|  ____                    __           ____        __
|")
      fmt.Println("| /\\  _`\\                 /\\ \\         /\\  _`\\
__/\\ \\                        |")
      fmt.Println("| \\ \\ \\ \\L\\ \\__  __     ___\\ \\ \\/'\\      \\
\\,\\L\\_\\/\\_\\ \\ \\____      __   _ __   |")
      fmt.Println("|  \\ \\ ,__/\\ \\/\\ \\ /' _ `\\ \\ , <      \\/_\\__
\\\\/\\ \\ \\ '__`\\  /'__`\\/\\`'__\\ |")
      fmt.Println("|   \\ \\ \\/\\ \\ \\_\\ \\/\\ \\/\\ \\ \\ \\\\`\\
/\\ \\L\\ \\ \\ \\ \\L\\ \\/\\  __/\\ \\ \\/  |")
      fmt.Println("|    \\ \\_\\ \\ \\____/\\ \\_\\ \\_\\ \\_\\ \\_\\    \\
`\\____\\ \\_\\ \\_,__/\\ \\____\\\\ \\_\\  |")
      fmt.Println("|     \\/_/  \\/___/  \\/_/\\/_/\\/_/\\/_/
\\/_____/\\/_/\\/___/  \\/____/ \\/_/  |")
      fmt.Println("|
|")
      fmt.Println("+=========================================================
==================+")
      fmt.Println()

      play := true

      for play {
            var paths listOfPath

            bufferSize, tokenMatrix, sequences, filePath := initiate()
            startTime := time.Now()
            findAllPaths(tokenMatrix, bufferSize, &paths)
            optimalPath, lastIdx, reward := findOptimalPath(bufferSize,
tokenMatrix, paths, sequences)
            doSave, runtime := outputResult(tokenMatrix, reward,
optimalPath, lastIdx, startTime)
            if doSave {
                  saveResult(tokenMatrix, reward, optimalPath, lastIdx,
runtime, filePath)
            }
```

