Laporan Tugas Kecil 1

Penyelesaian pola paling optimal untuk *minigames*

Breach Protocol Cyberpunk 2077

Disusun oleh:

Zaki Yudhistira Candra 13522031 (K01)

Mata Kuliah IF 2211 - Strategi Algoritma

Program Studi S1 Teknik Informatika

Sekolah Teknik Elektro dan Informatika

A. Algoritma Brute Force dalam mencari pola optimal

Akan digunakan teknik *exhaustive search* dalam mencari solusi paling optimal dari *minigames breach protocol Cyberpunk 2077.* Adapun langkah-langkah algoritma yang diambil adalah sebagai berikut :

1. Akan dipilih elemen matriks paling kiri atas sebagai titik permulaan.
2. Algoritma akan melakukan *traversal* pada setiap elemen dan arah proses tergantung dari *state* horizontalitasnya. Apabila dalam fase horizontal, *traversal* akan dilakukan mendatar atau dengan elemen dalam baris yang sama. Sebaliknya, bila pemrosesan mendatar selesai, akan dilakukan pemrosesan secara vertikal, atau elemen dalam kolom yang sama.
3. Horizontalitas pemrosesan akan dilakukan secara bergantian.
4. Setelah memilih elemen, akan dilakukan *traversal* secara vertikal terlebih dahulu dan pada setiap elemen, akan dilakukan pemanggilan fungsi secara rekursif. Setiap elemen pada kolom atau baris akan diproses.
5. Basis dari rekursi adalah ketika *buffer* atau panjang token mencapai batas maksimal.
6. Akan diulangi algoritma ini hingga seluruh pola yang mungkin dapat terbentuk. Baris pertama merupakan titik awal dari algoritma.

Dalam algoritma ini, akan digunakan 3 jenis data, yaitu *sequence, matrix,* dan *token.* Berikut adalah kegunaan dari tipe data terkait :

1. Sequence : Terdiri dari pola token dan *score* yang diperoleh bila berhasil memecahkannya.
2. Matrix : Terdiri dari konten matriks yaitu token, jumlah baris, dan jumlah kolom.
3. Token : Merupakan elemen satuan dari matriks, berisi *string* token, koordinat x, dan koordinat y relatif terhadap matriks.

B. Source code program

```
# Creator : Zaki Yudhistira Candra
# NIM     : 13522031
# Cyberpunk breach protocol solver

import random as rd
import time as tm

# Global variable
path = "test/Input/"
save_path = "test/Solution"

class Sequence: # Contains sequence code and its score
    def __init__(self, sequence, points):
        self.sequence = sequence
        self.points = points

    def printData(self):
        first = True
        for seq in self.sequence:
            if first:
                print(seq, end="")
                first = False
            else:
                print(" " + seq, end="")
        print("\nPoints : " + str(self.points))

def printSequences(sequences): # For displaying sequences
    first = True
    print(">>> Sequences <<<")
    for i in range (len(sequences)):
        if first:
            Sequence.printData(sequences[i])
            first = False
        else:
            print("")
            Sequence.printData(sequences[i])

def generateSequence(tokens, max, number_of_sequence):
    # Randomly generate sequences
    sequences = []
    for i in range(number_of_sequence):
        temp = []
        for j in range(rd.randint(1,max)):
            index = rd.randint(0, len(tokens)-1)
            temp.append(tokens[index])
        sequences.append(Sequence(temp, rd.randint(1,10)*10))
    return sequences
```

Gambar 2.1. Bagian inisialisasi *Class Sequence* dan Fungsi *printSequence*

```python
class Token: # Token class
    def __init__(self, token, x, y):
        self.token = token
        self.x = x
        self.y = y

def compareToken(tokens, seq):
    # Comparing 2 array of tokens and returns a boolean
    if len(seq) > len(tokens):
        return False
    else:
        for i in range(len(tokens) - len(seq) + 1):
            if seq[0] == tokens[i]:
                flag = True
                for j in range(1, len(seq)):
                    if seq[j] != tokens[i+j]:
                        flag = False
                        break
                if flag:
                    return True
        return False

def getTokenList(tokens):
    # Converting from an array of token into an array of strings
    return [token.token for token in tokens]
```

