TUGAS KECIL 1

STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun Oleh:

Evelyn Yosiana (13522083)

Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

DAFTAR ISI

DAFTAR ISI **2**
BAB 1 ..... **3**
BAB 2 ..... **6**
BAB 3 ..... **8**
3.1. function.py 8
3.2. inputOutput.py 10
3.3. gui.py 14
BAB 4 ..... **22**
4.1. Test case 1: input dari file txt 22
4.2. Test case 2: input dari CLI; matriks, sekuens, dan bobot sekuens diisi acak 23
4.3. Test case 3: input dari file melalui GUI 24
4.4. Test case 4: input secara manual melalui GUI 26
4.5. Test case 5: Kondisi ketika file masukan tidak ditemukan dan nama untuk file keluaran sudah ada 29
4.6. Test case 6: Kondisi ketika skor maksimal bernilai nol 31
4.7. Test case 7: Kondisi ketika masukan pada CLI tidak valid 32
4.8. Test case 8: Kondisi ketika token tidak valid pada pembacaan file 33
4.9. Test case 9: Kondisi ketika token tidak valid pada CLI 34
4.10. Test case 10 34
BAB 5 ..... **36**
DAFTAR PUSTAKA **37**
LAMPIRAN **38**
Github Repository 38
Checklist Keberhasilan Program 38

BAB 1

Deskripsi Masalah

Gambar 1 Permainan Breach Protocol

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

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.

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.

Ilustrasi kasus :

Diberikan matriks sebagai berikut dan ukuran buffernya adalah tujuh

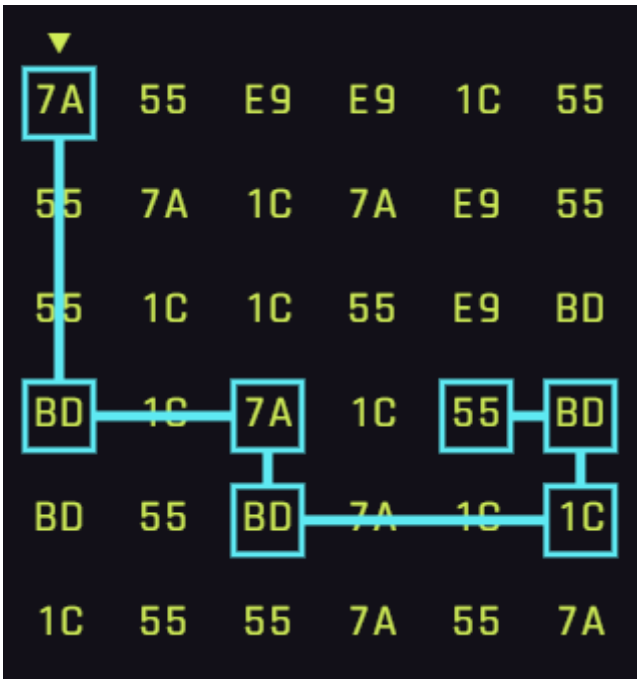
7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

Dengan sekuens sebagai berikut:

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi yang optimal untuk matriks dan sekuens yang diberikan adalah sebagai berikut:

- Total bobot hadiah : 50 poin
- Total langkah : 6 langkah



Gambar 2 Contoh Solusi

(Sumber: https://cyberpunk-hacker.com/)

Tugas anda adalah menemukan solusi dari **permainan Breach Protocol** yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan ***algoritma brute force***.

BAB 2

Teori Singkat dan Algoritma Brute Force

Algoritma merupakan urutan langkah untuk memecahkan suatu persoalan dengan memproses masukan menjadi keluaran. Kemangkusan dari suatu algoritma dapat diukur dengan parameter kompleksitas waktu (dinotasikan dengan T(n)) dan kompleksitas ruang (dinotasikan dengan S(n)) dengan n merupakan ukuran masukkan yang diproses.

Salah satu algoritma yang cukup terkenal ialah algoritma brute force. Algoritma brute force dapat digunakan untuk sebagian besar permasalahan. Algoritma brute force juga sederhana, baku, dan mudah dimengerti. Namun sayangnya algoritma ini kurang cocok digunakan untuk masukan yang berukuran besar karena jarang menghasilkan algoritma yang mangkus.Contoh permasalahan yang dapat menggunakan algoritma ini antara lain pencarian, pengurutan, pencocokan string, dan perkalian matriks. Contoh permainan yang dapat diselesaikan dengan menggunakan algoritma ini antara lain sudoku, permainan 24, *crossword puzzle*, kakurasu, futoshiki, kakuro dan juga *cyberpunk 2077 breach protocol*.

Permainan *cyberpunk 2077 breach protocol* dapat diselesaikan dengan mencari seluruh kemungkinan jalur kemudian mencari jalur dengan bobot nilai terbesar. Secara rinci, langkah-langkah penyelesaian permainan *cyberpunk 2077 breach protocol* dengan menggunakan algoritma brute force adalah sebagai berikut.

1. Pertama-tama, dibuat matriks yang menggambarkan koordinat berdasarkan indeks matriks. Misalkan untuk matriks berukuran 3×4, isi matriks adalah sebagai berikut.

   00 01 02 03

   10 11 12 13

   20 21 22 23

2. Kemudian dibuat *list* untuk menyimpan seluruh kemungkinan jalur. Awalnya (penambahan ke-0) *list* diisi dengan seluruh elemen matriks pada langkah nomor 1 dengan indeks [0][XX] dengan XX bilangan bulat yang diiterasi dari 0 sampai jumlah kolom matriks. Untuk penambahan ke-n, tiap elemen *list* yang memiliki n elemen

ditambahkan dengan seluruh elemen matriks dengan indeks [YY][indeks kolom elemen] dengan YY bilangan bulat yang diiterasi dari 0 sampai jumlah baris matriks, untuk n bilangan ganjil atau ditambahkan dengan seluruh elemen matriks dengan indeks [indeks baris elemen][XX] untuk n bilangan genap. Hasil penambahan akan ditambahkan pada *list* pada bagian akhir. Tidak lupa juga untuk mengecek apakah matriks dengan indeks yang sama telah digunakan atau belum. Jika sudah maka penambahan pada bagian tersebut dapat dilewati. Penambahan dilakukan berulang-ulang (bergantian secara vertikal dan horizontal) sampai n sama dengan ukuran maksimal *buffer*.

