



**Institut Teknologi Bandung**  
Sekolah Teknik Elektro dan Informatika  
Program Studi Informatika Semester II 2023/2024

Laporan Tugas Kecil

**Penyelesaian Cyberpunk 2077 Breach Protocol dengan  
Algoritma Brute Force**

IF2211 Strategi Algoritma

Zahira Dina Amalia K01 - 13522085

## Daftar Isi

Daftar Isi.....	2
Algoritma Brute Force.....	3
Source Program.....	6
Tangkapan Layar Input dan Output.....	16

## Algoritma *Brute Force*

Algoritma *brute force* adalah salah satu strategi untuk menyelesaikan masalah dengan mencoba semua kemungkinan solusi. Pendekatan ini tidak menggunakan strategi pintar atau heuristik untuk mempercepat proses, melainkan secara langsung mengevaluasi setiap kemungkinan. Pada program dengan strategi *brute force* kali ini, secara garis besar terdapat tiga tahap pemrosesan yaitu, tahap membaca *input*, pencarian solusi, dan *output* solusi. Berikut deskripsi dari setiap tahapan tersebut.

### 1. Membaca *input*

Pada pembacaan input, terdapat dua pilihan cara yaitu dengan membaca sebuah file berekstensi '.txt' atau dengan membaca masukan dari pengguna melalui command line interface (CLI). Jika pembacaan dilakukan dengan membaca sebuah file, program akan meminta pengguna untuk memasukkan nama folder dan file yang akan dibaca. Kemudian membaca file sebagai berikut.

Isi file	Keterangan
7	→ Ukuran buffer
6 6	→ Kolom matriks dan Baris matriks
7A 55 E9 E9 1C 55	→ Elemen-elemen matriks
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	→ Jumlah sekuens
BD E9 1C	→ Sekuens 1
15	→ Bobot sekuens 1
BD 7A BD	→ Sekuens 2
20	→ Bobot sekuens 2
. . .	. . .
BD 1C	→ Sekuens n
30	→ Bobot sekuens n

Sebanyak n sekuens akan disimpan ke dalam sebuah array of array yang akan disimpan. Bobot tiap sekuens juga akan disimpan pada sebuah array of integer. Matriks juga akan disimpan dalam sebuah variabel dengan tipe data matriks. Pembacaan akan berhenti jika file menemui akhir baris yang juga bobot sekuens terakhir atau ketika kode pertama kali menemukan baris yang

hanya berisi “”. Sebaliknya, jika pengguna memilih menggunakan CLI, program akan meminta pengguna untuk memasukkan jumlah token unik, token unik yang diinginkan sebanyak jumlah token unik, ukuran buffer, ukuran matriks, jumlah sekuens, dan ukuran maksimal sekuens. Program kemudian akan membuat matriks dan sekuens unik secara random berdasarkan informasi yang didapatkan. Dengan memberikan dua pilihan ini, program menjadi lebih fleksibel karena dapat menyesuaikan dengan preferensi pengguna dalam memasukkan data masukan.

## 2. Pencarian solusi

Tahap ini dilakukan setelah mendapatkan data-data yang dibutuhkan dari tahap pertama. Berdasarkan deskripsi permainan pada spesifikasi tugas, pemain hanya dapat bergerak dengan pola horizontal, vertikal, horizontal, vertikal secara bergantian hingga semua sekuens berhasil dicocokkan atau buffer mencapai kapasitas maksimal. Pemain memulai dengan memilih satu token dari posisi baris paling atas dalam matriks, lalu mencocokkan sekuens pada token-token yang berada di buffer. Satu token dapat digunakan untuk lebih dari satu sekuens, dan setiap sekuens memiliki bobot hadiah atau reward yang beragam. Adapun syarat minimal untuk sebuah sekuens adalah memiliki panjang dua token. Pada masukan melalui CLI, pemain dapat menentukan maksimal panjang sekuens.

