

Tugas Kecil 1 IF2211
Strategi Algoritma Semester II tahun 2023/2024
Penyelesaian Cyberpunk 2077 Breach Protocol dengan
Algoritma Brute Force



Disusun oleh:
Fedrianz Dharma 13522090
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Daftar Isi

Bab I	3
Bab II	4
Langkah 1	4
Langkah 2	5
Langkah 3	5
Bab III	6
Main.py	6
Function.py	7
1. Fungsi run()	7
2. Prosedur saving(maxscore, buffer, coordinate, finish, start)	11
3. Prosedur print_matrix(matrix, default = True)	11
4. Fungsi score(sequence, buffer)	12
5. Fungsi increment_array(arr, xlimit, ylimit)	12
6. Fungsi validateMax(arr, ylimit, xlimit)	13
7. Fungsi FinalList(final_coordinate, final_buffer)	13
8. Fungsi bruteForce(matrix, sequences, buffer_size)	13
Gui.py	16
Bab IV	18
Lampiran	28
Referensi	29

Bab I

Deskripsi Masalah

Cyberpunk 2077 Breach Protocol adalah minigame pada *video game* Cyberpunk 2077. Minigame ini merupakan simulasi peretasan jaringan local dari ICE (Intrusion Countermeasures Electronics) pada *game* 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.

(paragraf di atas dikutip dari Deskripsi Tugas pada Tugas Kecil 1 IF2211 Strategi Algoritma Semester II tahun 2023/2024 Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force)

Bab II

Algoritma Brute Force untuk Penyelesaian Permainan Breach Protocol

Algoritma *brute force* adalah pendekatan yang digunakan untuk memecahkan suatu masalah atau persoalan dengan mencoba semua kemungkinan solusi secara sistematis and terurut hingga menemukan solusi yang benar. Algoritma *brute force* dapat digunakan untuk menyelesaikan permainan Breach Protocol dengan mencoba semua kemungkinan solusi yang valid. Setelah itu, program akan mengeluarkan urutan solusi yang paling besar *rewards*-nya dan paling optimal.

Langkah 1

Pilih token yang berada paling kiri atas pada matrix dan cari semua kemungkinan solusi dengan membuat sebuah urutan pergerakan yang dapat dilakukan dari token tersebut untuk memilih token berikutnya. Panjang dari urutan pergerakan tersebut akan bergantung pada *buffer size*. Semakin tinggi *buffer size*, maka akan semakin panjang urutan pergerakan.

Urutan pergerakan dapat dibuat seperti pada *tally counter*, namun dengan *limit* yang tidak seragam pada setiap “slot” dan “slot” dimulai dari angka 1.



Gambar 1. Mechanical counter

“Slot” pada urutan ganjil memiliki *limit* sesuai dengan banyaknya baris pada matrix dikurangi 1, sedangkan “slot” pada urutan genap memiliki *limit* sesuai dengan banyaknya kolom pada matrix dikurangi 1. “Slot” pada urutan ganjil akan mengatur banyaknya langkah vertikal, sedangkan “slot” pada urutan genap akan mengatur

banyaknya langkah horizontal. Ketika “slot” sudah melewati *limit*, “slot” akan kembali menjadi angka 1 dan “slot” yang berada di kirinya nilainya akan ditambah 1. Contoh kemungkinan urutan pergerakan dengan *buffer size* 5 adalah 1 1 1 1.

Urutan pergerakan yang dihitung kemungkinannya hanyalah pergerakan ke bawah dan ke kanan. Pergerakan ke atas dan ke kiri akan memanfaatkan batas matrix. Jika sudah melewati batas matrix paling bawah, posisi akan kembali lagi ke paling atas dan jika sudah melewati batas matrix paling kanan, posisi akan kembali lagi ke paling kiri. Hal ini sama saja dengan melakukan pergerakan ke atas ataupun ke kiri sebanyak jumlah baris atau kolom dikurangi jumlah pergerakan.

Langkah 2

Setelah membuat sistem urutan pergerakannya, lakukan pencatatan token-token yang dipilih berdasarkan urutan pergerakan. Setiap penambahan satu token, cek jumlah *reward* yang bisa didapatkan oleh urutan token pada *buffer* saat ini. Jika *reward* yang didapat lebih besar dari *reward* yang ada pada catatan, hapus catatan yang ada. Namun, jika *rewardnya* sama, catatan sebelumnya tidak dihapus. Catat urutan token pada *buffer* tersebut beserta *reward* yang didapat. Jika pada saat melakukan pergerakan terdapat sel matrix yang terpilih 2 kali, maka langsung mencoba kemungkinan urutan pergerakan selanjutnya. Lakukan terus hingga urutan pergerakan saat ini selesai dan *buffer* penuh.

