

LAPORAN TUGAS KECIL 1 IF2211
STRATEGI ALGORITMA

*Penyelesaian Cyberpunk 2077 Breach Protocol
dengan Algoritma Brute Force*



Dosen Pengampu: Dr. Nur Ulfa Maulidevi, S.T, M.Sc

Disusun oleh:

Panji Sri Kuncara Wisma

(13522028)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

DAFTAR ISI.....	1
PENGECEKAN PROGRAM.....	2
DESKRIPSI MASALAH.....	3
A. Algoritma Brute Force.....	4
B. Source Program Python.....	5
C. Input dan Output.....	12
D. Repository.....	14

PENGECEKAN 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		✓

Tabel 1. Tabel Pengecekan Program

DESKRIPSI MASALAH

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:

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.

Terdapat aturan permainan Breach Protocol pada minigame Cyberpunk 2077 ini, yakni sebagai berikut.

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.

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

A. Algoritma Brute Force

Program utama yang digunakan untuk menyelesaikan masalah Cyberpunk 2077 Breach Protocol pada tugas kecil kali ini berada di file *main.py*. Semua fungsi-fungsi berada di dalam file tersebut. Bahasa python dipilih karena alasan familiar dan kemudahan membaca dan menulis file. Secara umum, ide dan langkah-langkah algoritma brute force yang saya gunakan untuk menyelesaikan persoalan ini adalah sebagai berikut:

1. Memilih salah satu blok atau elemen pada baris pertama matriks untuk memulai pergerakan. Dalam program, saya memilih baris ke-0 kolom ke-0 terlebih dahulu.
2. Objektif berikutnya adalah dengan mencari segala kemungkinan kombinasi token yang mungkin dengan ukuran buffer tertentu. Untuk ukuran buffer x , akan dicari segala kombinasi yang mungkin sesuai aturan yang sudah ditetapkan.
3. Idenya adalah mengunjungi setiap titik pada matriks secara selang-seling (vertikal, horizontal, vertikal,...) dengan menggunakan fungsi rekursif. Proses rekursif akan berhenti ketika sudah menyentuh basis. Pada masalah ini basis yang saya tetapkan adalah ketika buffer sudah penuh.
4. Untuk mengetahui gerakan yang mungkin, saya membuat sebuah fungsi *find_move(arah, titik, matriks, validasi)*. Arah digunakan untuk mengetahui pergerakan yang diperbolehkan. Titik digunakan untuk menandai lokasi saat ini. Validasi digunakan untuk menandai titik-titik yang sudah dikunjungi.
5. Penjelasan singkat untuk contoh kasus matriks 3×3 yang dimulai dari titik $(0,0)$ dan buffer berukuran 3. Mula-mula, sebuah array kosong diisapkan untuk menampung segala kemungkinan pergerakan. Arah mula-mula adalah vertikal. Titik-titik yang mungkin dikunjungi adalah $(1,0)$ dan $(2,0)$. Arah berikutnya adalah horizontal. Titik titik yang mungkin adalah $(1,1)$, $(1,2)$, $(2,1)$, dan $(2,2)$. Karena buffer berukuran 3, maka penelusuran hanya sampai sini. Koordinat titik-titik yang dilalui termasuk elemen matriks pada koordinat tersebut dimasukkan ke dalam suatu array.

6. Proses pada nomor 5 terjadi pada fungsi *bruteforce_combination*. Proses berikutnya sama seperti proses 1 - 5. Perbedaanya terletak pada titik awal dipilih. Hal tersebut terus dilanjutkan hingga semua titik pada baris pertama sudah dipilih sebagai titik awal.
7. Array yang berisi array kombinasi kemungkinan langkah untuk buffer berukuran x sudah didapatkan melalui proses 1 - 6. Langkah berikutnya adalah melakukan iterasi dari 1 sampai $x + 1$ untuk mencari segala kemungkinan untuk setiap buffer. Hal itu dilakukan karena mungkin saja ada hasil yang optimal dengan langkah yang sedikit. Ada kemungkinan Buffer terisi setengah penuh lebih optimal daripada buffer yang terisi penuh.
8. Solusi optimal didapat dengan mengecek untuk setiap kombinasi pada setiap buffer, apakah sekuens-sekuens yang tersedia merupakan subarray dari buffer tersebut. Jika iya, maka dihitung jumlah *reward* yang didapatkan dan dimasukkan ke dalam variabel. Jika tidak, maka kombinasi yang dimaksud akan dilewati dan dilanjutkan mengecek kemungkinan yang lain. Iterasi terus dilakukan hingga mendapat hasil yang optimal beserta jumlah poin dan koordinat pergerakannya.
9. Langkah terakhir adalah menuliskan jawabannya yang terdiri dari jumlah poin, kombinasi buffer, dan koordinat proses pergerakan.