Setelah mendapatkan data masukan, program memulai pencarian solusi dengan langkah-langkah tertentu. Langkah pertama adalah memilih token dari baris teratas matriks sebagai token awal. Selanjutnya, program mencoba untuk mengisi buffer dengan token-token dari matriks lalu dicocokkan dengan sekuens yang ada. Pencarian solusi dilakukan dengan mencoba semua kemungkinan langkah secara bergantian, yaitu dengan bergerak horizontal dan vertikal, hingga semua sekuens berhasil dicocokkan atau buffer mencapai kapasitas maksimal. Setiap langkah yang diambil selama pencarian dievaluasi untuk melihat apakah sekuens yang ada berhasil dicocokkan dengan token-token yang ada di buffer. Jika sekuens berhasil dicocokkan, bobot hadiahnya dihitung dan dibandingkan dengan solusi terbaik yang telah ada sejauh ini. Solusi terbaik akan disimpan untuk digunakan dalam tahap output.

Program akan memulai pencarian dengan membuat sebuah array yang akan menyimpan langkah dan token dari setiap kemungkinan. Pergerakan akan dilakukan sambil membandingkan array tersebut dengan sekuens yang ada dan menghitung bobot dari array tersebut. Program akan melakukan iterasi pada merangkai array yang mungkin ini dan menyimpan array dengan bobot tertinggi. Dengan pendekatan ini, program dapat menemukan solusi terbaik dengan mengevaluasi setiap kemungkinan secara langsung, memastikan bahwa solusi yang dihasilkan akurat dan efisien.

### 3. *Output* solusi

Setelah pencarian selesai, program akan menampilkan solusi terbaik yang ditemukan kepada pengguna dengan detail yang lengkap. Solusi terbaik akan mencakup informasi mengenai bobot hadiah yang berhasil diperoleh, urutan token yang membentuk sekuens yang berhasil dipasangkan, langkah-langkah yang diambil untuk mencapai solusi tersebut dalam bentuk koordinat matriks, serta waktu yang dibutuhkan untuk proses pencarian solusi. Pengguna akan diberikan opsi untuk menyimpan solusi dalam file teks jika diinginkan, sehingga solusi dapat diakses kembali di kemudian hari. Dengan tampilan solusi yang terperinci ini, pengguna dapat memahami secara jelas bagaimana program menyelesaikan permainan dan dapat memanfaatkan informasi tersebut untuk keperluan selanjutnya.

## Source Program

Pada program ini terdapat dua file yakni 'main.py' dan 'functions.py'. File 'main.py' adalah tempat program nantinya dieksekusi, sedangkan file 'functions.py' merupakan tempat disimpannya fungsi-fungsi yang digunakan pada file 'main.py'. Berikut *source code* dari setiap file tersebut dengan deskripsi singkatnya.

### File: 'main.py'

```
print("|-----|")
print("|----          SELAMAT DATANG          ----|")
print("|----      DI CYPERBUNK 2077 BREACH      ----|")
print("|----          PROTOCOL SOLVER          ----|")
print("|-----|\n")
print("Silakan pilih metode masukan yang diinginkan \n")
print("| 1 | Metode TXT: dengan memasukkan file '.txt'")
print("| 2 | Metode CLI: dengan masukan dari command
      line")

metode = 999
while metode != 1 and metode != 2:
    try:
        metode = int(input("\nMetode yang dipilih: "))
        if metode == 1:
            bufferSize, matrix, sequences, rewards =
read_txt()
        elif metode == 2:
            bufferSize, matrix, sequences, rewards =
read_cli()
        else:
            print("\nPilih antara 1 dan 2")
            print("| 1 | Metode TXT: dengan memasukkan
file '.txt'")
            print("| 2 | Metode CLI: dengan masukan dari
command line")
    except ValueError:
        print("\nMasukan harus berupa angka antara 1 dan
2. Silakan coba lagi.")

if bufferSize != None:
    start = time.time()
    reward_solution, array_solution, steps_solution =
```

```

    process(bufferSize, matrix, sequences, rewards)
end = time.time()
process_time = end - start

print("\nSolusi dari data yang telah didapatkan
    adalah:")
output_cli(reward_solution, array_solution,
    steps_solution, process_time)

save = input("\nApakah ingin menyimpan solusi? (Y/N)
    ")
if save.upper() == 'Y':
    name = input("\nMasukkan nama untuk file solusi
        yang hendak disimpan: ")
    output_txt(reward_solution, array_solution,
        steps_solution, process_time, name)
else:
    print("\n|----          SEE YOU SOON!!
    ----|")

