

LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma
Brute Force



Disusun oleh:

Agil Fadillah Sabri (13522006)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

DAFTAR ISI

DAFTAR ISI	i
DAFTAR GAMBAR	ii
DAFTAR TABEL	iii
BAB I DESKRIPSI MASALAH	1
BAB II LANGKAH PEMECAHAN MASALAH	3
BAB III IMPLEMENTASI PROGRAM	5
BAB IV UJI COBA	11
1. Tampilan Awal Program	11
2. Input Melalui File Berekstensi txt	12
3. Input Melalui CLI (Terminal)	17
DAFTAR PUSTAKA	22
LAMPIRAN	23

DAFTAR GAMBAR

Gambar 1. Solusi Optimal untuk Matriks berdasarkan Sekuens yang Diberikan	2
Gambar 2. Implementasi Program (1)	5
Gambar 3. Implementasi Program (2)	6
Gambar 4. Implementasi Program (3)	7
Gambar 5. Implementasi Program (4)	8
Gambar 6. Implementasi Program (5)	9
Gambar 7. Implementasi Program (6)	10
Gambar 8. Tampilan Awal Program	11
Gambar 9. Tampilan Input Tidak Valid	11
Gambar 10. Contoh Test 1 (test1.txt)	12
Gambar 11. Hasil Luaran Test 1	12
Gambar 12. Hasil Luaran Test 1 (<i>txt file</i>)	12
Gambar 13. Contoh Test 2 (test2.txt).....	13
Gambar 14. Hasil Luaran Test 2	13
Gambar 15. Hasil Luaran Test 2 (<i>txt file</i>)	13
Gambar 16. Contoh Test 3 (test3.txt).....	14
Gambar 17. Hasil Luaran Test 3	14
Gambar 18. Hasil Luaran Test 3 (<i>txt file</i>)	14
Gambar 19. Contoh Test 4 (test4.txt).....	15
Gambar 20. Hasil Luaran Test 4	15
Gambar 21. Hasil Luaran Test 4 (<i>txt file</i>)	15
Gambar 22. Contoh Test 5 (test5.txt).....	16
Gambar 23. Hasil Luaran Test 5	16
Gambar 24. Hasil Luaran Test 5 (<i>txt file</i>)	16
Gambar 25. Contoh Test 6 (Terminal)	17
Gambar 26. Hasil Luaran Test 6	17
Gambar 27. Hasil Luaran Test 6 (<i>txt file</i>)	17
Gambar 28. Contoh Test 7 (Terminal)	18
Gambar 29. Hasil Luaran Test 7	18
Gambar 30. Hasil Luaran Test 7 (<i>txt file</i>)	18
Gambar 31. Contoh Test 8 (Terminal)	19
Gambar 32. Hasil Luaran Test 8	19
Gambar 33. Hasil Luaran Test 8 (<i>txt file</i>)	19
Gambar 34. Contoh Test 9 (Terminal)	20
Gambar 35. Hasil Luaran Test 9	20
Gambar 36. Hasil Luaran Test 9 (<i>txt file</i>)	20
Gambar 37. Contoh Test 10 (Terminal)	21
Gambar 38. Hasil Luaran Test 10	21
Gambar 39. Hasil Luaran Test 10 (<i>txt file</i>)	21

DAFTAR TABEL

Tabel 1. Contoh Matriks pada <i>Cyberpunk 2077 Breach Protocol</i>	1
Tabel 2. Contoh Masukan 1 dari File Berekstensi txt	12
Tabel 3. Contoh Masukan 2 dari File Berekstensi txt.....	13
Tabel 4. Contoh Masukan 3 dari File Berekstensi txt.....	14
Tabel 5. Contoh Masukan 4 dari File Berekstensi txt	15
Tabel 6. Contoh Masukan 5 dari File Berekstensi txt	16
Tabel 7. Contoh Masukan 1 dari Terminal	17
Tabel 8. Contoh Masukan 2 dari Terminal	18
Tabel 9. Contoh Masukan 3 dari Terminal	19
Tabel 10. Contoh Masukan 4 dari Terminal	20
Tabel 11. Contoh Masukan 5 dari Terminal	21

BAB I

DESKRIPSI MASALAH

Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video *Cyberpunk 2077*. Minigame ini merupakan simulasi peretasan jaringan lokal 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.