Gambar 2.2 Inisialisasi dari *Class Token* dan fungsi terkait

```python
class Matrix: # Matrix class where tokens are placed
    def __init__(self, content, row, column):
        self.content = content
        self.column = column
        self.row = row
        self.size = row*column

    def printMatrix(self):
        for i in range(self.row):
            first = True
            for j in range(self.column):
                if first:
                    print(self.content[i][j].token, end="")
                    first = False
                else:
                    print(" " + self.content[i][j].token, end="")
            print("")

def generateMatrix(tokens, row, column):
    # Randomly matrix generation
    matrix = []
    for i in range(row):
        line = []
        for j in range(column):
            index = rd.randint(0, len(tokens) - 1)
            line.append(Token(tokens[index],i,j))
        matrix.append(line)
    return Matrix(matrix, row, column)
```

Gambar 2.3 Inisialisasi dari *Class Matrix* dan fungsi terkait

```python
def getScore(sequences, tokens):
    # Calculate possible score from the sequences and a solution
    score = 0
    for seq in sequences:
        if compareToken(getTokenList(tokens), seq.sequence):
            score += seq.points
    return score

def getSolution(solutions, sequences):
    # Solving algorithm for finding the most optimal path / sequence
    max = getScore(sequences, solutions[0])
    solve = solutions[0]
    for i in range(1, len(solutions)):
        if getScore(sequences, solutions[i]) > max:
            max = getScore(sequences, solutions[i])
            solve = solutions[i]
    return max, solve
```

Gambar 2.4 Fungsi tambahan

```python
print("\n>>> BREACH PROTOCOL <<<")
Matrix.printMatrix(main_matrix)
print("")
printSequences(sequences)

print("\nBuffer size :", buffer_size)
```

Gambar 2.5 *Display* matrix dan sikuens

```
# Welcome menu
print("|| BREACH PROTOCOL SOLVER ||")
print("----------------------------")
print("||   By Zaki Yudhistira   ||")
print("----------------------------")

# Option to load a custom breach protocol
initiation = input("Do you want to load a custom protocol ? (y/n) : ")
initiation = initiation.lower()
if initiation == "y":
    # File handling
    initiation = input("Please enter file name  : ")
    path = path + initiation
    try:
        file = open(path, 'r')
    except FileNotFoundError:
        print(initiation + " is not found, please recheck your filename.")
        exit()

    # Initialization
    buffer_size = int(file.readline())
    matrix_WnH = file.readline()
    matrix_WnH = matrix_WnH.split(" ")
    matrix_length = int(matrix_WnH[0])
    matrix_width = int(matrix_WnH[1])
    main_matrix = []
    for i in range(matrix_length):
        lines = file.readline()
        lines = lines.rstrip('\n')
        lines = lines.split(" ")
        main_matrix.append(lines)
    for i in range(matrix_width):
        for j in range(matrix_length):
            main_matrix[i][j] = Token(main_matrix[i][j], i, j)
    main_matrix = Matrix(main_matrix, matrix_length, matrix_width)
    sequence_count = int(file.readline())
    sequences = []
    for i in range(sequence_count*2):
        line = file.readline()
        if i % 2 == 0:
            s_temp = (line.rstrip('\n')).split(" ")
        else:
            sequences.append(Sequence(s_temp, int(line)))

elif initiation == "n":
    # Data input
    number_of_token = int(input("Please input the number of tokens : "))
    print("Please provide the tokens below :")
    tokens = str(input())
    tokens = tokens.upper()
    tokens = tokens.split(" ")
    if len(tokens) != number_of_token:
        print("Token invalid, exiting program.")
        exit()
    buffer_size = int(input("| Enter buffer size : "))
    sequence_count = int(input("| Enter sequence count   : "))
    max_sequence_size = int(input("| Enter maximum sequence length : "))
    matrix_WnH = input("Enter matrix row and column (row column) : ")
    matrix_WnH = matrix_WnH.split(" ")
    row = int(matrix_WnH[0])
    column = int(matrix_WnH[1])

    # Generation
    print("Generating breach protocol...")
    main_matrix = generateMatrix(tokens, row, column)
    sequences = generateSequence(tokens, max_sequence_size, sequence_count)
else:
    print("Command is not recognized, exiting program...")
    exit()
```