```

## File: 'functions.py'

### 1. Fungsi Utama dalam Membaca *Input*

→ read\_txt(): Untuk membaca dari file berekstensi ".txt"

```

def read_txt():
    buffer_size = 0
    matrix_cols = 0
    matrix_rows = 0
    matrix = []
    number_seq = 0
    sequences = []
    rewards = []
    belum = True

    while belum:
        file = input("Masukan nama file dengan format
            '.txt': ")
        path = os.path.join("test", file)

```





```

sequences, rewards)

    except FileNotFoundError:
        print(f"Error: File {path} tidak dapat
dibaca, pastikan file ada dan dalam bentuk '.txt'!")

    else:
        print(f"\nError: File dengan nama '{file}'
tidak ditemukan!")
        jawaban = input("Apakah ingin mencoba lagi?
(Y/N) ")
        answer = jawaban.upper()
        print("")
        if answer != 'Y':
            belum = False
            print("\n|----          SEE YOU SOON!!
----|")
            return None, None, None, None

return buffer_size, matrix, sequences, rewards

```

→ read\_cli(): Untuk membaca dari *command line interface* (CLI)

```

def read_cli():
    # Token
    number_token = int(input("Jumlah token unik yang
diinginkan: "))
    list_token = []
    for i in range(number_token):
        token = False
        while not token:
            token_i = input(f"Token ke-{i+1}: ")
            if len(token_i) == 2 and token_i.isalnum():
                token = True
                list_token.append(token_i)
            else:
                print("\nMasukan token dengan panjang 2
dan berisi alfanumerik!")

    # Buffer
    buffer_size = int(input("Masukan ukuran buffer yang
diinginkan: "))

```

```

# Matriks
aman = False
while not aman:
    while True:
        try:
            matrix_size = input("Masukan ukuran
matriks yang diinginkan (kolom baris): ")
            matrix_col = int((matrix_size.split())[0])
            matrix_row = int((matrix_size.split())[1])
            break
        except ValueError:
            print("Masukan harus berupa bilangan
bulat. Silakan coba lagi.")
    if matrix_col <= 8 and matrix_row <= 8:
        matrix = [[random.choice(list_token) for _ in
range(matrix_col)] for _ in range(matrix_row)]
        aman = True
    else:
        print("\nProgram akan berjalan sangat lama
untuk untuk matriks lebih besar dari 8x8, coba
masukan ukuran lain!")

# Sekuens
number_seq = int(input("Masukkan jumlah sekuens yang
diinginkan: "))
maximal_size_seq = int(input("Masukkan ukuran maksimal
sekuens yang diinginkan: "))
sequences = []
for j in range(number_seq):
    sequence = random_sequences(list_token,
maximal_size_seq)
    sequences.append(sequence)

# Rewards
rewards = []
for k in range(number_seq):
    reward = random.randint(1, 50)
    rewards.append(reward)
print_data(buffer_size, matrix, sequences, rewards)

return buffer_size, matrix, sequences, rewards

```

## 2. Fungsi Utama dalam Pencarian Solusi

→ `startMove0N(matrix, n)`: Untuk memulai merangkai array kemungkinan solusi dari baris pertama

```
def startMove0N(matrix, n):
    currentArray = []
    currentSteps = []
    for i in range(1, len(matrix)):
        array = []
        steps = []
        array.append(matrix[0][n])
        steps.append([0,n])
        array.append(matrix[i][n])
        steps.append([i,n])
        currentArray.append(array)
        currentSteps.append(steps)
    return currentArray, currentSteps
```