```
            var input string
            for input != "y" && input != "n" {
                  fmt.Print("\nApakah ingin bermain lagi? (y/n) ")
                  fmt.Scanln(&input)
            }
            if input == "n" {
                  play = false
                  fmt.Println()
                  fmt.Println("  ____
")
                  fmt.Println(" /\\  _`\\
__     ")
                  fmt.Println(" \\ \\,\\L\\_\\     __      ___ ___   _____
__   /\\_\\   ")
                  fmt.Println("  \\/_\\__ \\   /'__`\\  /' __` __`\\/\\
'__`\\  /'__`\\ \\/\\ \\  ")
                  fmt.Println("    /\\ \\L\\ \\/\\ \\L\\.\\_/\\ \\/\\ \\/\\
\\ \\ \\L\\ \\/\\ \\L\\.\\_\\ \\ \\ ")
                  fmt.Println("    \\ `\\____\\ \\__/.\\_\\ \\_\\ \\_\\
\\_\\ \\ ,__/\\ \\__/.\\_\\\\ \\_\\")
                  fmt.Println("     \\/_____/\\/__/\\/_/\\/_/\\/_/\\/_/\\
\\ \\/  \\/__/\\/_/ \\/_/")
                  fmt.Println("                                    \\ \\_\\
")
                  fmt.Println("                                     \\/_/
")
                  fmt.Println("  _____
__        ")
                  fmt.Println(" /\\___ \\
/\\ \\      ")
                  fmt.Println(" \\/__/\\ \\  __  __    ___ ___   _____
__  \\ \\ \\     ")
                  fmt.Println("    _\\ \\ \\/\\ \\/\\ \\ /' __` __`\\/\\
'__`\\  /'__`\\ \\ \\ \\    ")
                  fmt.Println("   /\\ \\_\\ \\ \\ \\ \\_\\ \\/\\ \\/\\ \\/\\
\\ \\ \\L\\ \\/\\ \\L\\.\\_\\ \\_\\   ")
                  fmt.Println("   \\ \\____/\\ \\____/\\ \\_\\ \\_\\ \\_\\
\\ ,__/\\ \\__/.\\_\\\\\/\\_\\  ")
                  fmt.Println("    \\/___/  \\/___/  \\/_/\\/_/\\/_/\\ \\
\\/  \\/__/\\/_/ \\/_/  ")
                  fmt.Println("                                    \\ \\_\\
")
                  fmt.Println("                                     \\/_/
")
                  fmt.Println()
            } else {
                  fmt.Println()
            }
      }
}
```

BAB IV
UJI COBA PROGRAM

Program untuk menyelesaikan permainan *Cyberpunk 2077 Breach Protocol* dikompilasi dengan menjalankan “go build -o bin/main src/main.go src/functions.go src/txtprocessor.go src/type.go” pada direktori utama, jika digunakan sistem operasi Linux. Setelah dikompilasi, jalankan printah “.\bin\main” lalu program akan mengeluarkan tampilan awal. Agar lebih sederhana, cukup jalankan perintah “./run.sh” pada direktori utama.



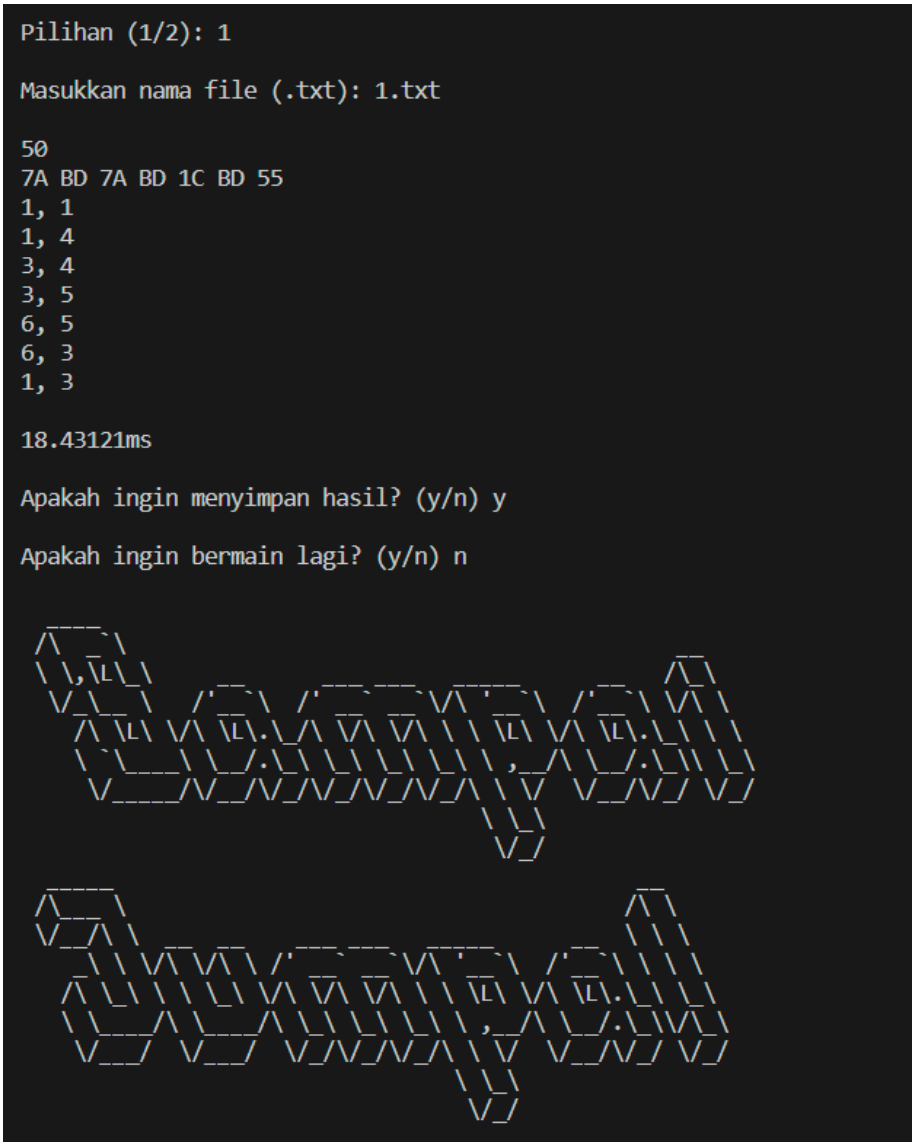
Gambar 5.1 Tampilan Awal Program

Berikutnya pengguna dapat memilih sesuai pilihan yang tersedia pada menu. Berikut beberapa data uji sebagai contoh input output program.

1. Data Uji 1

```
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 5.2 Masukkan Data Uji 1



Gambar 5.3 Keluaran Data Uji 1

2. Data Uji 2

```
7
3 3
55 1C 55
BD 7A BD
E9 BD 7A
2
1C 7A BD E9 7A 55 55
10
55 BD
5
```

Gambar 5.4 Data Uji 2

```
Pilihan (1/2): 1

Masukkan nama file (.txt): 2.txt

10
1C 7A BD E9 7A 55 55
2, 1
2, 2
1, 2
1, 3
3, 3
3, 1
1, 1

67.5µs

Apakah ingin menyimpan hasil? (y/n) y

Apakah ingin bermain lagi? (y/n) n
```

Gambar 5.5 Keluaran Data Uji 2

3. Data Uji 3

```
6
5 5
7A 1C 1C BD E9
1C BD BD 1C E9
E9 7A 1C E9 BD
1C BD BD E9 BD
1C 1C 7A E9 7A
2
7A 1C 1C
10
BD E9 BD 7A
30
```

Gambar 5.6 Data Uji 3

```
Pilihan (1/2): 1

Masukkan nama file (.txt): 3.txt

40
BD E9 BD 7A 1C 1C
4, 1
4, 3
5, 3
5, 5
1, 5
1, 2

557.418µs

Apakah ingin menyimpan hasil? (y/n) y

Apakah ingin bermain lagi? (y/n) n
```