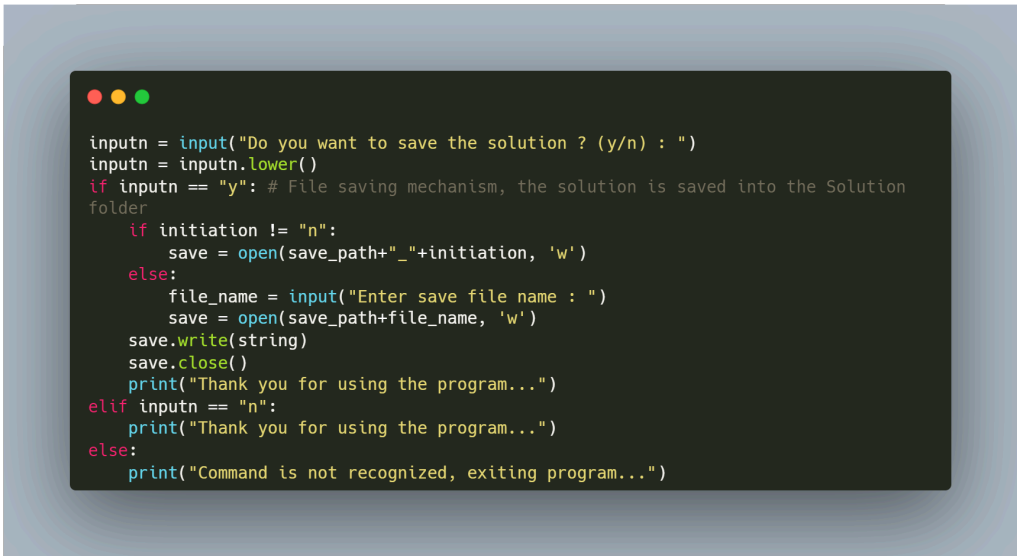
Gambar 2.6 Pembacaan *file* dan *random matrix generation*

```
# Solving algorithm
def isIn(row, column, stack):
# isIn function to detect wether a token is already selected or not
    if stack != None:
        for token in stack:
            if (token.x, token.y) == (column, row):
                return True
    return False

def searchSequence(matrix : Matrix, stack, row, column, buffer, solution, horizontal):
# Solution finding algorithm
    if buffer == 1:
        solution.append(list(stack))
        # Recursion basis
    else:
        if horizontal:
        # Horizontal (column) traversal
            for i in range(matrix.column):
                if not isIn(row, i, stack):
                    stack.append(matrix.content[row][i])
                    searchSequence(matrix, stack, row, i, buffer - 1, solution, False)
                    stack.pop()
        else:
        # Vertical (row) traversal
            for i in range(matrix.row):
                if not isIn(i, column, stack):
                    stack.append(matrix.content[i][column])
                    searchSequence(matrix, stack, i, column, buffer - 1, solution, True)
                    stack.pop()
```

Gambar 2.7 Algoritma *brute force exhaustive search*

```
# Main execution
horizontal = True
stack = []
solution = []
start = tm.time()
searchSequence(main_matrix, stack, 0, 0, buffer_size+1, solution, True)
a,b = getSolution(solution, sequences) # Retrieving results in tuple form
string ="" # To be saved string
end = tm.time()
if a != 0:
    print(a)
    first = True
    for i in b:
        print(i.token, end=' ')
        if not first:
            string += " "+i.token
        else:
            string += i.token
            first = False
    print("")
    for i in b:
        print(str(i.y+1)+", "+str(i.x+1))
        string += '\n' + str(i.y+1)+", "+str(i.x+1)
else:
    print("No optimum solution found")

print("")
print(int((end-start)*1000), "ms")
string += "\n\nRuntime : " + str(int((end-start)*1000)) + " ms"
```