Beberapa hal pada program ini tidak memiliki penanganan atau *handling*. Dengan asumsi bahwa input dan format selalu benar, kemungkinan besar program juga akan berjalan dengan benar. Beberapa *output* diberi penjelasan tambahan agar terlihat lebih jelas. Penjelasan mengenai fungsi-fungsi yang ada akan dijelaskan pada poin B.

B. Source Program Python

1. **find_move** digunakan untuk mencari segala pergerakan kemungkinan yang ada pada arah tertentu. Fungsi ini mengembalikan array of tuple

```

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

"""
PANJI SRI KUNCARA WISMA - 13522028
"""

import os
import random
import time

#Fungsi ini dipake buat nentuin arah gerak
def find_move(arah, titik, matriks, validasi):
    baris = len(matriks)
    kolom = len(matriks[0])
    move = []
    if arah == 'V':
        for i in range(baris):
            if validasi[i][titik[1]] == False and (i, titik[1]) != titik:
                move.append((i, titik[1]))
    elif arah == 'H':
        for j in range(kolom):
            #print("ini j", j)
            if validasi[titik[0]][j] == False and (titik[0], j) != titik:
                move.append((titik[0], j))
    return move

```

Gambar 1. Code 1

2. **brute_force_combination** digunakan untuk mencari segala kombinasi buffer yang memungkinkan sesuai aturan yang ada jika dimulai pada titik awal tertentu.

```

#Fungsi ini dipake buat nentuin arah gerak
def brute_force_combination(titik, buffer, arah, validasi, matriks, list_kombinasi_global, arr_kombinasi, koordinat, arr_koordinat):
    baris = len(matriks)
    kolom = len(matriks[0])

    if buffer == 1:
        arr_kombinasi.append(matriks[titik[0]][titik[1]])
        arr_koordinat.append((titik[0], titik[1]))

        validasi[titik[0]][titik[1]] = True

        #print(validasi)

        #validasi = [[False for k in range(kolom)] for j in range(baris)]

        list_kombinasi_global.append(arr_kombinasi.copy())
        koordinat.append(arr_koordinat.copy())

        arr_kombinasi.pop()
        arr_koordinat.pop()
    else:
        arr_kombinasi.append(matriks[titik[0]][titik[1]])
        arr_koordinat.append((titik[0], titik[1]))

        validasi[titik[0]][titik[1]] = True

        list_move = find_move(arah, titik, matriks, validasi)

        #print(validasi)

        if arah == 'V':
            arah_berikutnya = 'H'
        elif arah == 'H':
            arah_berikutnya = 'V'

        for titik_lanjutan in list_move:
            brute_force_combination(titik_lanjutan, buffer-1, arah_berikutnya, validasi, matriks, list_kombinasi_global, arr_kombinasi, koordinat, arr_koordinat)
            # validasi = [[False for k in range(kolom)] for j in range(baris)]
            validasi[titik_lanjutan[0]][titik_lanjutan[1]] = False

        arr_kombinasi.pop()
        arr_koordinat.pop()