Langkah 3

Setelah semua pergerakan pada urutan pergerakan dilakukan, nilai pada “slot” paling kanan akan bertambah 1. Lakukan kembali **Langkah 2** hingga semua kemungkinan urutan pergerakan telah dicoba, yaitu ketika nilai pada setiap “slot” telah mencapai nilai maksimalnya.

Setelah itu, cek catatan dan cari isi *buffer* yang memiliki panjang terpendek. Isi *buffer* tersebut merupakan solusi paling optimal, yaitu solusi dengan *reward* terbesar dan panjang *buffer* terpendek untuk permainan Breach Protocol.

Bab III

Implementasi Program dengan Python

Algoritma pada Bab II dapat diimplementasikan dalam sebuah program dengan menggunakan bahasa Python. Program dapat dibagi menjadi 2 bagian, yaitu **main.py** yang memuat program utama dan **function.py** yang berisi fungsi-fungsi yang akan digunakan pada program utama untuk menyelesaikan permainan Breach Protocol.

Main.py

[illegible]

Main.py berisi splash screen dan juga berguna sebagai program utama dari aplikasi dengan menjalankan fungsi-fungsi yang ada pada **function.py**. Pada **main.py** juga terdapat **timer** yang digunakan untuk menghitung waktu eksekusi program.

Function.py

Function.py berisi fungsi dan prosedur yang digunakan untuk berinteraksi dengan I/O pengguna dan file dan juga fungsi-fungsi yang digunakan untuk menyelesaikan permainan Breach Protocol.

1. Fungsi run()

```
def run():
    fromFile = input("\nMasukkan dari file txt? (y/n): ").lower()
    while fromFile != 'y' and fromFile != 'n':
        fromFile = input("Masukkan dari file txt? (y/n): ").lower()
    if fromFile == 'y':
        foldertest = input("File pada folder test? (y/n): ")
        while foldertest != 'y' and foldertest != 'n':
            foldertest = input("File pada folder test? (y/n): ")
        error = True
        while error:
            if foldertest == 'y':
                filename = input("Masukkan nama file txt: ")
                filename = "../test/" + filename
                while not(os.path.isfile(filename)):
                    print("\nFile tidak ditemukan.\n")
                    filename = input("Masukkan nama file txt: ")
                    filename = "../test/" + filename
                file = open(filename, 'r')
            else:
                path = input("Masukkan absolute path dari file txt: ")
                while not(os.path.isfile(path)):
                    print("\nFile tidak ditemukan.\n")
                    path = input("Masukkan absolute path dari file txt: ")
                file = open(path, 'r')
        print()

    while True:
        try:
            buffer_size = int(file.readline())
        except:
            print("Buffer size harus berupa integer\n")
            break
        try:
            matrix_size = file.readline().split()
            matrix_width = int(matrix_size[0])
            matrix_height = int(matrix_size[1])
        except:
            print("Ukuran matrix harus berupa integer\n")
```

```

        break
    try:
        matrix = []
        for i in range(matrix_height):
            cell = file.readline().split()
            isAlphaNum = all(element.isalnum() for element in cell)
            isLen2 = all(len(element) == 2 for element in cell)
            if isAlphaNum and isLen2:
                matrix.append(cell)
                if len(matrix[i]) != matrix_width:
                    print("Jumlah kolom matrix tidak sesuai.\n")
                    raise Exception()
            else:
                print("Cell matrix harus terdiri dari token dengan 2 karakter
alphanumeric.\n")
                raise Exception()
        except:
            break
    try:
        number_of_sequences = int(file.readline().strip('\n'))
    except:
        print("Jumlah sequences harus berupa integer.\n")
        break
    try:
        matrix_sequence = []
        for i in range(number_of_sequences):
            seq = file.readline().split()
            isAlphaNum = all(element.isalnum() for element in seq)
            isLen2 = all(len(element) == 2 for element in seq)
            if isAlphaNum and isLen2:
                matrix_sequence.append(seq)
            else:
                print("Sequence harus terdiri dari token dengan 2 karakter
alphanumeric.\n")
                raise Exception()
        try:
            matrix_sequence[i].append(int(file.readline().strip('\n')))
        except:
            print("Reward harus berupa integer\n")
            raise Exception()
    except:
        break
    else:
        error = False
        break
    return buffer_size, matrix, matrix_sequence
else:
    while True:
        while True:
            try:
                token_unik = int(input("\nMasukkan jumlah token unik: "))
                if token_unik < 2:
                    raise Exception()

