

LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

“Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force”



Disusun oleh:

Shulha K01 13522087

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022-2023

DAFTAR ISI

DAFTAR ISI	2
BAB 1	3
BAB 2	4
BAB 3	6
BAB 4	11
BAB 5	19
DAFTAR REFERENSI	20

BAB I

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 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

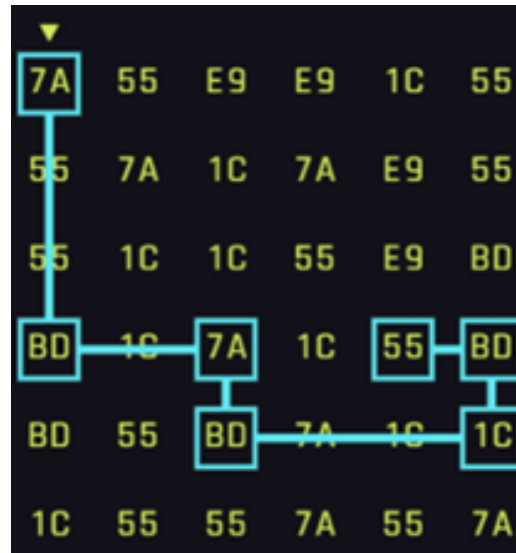
1C 55 55 7A 55 7A

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 1. Contoh Permainan Cyberpunk 2077

BAB II

LANGKAH-LANGKAH ALGORITMA BRUTE FORCE

Algoritma yang digunakan untuk menyelesaikan permasalahan *Cyberpunk 2077 Breach Protocol* adalah metode Brute Force. Program akan mengecek semua jenis kemungkinan susunan token untuk dapat memenuhi sequence yang diberikan dengan reward terbesar dan langkah minimal.

Langkah-langkah algoritma dalam program untuk menyelesaikan permasalahan tersebut adalah sebagai berikut:

1. Menggenerasi beberapa kemungkinan urutan sequence

Program menggunakan permutasi untuk menggenerasi kemungkinan urutan sequence. Misalnya untuk 2 sequence maka ada 2 kemungkinan yaitu [1,2] dan [2,1]. Jika terdapat 3 sequence maka ada $3! = 6$ kemungkinan urutan sequence, dan seterusnya. Kemungkinan tersebut kemudian disusun dalam bentuk urutan tokennya.

2. Menggenerasi seluruh kemungkinan dari setiap urutan sequence

Dalam setiap kemungkinan urutan sequence, sebenarnya ada lebih banyak lagi kemungkinan. Yaitu, apabila token awalan pada sequence selanjutnya sama atau berhimpitan dengan sequence sebelumnya. Oleh karena itu, dari setiap urutan sequence dibuat semua kemungkinan urutan token termasuk yang berhimpitan/sama. Seluruh kombinasi kemudian disesuaikan agar memiliki panjang maksimal sebesar buffer yang sudah ditentukan, serta agar tidak ada sequence yang sama atau terduplikasi.

3. Melakukan brute force dengan berjalan sepanjang papan matriks

Setelah memiliki semua kemungkinan urutan sequence. Program brute force mulai berjalan menyusuri papan matriks dimulai dari kolom pertama pada baris pertama, dan kemudian menyusuri kolom-kolom selanjutnya pada baris pertama.

Pada setiap kolom, program melakukan perjalanan dengan panduan berupa kemungkinan urutan sequence. Jika jalan buntu, program akan mencari jalan agar bisa memenuhi urutan sequence tersebut. Jika jalan benar-benar buntu, artinya jalan sequence tersebut gagal disusuri. Selama berhasil disusuri, program akan mengecek reward yang didapat oleh sequence yang sudah disusuri tersebut. Jika mendapatkan reward tertinggi dengan langkah minimal, hasil akan diperbarui.

Catatan: algoritma ini mungkin berbeda dengan hasil yang digenerasi oleh referensi solusi. Menurut asumsi pada program ini, jika selama menyusuri papan matriks token yang sudah terbaca telah sesuai atau memenuhi sequence yang ada, maka reward akan bertambah. Algoritma program ini sangat memungkinkan urutan sequence yang berhimpitan, sehingga seringkali solusi yang diberikan lebih optimal (reward lebih besar, langkah lebih kecil) dari referensi solusi. Referensi solusi (berdasarkan tinjauan penulis) kadang tidak memperhitungkan token yang sudah terbaca namun berhimpitan.