Gambar 2.8 Eksekusi utama program



```
inputn = input("Do you want to save the solution ? (y/n) : ")
inputn = inputn.lower()
if inputn == "y": # File saving mechanism, the solution is saved into the Solution
folder
    if initiation != "n":
        save = open(save_path+"_"+initiation, 'w')
    else:
        file_name = input("Enter save file name : ")
        save = open(save_path+file_name, 'w')
    save.write(string)
    save.close()
    print("Thank you for using the program...")
elif inputn == "n":
    print("Thank you for using the program...")
else:
    print("Command is not recognized, exiting program...")
```

Gambar 2.9 Bagian penyimpanan *file*

C. Cara menjalankan program

Sebelum menjalankan program, pengguna harus menggunakan sistem operasi minimal windows 7 ataupun sistem operasi linux yang sudah terinstall python versi 3.9.x. Disarankan untuk menggunakan python dengan versi yang lebih baru demi performa yang lebih baik. Berikut adalah cara menjalankan program.

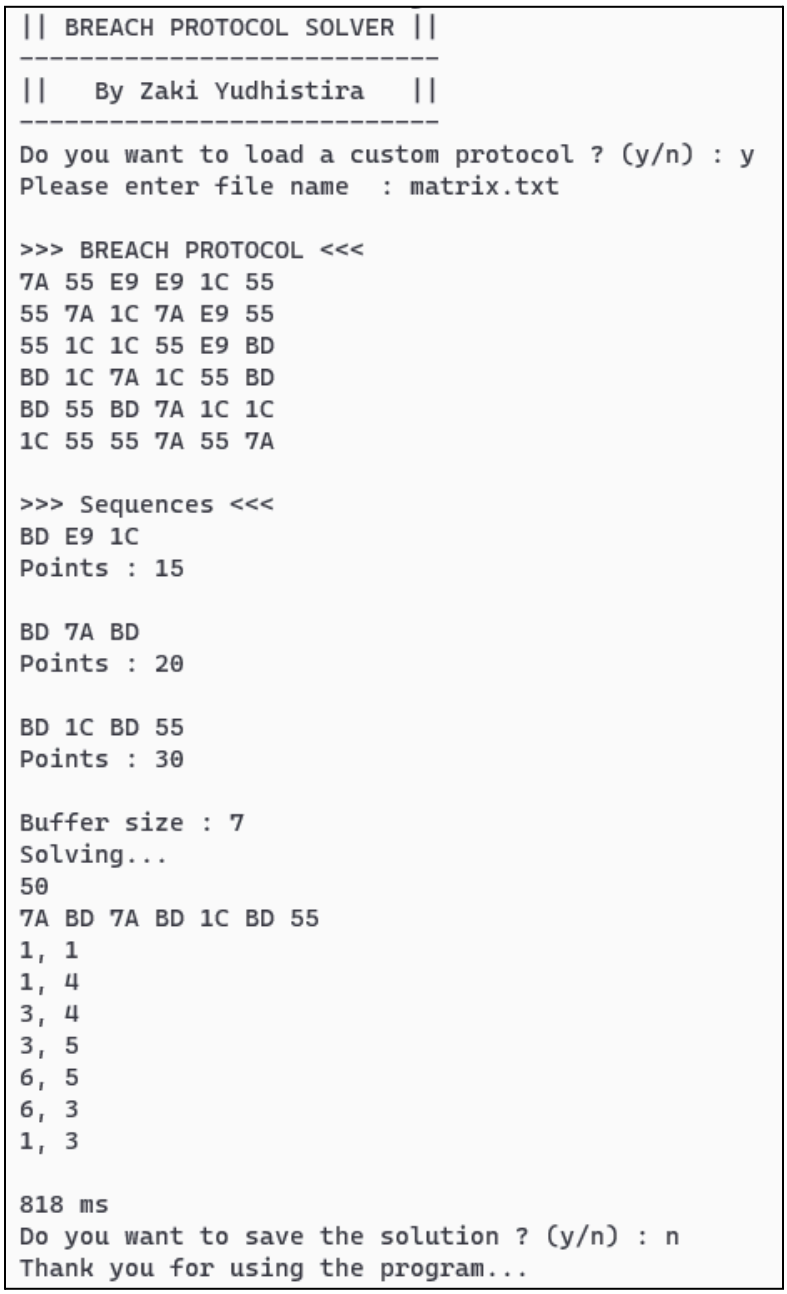
1. Buka terminal pada direktori program.
2. Untuk pengguna windows, silahkan ketik "`python src/Main.py`" pada terminal.
3. Anda dapat menggunakan file *custom* dengan menaruh file pada direktori "`test/Input`" atau menciptakan matriks dan *sequence* baru secara acak.
4. Ikuti arahan program.
5. File yang disimpan dapat dibuka pada *folder* dengan direktori "`test/Solution`"

```
PS F:\Documents Archive\Tugas Sekolah\ITB\SEM 4\IF2211 Strategi Algoritma\Tubes Tucil\Repo Tucil\Tucil1_13522031> python src/Main.py
|| BREACH PROTOCOL SOLVER ||
----------------------------
||   By Zaki Yudhistira   ||
----------------------------
Do you want to load a custom protocol ? (y/n) :
```