```



```

        except:
            print("Harus berupa angka dan minimal 2.")
        else:
            break
    while token_unik <= 0:
        print("Jumlah token unik tidak boleh 0 atau negatif.")
        token_unik = int(input("Masukkan jumlah token unik: "))

    token = input("Masukkan token unik (alphanumeric): ").split()
    isAlphaNum = all(element.isalnum() for element in token)
    isLen2 = all(len(element) == 2 for element in token)
    while len(set(token)) != token_unik or not(isAlphaNum) or not(isLen2):
        print("Token harus berupa alphanumeric dan berjumlah sama dengan jumlah token
unik")

        token = input("Masukkan token unik: ").split()
        isAlphaNum = all(element.isalnum() for element in token)
        isLen2 = all(len(element) == 2 for element in token)
    while True:
        try:
            buffer_size = int(input("Masukkan ukuran buffer: "))
        except:
            print("Harus berupa angka.")
        else:
            break
    while buffer_size < 2:
        print("Ukuran buffer minimal adalah 2.")
        buffer_size = int(input("Masukkan ukuran buffer: "))
    ukuran_matrix = input("Masukkan ukuran matrix (col,row): ").split()
    while len(ukuran_matrix) != 2 :
        print("Harus memasukkan kolom dan baris.")
        ukuran_matrix = input("Masukkan ukuran matrix (col,row): ").split()
        while int(ukuran_matrix[0]) <= 0 or int(ukuran_matrix[1]) <= 0 or
((int(ukuran_matrix[0]) * int(ukuran_matrix[1])) < token_unik):
            print("Ukuran matrix tidak boleh 0 ataupun negatif dan ukuran matrix harus
lebih besar dari jumlah token unik.")
            ukuran_matrix = input("Masukkan ukuran matrix (col,row): ").split()

    jumlah_seq = int(input("Masukkan jumlah sequences: "))
    while jumlah_seq <= 0:
        print("Jumlah sequence minimal 1.")
        jumlah_seq = int(input("Masukkan jumlah sequences: "))
    ukuran_max_seq = int(input("Masukkan ukuran maksimal sequences: "))
    while ukuran_max_seq < 2:
        print("Panjang minimal sequence adalah 2.")
        ukuran_max_seq = int(input("Masukkan ukuran maksimal sequences: "))
    print()
    matrix = [[0 for i in range(int(ukuran_matrix[0]))] for j in
range(int(ukuran_matrix[1]))]
    minimumlength = min(ukuran_matrix[0], ukuran_matrix[1])
    for item in token:
        x = random.randint(0, len(matrix)-1)
        y = random.randint(0, len(matrix[0])-1)
        while matrix[x][y] != 0:
            x = random.randint(0, len(matrix)-1)

```

```

        y = random.randint(0, len(matrix[0])-1)
        matrix[x][y] = item
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] != 0:
                continue
            else:
                matrix[i][j] = random.choice(token)
    isUnique = False
    matrix_sequence = [[random.choice(token) for i in
range(random.randint(2,ukuran_max_seq))] for j in range(jumlah_seq)]
    reroll = False
    count = 0
    while not(isUnique):
        for i in range(len(matrix_sequence)-1):
            if matrix_sequence[i] in matrix_sequence[i+1:]:
                matrix_sequence = [[random.choice(token) for bla in
range(random.randint(2,ukuran_max_seq))] for blabla in range(jumlah_seq)]
                reroll = True
                count += 1
                break
        if not reroll:
            isUnique = True
        reroll = False
        if count >= 500:
            print("Jumlah token unik tidak cukup untuk membuat sequence yang unik.")
            break
    if count < 500:
        break
    for item in matrix_sequence:
        item.append(random.randint(1,100))
    print_matrix(matrix)
    print_matrix(matrix_sequence, False)
    print()
    return buffer_size, matrix, matrix_sequence

```

Fungsi `run()` digunakan untuk membaca input dari file atau input dari pengguna dan mengembalikan ***buffer_size***, ***matrix***, dan ***matrix_sequence***. Matrix berisi token-token yang akan dipilih untuk menyusun urutan kode. Matrix sequence berisi rangkaian token yang harus dicocokkan beserta *rewardnya*. Fungsi `run()` juga melakukan validasi, baik input dari file maupun input dari pengguna. Misalnya seperti, input token “AAA” atau token yang mengandung *special characters* “!@”, maka program akan meminta input ulang. Untuk input dari pengguna, fungsi `run()` juga melakukan pengisian ***matrix*** dan ***matrix_sequence*** secara random sesuai dengan spesifikasi yang telah di input oleh pengguna dan menampilkan ***matrix*** dan ***matrix_sequence*** yang telah terisi ke layar.