BAB III

IMPLEMENTASI PROGRAM DALAM PYTHON

1. File Input

Terdapat dua file yang bertugas membaca dan menerima input baik dari file .txt maupun yang dapat digenerasi secara otomatis dengan meminta masukan melalui command line interface (CLI). Fungsi pada kedua file ini akan mengembalikan nilai-nilai matriks, sequences, dan rewards kepada program utama. Untuk generasi masukan otomatis, reward digenerasi secara otomatis juga dengan rentang nilai 5 s.d. 30.

```
src > read_txt.py > read_txt
3 def read_txt(filename):
4
5     #Opening .txt file
6     filepath = os.path.join(os.path.dirname(__file__), '..', 'test', 'input', filename)
7     if not (os.path.exists(filepath)):
8         print("The file does not exist.")
9         return
10    else:
11        requirements = open(filepath, "r")
12        lines = [line.strip() for line in requirements]
13
14        #Get Buffer Size
15        buffer_size = int(lines[0])
16
17        #Get Matrix Size
18        line1 = lines[1].split()
19        matrix_width = int(line1[0])
20        matrix_height = int(line1[1])
21
22        #Read Matrix
23        matrix = []
24        for i in range(2, 2+matrix_height):
25            line = lines[i].split()
26            matrix_row = []
27            for j in range(matrix_width):
28                matrix_row.append(line[j])
29            matrix.append(matrix_row)
30
31        #Read Sequences
32        seq_numline = 2+matrix_height
33        number_of_sequences = int(lines[seq_numline])
34
35        sequences = []
36        rewards = []
37
38        for i in range(number_of_sequences):
39            sequence = []
40            sequence_tokens = lines[seq_numline+1+2*i].split()
41            sequences.append(sequence_tokens)
42
43            reward = int(lines[seq_numline+2*(1+i)])
44            rewards.append(reward)
45        return buffer_size, matrix_width, matrix_height, matrix, number_of_sequences, sequences, rewards
```

Gambar 2. File input 'read_txt.py'

```

src > read_cli.py > read_cli
1 import numpy as np
2
3 def read_cli():
4     #Get Unique Tokens
5     num_unique_tokens = int(input("Masukkan jumlah token unik: "))
6     print("Token haruslah dua karakter alfanumerik.")
7
8     tokens=[]
9     for i in range(num_unique_tokens):
10         token = input(f"Masukkan token {i+1}: ")
11         tokens.append(token)
12     #Get Buffer Size
13     buffer_size = int(input("Masukkan ukuran buffer: "))
14     #Generate Matrix
15     matrix_height = int(input("Masukkan tinggi matriks: "))
16     matrix_width = int(input("Masukkan lebar matriks: "))
17     matrix = np.random.choice(tokens, size=(matrix_height, matrix_width))
18     #Read Sequence Req
19     number_of_sequences = int(input("Masukkan jumlah sequences: "))
20     seq_max_length = int(input("Berapa panjang maksimal sequences-nya?: ")) + 1
21     #Generate Sequences
22     sequences = []
23     rewards = []
24     for i in range(number_of_sequences):
25         sequence = [np.random.choice(tokens) for j in range(np.random.randint(2, seq_max_length))]
26         sequences.append(sequence)
27         rewards.append(np.random.randint(5,30))
28
29     #Information
30     print("\nMatriks yang digunakan adalah:")
31     for i in range(matrix_height):
32         for j in range(matrix_width):
33             print(matrix[i][j], end=' ')
34         print()
35
36     print("\nSequences yang akan dicocokkan adalah:")
37     for seq in range(len(sequences)):
38         print(f"{seq+1}.", end=' ')
39         for tok in range(len(sequences[seq])):
40             print(sequences[seq][tok], end=' ')
41         print(f"(Reward = {rewards[seq]})")
42
43     return buffer_size, matrix_width, matrix_height, matrix, number_of_sequences, sequences, rewards

```

Gambar 3. File input 'read_cli.py'