→ `horizontal_move(arrayBefore, stepsBefore, matrix)`: Untuk memulai meneruskan rangkaian array kemungkinan solusi untuk saat pergerakan horizontal

```
def horizontal_move(arrayBefore, stepsBefore, matrix):
    currentArray = []
    currentSteps = []

    for p in range(len(stepsBefore)):
        i = stepsBefore[p][len(stepsBefore[p])-1][0]
        for j in range(len(matrix[i])):
            old_array = arrayBefore[p]
            old_steps = stepsBefore[p]
            if [i,j] not in old_steps:
                new_steps = old_steps.copy()
                new_array = old_array.copy()
                new_array.append(matrix[i][j])
                new_steps.append([i,j])
                currentArray.append(new_array)
                currentSteps.append(new_steps)
            else:
                continue
```

```
return currentArray, currentSteps
```

→ `vertical_move(arrayBefore, stepsBefore, matrix)`: Untuk memulai meneruskan rangkaian array kemungkinan solusi untuk saat pergerakan vertikal

```
def vertical_move(arrayBefore, stepsBefore, matrix):  
    currentArray = []  
    currentSteps = []  
  
    for p in range(len(stepsBefore)):  
        j = stepsBefore[p][len(stepsBefore[p])-1][1]  
        for i in range(len(matrix)):  
            old_array = arrayBefore[p]  
            old_steps = stepsBefore[p]  
            if [i,j] not in old_steps:  
                new_steps = old_steps.copy()  
                new_array = old_array.copy()  
                new_array.append(matrix[i][j])  
                new_steps.append([i,j])  
                currentArray.append(new_array)  
                currentSteps.append(new_steps)  
            else:  
                continue  
  
    return currentArray, currentSteps
```

→ `is_sequence_in_array(sequence, array)`: Untuk mengecek apakah terdapat sebuah sekuens dalam array kemungkinan tersebut

```
def is_sequence_in_array(sequence, array):  
    lengthSequence = len(sequence)  
    lengthArray = len(array)  
  
    if lengthSequence > lengthArray:  
        return False  
  
    for i in range(lengthArray - lengthSequence + 1):  
        if array[i:i+lengthSequence] == sequence:  
            return True  
  
    return False
```

→ process(bufferSize, matrix, sequences, rewards): Untuk menghitung bobot hadiah maksimal apabila terdapat sebuah sekuens dalam array kemungkinan tersebut dan menyimpan token-tokennya, langkah-langkahnya, dan bobotnya

```
def process(bufferSize, matrix, sequences, rewards):
    saved_array = []
    saved_steps = []
    saved_reward = 0
    max_rewards = sum(rewards)

    for start in range(len(matrix[0])):
        arrayNow, stepsNow = startMove0N(matrix, start)
        for i in range(1, bufferSize-1):
            if i % 2 == 0:
                arrayNow, stepsNow =
vertical_move(arrayNow, stepsNow, matrix)
            else:
                arrayNow, stepsNow =
horizontal_move(arrayNow, stepsNow, matrix)

        for chance in range(len(arrayNow)):
            reward = 0
            for seq in range(len(sequences)):
                init =
is_sequence_in_array(sequences[seq],
arrayNow[chance])
                if init:
                    reward += rewards[seq]
            if reward > saved_reward:
                saved_reward = reward
                saved_array = arrayNow[chance]
                saved_steps = stepsNow[chance]
            if reward == max_rewards:
                saved_reward = reward
                saved_array = arrayNow[chance]
                saved_steps = stepsNow[chance]
                break

    return saved_reward, saved_array, saved_steps
```

### 3. Fungsi Utama dalam *Output* Solusi

→ `output_txt(reward, tokens, steps, time, name)`: Untuk menyimpan solusi pada sebuah file berekstensi “.txt”

```
def output_txt(reward, tokens, steps, time, name):
    folder = "test"
    file = os.path.join(folder, name)

    with open(file, 'w') as file:
        file.write(str(reward) + "\n")
        sequence = " ".join(tokens)
        file.write(sequence)
        file.write("\n")
        print_steps(steps, file=file)
        file.write(f"\n{str(int(time*1000))} ms")

    print(f"Solusi telah di simpan pada '{name}.txt'.")
    print("\n|---- SEE YOU SOON!!")
    print("----|")
    return
```