2. Prosedur saving(maxscore, buffer, coordinate, finish, start)

```
def saving(maxscore, buffer, coordinate, finish, start):
    saveOrNo = input("Apakah ingin menyimpan solusi? (y/n): ")
    while saveOrNo != 'y' and saveOrNo != 'n':
        saveOrNo = input("Apakah ingin menyimpan solusi? (y/n): ").lower()
    if saveOrNo == 'y':
        while True:
            filename = input("Masukkan nama file (termasuk .txt): ")
            filename = "../test/" + filename
            print()
            isExistAndFile = os.path.isfile(filename)
            if isExistAndFile:
                print("File dengan nama tersebut sudah ada.")
                print("Apakah Anda ingin mengubah nama file atau rewrite?\n")
                timpa = input("r untuk rewrite dan u untuk ubah nama file: ").lower()
                print()
                while timpa != 'r' and timpa != 'u':
                    timpa = input("r untuk rewrite dan u untuk ubah nama file: ").lower()
                if timpa == 'u':
                    continue
                else:
                    break
            else:
                break
        file = open(filename, "w")
        file.write(str(maxscore) + '\n')
        if maxscore == 0:
            file.write("Tidak memiliki kombinasi dengan score positif.\n")
        else:
            file.write(buffer + '\n')
            for tuple in coordinate:
                x,y = tuple
                file.write(str(x) + ", " + str(y) + '\n')
            file.write('\n')
        file.write(str((finish - start)*1000) + " ms")
        file.write('\n')
```

Prosedur **saving(maxscore, buffer, coordinate, finish, start)** digunakan untuk memberikan *prompt* melakukan penyimpanan solusi yang telah didapatkan beserta waktu eksekusi program ke dalam sebuah file berekstensi *.txt*.

3. Prosedur print_matrix(matrix, default = True)

```
def print_matrix(matrix, default=True):
    if default: # scorenya disamping
        for i in range(len(matrix)):
            for j in range(len(matrix[i])):
                if j != len(matrix[i])-1:
                    print(matrix[i][j], end=" ")
                else:
                    print(matrix[i][j])
            print()
```

```

else: # scorenya di newline
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if j < len(matrix[i])-2:
                print(matrix[i][j], end=" ")
            else:
                print(matrix[i][j])

```

Prosedur **print_matrix(matrix, default=True)** digunakan untuk menampilkan **matrix** dan **matrix_sequence** yang terbentuk dari input CLI. **Matrix** akan ditampilkan dengan default = True, sedangkan untuk menampilkan **matrix_sequence** default = False.

4. Fungsi score(sequence, buffer)

```

def score(sequence, buffer):
    # strbuffer = " ".join(buffer) # kalo buffernya dalam bentuk array
    strbuffer = buffer
    score = 0
    strsequence = []
    for i in range(len(sequence)):
        strsequence.append(" ".join(sequence[i][: -1]))
    for i in range(len(strsequence)):
        if strsequence[i] in strbuffer:
            score += int(sequence[i][ -1])
    return score

```

Fungsi **score(sequence, buffer)** digunakan untuk menghitung jumlah **reward** yang didapatkan untuk **buffer** saat ini.

5. Fungsi increment_array(arr, xlimit, ylimit)

```

def increment_array(arr, xlimit, ylimit):
    for i in range(len(arr) - 1, -1, -1):
        if (i+1) % 2 == 1: # ganjil berarti yang terakhir adalah vertikal
            arr[i] = ((arr[i]) % ylimit) + 1
        else:
            arr[i] = ((arr[i]) % xlimit) + 1
        if arr[i] != 1:
            break
    return arr

```

Fungsi **increment_array(arr, xlimit, ylimit)** digunakan untuk menambahkan nilai pada index terakhir array sebanyak 1. Jika nilai pada suatu index pada array tersebut telah melebihi batas, nilai pada index yang sebelumnya akan ditambahkan satu juga.