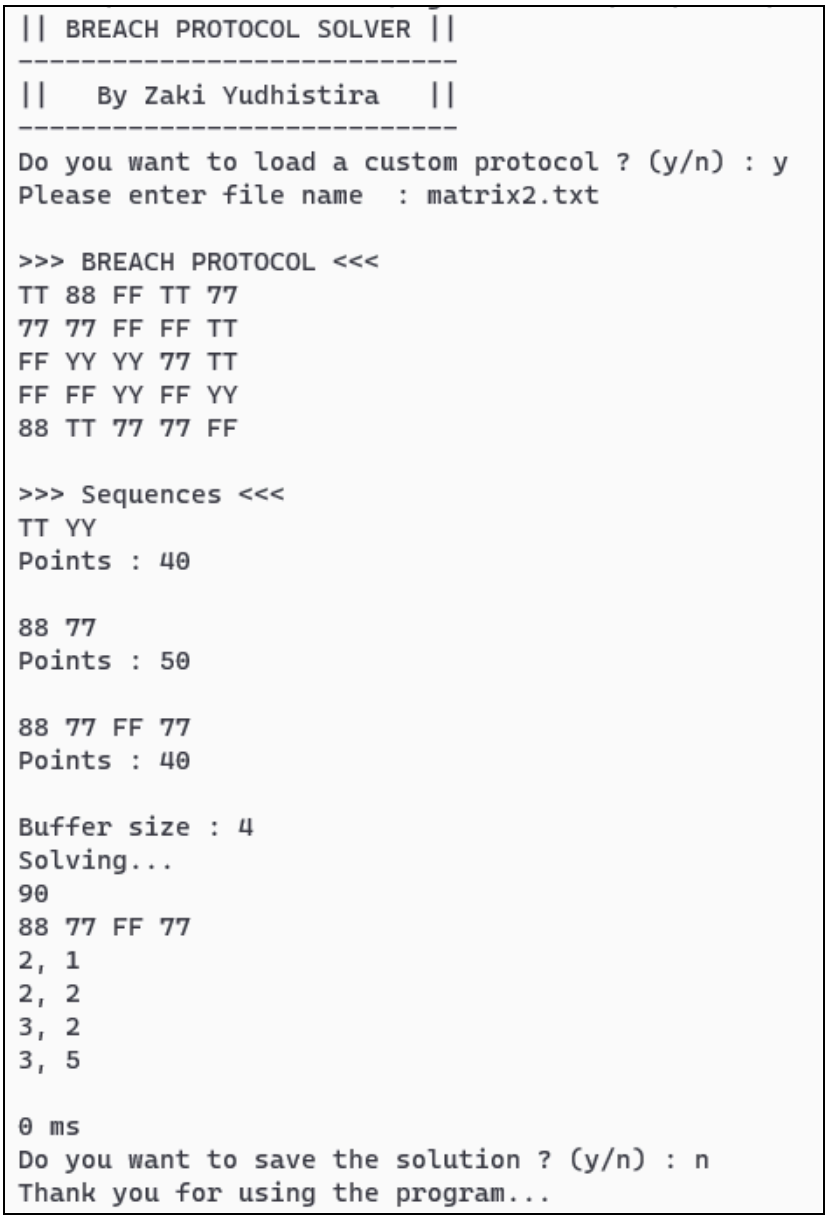
Gambar 3.1 Contoh eksekusi program di terminal windows