```

Gambar 2. Code 2

3. **origin_bruteforce_combination** digunakan untuk melakukan iterasi pada titik-titik awal dan mencatat kombinasi untuk setiap titik tersebut. **is_subarray** digunakan untuk mengecek apakah suatu array merupakan subarray dari array lain. **count_subarray** digunakan untuk mengetahui seberapa banyak suatu array muncul di array lain.

```
#Fungsi gerbang buat masuk fungsi rekursif bruteforce_combination
def origin_bruteforce_combination(matriks, buffer):

    baris = len(matriks)
    kolom = len(matriks[0])
    list_kombinasi_global = []
    koordinat = []

    for i in range(kolom):
        validasi = [[False for k in range(kolom)] for j in range(baris)]
        bruteforce_combination((0,i), buffer, 'V', validasi, matriks, list_kombinasi_global, [], koordinat, [])

    return list_kombinasi_global, koordinat

#Fungsi ini dipake buat ngecek apakah suatu array itu ada di array yang lain
def is_subarray(subarray, array):
    len_subarray = len(subarray)
    len_array = len(array)

    for i in range(len_array - len_subarray + 1):
        if array[i:i + len_subarray] == subarray:
            return True

    return False

#Fungsi ini dipake buat ngecek ada berapa banyak suatu array yang ada di array yang lain
def count_subarrays(subarray, array):
    len_subarray = len(subarray)
    len_array = len(array)
    count = 0

    for i in range(len_array - len_subarray + 1):
        if array[i:i + len_subarray] == subarray:
            count += 1

    return count
```

Gambar 3. Code 3

4. **find_best_combines** digunakan untuk melakukan iterasi untuk setiap buffer, mencatat semuanya, dan mencari solusi kombinasi yang paling optimal termasuk

jumlah nilai yang dihasilkan dan koordinatnya.

```
def find_best_combines(matrics, buffer, matrics_sekuens):  
    jumlah_pembanding = 0  
    matrics_sekuens_terpilih = []  
    list_kombinasi_global = []  
    list_koordinat = []  
    for i in range(1, buffer+1):  
        list_kombinasi_global_sementara, list_koordinat_sementara = origin_bruteforce_combination(matrics, i)  
        list_kombinasi_global.extend(list_kombinasi_global_sementara.copy())  
        list_koordinat.extend(list_koordinat_sementara.copy())  
        list_kombinasi_global_sementara = []  
        list_koordinat_sementara = []  
    # for i in list_kombinasi_global:  
    #     if(i[0] == 'A'):  
    #         print(i)  
    for kombinasi in list_kombinasi_global:  
        jumlah_sementara = 0  
        for sekuens in matrics_sekuens:  
            print(sekuens)  
            if is_subarray(sekuens[0], kombinasi):  
                count = count_subarrays(sekuens[0], kombinasi)  
                temp = count * sekuens[1]  
                jumlah_sementara = jumlah_sementara + temp  
        if jumlah_sementara > jumlah_pembanding:  
            # print(kombinasi, jumlah_sementara)  
            jumlah_pembanding = jumlah_sementara  
            matrics_sekuens_terpilih = kombinasi  
    if matrics_sekuens_terpilih != []:  
        index = list_kombinasi_global.index(matrics_sekuens_terpilih)  
        langkah_koordinat = list_koordinat[index]  
    else:  
        langkah_koordinat = []  
    return matrics_sekuens_terpilih, jumlah_pembanding, langkah_koordinat
```

Gambar 4. Code 4

5. **read_input** digunakan untuk melakukan pembacaan dari teks.
hasilkan_sekuens digunakan untuk menghasilkan sekuens secara unik, acak dan otomatis.

```

main.py > read_input
def read_input(file_path):
    file_path = os.path.join '..', file_path

    with open(file_path, 'r') as file:
        #ukuran buffer
        ukuran_buffer = int(file.readline().strip())

        # kolom
        kolom, baris = map(int, file.readline().strip().split())

        #matrix
        matrix = [list(file.readline().strip().split()) for _ in range(baris)]

        #banyak sekuens
        banyak_sekuens = int(file.readline().strip())

        arr_sekuens = []

        for _ in range(banyak_sekuens):
            sekuens = file.readline().strip().split()
            reward = int(file.readline().strip())
            arr_sekuens.append((sekuens, reward))

    return ukuran_buffer, kolom, baris, matrix, banyak_sekuens, arr_sekuens

#Generate Sekuens
def hasilkan_sekuens(tokens, jumlah_sekuens, ukuran_maksimal_sekuens):
    sekuens_unik = []

    def hasilkan_kombinasi(sekuens, sisa_ukuran, panjang_saat_ini):
        if sisa_ukuran == 0:
            nilai_acak = random.randint(-100, 100)
            sekuens_unik.append((sekuens[:], nilai_acak))
            return
        if panjang_saat_ini >= ukuran_maksimal_sekuens:
            return
        for token in tokens:
            sekuens_baru = sekuens + [token]
            hasilkan_kombinasi(sekuens_baru, sisa_ukuran - 1, panjang_saat_ini + 1)

    for panjang in range(2, ukuran_maksimal_sekuens + 1):
        hasilkan_kombinasi([], panjang, 0)

    sekuens_terpilih = random.sample(sekuens_unik, jumlah_sekuens)

    return sekuens_terpilih