6. Fungsi validateMax(arr, ylimit, xlimit)

```
def validateMax(arr, ylimit, xlimit):
    isMax = True
    for i in range(len(arr)):
        if (i+1) % 2 == 1:
            if arr[i] != ylimit:
                isMax = False
        else:
            if arr[i] != xlimit:
                isMax = False
    return isMax
```

Fungsi validateMax(arr, ylimit, xlimit) digunakan untuk mengetahui apakah setiap index pada array telah mencapai batas yang sudah ditentukan. Jika setiap index sudah mencapai batas, fungsi akan mengembalikan nilai True, sebaliknya jika belum mencapai nilai batas, fungsi akan mengembalikan nilai False.

7. Fungsi FinalList(final_coordinate, final_buffer)

```
def FinalList(final_coordinate, final_buffer):
    if len(final_buffer) == 0:
        return final_buffer, final_coordinate
    if all(buffer == final_buffer[0] for buffer in final_buffer):
        return final_buffer[0], final_coordinate[0]
    else:
        buffer = min(final_buffer, key=len)
        coordinate = np.where(np.array(final_buffer) == buffer)
        return buffer, final_coordinate[coordinate[0][0]]
```

Fungsi FinalList(final_coordinate, final_buffer) digunakan untuk mengembalikan solusi akhir yang paling optimal dari beberapa solusi yang ada dengan mencari solusi yang memiliki panjang buffer paling pendek. Selain itu, fungsi FinalList juga akan mengembalikan koordinat dari setiap token yang ada pada solusi optimal yang terpilih secara berurutan.

8. Fungsi bruteForce(matrix, sequences, buffer_size)

```
def bruteForce(matrix, sequences, buffer_size):
    maxscore = 0
    buffer_coordinate = []
```

```

final_coordinate = []
final_buffer = []
ymax = len(matrix) - 1
xmax = len(matrix[0]) - 1
for i in range(len(matrix[0])):
    movement = [1 for i in range(buffer_size-1)]
    is_horizontal = False
    string = matrix[0][i]
    y = 0
    x = i
    buffer_coordinate.append((x+1,y+1))
    while not(validateMax(movement, ymax, xmax)): # ngitung semua kemungkinan
        for j in range(len(movement)): # melakukan pergerakan sesuai dengan kemungkinan
            movement saat ini
            if is_horizontal:
                x = (x + movement[j]) % len(matrix[0])
                if (x+1,y+1) in buffer_coordinate: # kalo movement yang dipakai terdapat 2
                    cell matrix yang dilewati
                    break # maka langsung keluar loop dan mencari kemungkinan movement
                    yang lain
            else:
                string += " " + matrix[y][x]
                buffer_coordinate.append((x+1,y+1))
                is_horizontal = False
                curr_score = score(sequences, string) # hitung score saat ini
                if curr_score > maxscore:
                    final_coordinate = []
                    final_buffer = []
                    maxscore = score(sequences, string)
                    final_buffer.append(copy.deepcopy(string))
                    final_coordinate.append(copy.deepcopy(buffer_coordinate))
                elif curr_score >= maxscore: # bisa dioptimasi dengan menghapus bagian
                    ini mungkin? Gak jadi karena yang optimal bisa dibelakangan
                    maxscore = score(sequences, string)
                    final_buffer.append(copy.deepcopy(string))
                    final_coordinate.append(copy.deepcopy(buffer_coordinate))
            else:
                y = (y + movement[j]) % len(matrix)
                if (x+1,y+1) in buffer_coordinate:
                    break
            else:
                string += " " + matrix[y][x]
                buffer_coordinate.append((x+1,y+1))
                is_horizontal = True
                curr_score = score(sequences, string) # hitung score saat ini
                if curr_score > maxscore:
                    final_coordinate = []
                    final_buffer = []
                    maxscore = score(sequences, string)
                    final_buffer.append(copy.deepcopy(string))
                    final_coordinate.append(copy.deepcopy(buffer_coordinate))
                elif curr_score >= maxscore:
                    maxscore = score(sequences, string)
                    final_buffer.append(copy.deepcopy(string))

```

```

        final_coordinate.append(copy.deepcopy(buffer_coordinate))
    curr_score = score(sequences, string)
    if curr_score > maxscore:
        final_coordinate = []
        final_buffer = []
        maxscore = score(sequences, string)
        final_buffer.append(copy.deepcopy(string))
        final_coordinate.append(copy.deepcopy(buffer_coordinate))
    elif curr_score >= maxscore:
        maxscore = score(sequences, string)
        final_buffer.append(copy.deepcopy(string))
        final_coordinate.append(copy.deepcopy(buffer_coordinate))

    y = 0
    x = i
    is_horizontal = False
    string = matrix[0][i]
    buffer_coordinate = []
    buffer_coordinate.append((x+1,y+1))
    if xmax == 0 or ymax == 0:
        break
    increment_array(movement, xmax, ymax)
    buffer, coordinate = FinalList(final_coordinate, final_buffer, maxscore, sequences)
    print(maxscore)
    if maxscore > 0:
        print(buffer)
        for tuple in coordinate:
            x,y = tuple
            print(str(x) + ", " + str(y))
    else:
        print("Tidak ada solusi yang menghasilkan nilai positif")
    return maxscore, buffer, coordinate