2. File Output

File 'output.py' ini berisi fungsi untuk menampilkan solusi optimal program ke layar dan juga fungsi untuk menyimpan solusi ke file .txt.

```

output.py > save_txt
#File ini berisi fungsi untuk melakukan output

import os

def print_to_console(max_reward, sequence, final_coordinates, time):
    print()
    print("-----")
    print()
    print("Solusi paling optimal adalah:")
    print()
    print(max_reward)
    # Most optimal Sequence
    for i in range(len(sequence)):
        print(sequence[i], end=' ')
    print()
    # Most Optimal Coordinates
    for i in range(len(final_coordinates)):
        print(f"{final_coordinates[i][0]}, {final_coordinates[i][1]}")
    print()
    print(time, " ms")

    print("Anda dapat menyimpan solusi ini dalam file.txt")

```

Gambar 4. Fungsi menampilkan solusi ke layar


```

output.py > save_txt
print("Anda dapat menyimpan solusi ini dalam file.txt")

def save_txt(filename, max_reward, sequence, final_coordinates, time):
    filepath = os.path.join(os.path.dirname(__file__), '..', 'test', 'output', filename)
    f = open(filepath, "w")

    f.write(str(max_reward) + '\n')
    for i in range(len(sequence)):
        f.write(sequence[i])
        f.write(' ')
    f.write('\n')
    for i in range(len(final_coordinates)):
        f.write(f"{final_coordinates[i][0]},{final_coordinates[i][1]}\n")
    f.write('\n')
    f.write(str(time))
    f.write(" ms")

    f.close()

    print(f"\nSolusi Anda telah disimpan pada file {filename}")

```

Gambar 5. Fungsi menyimpan solusi ke file .txt

3. File Generasi Kombinasi Sequence

File ‘combination.py’ ini melakukan generasi kemungkinan kombinasi sequence sebagaimana yang dijelaskan pada algoritma langkah 1 dan 2.

```

combination.py > generate_combinations
6 def generate_permutations(n):
7     numbers = list(range(1, n+1)) # Create a list of numbers from 1 to n
8     all_permutations = list(permutations(numbers)) # Generate permutations
9     all_permutations_as_lists = [list(permutation) for permutation in all_permutations] # Convert
10    return all_permutations_as_lists
11
12 def tokens_of_permutations(all_permutations, sequences):
13     sequence_combination = []
14     for perm in range(len(all_permutations)):
15         combination = []
16         for i in range(len(all_permutations[0])):
17             combination.append(sequences[all_permutations[perm][i]-1])
18         sequence_combination.append(combination)
19     return sequence_combination
20
21 def get_unions(list1, list2):
22     combinations = [list1+list2]
23     for i in range(min(len(list1), len(list2)) + 1):
24         if list1[-i:] == list2[:i]:
25             combinations.append(list1 + list2[i:])
26     return combinations
27
28 def generate_combinations(list, all_permutations):
29     each_combinations = get_unions(list[0], list[1])
30     combinations = each_combinations
31     if (len(list)>2):
32         for j in range(2, len(all_permutations[0])):
33             each_combinations = combinations
34             for k in range(len(each_combinations)):
35                 each_combinations[k] = get_unions(each_combinations[k], list[j])
36
37     combinations = []
38     for sublist in each_combinations:
39         combinations.extend(sublist)
40     return combinations
41
42 def generate_all_combinations(sequence_combination, all_permutations):
43     all_combinations = []
44     for i in range(len(all_permutations)):
45         each_combinations = generate_combinations(sequence_combination[i], all_permutations)
46         all_combinations.extend(each_combinations)
47     return all_combinations
48

```

Gambar 6. File ‘combination.py’

4. File Utama

Berisi kode program untuk melakukan perjalanan pada papan matriks sebagaimana dijelaskan pada algoritma langkah ketiga.

```
from read_txt import read_txt
from read_cli import read_cli
from util import intro
import math
import time
import combination
import output

#Pemilihan Masukan dan Load Variabel yang Dibutuhkan
intro()
choice = int(input("Masukan file dari (1) file .txt (2) otomatis / CLI : "))
if choice==1:
    filename = input("Masukkan nama file dalam <namafile>.txt: ")
    buffer_size, matrix_width, matrix_height, matrix , number_of_sequences,
elif choice==2:
    buffer_size, matrix_width, matrix_height, matrix , number_of_sequences,

#Inisialisasi Nilai
start_time = time.time()
max_reward = 0
min_steps = 999999
current_steps = 0
current_coordinate = []
ongoing_coordinates = []
final_coordinates = []
current_reward = 0
state = "hor"
col_win = -1
```