3. Tiap nilai elemen *list* pada nomor 2 diganti dengan elemen matriks berisi token sesuai dengan indeksnya. Misalnya elemen “00” diganti dengan nilai elemen matriks token ke [0][0].
4. Untuk setiap elemen pada *list* yang sudah berisikan token, akan dicek apakah setiap sekuens yang ada merupakan subset dari elemen tersebut. Jika ya, maka bobot dari elemen tersebut akan ditambah sebesar bobot yang sekuens terkait. Kemudian dari bobot elemen-elemen pada *list* tersebut akan dicari bobot yang paling besar. Elemen dengan bobot paling besar merupakan jalur paling optimal. Pengecekan akan dilakukan mulai dari elemen *list* paling depan untuk mendapatkan jalur dengan bobot nilai terbesar namun dengan jumlah *buffer* paling sedikit (optimal).
5. Untuk mendapatkan koordinat dari jalur optimal, dicari elemen *list* yang merupakan jalur dari hasil optimal (berupa angka seperti pada langkah nomor 1). Indeks kolom didapatkan dari nilai elemen pertama (indeks[0]) ditambah dengan satu, sedangkan indeks baris didapatkan dari nilai elemen kedua (indeks[1]) ditambah dengan satu. Misalkan elemen berisi angka “01”, maka koordinatnya adalah (1+1, 0+1) atau (2,1).

BAB 3

Source Code

3.1. function.py

3.1.1. Fungsi isSubset menerima parameter dua buah array (buffer dan sequence) dan mengembalikan nilai *true* apabila array sequence merupakan subset dari array buffer dan mengembalikan nilai *false* apabila array sequence bukan merupakan subset dari array buffer.

```
def isSubset(buffer, sequence):
    for i in range(len(buffer) - len(sequence) + 1):
        isFound = True
        for j in range(len(sequence)):
            if buffer[i + j] != sequence[j]:
                isFound = False
                break
        if isFound:
            return True
    return False
```

3.1.2. Fungsi solver menerima parameter tiga buah bilangan bulat (matrixWidth, matrixHeight, dan bufferSize), dua buah matriks (sequence dan matrix), serta sebuah array (sequenceReward) kemudian akan mengembalikan tiga buah nilai, yaitu sebuah bilangan bulat (maxScore) dan dua buah array (optimalBuffer dan path).

```
def solver(matrixWidth, matrixHeight, bufferSize, sequence,
sequenceReward, matrix):
    matrixIndex = [['' for _ in range(matrixWidth)] for _ in
range(matrixHeight)]
    for i in range (matrixHeight):
        for j in range (matrixWidth):
            matrixIndex[i][j] = str(i) + str(j)

    isVertical = True
    numberOfBufferNow = 1
    buffer = [[x] for x in matrixIndex[0]]
```

```
    while (numberOfBufferNow < bufferSize):
        if (isVertical):
            for i in buffer:
                if (len(i) == numberOfBufferNow):
                    for j in range (matrixHeight):
                        if (matrixIndex[j][int(i[-1][1])] not in
i):
                            buffer.append(i +
[matrixIndex[j][int(i[-1][1])]])
                else: continue
            isVertical = False
        else:
            for i in buffer:
                if (len(i) == numberOfBufferNow):
                    for j in range (matrixWidth):
                        if (matrixIndex[int(i[-1][0])][j] not in
i):
                            buffer.append(i +
[matrixIndex[int(i[-1][0])][j]])
                else: continue
            isVertical = True
        numberOfBufferNow += 1

    bufferToken = copy.deepcopy(buffer)

    for i in range(len(bufferToken)):
        for j in range(len(bufferToken[i])):
            bufferToken[i][j] =
matrix[int(bufferToken[i][j][0])][int(bufferToken[i][j][1])]

    maxScore = 0
    optimalBuffer = []
    for item in bufferToken:
        score = 0
        for j in sequence:
            if isSubset(item, j):
                score += int(sequenceReward[sequence.index(j)])
        if score > maxScore:
            maxScore = score
```

```
            optimalBuffer = item
    path = buffer[bufferToken.index(item)]
    for i in range (len(path)):
        temp1 = path[i][1]
        temp2 = path[i][0]
        path[i] = str(int(temp1)+1) + "," + str(int(temp2)+1)
    if (maxScore == 0):
        optimalBuffer = []
        path = []
    return (optimalBuffer, maxScore, path)
```

3.2. inputOutput.py

File ini berisi kode untuk menerima input (baik dari file txt maupun CLI (*command line interface)*), mengisi matriks, sekuens, dan bobot sekuens secara random jika masukkan dari CLI, menampilkan hasil dalam CLI, serta dapat menyimpan hasil dalam ekstensi txt dan juga menghitung dan menampilkan *runtime*.

```
# KAMUS UMUM
sequence = []
sequenceReward = []
```

```
# INPUT
print("Apakah ingin menginput masukan melalui file atau CLI?")
print("(1) File txt")
print("(2) CLI")
choosenInput = int(input())
```

Program hanya dapat memproses input dari file txt dan CLI, jika input untuk memilih metode untuk menginput file tidak valid, maka program akan meminta pengguna untuk memasukkan pilihan metode lagi sampai valid.

```
# Validate read from file or CLI
while choosenInput < 1 or choosenInput > 2:
    print("Input invalid, silakan input ulang!")
    print("Apakah ingin menginput masukan melalui file atau CLI? (ketik
angkanya saja)")
    print("1. File txt")
    print("2. CLI")
    choosenInput = int(input())
```

Untuk input dari file, program akan meminta pengguna untuk memasukkan nama file, dan jika file tidak ditemukan, program akan meminta pengguna untuk memasukkan kembali nama file sampai nama file tersebut ditemukan.

```
# Read from file
if (choosenInput == 1):
    while True:
        file_name = input("Silakan masukkan nama file input (disertai
dengan .txt): ")
        try:
            with open(file_name, 'r') as inputFile:
                bufferSize = int(inputFile.readline())
                matrixWidth, matrixHeight = map(int,
inputFile.readline().split())
                matrix = [inputFile.readline().split() for _ in
range(matrixHeight)]
                numberOfSequences = int(inputFile.readline())
                line_number = 0
                sequence = []
                sequenceReward = []
                for line in inputFile:
                    if line_number % 2 == 0:
                        sequence.append(line.split())
                    else:
                        sequenceReward.append(line.strip())
                    line_number += 1
                break
        except FileNotFoundError:
            print("File tidak ditemukan, silahkan coba lagi.")
```