```

Gambar 5. Code 5

6. **hasilkan_matriks** digunakan untuk menghasilkan matriks secara otomatis.
masukkan_CLI digunakan untuk penanganan masukan secara CLI.

```

#Generate Matriks
def hasilkan_matriks(tokens, jumlah_baris, jumlah_kolom):
    matriks = [[' ' for _ in range(jumlah_kolom)] for _ in range(jumlah_baris)]
    random.shuffle(tokens)
    indeks_token = 0

    for baris in range(jumlah_baris):
        for kolom in range(jumlah_kolom):
            matriks[baris][kolom] = tokens[indeks_token]
            indeks_token = (indeks_token + 1) % len(tokens)
            random.shuffle(tokens)

    return matriks

#Masukkan via CLI
def masukkan_CLI():
    jumlah_token = int(input("Masukkan banyak token: "))

    while True:
        token = input("Masukkan token (tolong pisahkan dengan spasi ya): ").split()
        if jumlah_token == len(token):
            break
        else:
            print("Jumlah token tidak sesuai")

    buffer = int(input("Masukkan ukuran buffer: "))

    kolom, baris = map(int, input("Masukkan ukuran matriks (format : kolom baris, contoh: 6 6): ").split())

    while True:
        jumlah_sekuens = int(input("Masukkan jumlah sekuens: "))
        ukuran_maksimal_sekuens = int(input("Masukkan ukuran maksimal sekuens: "))

        if jumlah_sekuens <= jumlah_token ** ukuran_maksimal_sekuens:
            break
        else:
            print("Jumlah Sekuens yang diminta melebihi jumlah maksimal sekuens yang dapat dibentuk. Silakan masukkan ulang.")

    sekuens = hasilkan_sekuens(tokens, jumlah_sekuens, ukuran_maksimal_sekuens)

    matrix = hasilkan_matriks(token, baris, kolom)
    return matrix, buffer, sekuens

```

Gambar 6. Code 6

7. **write_to_file** digunakan untuk menulis ke dalam file. **main** adalah program yang menyatukan semuanya

```

#Mode buat tulis file
def write_to_file(filename, content):
    with open(filename, 'w') as file:
        file.write(content)

def main():
    print("Tucil 1 STIMA")
    while True:
        print("Pilih salah satu")
        print("1. Masukkan Via CLI")
        print("2. Masukkan Via Input Text")
        masukkan = input("Tuliskan Angka Pilihan (1 atau 2): ")

        if masukkan == '1' or masukkan == '2':
            break
        else:
            print("Tolong inputkan dengan benar")

    if (masukkan == '1'):
        matriks, buffer, sekuens = masukkan_CLI()

        start_time = time.time()
        matriks_sekuens_terpilih, jumlah_pembanding, langkah_koordinat = find_best_combines(matriks, buffer, sekuens)
        end_time = time.time()

        print("-----Ini Matriksnya-----")
        for i in range(len(matriks)):
            for j in range(len(matriks[0])):
                print(f"{matriks[i][j]}", end=" ")
            print()

        print("-----Ini sekuensnya-----")
        for sekuens in sekuens:
            for j in sekuens[0]:
                print(j, end=" ")
            print()
            print("bobot: ", sekuens[1])
        print("-----Ini buffer-----")
        print("Ukuran Buffer= ", buffer)
        print("-----Solusi-----")
        print(jumlah_pembanding)

        for i in matriks_sekuens_terpilih:
            print(i, end=" ")
            print()