```

Fungsi `bruteForce(matrix, sequences, buffer_size)` merupakan implementasi dari algoritma pada Bab II yang melakukan brute force semua kemungkinan yang ada dan menghitung *reward* yang didapatkan dari setiap kemungkinan tersebut.

Fungsi ini akan menerima `matrix` permainan, `sequences` yang harus dicocokkan, dan juga ukuran `buffer`. Fungsi ini mengembalikan dan menampilkan bobot hadiah, solusi paling optimal, serta koordinat dari masing-masing token yang ada pada solusi.

Fungsi `bruteForce` akan menghitung setiap kemungkinan urutan pergerakan yang dapat dilakukan. Dari urutan pergerakan tersebut fungsi akan memilih token-token yang ada pada `matrix` ke dalam `buffer`. Setiap memasukkan 1 token ke dalam `buffer`, fungsi akan menghitung *reward* yang bisa didapatkan dengan kondisi `buffer` saat ini. Setelah itu, fungsi akan melanjutkan pemilihan token berdasarkan urutan pergerakan. Setelah urutan

pergerakan selesai dilakukan, fungsi bruteForce akan memanggil fungsi increment_array untuk mendapatkan kemungkinan urutan pergerakan yang berikutnya.

Setelah semua kemungkinan gerakan telah dilakukan, fungsi akan memanggil fungsi FinalList untuk mendapat solusi yang memiliki panjang buffer terpendek dengan *reward* terbesar. Setelah itu fungsi akan menampilkan dan mengembalikan boboh hadiah yang didapat, solusi paling optimal, dan koordinat dari setiap token yang ada di solusi.

Gui.py

```
Window = Tk()
window.geometry("900x800")
window.title("Breach Protocol")
window.config(bg="Black")
frame = Frame(window)
frame.config(bg="white")
frame2 = Frame(window)
frame2.config(bg="black")

frame3 = Frame(window)
frame3.config(bg='black')

canvas = Canvas(frame3, width=400, height=400)
canvas.pack()
frame4 = Frame(window)
frame4.config(bg='black')

frame5 = Frame(window)
frame5.config(bg='white')

labelseq = Label(frame5, text="Sequence: ", width=20)
labelseq.pack(side=TOP)
canvas2 = Canvas(frame5, width=200, height=300)
canvas2.pack()

entry1 = Entry(frame4)
entry1.grid(row=0,column=1)
labelentry1 = Label(frame4, text='Jumlah token: ', width=25).grid(row=0,column=0, padx=5)
entry2 = Entry(frame4)
entry2.grid(row=1,column=1)
labelentry2 = Label(frame4, text='Token unik: ', width=25).grid(row=1,column=0, padx=5)
entry3 = Entry(frame4)
entry3.grid(row=2,column=1)
labelentry3 = Label(frame4, text='Ukuran buffer: ', width=25).grid(row=2,column=0, padx=5)
entry4 = Entry(frame4)
entry4.grid(row=3,column=1)
labelentry4a = Label(frame4, text='Panjang kolom: ', width=25).grid(row=3,column=0, padx=5)
entry4b = Entry(frame4)
entry4b.grid(row=4,column=1)
```



```

labelentry4b = Label(frame4, text='Panjang baris: ', width=25).grid(row=4,column=0, padx=5)
entry5 = Entry(frame4)
entry5.grid(row=5,column=1)
labelentry5 = Label(frame4, text='Jumlah sequence: ', width=25).grid(row=5,column=0, padx=5)
entry6 = Entry(frame4)
entry6.grid(row=6,column=1)
labelentry6 = Label(frame4, text='Panjang maksimal sequence: ', width=25).grid(row=6,column=0, padx=5)

fileButton = Button(frame, text="BROWSE", command=openFile, width=5, padx=10).grid(column=0, row= 0,
pady=5, padx=10)
saveButton = Button(frame, text='Save', command=saveFile,width=5, padx=10).grid(column=0, row= 5,
pady=5, padx=10)
saveButton = Button(frame4, text='Submit', command=submit,width=5, padx=10).grid(column=0, row= 7,
pady=5, padx=10)

maxscore2 = StringVar()
labelmaxscore = Label(frame2, text='Reward: ', fg="green", width=10).grid(column=0, row= 0, pady=5,
padx=10)
label1 = Label(frame2, textvariable=maxscore2, width=10).grid(column=0, row= 3, pady=(5,5), padx=10)

buffer2 = StringVar()
labelbuffer = Label(frame2, text='Solution: ', fg="green", width=30).grid(column=5, row= 0, pady=5,
padx=10)
label2 = Label(frame2, textvariable=buffer2, width=30).grid(column=5, row= 3, pady=(5,5), padx=10)

time2 = StringVar()
labeltime = Label(frame2, text='Time: ', fg="green", width=10).grid(column=10, row= 0, pady=5,
padx=(10,20))
label3 = Label(frame2, textvariable=time2, width=10).grid(column=10, row= 3, pady=(5,5), padx=(10,20))

frame.grid(column=0, row=2, padx=20)
frame2.grid(column=0, row=0, pady=20)
frame3.grid(column=0, row=1, ipadx=5, ipady=5)
frame4.grid(column=1, row=0, pady=20, padx=20)
frame5.grid(column=1, row=1)

window.mainloop()