D. Hasil tangkapan layar eksekusi program

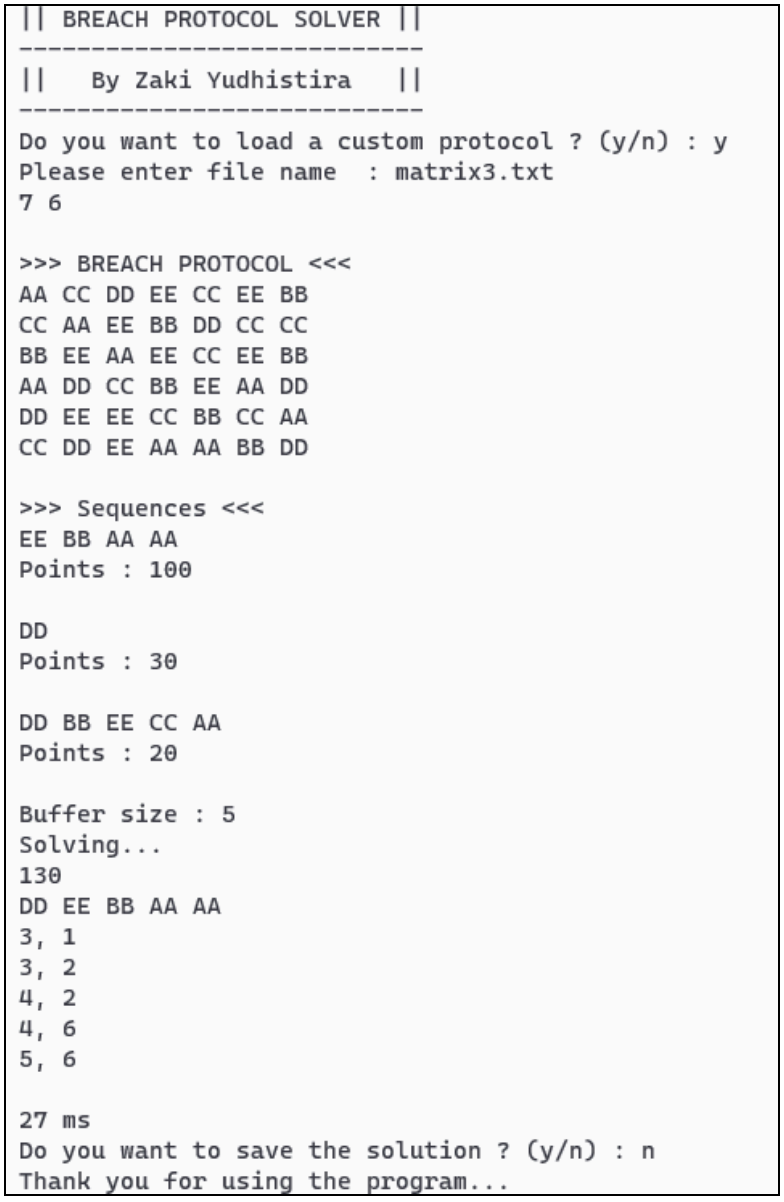
Berikut adalah hasil eksekusi program menggunakan sistem operasi Windows 11 dengan spesifikasi prosesor Ryzen 5 3600 serta RAM 16gb DDR4 3200Mhz.