Untuk input dari CLI, program akan meminta pengguna untuk memasukkan kembali nilai yang diminta sampai masukan valid.

```
# Read from CLI
else:
    jumlahTokenUnik = int(input("Masukkan jumlah token unik: "))
    while (jumlahTokenUnik < 1):
        jumlahTokenUnik = int(input("Jumlah token unik harus bilangan
bulat positif! Masukkan jumlah token unik: "))
    preToken = input("Masukkan token dipisahkan dengan spasi: ")
    while (len(preToken.split()) != jumlahTokenUnik):
        preToken = input("Jumlah token tidak sesuai! Silakan masukkan
token lagi: ")
```

```
    bufferSize = int(input("Masukkan ukuran buffer: "))
    while (bufferSize < 0):
        bufferSize = int(input("Ukuran buffer harus lebih dari sama
dengan nol! Masukkan ukuran buffer: "))
    matrixHeight, matrixWidth = map(int, input("Masukkan ukuran matrix:
").split())
    while (matrixWidth < 1 or matrixHeight < 1):
        matrixHeight, matrixWidth = map(int, input("Ukuran matriks
minimal 1x1! Masukkan ukuran matrix: ").split())
    numberOfSequences = int(input("Masukkan jumlah sekuens: "))
    while (numberOfSequences < 1):
        numberOfSequences = int(input("Jumlah sekuens harus lebih dari
nol! Masukkan jumlah sekuens: "))
    sequenceMax = int(input("Masukkan ukuran maksimal sekuens: "))
    while (sequenceMax < 1):
        sequenceMax = int(input("Ukuran maksimal sekuens harus lebih
dari nol! Masukkan ukuran maksimal sekuens: "))

    # Process token array, random matrix, random sequences, and random
sequences reward
    token = preToken.split()
    matrix = [[random.choice(token) for _ in range(matrixWidth)] for _
in range(matrixHeight)]
    sequence = [[random.choice(token) for _ in range(random.randint(2,
sequenceMax))] for _ in range(numberOfSequences)]
    sequenceReward = [random.randint(5, 100) for _ in
range(numberOfSequences)]

    # Print matrix, sequences, and sequences reward
    print()
    print("Matriks: ")
    for i in matrix:
        for j in i:
            print(j, end=" ")
        print()
    print()
    print("Sekuens: ")
    for i in range (len(sequence)):
        for j in sequence[i]:
            print(j, end=" ")
        print(" reward = ", sequenceReward[i])
```

Program akan memvalidasi panjang buffer, ukuran matriks, dan token dalam matriks dan sekuens. Jika ada yang tidak valid maka program akan berhenti

```
# VALIDATE INPUT
isInputValid = True
if bufferSize < 0 or matrixHeight < 0 or matrixWidth < 0 or
numberOfSequences < 0:
    print("Input invalid!")
    exit(1)
for row in matrix:
    for elmt in row:
        if (len(elmt) != 2):
            print("Input invalid!")
            exit(1)
for row in sequence:
    for elmt in row:
        if (len(elmt) != 2):
            print("Input invalid!")
            exit(1)
```

```
# PRINT RESULT
result, score, coordinate = func.solver(matrixWidth, matrixHeight,
bufferSize, sequence, sequenceReward, matrix)
print(score)
for i in result:
    print(i, end=' ')
print()
for i in coordinate:
    print(i)
print()
```

```
# SAVE RESULT TO TXT
wantToSave = input("Apakah ingin menyimpan solusi? (y/n) ")
if wantToSave == 'y':
    fileOutput = input("Silakan masukkan nama file output (tanpa .txt):
")
    while True:
        try:
            with open(fileOutput + ".txt", 'r') as outputFile:
                print("File dengan nama tersebut sudah ada, silakan
coba lagi!")
                fileOutput = input("Silakan masukkan nama file output
(tanpa .txt): ")
```

```
        except FileNotFoundError:
            with open(fileOutput + ".txt", 'a') as outputFile:
                outputFile.write(str(score))
                outputFile.write('\n')
                for i in result:
                    outputFile.write(str(i) + ' ')
                outputFile.write('\n')
                for i in coordinate:
                    outputFile.write(str(i))
                    outputFile.write('\n')
                print(fileOutput)
                print("Hasil berhasil disimpan dalam file", fileOutput
+ "." + "txt!")
                break
```

3.3. gui.py

```
def main(page: ft.Page):
    matrix = []
    sequences = []
    sequences_score = []
```

Fungsi build_matrix bertujuan untuk membentuk matriks berukuran height × width. Fungsi ini akan dijalankan ketika fungsi update_size berjalan.

```
    def build_matrix(width, height):
        matrix.clear()
        for _ in range(height):
            row = []
            for _ in range(width):
                field = ft.TextField(text_align="center", width=50)
                row.append(field)
            matrix.append(row)
        page.update()
```

Fungsi build_sequences bertujuan untuk membentuk matriks berukuran height × width. Fungsi ini akan dijalankan ketika fungsi update_size berjalan.

```
    def build_sequences(width, height):
        sequences.clear()
        for _ in range(height):
```

```
            row = []
            for _ in range(width):
                field = ft.TextField(text_align="center", width=50)
                row.append(field)
            sequences.append(row)
        page.update()
```

Fungsi build_sequences_score bertujuan untuk membentuk matriks berukuran height. Fungsi ini akan dijalankan ketika fungsi update_size berjalan.

```
    def build_sequences_score(height):
        sequences_score.clear()
        for _ in range(height):
            row = []
            for _ in range(1):
                field = ft.TextField(text_align="center", width=50)
                row.append(field)
            sequences_score.append(row)
        page.update()
```

Fungsi update_size akan dijalankan ketika tombol “fix size” di-*click*. Fungsi ini berfungsi untuk meng-*update* halaman sehingga dapat menampilkan *field text* matriks, sekuens, dan bobot sekuens untuk diisi secara manual. Setelah di-*click*, tombol fix size akan dihapus sehingga ukuran matriks, sekuens, dan bobot sekuens tidak dapat diubah lagi.

```
    def update_size(e):
        matrixWidth = int(width_input.value)
        matrixHeight = int(height_input.value)
        sequencesWidth = int(sequences_width_input.value)
        sequencesHeight = int(sequences_height_input.value)
        build_matrix(matrixWidth, matrixHeight)
        build_sequences(sequencesWidth, sequencesHeight)
        build_sequences_score(sequencesHeight)
        page.controls.remove(matrix_view)
        page.update()
        buffer_input = ft.Row([buffer_size_input],
alignment=ft.MainAxisAlignment.CENTER, disabled=True)
        new_matrix_view = ft.Row(
            [
                ft.Column(
                    [
```

```
                        ft.Row([width_input, height_input],
alignment=ft.MainAxisAlignment.CENTER, disabled=True),
                        ft.Container(height=2),
                        ft.Container(height=10),
                        *[ft.Row(row,
alignment=ft.MainAxisAlignment.CENTER) for row in matrix]
                    ]
                ),
                ft.Container(width=50),
                ft.Column(
                    [
                        ft.Row([sequences_width_input,
sequences_height_input], alignment=ft.MainAxisAlignment.CENTER,
disabled=True),
                        ft.Container(height=2),
                        ft.Container(height=10),
                        *[ft.Row(row,
alignment=ft.MainAxisAlignment.CENTER) for row in sequences]
                    ]
                ),
                ft.Container(width=20),
                ft.Column(
                    [
                        ft.Container(height=50),
                        ft.Text(value="Score", size=19),
                        *[ft.Row(row,
alignment=ft.MainAxisAlignment.CENTER) for row in sequences_score]
                    ]
                )
            ],
            alignment=ft.MainAxisAlignment.CENTER
        )
        solver = ft.Row(
            [
                ft.Row([solve_button],
alignment=ft.MainAxisAlignment.CENTER)
            ],
            alignment=ft.MainAxisAlignment.CENTER
        )
        page.controls.append(buffer_input)
        page.controls.append(ft.Container(height=20))
        page.controls.append(new_matrix_view)
        page.controls.append(ft.Container(height=20))
```

```
        page.controls.append(solver)
        page.update()
```