Gambar 5.7 Keluaran Data Uji 3

4. Data Uji 4

```
5
5 4
7A BD 1C 55 E9
E9 1C BD 7A 1C
7A 1C E9 BD BD
BD E9 1C 7A 55
2
BD 7A
5
E9 55 1C BD
20
```

Gambar 5.8 Data Uji 4

```
Pilihan (1/2): 1

Masukkan nama file (.txt): 4.txt

25
E9 55 1C BD 7A
5, 1
5, 4
3, 4
3, 2
4, 2

133.675µs

Apakah ingin menyimpan hasil? (y/n) y

Apakah ingin bermain lagi? (y/n) n
```

Gambar 5.9 Keluaran Data Uji 4

5. Data Uji 5

```
Pilihan (1/2): 2

Masukkan jumlah token: 5
Masukkan token: AA BB CC DD EE
Masukkan ukuran buffer: 7
Masukkan lebar dan tinggi matriks: 3 9
Masukkan jumlah sekuens: 3
Masukkan panjang sekuens maksimal (minimal 2): 2

Matriks:
DD BB DD
AA AA CC
AA AA BB
CC AA CC
CC EE EE
EE AA CC
AA DD DD
AA DD EE
EE BB BB

Sekuens:
CC BB Reward: 40
CC CC Reward: 40
EE CC Reward: 40
```

Gambar 5.10 Data Uji 5

```
120
DD CC CC BB EE CC
1, 1
1, 4
3, 4
3, 9
1, 9
1, 5

2.252298ms

Apakah ingin menyimpan hasil? (y/n) y
Masukkan nama file untuk menyimpan hasil: 5-result

Apakah ingin bermain lagi? (y/n) n
```

Gambar 5.11 Keluaran Data Uji 5

6. Data Uji 6

```
Pilihan (1/2): 2

Masukkan jumlah token: 4
Masukkan token: IF TK SF TI MA
Token tidak sesuai! Masukkan token (Sebanyak 4 dan alfanumerik dua karakter): IF TK SF MA
Masukkan ukuran buffer: 6
Masukkan lebar dan tinggi matriks: 3 7
Masukkan jumlah sekuens: 3
Masukkan panjang sekuens maksimal (minimal 2): 2

Matriks:
MA MA IF
TK TK TK
SF IF MA
MA IF SF
MA MA MA
SF IF TK
MA IF MA
```

Gambar 5.12 Data Uji 6

```
40
MA IF TK MA
2, 1
2, 6
3, 6
3, 7

700.775µs

Apakah ingin menyimpan hasil? (y/n) y
Masukkan nama file untuk menyimpan hasil: 6-result

Apakah ingin bermain lagi? (y/n) n
```

Gambar 5.13 Keluaran Data Uji 6

Program juga melakukan validasi terhadap input user. Berikut beberapa tampilan validasi yang dilakukan program.

```
Pilih cara memberi masukkan
1. Melalui file ".txt"
2. Melalui Command Line (Matriks dan sekuens akan dihasilkan secara acak)

Pilihan (1/2): 5

Pilihan tidak tersedia!
Pilih cara memberi masukkan
1. Melalui file ".txt"
2. Melalui Command Line (Matriks dan sekuens akan dihasilkan secara acak)

Pilihan (1/2): 1
```

Gambar 5.14 Validasi Pilihan *Input*

```
Pilihan (1/2): 2

Masukkan jumlah token: 4
Masukkan token: AA BB CC DD
Masukkan ukuran buffer: 6
Masukkan lebar dan tinggi matriks: 3 3
Masukkan jumlah sekuens: 4
Masukkan panjang sekuens maksimal (minimal 2): 1
Panjang maksimal kurang! Masukkan panjang sekuens maksimal (minimal 2): 3
```

Gambar 5.15 Validasi Ukuran Sekuens Agar Sekuens Unik

```
Apakah ingin bermain lagi? (y/n) y

Pilih cara memberi masukkan
1. Melalui file ".txt"
2. Melalui Command Line (Matriks dan sekuens akan dihasilkan secara acak)

Pilihan (1/2):
```

Gambar 5.16 Validasi Jika *User* Ingin Bermain Lagi

REFERENSI

R. Munir, “Algoritma Brute Force (Bagian I),” *IF2211 Strategi Algoritma*, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/20212022/Algoritma-Brute-Force-(2022)-Bag1.pdf

LAMPIRAN

Tabel Hasil

Poin	**Ya**	**Tidak**
Program berhasil dikompilasi tanpa kesalahan	√	
Program berhasil dijalankan	√	
Program dapat membaca masukkan berkas .txt	√	
Program dapat menghasilkan masukkan secara acak	√	
Solusi yang diberikan program optimal	√	
Program dapat menyimpan solusi dalam berkas .txt	√	
Program memiliki GUI		√

Pranala Github

Repository Github dapat diakses melalui pranala berikut:

https://github.com/julianchandras/Tucil1_13522080.git