Gambar 7. Load masukan dan Inisialisasi nilai-nilai

```
c > main.py > [e] cols
29
30 #Definisi Fungsi-Fungsi
31 > def get_token_combinations(): ...
36
37 > def move_vertical(next_token, row_coordinate=0): ...
61
62 > def move_horizontal(next_token, col_coordinate=0): ...
86
87 > def reset(): ...
96
97 > def matching_check(ongoing_coordinates): ...
114
115 > def coord_to_token(coord): ...
121
122 > def adjust_final_coordinates(coord, steps, i): ...
132
133 > def adjust_allcomb(allcomb, buffer): ...
143
144 > def teso(coord): ...
```

Gambar 8. Definisi berbagai fungsi untuk penyusunan

```

main.py > cols buffer_size, matrix_width, matrix_height, matrix , number_of_sequences, sequences,
allcomb = get_token_combinations() Full name: src.main.buffer_size
allcomb = adjust_allcomb(allcomb, buffer_size)

for cols in range(matrix_width): #Brute force dari setiap sel pada baris pertama
    starting_cell = matrix[0][cols]

    for i in range (len(allcomb)): #Brute force dari seluruh kombinasi sequence yang mungkin
        reset()
        state="hor"
        j = 0
        current_reward = 0
        starting_token = allcomb[i][j]
        current_coordinate = [1, cols+1]
        ongoing_coordinates.append(current_coordinate)

        #Initiation
        if (starting_cell == starting_token):
            j+=1

        #Looping
        fail = False
        prevfail = False
        while (len(ongoing_coordinates)>0) and (0<=current_steps<buffer_size-1) and (j<len(allcomb[i])):
            current_coordinate = ongoing_coordinates[-1] #memastikan current_coordinate adalah koordinat terakhir
            next_token = allcomb[i][j]

            if state=="hor": ...

            elif state=="ver": ...

            if not fail:
                prevfail=False
                j+=1 #continue to next token
                if matching_check(ongoing_coordinates)!=0: #if already met the end of the sequence
                    current_reward = matching_check(ongoing_coordinates) #save current rewards

                #Updating if set a new record
                if current_reward > max_reward:
                    col_win = cols
                    max_reward = current_reward
                    final_coordinates = ongoing_coordinates
                    min_steps = current_steps
                elif current_reward == max_reward:
                    if len(ongoing_coordinates)-1 < min_steps:
                        col_win = cols
                        min_steps = len(ongoing_coordinates)-1
                        final_coordinates = ongoing_coordinates[:min_steps+1]

            if (current_steps<0) or (current_steps>=buffer_size-1):
                continue