Fungsi solve digunakan untuk memproses matriks, sekuens, dan bobot sekuens nilainya diambil dari *field* yang tersedia. Fungsi ini akan berjalan ketika pengguna mengeklik tombol “SOLVE AND SAVE”. Setelah diproses, hasilnya akan ditampilkan pada gui serta file bernama “result.txt”. Jika sebelumnya sudah ada file bernama “result.txt”, maka file yang lama akan ditimpa dengan file yang baru. Ketika tombol “SOLVE AND SAVE” diklik sebanyak n kali, maka hasil juga akan ditampilkan sebanyak n dengan hasil ke n akan berada di bawah hasil n-1.

```
    def solve(e):
        matrixWidth = int(width_input.value)
        matrixHeight = int(height_input.value)
        bufferSize = int(buffer_size_input.value)
        matrix_value = []
        for row in matrix:
            row_value = []
            for field in row:
                row_value.append(field.value)
            matrix_value.append(row_value)
        sequence_value = []
        for row in sequences:
            row_value = []
            for field in row:
                row_value.append(field.value)
            sequence_value.append(row_value)
        sequences_score_value = []
        for row in sequences_score:
            for field in row:
                sequences_score_value.append(int(field.value))
        result, score, coordinate = func.solver(matrixWidth,
matrixHeight, bufferSize, sequence_value
        , sequences_score_value, matrix_value)
        print(sequences_score_value)
        print(result, score, coordinate)
        with open("result.txt", 'w') as outputFile:
            outputFile.write(str(score) + '\n')
            outputFile.write(' '.join(map(str, result)) + '\n')
            outputFile.write('\n'.join(map(str, coordinate)) + '\n')
        result_output = ft.Text(value=f"Optimal Buffer: {',
'.join(result)}", size=17)
```

```
        score_output = ft.Text(value=f"Optimal Score: {score}",
size=17)
        coordinate_output = ft.Text(value=f"Coordinate: {',
'.join(coordinate)}", size=17)
        success_message = ft.Text(value="Succed to save!")
        result_row = ft.Row(
            [
                ft.Column(
                    [
                        ft.Row([result_output],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Container(height=10),
                        ft.Row([score_output],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Container(height=10),
                        ft.Row([coordinate_output],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Container(height=10),
                        ft.Row([success_message],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Container(height=10),
                    ]
                )
            ],
            alignment=ft.MainAxisAlignment.CENTER
        )
        page.controls.append(ft.Container(height=20))
        page.controls.append(result_row)
        page.update()
```

Fungsi solve_from_txt digunakan untuk membaca file yang nama filenya akan didapatkan dari *field* “or input file name here…”, kemudian memproses matriks, sekuens, dan bobot sekuens nilainya diambil dari file tersebut. Fungsi ini akan berjalan ketika pengguna meng-*click* tombol “SOLVE AND SAVE” yang ada di sebelah tombol “or input file name here…”. Setelah diproses, hasilnya akan ditampilkan pada gui serta file bernama “result.txt”. Jika sebelumnya sudah ada file bernama “result.txt”, maka file yang lama akan ditimpa dengan file yang baru. Ketika tombol “SOLVE AND SAVE” diklik sebanyak n kali, maka hasil juga akan ditampilkan sebanyak n dengan hasil ke n akan berada di bawah hasil n-1.

```
    def solve_from_txt(e):
        with open(upload_file_input.value, 'r') as inputFile:
```

```
                bufferSize = int(inputFile.readline())
                matrixWidth, matrixHeight = map(int,
inputFile.readline().split())
                matrix = [inputFile.readline().split() for _ in
range(matrixHeight)]
                numberOfSequences = int(inputFile.readline())
                line_number = 0
                sequence = []
                sequenceReward = []
                for line in inputFile:
                    if line_number % 2 == 0:
                        sequence.append(line.split())
                    else:
                        sequenceReward.append(line.strip())
                    line_number += 1
        result, score, coordinate = func.solver(matrixWidth,
matrixHeight, bufferSize, sequence, sequenceReward, matrix)
        with open("result.txt", 'w') as outputFile:
            outputFile.write(str(score) + '\n')
            outputFile.write(' '.join(map(str, result)) + '\n')
            outputFile.write('\n'.join(map(str, coordinate)) + '\n')
        file_input_name = ft.Text(value=f"File Input:
{(upload_file_input.value)}", size=17)
        result_output = ft.Text(value=f"Optimal Buffer: {',
'.join(result)}", size=17)
        score_output = ft.Text(value=f"Optimal Score: {score}",
size=17)
        coordinate_output = ft.Text(value=f"Coordinate: {',
'.join(coordinate)}", size=17)
        success_message = ft.Text(value="Succed to save!")
        result_row = ft.Row(
            [
                ft.Column(
                    [
                        ft.Row([file_input_name],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Row([result_output],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Row([score_output],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Row([coordinate_output],
alignment=ft.MainAxisAlignment.CENTER),
```

```
                        ft.Row([success_message],
alignment=ft.MainAxisAlignment.CENTER),
                    ]
                )
            ],
            alignment=ft.MainAxisAlignment.CENTER
        )
        page.controls.append(ft.Container(height=20))
        page.controls.append(result_row)
        page.update()
```

```
    #Button, text, and field
    title = ft.Text(value="Cyberpunk 2077 Breach Protocol", size=50)
    width_input = ft.TextField(label="Matrix Width", width=130)
    height_input = ft.TextField(label="Matrix Height", width=130)
    buffer_size_input = ft.TextField(label="Buffer size", width=130)
    fix_size = ft.FilledButton(text="Fix Size", on_click=update_size,
height=35)
    sequences_width_input = ft.TextField(label="Sequences Max",
width=170)
    sequences_height_input = ft.TextField(label="Number of Sequences",
width=170)
    solve_button = ft.FilledButton(text="SOLVE AND SAVE",
on_click=solve)
    upload_file_input = ft.TextField(label="or input file name
here...", width=300)
    upload_file_button = ft.FilledButton(text="SOLVE AND SAVE",
on_click=solve_from_txt, height=35)

    matrix_view = ft.Row(
            [
                ft.Column(
                    [
                        ft.Row([width_input, height_input,
sequences_width_input, sequences_height_input, buffer_size_input],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Container(height=7),
                        ft.Row([fix_size],
alignment=ft.MainAxisAlignment.CENTER),
                        ft.Container(height=7),
                        ft.Row([upload_file_input, upload_file_button],
alignment=ft.MainAxisAlignment.CENTER),
```

```
                    ]
                ),
            ],
            alignment=ft.MainAxisAlignment.CENTER
        )
```

```
    page.add(
        ft.Container(height=7),
        ft.Row([title], alignment=ft.MainAxisAlignment.CENTER),
        ft.Container(height=7)
    )
    page.controls.append(matrix_view)
    page.add(
        ft.Container(height=2),
    )
```

```
    page.scroll = "always"
```

```
ft.app(target=main)
```