```

Gambar 7. Code 7

```

        for i in matriks_sekuens_terpilih:
            print(i, end=" ")
            print()

        result = [(y+1, x+1) for x, y in langkah_koordinat]

        for point in result:
            print(f"{point[0]}, {point[1]}")

        elapsed_time = end_time - start_time
        print()
        print(f"{elapsed_time * 1000:.2f} ms")
        print()

        save_solution = input("Apakah ingin menyimpan solusi? (y/n): ").lower()

        if save_solution == 'y':
            filename = input("Masukkan nama file (termasuk ekstensi .txt): ")

            content = f"{jumlah_pembanding}\n"
            content += ' '.join(map(str, matriks_sekuens_terpilih)) + '\n'
            content += '\n'.join([f"{x[0]}, {x[1]}" for x in result]) + '\n'
            content += "\n"
            content += f"{elapsed_time * 1000:.2f} ms\n"

            file_path = os.path.join('..', 'test', filename)
            write_to_file(file_path, content)
            print(f"Hasil telah ditulis ke dalam file {filename}")
        else:
            print("Solusi tidak disimpan.")

    else:
        print("\n----Pastikan file sudah ada dan berada sejajar di path yang sama dengan folder src, doc, bin, dan test----\n")

        file_path = input("Masukkan nama file dan ekstensinya (contoh: input.txt): ")

        buffer, kolom, baris, matriks, banyak_sekuens, sekuens = read_input(file_path)

        start_time = time.time()
        matriks_sekuens_terpilih, jumlah_pembanding, langkah_koordinat = find_best_combines(matriks, buffer, sekuens)
        end_time = time.time()

        print(jumlah_pembanding)

```

Gambar 8. Code 8

```

else:
    print("\n-----Pastikan file sudah ada dan berada sejajar di path yang sama dengan folder src, doc, bin, dan test-----\n")

    file_path = input("Masukkan nama file dan ekstensinya (contoh: input.txt): ")

    buffer, kolom, baris, matriks, banyak_sekuens, sekuens = read_input(file_path)

    start_time = time.time()
    matriks_sekuens_terpilih, jumlah_pembanding, langkah_koordinat = find_best_combines(matriks, buffer, sekuens)
    end_time = time.time()

    print(jumlah_pembanding)

    for i in matriks_sekuens_terpilih:
        print(i, end=" ")
    print()

    result = [(y+1, x+1) for x, y in langkah_koordinat]

    for point in result:
        print(f"{point[0]}, {point[1]}")

    elapsed_time = end_time - start_time
    print()
    print(f"{elapsed_time * 1000:.2f} ms")
    print()

    save_solution = input("Apakah ingin menyimpan solusi? (y/n): ").lower()

    if save_solution == 'y':
        filename = input("Masukkan nama file (termasuk ekstensi .txt): ")

        content = f"{jumlah_pembanding}\n"
        content += ' '.join(map(str, matriks_sekuens_terpilih)) + '\n'
        content += '\n'.join([f"{x[0]}, {x[1]}" for x in result]) + '\n'
        content += "\n"
        content += f"{elapsed_time * 1000:.2f} ms\n"

        file_path = os.path.join('..', 'test', filename)
        write_to_file(file_path, content)
        print(f"Hasil telah ditulis ke dalam file {filename}")
    else:
        print("Solusi tidak disimpan.")

```

Gambar 9. Code 9

```

if __name__ == "__main__":
    main()

```

Gambar 10. Code 10

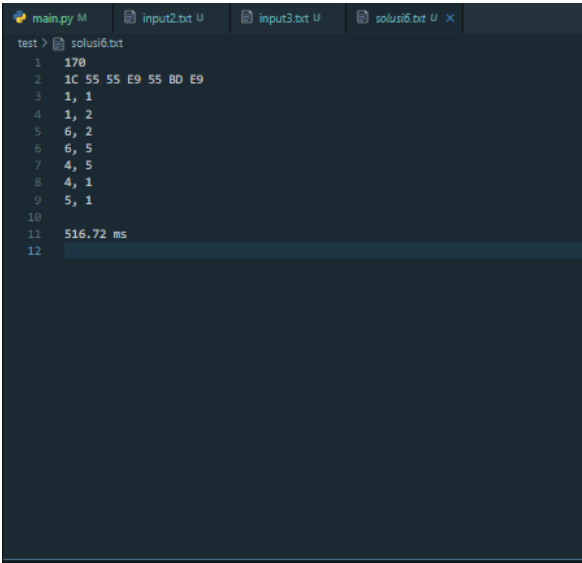
C. Input dan Output

Keterangan tambahan: matrix_width dianggap sebagai banyak kolom, jadi 6 7 adalah 6 kolom dan 7 baris.