```

Gambar 9. Brute force penyusuran untuk setiap kolom dan kombinasi sequence

```

main.py > cols
224
225 if (len(final_coordinates)==0):
226     print("Tidak ada solusi")
227 else:
228     final_coordinates_in_tokens, final_coordinates = adjust_final_coordinates(final_coordinates, min_steps, col_win)
229     end_time = time.time()
230     execution_time = 1000*(end_time-start_time)
231     output.print_to_console(max_reward, final_coordinates_in_tokens, final_coordinates, execution_time)
232
233     save_answer = input("\nApakah Anda ingin menyimpannya dalam file .txt? (y/n) ")
234     if save_answer=="y":
235         filename = input("Masukkan nama file Anda ingin menyimpan solusi di atas? (Masukkan dalam format <namafile>.txt): ")
236         output.save_txt(filename, max_reward, final_coordinates_in_tokens, final_coordinates, execution_time)
237     elif save_answer=="n":
238         print("Program selesai. Silakan jalankan program ini kembali jika Anda ingin menggunakannya lagi.")

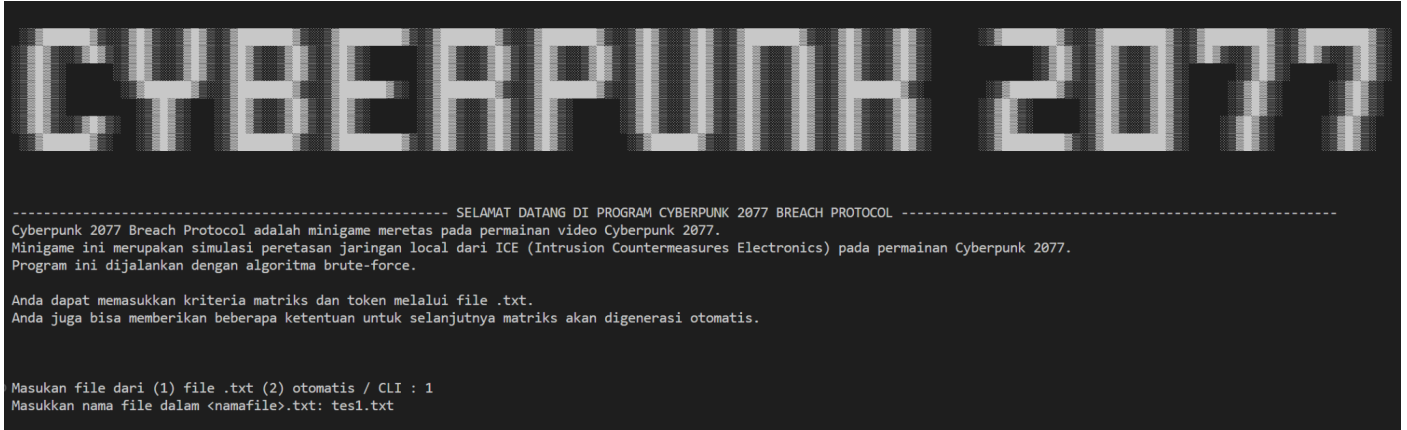
```

Gambar 10. Menampilkan solusi

BAB IV

EKSPERIMEN

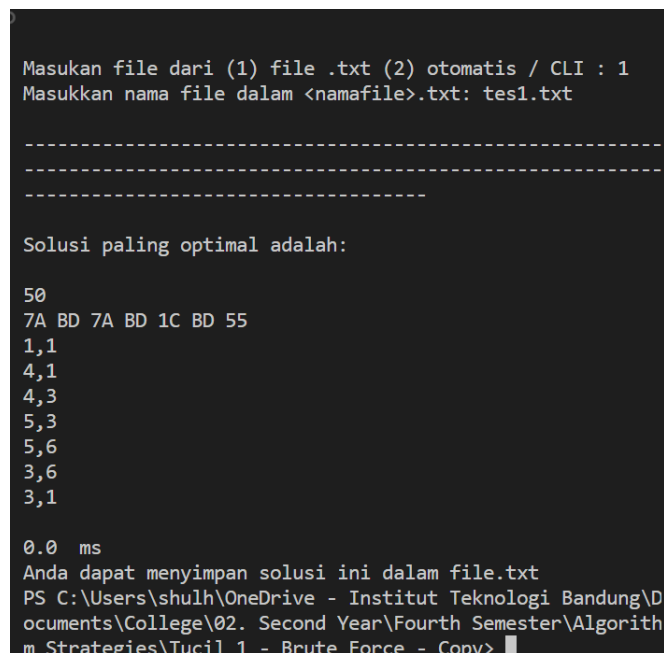
1. Inisialisasi Program



Gambar 11. Tampilan Awal Program

2. Eksperimen 1

Contoh ini merupakan contoh default pada website referensi dan juga yang tertera di deskripsi tugas.