BAB 4

Testing

4.1. Test case 1: input dari file txt

```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
1
Silakan masukkan nama file input (disertai dengan .txt): test.txt
50
7A BD 7A BD 1C BD 55
6,1
6,6
2,6
2,3
3,3
3,5
2,5

Apakah ingin menyimpan solusi? (y/n) y
Silakan masukkan nama file output (tanpa .txt): result
Hasil berhasil disimpan dalam file result.txt!
24756.409645080566 ms
```

Gambar 4.1.1 Tampilan CLI ketika input dari file txt

src > test.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  30
16
```

Gambar 4.1.2 File masukan

src > result.txt

```
50
7A BD 7A BD 1C BD 55
6,1
6,6
2,6
2,3
3,3
3,5
2,5
```

Gambar 4.1.3 File keluaran

4.2. Test case 2: input dari CLI; matriks, sekuens, dan bobot sekuens diisi acak

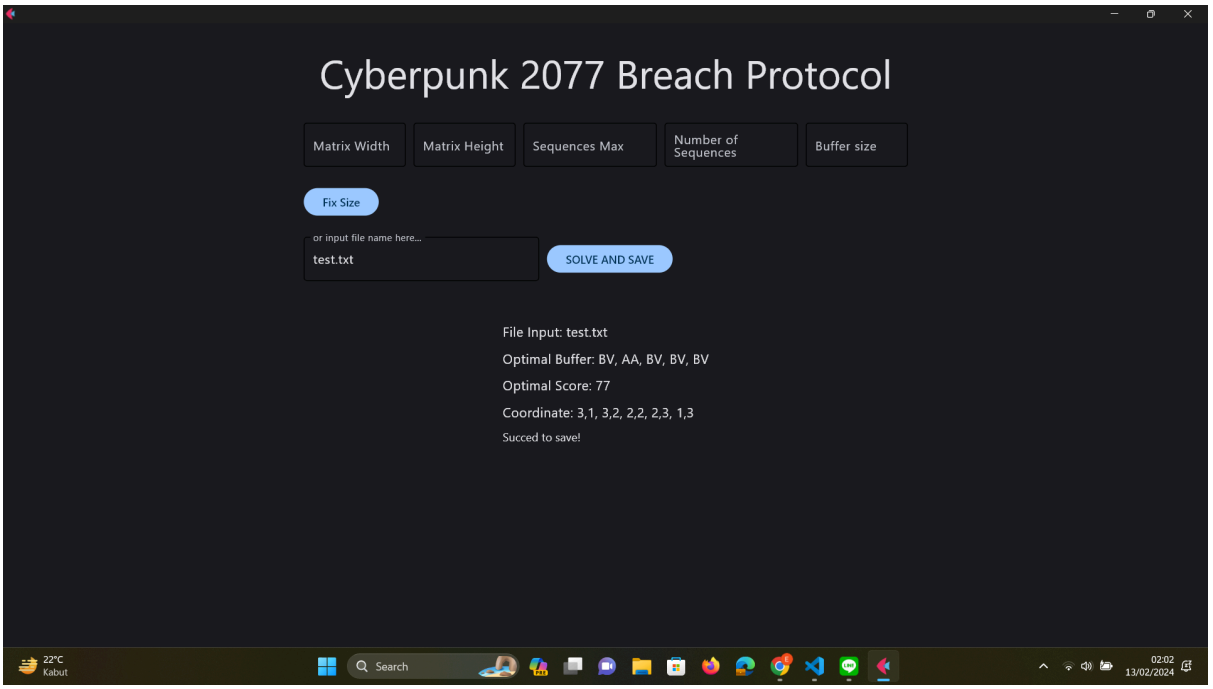
```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
2
Masukkan jumlah token unik: 5
Masukkan token dipisahkan dengan spasi: AA BB CC DD EE
Masukkan ukuran buffer: 5
Masukkan ukuran matrix: 5 5
Masukkan jumlah sekuens: 5
Masukkan ukuran maksimal sekuens: 5

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

Sekuens:
BB AA BB AA  reward =  18
EE EE BB  reward =  63
CC AA EE  reward =  25
BB DD BB CC EE  reward =  61
BB CC DD  reward =  6
```

Gambar 4.2.1 Tampilan CLI ketika input dari CLI

```
Hasil:
69
EE EE BB CC DD
4,1
4,3
2,3
2,4
4,4

10934.777021408081 ms

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

Gambar 4.2.2 Tampilan CLI ketika input dari CLI

4.3. Test case 3: input dari file melalui GUI



Gambar 4.3.1 Tampilan GUI setelah tombol “SOLVE AND SAVE” diklik

src > test.txt

```
5
5 5
AA AA BV BV BV
BV AA AA AA AA
AA BV BV AA AA
BV BV BV BV AA
AA AA BV AA AA
10
BV AA BV BV BV
65
BV AA BV BV BV
-48
AA BV BV AA AA
-16
BV AA
-44
BV BV AA BV
-13
AA AA BV
-8
BV BV
-9
AA AA BV AA BV
9
BV AA BV BV AA
-26
BV AA BV AA AA
-7
```