Gambar 4.1 Eksekusi program menggunakan *file* kustom matrix.txt tanpa menyimpan hasil



Gambar 4.2 Eksekusi program menggunakan *file* kustom matrix2.txt tanpa menyimpan hasil

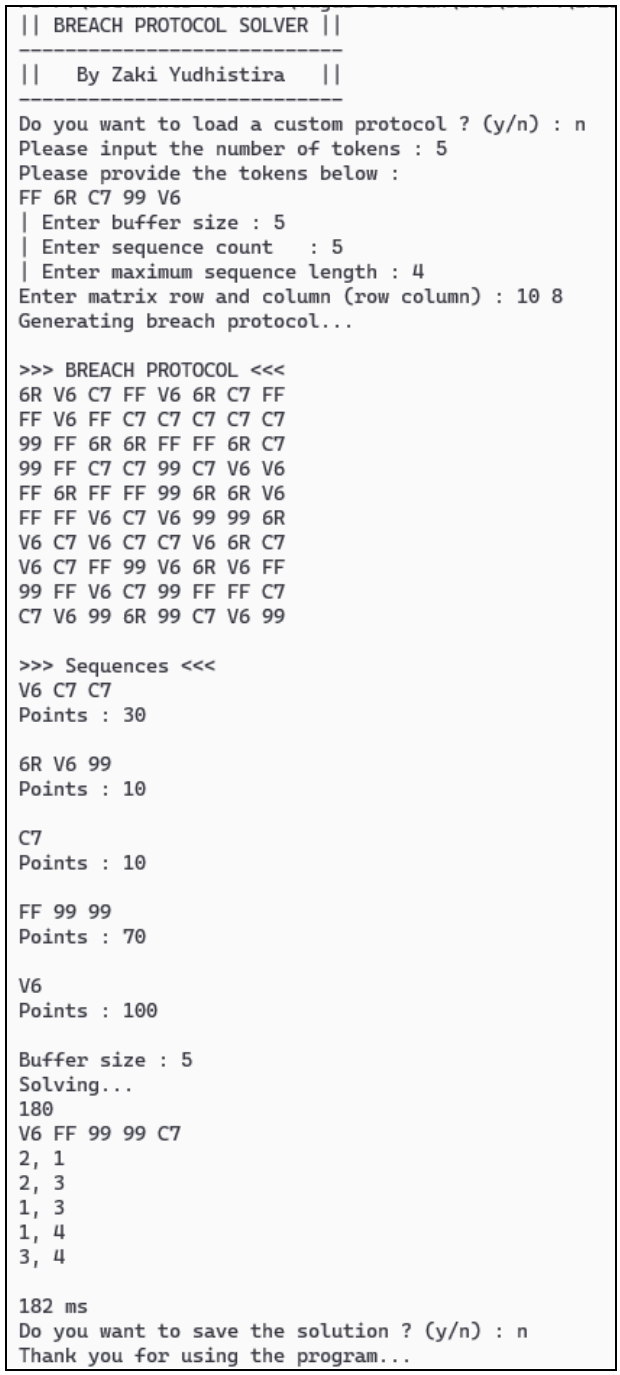


Gambar 4.3 Eksekusi program menggunakan *file* kustom matrix3.txt tanpa menyimpan hasil

```
|| BREACH PROTOCOL SOLVER ||
----------------------------
||   By Zaki Yudhistira   ||
----------------------------
Do you want to load a custom protocol ? (y/n) : n
Please input the number of tokens : 4
Please provide the tokens below :
QQ WW EE RR
| Enter buffer size : 5
| Enter sequence count   : 4
| Enter maximum sequence length : 5
Enter matrix row and column (row column) : 7 8
Generating breach protocol...

>>> BREACH PROTOCOL <<<
WW EE WW WW QQ RR EE EE
QQ RR QQ EE WW RR WW QQ
EE WW RR RR RR EE QQ EE
EE QQ EE QQ RR QQ QQ RR
EE QQ EE RR WW QQ EE WW
WW QQ EE EE QQ RR QQ RR
WW WW RR RR WW RR EE QQ

>>> Sequences <<<
RR EE
Points : 90

RR RR
Points : 90

RR
Points : 70

EE RR
Points : 100

Buffer size : 5
Solving...
350
WW EE RR RR EE
1, 1
1, 3
3, 3
3, 7
7, 7

65 ms
Do you want to save the solution ? (y/n) : y
Enter save file name : sol1.txt
Thank you for using the program...
```

```
350
WW EE RR RR EE
1, 1
1, 3
3, 3
3, 7
7, 7

Runtime : 65 ms
```