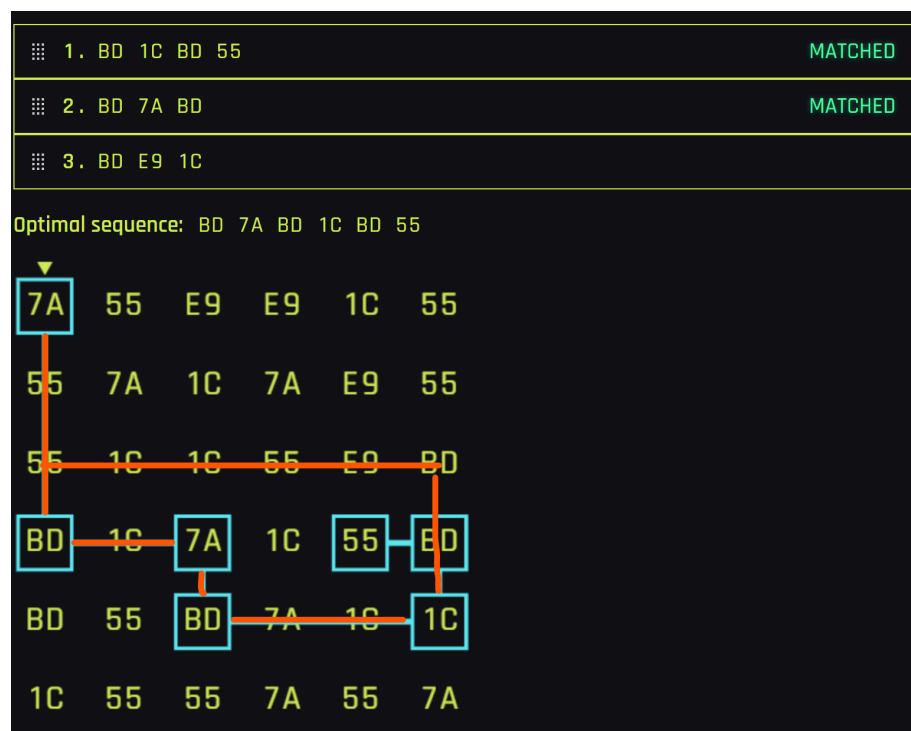
Gambar 12. Output Eksperimen 1

```

tes1.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

```

Gambar 13. Input Eksperimen 1



Gambar 14. Solusi pada referensi (biru); Solusi program saya (jingga)

3. Eksperimen 2

Modifikasi dari contoh default dengan menambah buffer hingga 10.

```

Masukan file dari (1) file .txt (2) otomatis / CLI : 1
Masukkan nama file dalam <namafilename>.txt: tes1.txt

-----

Solusi paling optimal adalah:

65
55 BD E9 1C BD 7A BD 1C BD 55
1,6
3,6
3,5
5,5
5,3
4,3
4,6
5,6
5,1
2,1

0.0 ms
Anda dapat menyimpan solusi ini dalam file.txt

```

Gambar 15. Output Eksperimen 2

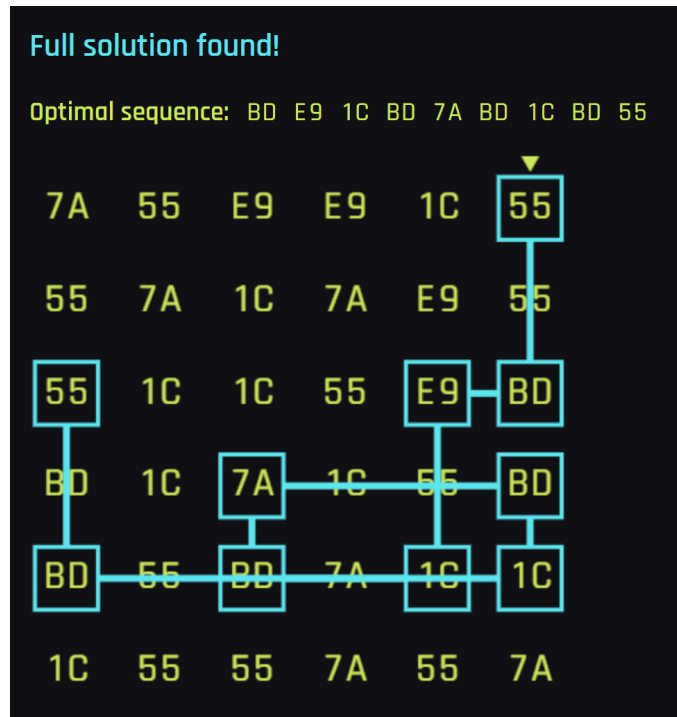
```

tes1.txt

1 10
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

```

Gambar 16. Input Eksperimen 2



Gambar 16. Solusi pada referensi (sama, koordinat mirip)