```

Bagian ini merupakan implementasi program utama dengan menggunakan GUI. Pada *gui.py* terdapat 3 fungsi untuk membantu dalam implementasi, yaitu fungsi untuk membaca masukkan dari file txt, fungsi untuk membaca dari input manual, dan fungsi untuk melakukan save ke dalam file txt.

Bab IV

Test

1. Run program



2. Input dari file

```
Masukkan dari file txt? (y/n): y
Masukkan nama file txt: file.txt
```

```
50
```

```
7A BD 7A BD 1C BD 55
```

```
1, 1
```

```
1, 4
```

```
3, 4
```

```
3, 5
```

```
6, 5
```

```
6, 3
```

```
1, 3
```

```
779.2339324951172 ms
```

3. Input manual

```
Masukkan dari file txt? (y/n): n
```

```
Masukkan jumlah token unik: a
Harus berupa angka.
```

```
Masukkan jumlah token unik: 3
```

```
Masukkan token unik (alphanumeric): 1 1 1
```

```
Token harus berupa alphanumeric dan berjumlah sama dengan jumlah token unik
```

```
Masukkan token unik: ! ! !
```

```
Token harus berupa alphanumeric dan berjumlah sama dengan jumlah token unik
```

```
Masukkan token unik: 11 22 33
```

```
Masukkan ukuran buffer: 4
```

```
Masukkan ukuran matrix (col,row): 4 4
```

```
Masukkan jumlah sequences: 4
```

```
Masukkan ukuran maksimal sequences: 1
```

```
Panjang minimal sequence adalah 2.
```

```
Masukkan ukuran maksimal sequences: 3
```

```
Matrix:
```

```
33 22 11 11
```

```
33 11 33 11
```

```
22 33 33 22
```

```
22 11 22 22
```

```
Sequence 1: 22 11 33
```

```
Reward: 33
```

```
Sequence 2: 22 33
```

```
Reward: 80
```

```
Sequence 3: 11 11 22
```

```
Reward: 41
```

```
Sequence 4: 11 22 33
```

```
Reward: 4
```

```
84
```

```
11 22 33
```

```
4, 1
```

```
4, 3
```

```
2, 3
```

```
15.302181243896484 ms
```

4. Saving

```
Masukkan dari file txt? (y/n): y
Masukkan nama file txt: file3.txt

0
Tidak ada solusi yang menghasilkan nilai positif

1.0247230529785156 ms

Apakah ingin menyimpan solusi? (y/n): y
Masukkan nama file (termasuk .txt): out3.txt
```

```
test > ≡ out3.txt
1  0
2  Tidak memiliki kombinasi dengan score positif.
3
4  1.0247230529785156 ms
5
```

5. Test Case 1

Input:

```
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
```

Output:

```
Masukkan dari file txt? (y/n): y
Masukkan nama file txt: file.txt
```

```
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

779.2339324951172 ms
```

6. Test Case 2

Input:

```
test > file2.txt
1 | 7
2 | 1 1
3 | 7A
4 | 1
5 | 7A
6 | 15
```

Output:

```
Masukkan dari file txt? (y/n): y
Masukkan nama file txt: file2.txt

15
7A
1, 1

0.0 ms
```