No.	Input	Output
-----	-------	--------

<p>1.</p>	<p>input1.txt</p> <pre> main.py M input.txt X input.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 </pre>	<pre> PS D:\VA Berpakaian ITB\Meta Kuliah\Semester 4\STDMA\Tucil1_13522028\src> python main.py Tucil 1 STDMA Pilih salah satu 1. Masukkan Via CLI 2. Masukkan Via Input Text Tuliskan Angka Pilihan (1 atau 2): 2 -----Pastikan file sudah ada dan berada sejajar di path yang sama dengan folder src, doc, bin, dan test----- Masukkan nama file dan ekstensinya (contoh: input.txt): input1.txt 50 7A BD 7A BD 1C BD 55 1, 1 1, 4 3, 4 3, 5 6, 5 6, 3 1, 3 483.93 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (termasuk ekstensi .txt): solusi1.txt Hasil telah ditulis ke dalam file solusi1.txt </pre> <p>solusi1.txt</p> <pre> test > solusi1.txt 1 50 2 7A BD 7A BD 1C BD 55 3 1, 1 4 1, 4 5 3, 4 6 3, 5 7 6, 5 8 6, 3 9 1, 3 10 11 483.93 ms 12 </pre>
<p>2.</p>	<p>input2.txt</p> <pre> input2.txt 1 8 2 7 6 3 BD 55 E9 E9 1C 55 7A 4 7A 7A 1C 7A E9 55 7A 5 E9 1C 1C 55 E9 BD 1C 6 55 1C 7A 1C 55 BD 1C 7 1C 55 BD 7A 1C 1C 1C 8 1C 55 55 7A 55 7A 7A 9 4 10 BD E9 1C 11 15 12 BD 7A BD 13 20 14 BD 1C BD 55 15 30 16 BD E9 BD 17 40 18 </pre>	<pre> Tucil 1 STDMA Pilih salah satu 1. Masukkan Via CLI 2. Masukkan Via Input Text Tuliskan Angka Pilihan (1 atau 2): 2 -----Pastikan file sudah ada dan berada sejajar di path yang sama dengan folder src, doc, bin, dan test----- Masukkan nama file dan ekstensinya (contoh: input.txt): input2.txt 80 55 BD E9 BD E9 BD 6, 1 6, 3 1, 3 1, 1 3, 1 3, 5 6535.07 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (termasuk ekstensi .txt): solusi2.txt Hasil telah ditulis ke dalam file solusi2.txt </pre> <p>solusi2.txt</p> <pre> main.py M input2.txt U solusi2.txt U X test > solusi2.txt 1 80 2 55 BD E9 BD E9 BD 3 6, 1 4 6, 3 5 1, 3 6 1, 1 7 3, 1 8 3, 5 9 10 6535.07 ms 11 </pre>