→ `output_cli(reward, tokens, steps, time)`: Untuk memberikan solusi pada CLI

```
def output_cli(reward, tokens, steps, time):
    if reward == 0:
        print("Tidak ada solusi yang memenuhi.")
    else:
        print(reward)
        sequence = " ".join(tokens)
        print(sequence)
        print_steps(steps)
        print("")
        print(f"{int(time*1000)} ms")
    return
```

### 4. Fungsi Tambahan

→ `random_sequences(tokens, maxSeq)`: Untuk membuat sekuens secara acak pada masukan CLI

```
def random_sequences(tokens, maxSeq):
    length = random.randint(2, maxSeq)
```

```
return random.choices(tokens, k=length)
```

→ `random_reward()`: Untuk menentukan besar bobot secara acak untuk tiap sekuen

```
def random_reward():  
    num = random.randint(1, 100)  
    return num
```

→ `print_matrix(matrix)`: Untuk menampilkan matriks

```
def print_matrix(matrix):  
    for row in matrix:  
        for elem in row:  
            print(elem, end=" ")  
        print()  
    return
```

→ `print_sequences()`: Untuk mengetahui sekuen

```
def print_sequences(sequences, rewards):  
    for i in range(1, len(sequences)+1):  
        print(f"{i}. ")  
        for j in i:  
            print(j, end=" ")  
        print(rewards[i])  
    return
```

→ `print_data()`: Untuk menampilkan informasi/data yang didapatkandari file berekstensi “.txt” atau CLI

```
def print_data(buffer_size, matrix, sequences, rewards):  
    print("\n----- DATA YANG DIDAPATKAN -----")  
    print("- Ukuran buffer :", buffer_size)  
    print("- Ukuran matriks:", len(matrix), "x",  
          len(matrix[0]))  
    print("- Matriks      :")  
    print_matrix(matrix)  
    print("- Jumlah sekuens:", len(sequences))  
    print("- Daftar sekuens:")  
    for i, (sequence, reward) in enumerate(zip(sequences,  
                                              rewards), start=1):  
        print(f"  {i}. {' '.join(sequence)} memiliki bobot
```

```
    sebesar {reward}.")  
    return
```

→ print\_steps(): Untuk membaca dari file berekstensi “.txt”

```
def print_steps(arraySteps, file=None):  
    for i in range(len(arraySteps)):  
        r = arraySteps[i][0] + 1  
        c = arraySteps[i][1] + 1  
        if file:  
            print("{}, {}".format(r, c), file=file)  
        else:  
            print("{}, {}".format(r, c))  
    return
```



## Tangkapan Layar *Input* dan *Output*

### 1. Test case 1

→ Membaca file 'testcase1.txt' dengan menyimpan jawaban

Input (txt)	Output
<pre> ≡ testcase1.txt 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 </pre> <p>File solusi: solusi1.txt</p> <pre> ≡ solusi1 50 7A BD 7A BD 1C BD 55 1, 1 4, 1 4, 3 5, 3 5, 6 3, 6 3, 1  360 ms </pre>	<pre>  -----   ----- SELAMAT DATANG -----   ----- DI CYPERBUNK 2077 BREACH -----   ----- PROTOCOL SOLVER -----   -----   Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 1 Masukan nama file dengan format '.txt': testcase  Error: File dengan nama 'testcase' tidak ditemukan! Apakah ingin mencoba lagi? (Y/N) y  Masukan nama file dengan format '.txt': testcase1  Error: File dengan nama 'testcase1' tidak ditemukan! Apakah ingin mencoba lagi? (Y/N) y  Masukan nama file dengan format '.txt': testcase1.txt  Apakah ingin melihat data yang dimuat? (Y/N) y  ----- DATA YANG DIDAPATKAN ----- - Ukuran buffer : 7 - Ukuran matriks: 6 x 6 - Matriks      : 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 - Jumlah sekuens: 3 - Daftar sekuens:   1. BD E9 1C memiliki bobot sebesar 15.   2. BD 7A BD memiliki bobot sebesar 20.   3. BD 1C BD 55 memiliki bobot sebesar 30.  Solusi dari data yang telah didapatkan adalah: 50 7A BD 7A BD 1C BD 55 1, 1 4, 1 4, 3 5, 3 5, 6 3, 6 3, 1  360 ms  Apakah ingin menyimpan solusi? (Y/N) y  Masukkan nama untuk file solusi yang hendak disimpan: solusi1 Solusi telah di simpan pada 'solusi1.txt'.   ----- SEE YOU SOON!! -----  </pre>