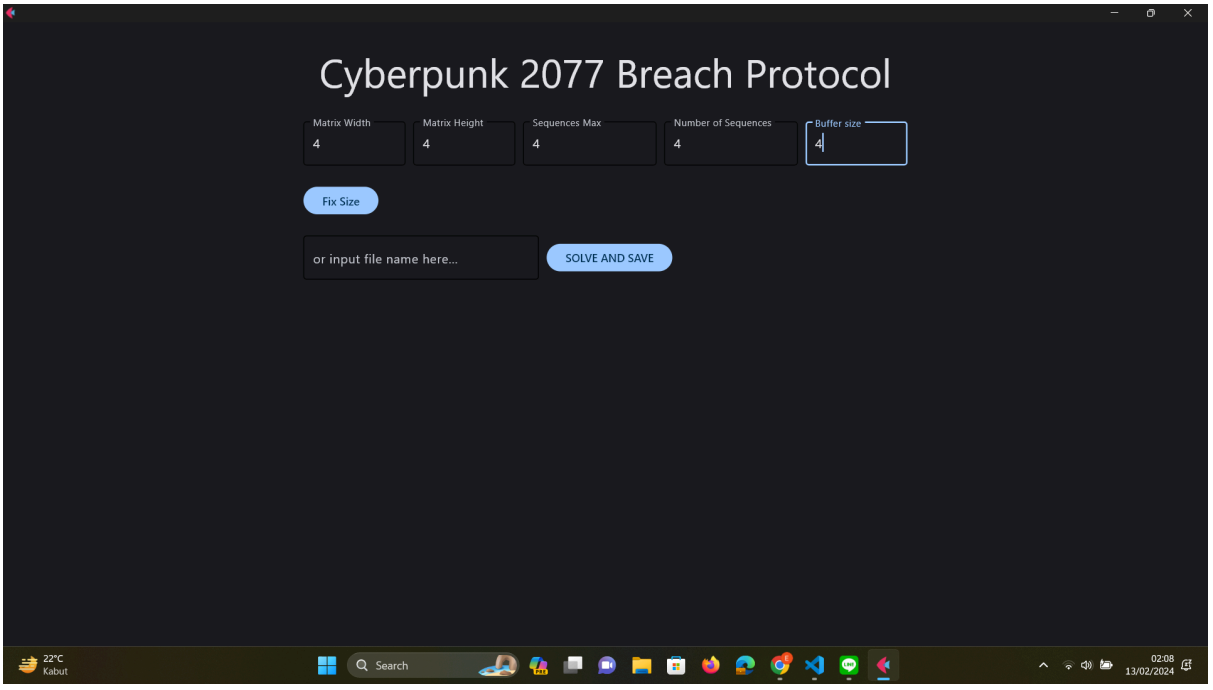
Gambar 4.3.2 File masukan

src > result.txt

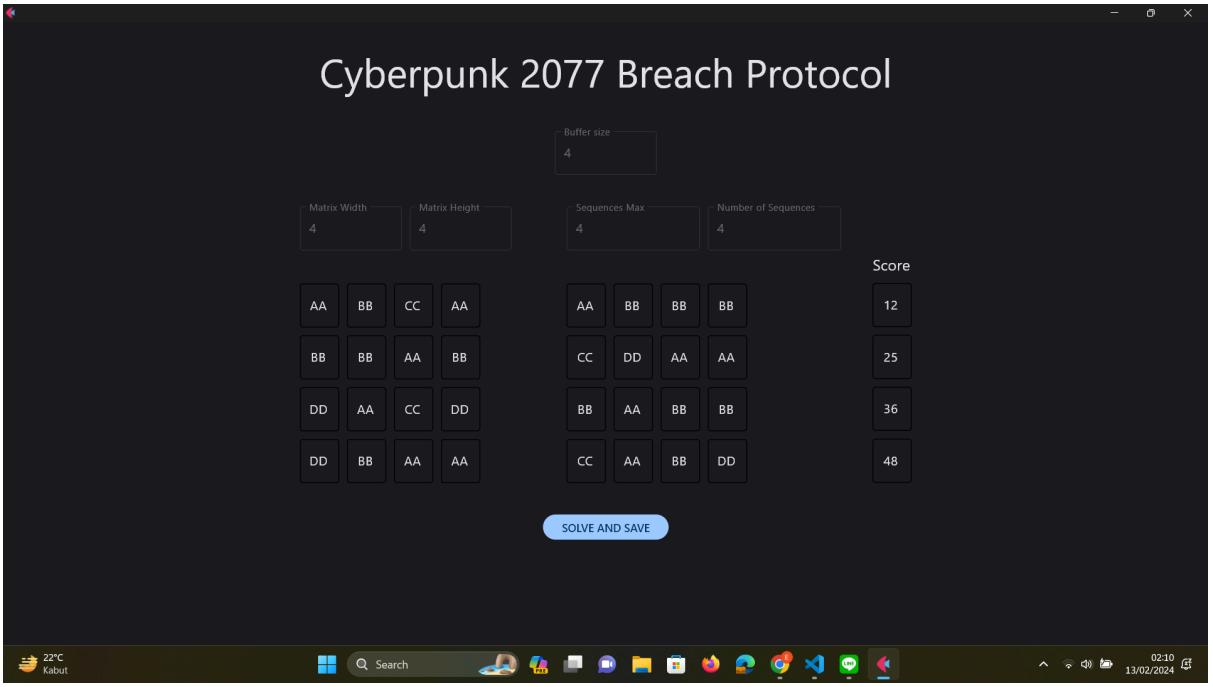
```
1  77
2  BV AA BV BV BV
3  3,1
4  3,2
5  2,2
6  2,3
7  1,3
8
```

Gambar 4.3.3 File keluaran

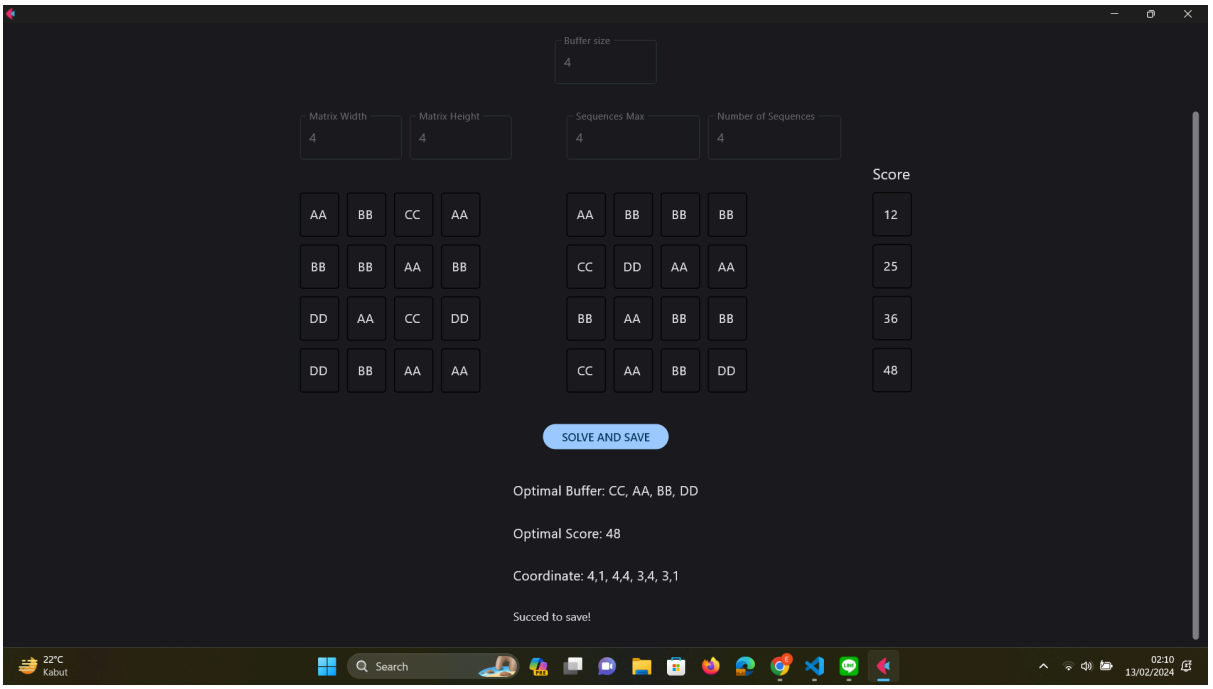
4.4. Test case 4: input secara manual melalui GUI