<p>3.</p>	<p>input3.txt</p> <pre> 1 5 2 6 6 3 AA 55 E9 E9 1C 55 4 55 AA 1C 7A E9 55 5 55 1C AA 55 E9 5D 6 5D 1C 7A AB 55 5D 7 5D 55 5D 7A AB 1C 8 1C 55 55 7A 55 AB 9 5 10 5D E9 1C 11 55 12 5D 7A 5D 13 20 14 5D 1C 5D 55 15 30 16 AA 55 AA 17 40 18 AB 5D AB 19 50 </pre>	<pre> Tucil 1 STIMA Pilih salah satu 1. Masukkan Via CLI 2. Masukkan Via Input Text Tuliskan Angka Pilihan (1 atau 2): 2 -----Pastikan file sudah ada dan berada sejajar di path yang sama dengan folder src, doc, bin, dan test----- Masukkan nama file dan ekstensinya (contoh: input.txt): input3.txt 50 E9 AB 5D AB 4, 1 4, 4 6, 4 6, 6 36.03 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (termasuk ekstensi .txt): solusi3.txt hasil telah ditulis ke dalam file solusi3.txt </pre> <p>solusi3.txt</p> <pre> test > solusi3.txt 1 50 2 E9 AB 5D AB 3 4, 1 4 4, 4 5 6, 4 6 6, 6 7 8 36.03 ms 9 </pre>
<p>4.</p>	<p>CLI</p> <pre> Tucil 1 STIMA Pilih salah satu 1. Masukkan Via CLI 2. Masukkan Via Input Text Tuliskan Angka Pilihan (1 atau 2): 1 Masukkan Angka Pilihan: 5 Masukkan ukuran buffer: 4 Masukkan ukuran matriks (format : kolom baris, contoh: 6 4): 6 4 Masukkan jumlah sekuen: 3 Masukkan urutan matriks sekuen: 1 -----Inj Metriknya----- 50 AA 50 50 50 50 AA 50 -----Inj sekuennya----- 50 AA 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 -----Inj buffer----- ukuran buffer: 4 -----Solusi----- 4 5.23 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (termasuk ekstensi .txt): solusi4.txt hasil telah ditulis ke dalam file solusi4.txt </pre> <p>solusi4.txt</p> <pre> test > solusi4.txt 1 0 2 3 4 5 5.23 ms 6 </pre>	<p>solusi4.txt</p> <pre> test > solusi4.txt 1 0 2 3 4 5 5.23 ms 6 </pre>
<p>5.</p>	<pre> Tucil 1 STIMA Pilih salah satu 1. Masukkan Via CLI 2. Masukkan Via Input Text Tuliskan Angka Pilihan (1 atau 2): 1 Masukkan Angka Pilihan: 5 Masukkan ukuran buffer: 5 Masukkan ukuran matriks (format : kolom baris, contoh: 6 4): 7 7 Masukkan jumlah sekuen: 4 Masukkan urutan matriks sekuen: 4 -----Inj Metriknya----- 50 50 AA 7F 55 5D 7F 5D 50 55 AA 7F 5D 50 7F 55 5D AA 50 7F 7F 55 5D 50 AA 7F 55 50 AA 5D 7F 55 AA 50 AA 55 7F 5D 50 AA 5D AA 55 7F 5D AA -----Inj sekuennya----- 5D 5D AA 7F 50 50 50 50 7F 50 50 7F 55 AA 50 7F 55 AA 50 5D 7F AA 50 5D 7F AA -----Inj buffer----- ukuran buffer: 5 -----Solusi----- 100 5D 5D AA 7F 2, 1 2, 2 2, 2 5, 1 45.54 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (termasuk ekstensi .txt): solusi5.txt hasil telah ditulis ke dalam file solusi5.txt </pre> <p>Solusi5.txt</p> <pre> test > solusi5.txt 1 100 2 5D 5D AA 7F 3 2, 1 4 2, 2 5 5, 2 6 5, 1 7 8 45.54 ms 9 </pre>	<p>Solusi5.txt</p> <pre> test > solusi5.txt 1 100 2 5D 5D AA 7F 3 2, 1 4 2, 2 5 5, 2 6 5, 1 7 8 45.54 ms 9 </pre>

6.	<pre> Tucil 1 S7904 Pilih salah satu 1. Masukkan Via CLI 2. Masukkan Via Input Test Tuliskan Angka Pilihan (1 atau 2): 1 Masukkan banyak token: 5 Masukkan token (tolong pisahkan dengan spasi ya): BD 1C 7A 5S E9 Masukkan ukuran buffer: 7 Masukkan ukuran matriks (format : kolom baris, contoh: 6 6): 6 6 Masukkan jumlah sekuen: 3 Masukkan ukuran maksimal sekuen: 4 -----Ini Metriknya----- 1C 5S 7A BD E9 1C 5S 7A BD 1C E9 5S E9 5S 1C 7A BD E9 1C 7A BD 5S E9 1C E9 7A 1C 5S BD E9 E9 1C 5S 7A BD E9 -----Ini sekuenya----- 5S E9 5S BD bobot: 100 BD E9 bobot: 70 E9 BD 7A E9 bobot: 57 -----Ini buffer----- Ukuran Buffer: 7 -----Solusi----- 170 1C 5S 5S E9 5S BD E9 1, 1 1, 2 6, 2 6, 5 4, 5 4, 1 5, 1 516.72 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (termasuk ekstensi .txt): solusi6.txt Hasil telah ditulis ke dalam file solusi6.txt </pre>	<div>solusi6.txt</div> 
----	--	---

Tabel 2. Input dan Output Program

D. Repository

Link Repository dari Tugas Kecil 01 IF2211 Strategi Algoritma Panji Sri Kuncara Wisma adalah sebagai berikut.

https://github.com/PanjiSri/Tucil1_13522028.git