## 2. Test case 2

→ Membaca file 'testcase2.txt' dengan tidak menyimpan jawaban

Input (txt)	Output
<pre> ≡ testcase2.txt 8 8 8 BD E9 BD 7A 1C 55 E9 BD 1C E9 7A 7A 55 7A 7A E9 E9 E9 1C E9 55 7A 55 1C E9 BD 55 E9 BD 55 BD 55 BD E9 1C 1C 1C E9 1C E9 BD 1C 55 55 BD 55 55 BD E9 1C BD 1C 1C E9 7A 1C 1C BD 7A BD 1C BD E9 BD 10 BD 1C -84 55 BD 77 E9 55 -37 7A 1C -72 55 7A 59 BD 7A 52 BD 55 6 E9 E9 96 7A 1C 7A -50 55 55 -88 </pre>	<pre> -----            SELAMAT DATANG                 DI CYPERBUNK 2077 BREACH                 PROTOCOL SOLVER           -----  Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 1 Masukan nama file dengan format '.txt': testcase2.txt  Apakah ingin melihat data yang dimuat? (Y/N) n  Solusi dari data yang telah didapatkan adalah: 290 BD 7A 55 BD 55 7A E9 E9 1, 3 2, 3 2, 5 4, 5 4, 3 8, 3 8, 7 1, 7  63969 ms Apakah ingin menyimpan solusi? (Y/N) n   -----           SEE YOU SOON!!           -----  </pre>

### 3. Test case 3

→ Membaca file 'testcase3.txt'

Input (txt)	Output
<pre> ≡ testcase2.txt 8 6 5 6t 45 t7 t7 2f 45 45 6t 2f 6t t7 45 45 2f 2f 45 t7 as as 2f 6t 2f 45 as as 45 as 6t 2f 2f 4 as t7 2f 15 as 6t as 20 as 2f as 45 30 45 t7 as 25 </pre> <p>File solusi: solusi3.txt</p> <pre> ≡ solusi3 75 E9 1C E9 55 BD 7A 55 1, 1 3, 1 3, 6 4, 6 4, 3 3, 3 3, 5  546 ms </pre>	<pre>  -----   ----- SELAMAT DATANG -----   ----- DI CYPERBUNK 2077 BREACH -----   ----- PROTOCOL SOLVER -----   -----   Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 1 Masukan nama file dengan format '.txt': testcase3.txt  Apakah ingin melihat data yang dimuat? (Y/N) y  ----- DATA YANG DIDAPATKAN ----- - Ukuran buffer : 7 - Ukuran matriks: 6 x 6 - Matriks      : E9 1C 55 7A 7A 55 55 1C 1C E9 55 BD 1C 7A 7A BD 55 E9 7A 1C BD 1C E9 55 BD 1C 1C 55 55 7A 55 7A E9 BD 55 1C - Jumlah sekuens: 5 - Daftar sekuens:   1. BD E9 1C memiliki bobot sebesar 15.   2. BD 7A BD memiliki bobot sebesar 20.   3. BD 1C BD 55 memiliki bobot sebesar 30.   4. BD 7A 55 memiliki bobot sebesar 35.   5. 1C E9 55 memiliki bobot sebesar 40.  Solusi dari data yang telah didapatkan adalah: 75 E9 1C E9 55 BD 7A 55 1, 1 3, 1 3, 6 4, 6 4, 3 3, 3 3, 5  Apakah ingin menyimpan solusi? (Y/N) y  Masukkan nama untuk file solusi yang hendak disimpan: solusi3 Solusi telah di simpan pada 'solusi3.txt'.   ----- SEE YOU SOON!! -----  </pre>