Gambar 4.4 Eksekusi program dengan matriks acak dengan menyimpan hasil dalam file .txt



```
|| BREACH PROTOCOL SOLVER ||
----------------------------
||   By Zaki Yudhistira   ||
----------------------------
Do you want to load a custom protocol ? (y/n) : n
Please input the number of tokens : 5
Please provide the tokens below :
FF 6R C7 99 V6
| Enter buffer size : 5
| Enter sequence count   : 5
| Enter maximum sequence length : 4
Enter matrix row and column (row column) : 10 8
Generating breach protocol...

>>> BREACH PROTOCOL <<<
6R V6 C7 FF V6 6R C7 FF
FF V6 FF C7 C7 C7 C7 C7
99 FF 6R 6R FF FF 6R C7
99 FF C7 C7 99 C7 V6 V6
FF 6R FF FF 99 6R 6R V6
FF FF V6 C7 V6 99 99 6R
V6 C7 V6 C7 C7 V6 6R C7
V6 C7 FF 99 V6 6R V6 FF
99 FF V6 C7 99 FF FF C7
C7 V6 99 6R 99 C7 V6 99

>>> Sequences <<<
V6 C7 C7
Points : 30

6R V6 99
Points : 10

C7
Points : 10

FF 99 99
Points : 70

V6
Points : 100

Buffer size : 5
Solving...
180
V6 FF 99 99 C7
2, 1
2, 3
1, 3
1, 4
3, 4

182 ms
Do you want to save the solution ? (y/n) : n
Thank you for using the program...
```

Gambar 4.5 Eksekusi program dengan matriks acak tanpa menyimpan hasil

```
|| BREACH PROTOCOL SOLVER ||
----------------------------
||   By Zaki Yudhistira   ||
----------------------------
Do you want to load a custom protocol ? (y/n) : n
Please input the number of tokens : 4
Please provide the tokens below :
qw er ty ui
| Enter buffer size : 6
| Enter sequence count   : 6
| Enter maximum sequence length : 6
Enter matrix row and column (row column) : 9 11
Generating breach protocol...

>>> BREACH PROTOCOL <<<
QW ER TY TY QW QW TY TY TY ER TY
ER QW TY ER UI QW UI QW ER QW TY
TY TY TY TY TY QW UI ER UI UI UI
UI UI TY TY UI ER TY QW TY UI UI
ER QW TY QW TY QW TY TY ER TY TY
TY QW ER UI QW UI QW ER UI ER ER
UI UI TY UI QW QW QW UI ER ER ER
QW UI UI UI QW UI TY ER TY TY UI
ER UI ER QW TY ER UI UI TY UI TY

>>> Sequences <<<
QW QW
Points : 50

QW QW TY QW UI
Points : 40

TY ER TY QW QW
Points : 30

QW UI
Points : 20

QW QW UI QW
Points : 70

TY ER
Points : 50

Buffer size : 6
Solving...
190
QW QW UI QW TY ER
1, 1
1, 8
2, 8
2, 2
3, 2
3, 6

4330 ms
Do you want to save the solution ? (y/n) : y
```

```
190
QW QW UI QW TY ER
1, 1
1, 8
2, 8
2, 2
3, 2
3, 6

Runtime : 4330 ms
```

Gambar 4.6 Eksekusi program dengan matriks acak dan menyimpan hasil

E. Lampiran

Tautan *repository github* : https://github.com/ZakiYudhistira/Tucil1_13522031

Tabel Spesifikasi Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	**V**	
2. Program berhasil dijalankan	**V**	
3. Program dapat membaca masukan berkas .txt	**V**	
4. Program dapat menghasilkan masukan secara acak	**V**	
5. Solusi yang diberikan program optimal		**V**
6. Program dapat menyimpan solusi dalam berkas .txt	**V**	
7. Program memiliki GUI		**V**

Catatan : program akan lebih optimal lagi apabila diimplementasikan menggunakan bahasa C++ atau C.