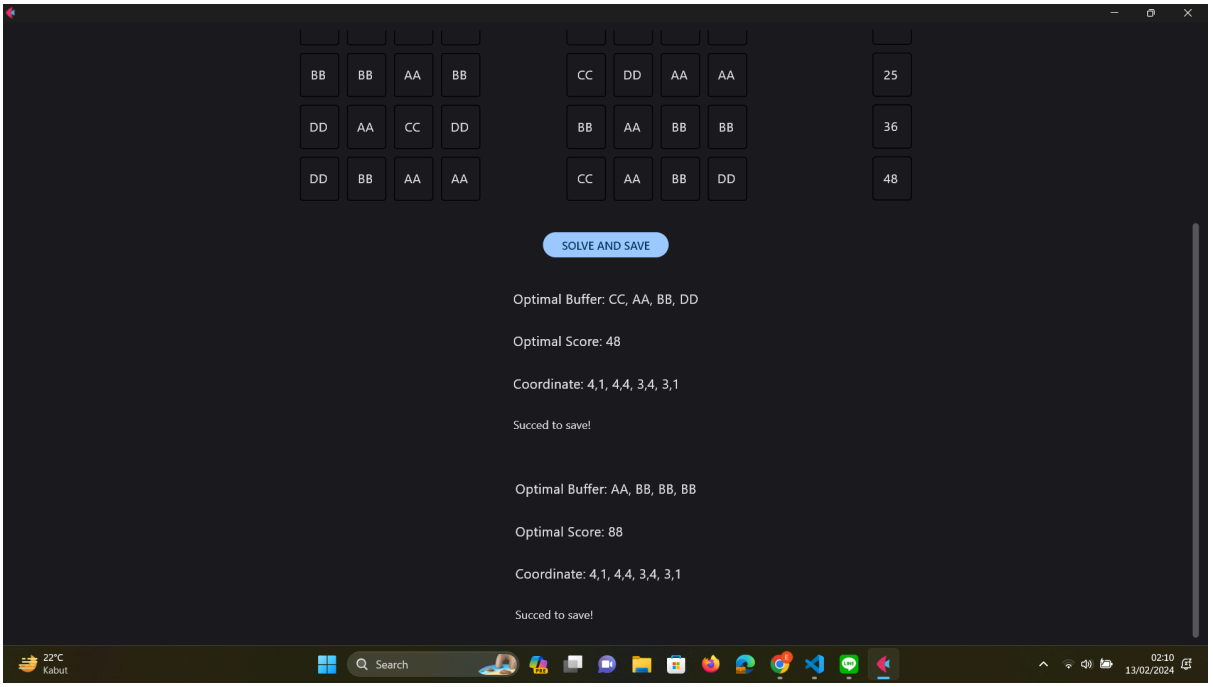
Gambar 4.4.1 Tampilan GUI sebelum tombol “Fix Size” diklik



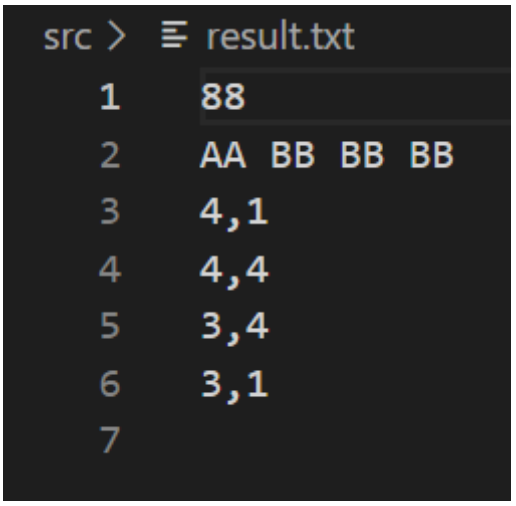
Gambar 4.4.2 Tampilan GUI setelah tombol “Fix Size” diklik dan matriks dan sekuens serta bobotnya diisi



Gambar 4.4.3 Tampilan GUI setelah tombol “SOLVE AND SAVE” diklik



Gambar 4.4.4 Tampilan GUI setelah input diedit dan tombol “SOLVE AND SAVE” diklik lagi



Gambar 4.4.5 File keluaran menampilkan hasil terakhir

4.5. Test case 5: Kondisi ketika file masukan tidak ditemukan dan nama untuk file keluaran sudah ada

```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
1
Silakan masukkan nama file input (disertai dengan .txt): text.txt
File tidak ditemukan, silahkan coba lagi.
Silakan masukkan nama file input (disertai dengan .txt): tes.txt
File tidak ditemukan, silahkan coba lagi.
Silakan masukkan nama file input (disertai dengan .txt): test.txt

Hasil:
4
BV AA BV BV BV AA AA BV AA
3,1
3,2
2,2
2,3
1,3
1,1
2,1
2,5
1,5
1,2

22459.39564704895 ms

Apakah ingin menyimpan solusi? (y/n) y
Silakan masukkan nama file output (tanpa .txt): result
File dengan nama tersebut sudah ada, silakan coba lagi!
Silakan masukkan nama file output (tanpa .txt): test
File dengan nama tersebut sudah ada, silakan coba lagi!
Silakan masukkan nama file output (tanpa .txt): result2
Hasil berhasil disimpan dalam file result2.txt!
```

Gambar 4.5.1 Tampilan pada CLI ketika file masukan tidak ditemukan dan nama file keluaran sudah ada

```
src > result2.txt
1   4
2   BV AA BV BV BV AA AA BV AA
3   3,1
4   3,2
5   2,2
6   2,3
7   1,3
8   1,1
9   2,1
10  2,5
11  1,5
12  1,2
13
```

Gambar 4.5.2 File keluaran

src > test.txt

```
10
5 5
AA AA BV BV BV
BV AA AA AA AA
AA BV BV AA AA
BV BV BV BV AA
AA AA BV AA AA
10
BV AA BV BV BV AA AA BV AA
65
BV AA BV BV BV BV AA
-48
AA BV BV AA AA
-16
BV AA
-44
BV BV AA BV
-13
AA AA BV
-8
BV BV
-9
AA AA BV AA BV BV BV
9
BV AA BV BV AA
-26
BV AA BV AA AA BV
-7
```