4. Eksperimen 3

Menggenerasi secara otomatis matriks dan sequence nya dengan input seperti contoh dalam deskripsi.

```
Masukan file dari (1) file .txt (2) otomatis / CLI : 2
Masukkan jumlah token unik: 5
Token haruslah dua karakter alfanumerik.
Masukkan token 1: 7A
Masukkan token 2: BD
Masukkan token 3: E9
Masukkan token 4: 1C
Masukkan token 5: 55
Masukkan ukuran buffer: 7
Masukkan tinggi matriks: 6
Masukkan lebar matriks: 6
Masukkan jumlah sequences: 3
Berapa panjang maksimal sequences-nya?: 4

Matriks yang digunakan adalah:
```

Gambar 17. Input Eksperimen 3

Matriks yang digunakan adalah:

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

Sequences yang akan dicocokkan adalah:

1. 55 7A 1C (Reward = 6)
2. 55 7A 7A 1C (Reward = 24)
3. E9 55 (Reward = 12)

Solusi paling optimal adalah:

36
 55 7A 7A 1C E9 55
 1,1
 5,1
 5,4
 3,4
 3,3
 1,3
 2.019166946411133 ms

Gambar 18. Output Eksperimen 3

1. 55 7A 7A 1C	MATCHED
2. E9 55	MATCHED
3. 55 7A 1C	

Optimal sequence: E9 55 55 7A 7A 1C

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

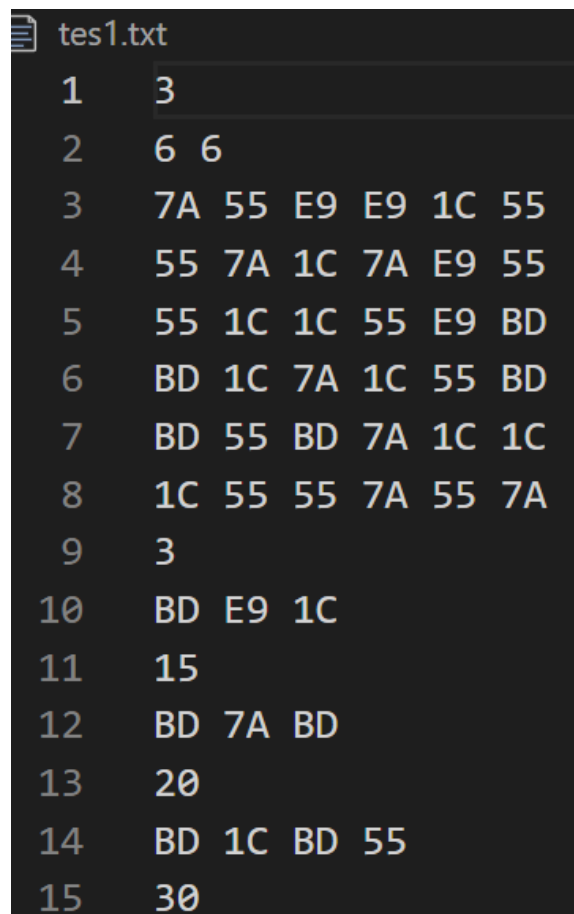
Gambar 18. Solusi pada referensi (berkebalikan urutan, nilai sama)

5. Eksperimen 4

Masukan default dengan buffer 3 yang tidak menghasilkan solusi.

```
Program ini dijalankan dengan algoritma brute-force.  
  
Anda dapat memasukkan kriteria matriks dan token melalui file .txt.  
Anda juga bisa memberikan beberapa ketentuan untuk selanjutnya matriks akan digenerasi otomatis.  
  
Masukan file dari (1) file .txt (2) otomatis / CLI : 1  
Masukkan nama file dalam <namafile>.txt: tes1.txt  
Tidak ada solusi  
PS C:\Users\shulh\OneDrive - Institut Teknologi Bandung\Documents\College\02. Second Year\Fourth Semester\Algorithm Strategies\Tucil 1 -
```

Gambar 19. Output Eksperimen 4



tes1.txt

1	3					
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					

Gambar 19. Input Eksperimen 4

No solutions found!

Change sequence priority by dragging:

1.	BD	7A	BD	
2.	BD	E9	1C	
3.	BD	1C	BD	55

Buffer size may be too small. Note that the solver currently allows only one wasted digit; at the first digit.