7. Test Case 3

Input:

```
test > file3.txt
1 7
2 1 1
3 7A
4 1
5 8A
6 15
```

Output:

```
Masukkan dari file txt? (y/n): y
Masukkan nama file txt: file3.txt

0
Tidak ada solusi yang menghasilkan nilai positif

1.0247230529785156 ms
```

8. Test Case 4

Input:

```
test > file4.txt
1 4
2 2 2
3 7A 8A
4 9A 1A
5 4
6 7A 8A
7 15
8 7A 9A
9 10
10 7A 9A 8A 1A
11 20
12 7A 9A 1A 8A
13 10
```

Output:

```
Masukkan dari file txt? (y/n): y
Masukkan nama file txt: file4.txt
```

```
20
7A 9A 1A 8A
1, 1
1, 2
2, 2
2, 1
```

```
0.0 ms
```

9. Test Case 5

Input:

```
Masukkan dari file txt? (y/n): n

Masukkan jumlah token unik: 5
Masukkan token unik (alphanumeric): 11 22 33 44 55
Masukkan ukuran buffer: 5
Masukkan ukuran matrix (col,row): 5 5
Masukkan jumlah sequences: 5
Masukkan ukuran maksimal sequences: 5

Matrix:
33 33 22 11 33
55 22 22 44 22
44 11 55 22 22
11 44 11 11 44
55 22 44 55 55

Sequence 1: 11 55 22
Reward: 42
Sequence 2: 33 44 44
Reward: 91
Sequence 3: 11 55 11
Reward: 55
Sequence 4: 55 55
Reward: 1
Sequence 5: 33 44
Reward: 29
```

Output:

```
121
33 44 44 55 55
2, 1
5, 5
1, 5

19.077539443969727 ms
```

10. Test Case 6

Input:

```
Masukkan dari file txt? (y/n): n

Masukkan jumlah token unik: 2
Masukkan token unik (alphanumeric): 11 22
Masukkan ukuran buffer: 2
Masukkan ukuran matrix (col,row): 2 2
Masukkan jumlah sequences: 2
Masukkan ukuran maksimal sequences: 2

Matrix:
11 11
22 22

Sequence 1: 22 11
Reward: 43
Sequence 2: 11 11
Reward: 93
```

Output:

```
0
Tidak ada solusi yang menghasilkan nilai positif

0.0 ms
```

11. Test Case 7

Input:


```
Masukkan dari file txt? (y/n): n

Masukkan jumlah token unik: 3
Masukkan token unik (alphanumeric): 11 22 33
Masukkan ukuran buffer: 3
Masukkan ukuran matrix (col,row): 3 3
Masukkan jumlah sequences: 3
Masukkan ukuran maksimal sequences: 3

Matrix:
33 11 11
33 22 33
22 22 33

Sequence 1: 11 22
Reward: 89
Sequence 2: 33 22 11
Reward: 71
Sequence 3: 22 11
Reward: 92
```

Output:

```
89
11 22
2, 1
2, 2

1.2485980987548828 ms
```

12. Test Case GUI

Breach Protocol

Reward:

Solution:

Time:

50

7A BD 7A BD 1C BD 55

0.72 ms

Jumlah token:

Token unik:

Ukuran buffer:

Panjang kolom:

Panjang baris:

Jumlah sequence:

Panjang maksimal sequence:

Submit

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

Sequence:

BROWSE

Save

13. Test Case GUI 2

Reward:

84

Solution:

33 44 44

Time:

0.00 ms

Jumlah token:

4

Token unik:

11 22 33 44

Ukuran buffer:

4

Panjang kolom:

4

Panjang baris:

4

Jumlah sequence:

4

Panjang maksimal sequence:

4

Submit

44	44	33	33
11	22	11	33
44	33	44	33
22	33	22	11

Sequence:

33	22	44	71
22	44	11	69
33	44	44	84
22	22	22	20

BROWSE

Save

Lampiran

Github repository:

https://github.com/FedrianzD/Tucil1_13522090

Tabel:

No	Poin	Yes	No
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	✓	

Referensi

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf)

<https://emojicombos.com/dot-art-generator>

<https://www.geeksforgeeks.org/print-colors-python-terminal/>

https://www.w3schools.com/python/python_try_except.asp

<https://patorjk.com/software/taag/#p=testall&f=BlurVision%20ASCII&t=Breach%20Protocol>

https://www.youtube.com/watch?v=TuLxsvK4svQ&t=10617s&ab_channel=BroCode