Gambar 4.5.3 File masukan

4.6. Test case 6: Kondisi ketika skor maksimal bernilai nol

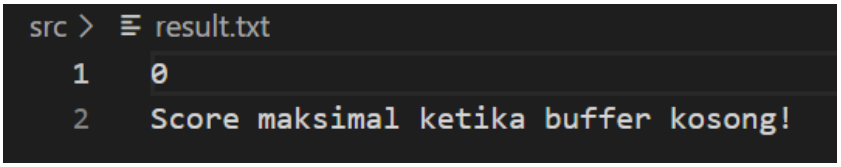
```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
1
Silakan masukkan nama file input (disertai dengan .txt): test.txt

Hasil:
0
Score maksimal ketika buffer kosong!

5428.492546081543 ms

Apakah ingin menyimpan solusi? (y/n) y
Silakan masukkan nama file output (tanpa .txt): result
Hasil berhasil disimpan dalam file result.txt!
```

Gambar 4.6.1 Tampilan pada CLI

src > test.txt

```
1   8
2   5 5
3   AA AA BV BV BV
4   BV AA AA AA AA
5   AA BV BV AA AA
6   BV BV BV BV AA
7   AA AA BV AA AA
8   8
9   BV AA BV BV BV BV AA
10  -48
11  AA BV BV AA AA
12  -16
13  BV AA
14  -44
15  BV BV AA BV
16  -13
17  AA AA BV
18  -8
19  BV BV
20  -9
21  BV AA BV BV AA
22  -26
23  BV AA BV AA AA BV
24  -7
```

Gambar 4.6.2 File masukan



Gambar 4.6.3 File keluaran

4.7. Test case 7: Kondisi ketika masukan pada CLI tidak valid

```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
2
Masukkan jumlah token unik: -1
Jumlah token unik harus bilangan bulat positif! Masukkan jumlah token unik: 3
Masukkan token dipisahkan dengan spasi: AA BB
Jumlah token tidak sesuai! Silakan masukkan token lagi: AA BB CC
Masukkan ukuran buffer: -1
Ukuran buffer harus lebih dari sama dengan nol! Masukkan ukuran buffer: 4
Masukkan ukuran matrix: -1 9
Ukuran matriks minimal 1x1! Masukkan ukuran matrix: 1 -9
Ukuran matriks minimal 1x1! Masukkan ukuran matrix: 4 4
Masukkan jumlah sekuens: -1
Jumlah sekuens harus lebih dari nol! Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: -1
Ukuran maksimal sekuens harus lebih dari nol! Masukkan ukuran maksimal sekuens: 3
```

Gambar 4.7.1 Tampilan pada CLI

```
Matriks:
CC BB CC BB
CC BB BB CC
AA AA CC CC
BB CC AA BB

Sekuens:
CC BB  reward =  53
CC CC BB  reward =  86
AA BB  reward =  74

Hasil:
139
CC CC BB
4,1
4,4
3,4
3,1

51765.72585105896 ms

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

Gambar 4.7.2 Tampilan pada CLI

4.8. Test case 8: Kondisi ketika token tidak valid pada pembacaan file

```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
1
Silakan masukkan nama file input (disertai dengan .txt): test.txt
Input invalid!
```

Gambar 4.8.1 Tampilan pada CLI

src > test.txt

```
1   8
2   5 5
3   AA AA BV BV BVB
4   BV AA AA AA AA
5   AA BV BV AA AA
6   BV BV BV BV AA
7   AA AA BV AA AA
8   8
9   BV AA BV BV BV BV AA
10  -48
11  AA BV BV AA AA
12  -16
13  BV AA
14  -44
15  BV BV AA BV
16  -13
17  AA AA BV
18  -8
19  BV BV
20  -9
21  BV AA BV BV AA
22  -26
23  BV AA BV AA AA BV
24  -7
```

Gambar 4.8.2 File masukan (terdapat token dengan panjang 3)

4.9. Test case 9: Kondisi ketika token tidak valid pada CLI

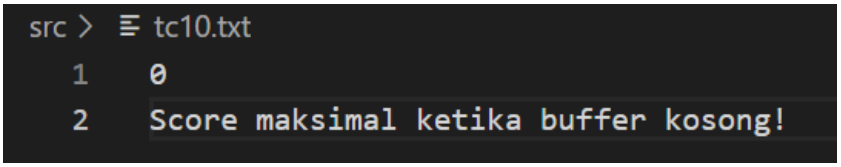
```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
2
Masukkan jumlah token unik: 4
Masukkan token dipisahkan dengan spasi: AAA BB CC DD
Masukkan ukuran buffer: 5
Masukkan ukuran matrix: 5 5
Masukkan jumlah sekuens: 5
Masukkan ukuran maksimal sekuens: 5

Matriks:
BB BB CC DD DD
DD BB DD BB BB
BB DD DD BB DD
DD CC DD AAA DD
BB DD DD DD CC

Sekuens:
CC AAA  reward =  56
DD CC BB  reward =  86
BB BB DD CC BB  reward =  93
CC AAA BB  reward =  6
CC CC AAA CC  reward =  13
Input invalid!
```

Gambar 4.9.1 Tampilan pada CLI (terdapat token dengan panjang 3)

4.10. Test case 10

```
evelyn@LAPTOP-MPQP26C1:/mnt/c/EVELYN/semester4/stima/tucil1_python/src$ python3 inputOutput.py
Apakah ingin menginput masukan melalui file atau CLI?
(1) File txt
(2) CLI
2
Masukkan jumlah token unik: 3
Masukkan token dipisahkan dengan spasi: AA BB CC
Masukkan ukuran buffer: 4
Masukkan ukuran matrix: 2 2
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 5

Matriks:
BB AA
BB CC

Sekuens:
BB BB CC BB  reward =  40
BB BB CC CC AA  reward =  30
BB BB BB  reward =  84

Hasil:
0
Score maksimal ketika buffer kosong!

35025.63285827637 ms

Apakah ingin menyimpan solusi? (y/n) y
Silakan masukkan nama file output (tanpa .txt): tc10
Hasil berhasil disimpan dalam file tc10.txt!
```

Gambar 4.10.1 Tampilan pada CLI



Gambar 4.10.2 File keluaran

BAB 5

Kesimpulan

Dari program penyelesaian permainan cyberpunk 2077 breach protocol dengan menggunakan algoritma brute force berbahasa python, dapat disimpulkan bahwa program ini dapat berjalan dengan baik. Program dapat menerima masukan dari CLI maupun file txt. Program berhasil mendeteksi input yang tidak valid, kemudian input akan diminta lagi atau program berhenti. Program berhasil menghasilkan buffer yang paling optimal. Program juga memiliki GUI yang dibuat dengan menggunakan flet, yang dapat menerima masukan secara manual maupun dari file txt.

DAFTAR PUSTAKA

Flet Documentation. Sumber: https://flet.dev/docs/

Munir, Rinaldi. 2023. Pengantar Strategi Algoritma. Bandung: Institut Teknologi Bandung Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Pengantar-Strategi-Algoritma-(2024).pdf

LAMPIRAN

Github Repository

https://github.com/evelynnn04/tucil1stima.git

Checklist Keberhasilan Program

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