#### 4. Test case 4

→ Membaca file 'testcase4.txt'

Input (txt)	Output
<pre> ≡ testcase4.txt 4 2 2 76 76 AB 76 4 76 ab 32 AB 76 27 ab ab 49 AB AB 49 </pre>	<pre>  -----   ----- SELAMAT DATANG -----   ----- DI CYPERBUNK 2077 BREACH -----   ----- PROTOCOL SOLVER -----   -----   Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 1 Masukan nama file dengan format '.txt': testcase4.txt  Apakah ingin melihat data yang dimuat? (Y/N) y  ----- DATA YANG DIDAPATKAN ----- - Ukuran buffer : 4 - Ukuran matriks: 2 x 2 - Matriks      : 76 76 AB 76 - Jumlah sekuens: 4 - Daftar sekuens:   1. 76 ab memiliki bobot sebesar 32.   2. AB 76 memiliki bobot sebesar 27.   3. ab ab memiliki bobot sebesar 49.   4. AB AB memiliki bobot sebesar 49.  Solusi dari data yang telah didapatkan adalah: 27 76 AB 76 76 1, 1 2, 1 2, 2 1, 2  0 ms  Apakah ingin menyimpan solusi? (Y/N) n   ----- SEE YOU SOON!! -----  </pre>

## 5. Test case 5

→ Membaca dari CLI

Input (CLI)	Output
<pre> ----- ---- SELAMAT DATANG ---- ---- DI CYPARBUNK 2077 BREACH ---- ---- PROTOCOL SOLVER ---- -----  Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 2 Jumlah token unik yang diinginkan: 4 Token ke-1: 3e Token ke-2: 4f Token ke-3: 3E Token ke-4: gn Masukan ukuran buffer yang diinginkan: 6 Masukan ukuran matriks yang diinginkan (kolom baris): 6 5 Masukkan jumlah sekuens yang diinginkan: 4 Masukkan ukuran maksimal sekuens yang diinginkan: 4  ----- DATA YANG DIDAPATKAN ----- - Ukuran buffer : 6 - Ukuran matriks: 5 x 6 - Matriks      : 3E gn 4f 3e 4f gn gn 3e gn 3e 3e 3E 4f 3E gn 4f 4f gn 3e 3E 3e gn 3e gn 3E 3e gn 4f 3e 4f - Jumlah sekuens: 4 - Daftar sekuens: 1. 3E 3E memiliki bobot sebesar 49. 2. 3E 4f 3e memiliki bobot sebesar 6. 3. 4f gn memiliki bobot sebesar 19. 4. 3e 3e 3E memiliki bobot sebesar 8. </pre>	<pre> Solusi dari data yang telah didapatkan adalah: 76 4f gn 3e 3e 3E 3E 1, 3 2, 3 2, 2 5, 2 5, 1 1, 1  57 ms  Apakah ingin menyimpan solus i? (Y/N) y  Masukkan nama untuk file sol usi yang hendak disimpan: te stcasecli1 Solusi telah di simpan pada 'testcasecli1.txt'.   ----- SEE YOU SOON! !-----  </pre>