Gambar 20. Solusi pada referensi

6. Eksperimen 5

Input digenerasi otomatis dengan meminta masukan dari CLI dan output disimpan ke file ans1.txt

```
Masukan file dari (1) file .txt (2) otomatis / CLI : 2
Masukkan jumlah token unik: 5
Token haruslah dua karakter alfanumerik.
Masukkan token 1: AB
Masukkan token 2: 12
Masukkan token 3: CD
Masukkan token 4: 34
Masukkan token 5: EF
Masukkan ukuran buffer: 7
Masukkan tinggi matriks: 6
Masukkan lebar matriks: 6
Masukkan jumlah sequences: 3
Berapa panjang maksimal sequences-nya?: 4

Matriks yang digunakan adalah:
CD AB CD 34 CD 34
AB CD 12 12 EF 34
EF EF EF 34 AB CD
AB AB AB 34 CD CD
AB EF EF AB CD 34
12 AB CD 12 EF CD

Sequences yang akan dicocokkan adalah:
1. CD AB 34 34 (Reward = 16)
2. 12 AB EF (Reward = 23)
3. AB EF 12 (Reward = 23)
-----
```

Gambar 21. Input Eksperimen 5

```
-----
Solusi paling optimal adalah:
46
CD AB EF 12 AB EF
1,1
5,1
5,3
2,3
2,1
3,1

7.352113723754883 ms
Anda dapat menyimpan solusi ini dalam file.txt
Apakah Anda ingin menyimpannya dalam file .txt? (y/n) y
Masukkan nama file Anda ingin menyimpan solusi di atas?
(Masukkan dalam format <namafile>.txt): ans1.txt
PS C:\Users\shulh\OneDrive - Institut Teknologi Bandung\

ans1.txt
1 46
2 CD AB EF 12 AB EF
3 1,1
4 5,1
5 5,3
6 2,3
7 2,1
8 3,1
9
10 7.352113723754883 ms
```

Gambar 22 dan 23. Output Eksperimen 5

7. Eksperimen 6

Pada solusi di bawah, terlihat bahwa solusi optimal yang digenerasi program menghasilkan nilai 45 dalam 4 langkah. Solusi program ini berbeda dengan solusi referensi yang menghasilkan nilai 45 dalam 6 langkah. Hal ini telah dijelaskan pada pada Bab II pada bagian Catatan.

```

Masukan file dari (1) file .txt (2) otomatis / CLI : 1
Masukkan nama file dalam <namafile>.txt: tes6.txt

-----
-----

Solusi paling optimal adalah:

45
55 7A 55 55 1C
1,3
6,3
6,5
5,5
5,2

16.286849975585938 ms
Anda dapat menyimpan solusi ini dalam file.txt
Apakah Anda ingin menyimpannya dalam file .txt? (y/n) ☐

```

Gambar 24. Output Eksperimen 6

```

test > input > tes6.txt
1 7
2 6 6
3 BD 7A 55 55 1C BD
4 55 55 BD E9 7A BD
5 7A BD 55 7A 7A BD
6 55 1C 7A E9 E9 BD
7 E9 1C E9 1C 55 7A
8 BD 55 7A E9 55 55
9 5
10 7A BD BD
11 5
12 7A 55 55 1C
13 10
14 55 55
15 15
16 55 7A
17 20
18 1C 7A 1C 7A 7A
19 25

```

Gambar 25. Input Eksperimen 6

Partial solution found!

Change sequence priority by dragging:

1. 1C 7A 1C 7A 7A	MATCHED
2. 55 7A	MATCHED
3. 55 55	
4. 7A 55 55 1C	
5. 7A BD BD	

Optimal sequence: 55 7A 1C 7A 1C 7A 7A

BD 7A 55 1C BD

55 55 BD E9 7A BD

7A BD 55 7A 7A BD

55 1C 7A E9 E9 BD

E9 1C E9 1C 55 7A

BD 55 7A E9 55 55

Gambar 26. Solusi referensi

BAB V

PENUTUP

5.1 Kesimpulan

Berbagai cara bisa dilakukan untuk melakukan brute force. Brute force memang membutuhkan waktu yang lama karena mencari jawaban dari semua kemungkinan yang ada.

5.2 Link Repository

Link repository untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut https://github.com/shulhajws/Tucil1_13522087

5.3 Tabel Checkpoint 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		