## 6. Test case 6

→ Membaca dari CLI

Input (CLI)	Output
<pre> -----  ----- SELAMAT DATANG -----   ----- DI CYPERBUNK 2077 BREACH -----   ----- PROTOCOL SOLVER -----   ----- Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: a  Masukan harus berupa angka antara 1 dan 2. Silakan coba lagi.  Metode yang dipilih: 3  Pilih antara 1 dan 2   1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 2 Jumlah token unik yang diinginkan: d Masukan harus berupa bilangan bulat. Silakan coba lagi. Jumlah token unik yang diinginkan: wq Masukan harus berupa bilangan bulat. Silakan coba lagi. Jumlah token unik yang diinginkan: 5 Token ke-1: av Token ke-2: av  Token sudah dimasukkan sebelumnya! Ulangi masukan! Token ke-2: avf  Masukan token dengan panjang 2 dan berisi alfanumerik! Token ke-2: 4r Token ke-3: fe Token ke-4: wq Token ke-5: 23  Masukan ukuran buffer yang diinginkan: 9 Masukan ukuran matriks yang diinginkan (kolom baris): 2 4 Masukan jumlah sekuens yang diinginkan: 6 Masukan ukuran maksimal sekuens yang diinginkan: 2  ----- DATA YANG DIDAPATKAN ----- - Ukuran buffer : 9 - Ukuran matriks: 4 x 2 - Matriks      : 4r 4r 23 av fe fe 23 23 - Jumlah sekuens: 6 - Daftar sekuens: 1. 4r av memiliki bobot sebesar 23. 2. fe fe memiliki bobot sebesar 7. 3. av wq memiliki bobot sebesar 41. 4. wq fe memiliki bobot sebesar 19. 5. 23 fe memiliki bobot sebesar 20. 6. av 23 memiliki bobot sebesar 35. </pre>	<pre> Solusi dari data yang telah didapatkan adalah: 85 4r av 23 fe fe 23 23 4r 1, 2 2, 2 2, 1 3, 1 3, 2 4, 2 4, 1 1, 1 0 ms  Apakah ingin menyimpan solusi? (Y/N) y  Masukkan nama untuk file solusi yang hendak disimpan: testcasecli3 Solusi telah di simpan pada 'testcasecli3.txt'.   ----- SEE YOU SOON!! -----  </pre>

## 7. Test case 7

→ Membaca dari CLI

Input (CLI)	Output
<pre>  -----   ----  SELAMAT DATANG  ----   ----  DI CYPERBUNK 2077 BREACH  ----   ----  PROTOCOL SOLVER  ----   -----   Silakan pilih metode masukan yang diinginkan    1   Metode TXT: dengan memasukkan file '.txt'   2   Metode CLI: dengan masukan dari command line  Metode yang dipilih: 2 Jumlah token unik yang diinginkan: 4 Token ke-1: vt Token ke-2: 56 Token ke-3: g5 Token ke-4: 3r Masukan ukuran buffer yang diinginkan: 8 Masukan ukuran matriks yang diinginkan (kolom baris): 8 9  Program akan berjalan sangat lama untuk untuk matriks lebih besar dari 8x8 (yang hasil kali baris dan kolomnya 64), coba masukan ukuran lain ! Masukan ukuran matriks yang diinginkan (kolom baris): 8 7 Masukan jumlah sekuens yang diinginkan: 6 Masukan ukuran maksimal sekuens yang diinginkan: 3  ----- DATA YANG DIDAPATKAN ----- - Ukuran buffer : 8 - Ukuran matriks: 7 x 8 - Matriks      : 56 vt vt 56 g5 vt 56 g5 vt 56 56 vt vt 3r g5 g5 g5 3r 56 vt 56 56 vt g5 g5 56 vt 56 56 vt 3r vt 56 vt 56 vt 56 vt g5 3r g5 g5 56 g5 56 3r vt vt vt 3r g5 vt g5 vt 3r 56 - Jumlah sekuens: 5 - Daftar sekuens: 1. 3r vt 3r memiliki bobot sebesar 50. 2. g5 vt memiliki bobot sebesar 49. 3. vt 56 56 memiliki bobot sebesar 44. 4. 56 56 g5 memiliki bobot sebesar 15. 5. vt vt memiliki bobot sebesar 20. </pre>	<pre> Solusi dari data yang telah didapatkan adalah: 163 g5 vt vt 56 56 3r vt 3r 1, 5 2, 5 2, 1 1, 1 1, 7 4, 7 4, 6 2, 6  24540 ms  Apakah ingin menyimpan solusi? (Y/N) y  Masukkan nama untuk file solusi yang hendak disimpan: solusi4 Solusi telah di simpan pada 'solusi4.txt'.   ----          SEE YOU SOON!!          ----  </pre>

## Lainnya

### A. Repository

→ <https://github.com/hiirr/Cyberpunk-Minigame-Solver.git>

### B. Checklist

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		✓