Berikut adalah ilustrasi kasus. Misal terdapat matriks sebagai berikut:

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

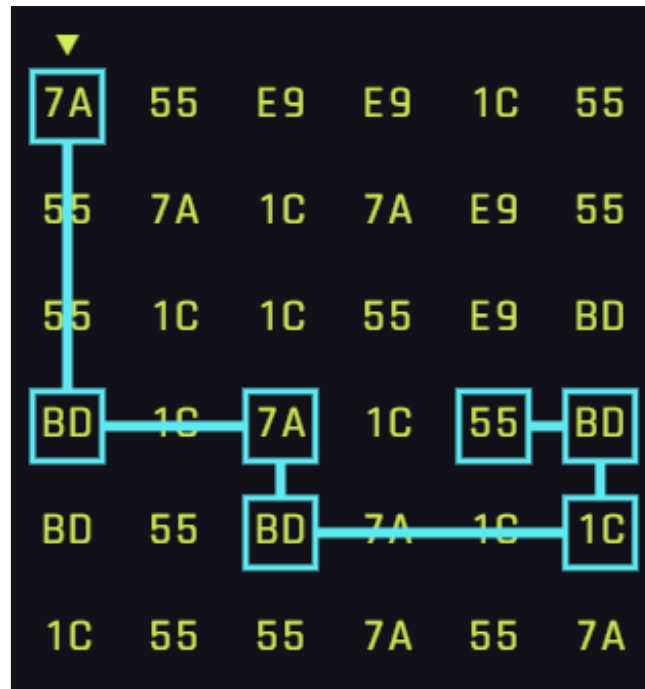
Tabel 1. Contoh Matriks pada *Cyberpunk 2077 Breach Protocol*

Jika panjang buffernya adalah tujuh, dan terdapat tiga 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:

1. Total bobot hadiah : 50 poin.
2. Total langkah : 6 langkah.



Gambar 1. Solusi Optimal untuk Matriks berdasarkan Sekuens yang Diberikan

BAB II

LANGKAH PEMECAHAN MASALAH

Metode algoritma yang digunakan untuk menyelesaikan permainan *Cyberpunk 2077 Breach Protocol* ini adalah metode *Brute Force*. Adapun langkah-langkah penyelesaiannya adalah sebagai berikut:

1. Pertama-tama, program menerima masukan dari pengguna, bisa melalui file berekstensi txt ataupun melalui *Command Line Interface* (CLI) pada terminal. Dari sini akan diperoleh ukuran *buffer*, isi matriks, sekuens beserta *reward* masing-masing sekuens. Sebelum dilakukan pencarian, terlebih dahulu dihitung *reward* maksimum yang mungkin bisa diperoleh dari sekuens yang ada (dengan menjumlahkan semua *reward* sekuens yang bernilai positif). Jika semua sekuens memiliki *reward* negatif atau 0, maka *reward* maksimum yang bisa diperoleh dapat dianggap sama dengan 0 (karena $0 \geq$ semua bilangan negatif). Hal ini berguna nantinya untuk meningkatkan efisiensi pencarian sekuens.
2. Karena permainan selalu dimulai dengan memilih salah satu token yang ada pada baris pertama matriks, maka pada *main program*, proses pencarian diawali dengan melakukan perulangan (*looping*) sebanyak jumlah kolom untuk mengiterasi satu-persatu kemungkinan yang ada pada setiap kolom pada baris pertama. Pada tahap ini, *buffer* pada posisi pertama telah terisi.
3. Selanjutnya, dalam setiap perulangan pada program utama tersebut, akan dilakukan pengisian *buffer* pada posisi kedua, dengan melakukan pergerakan secara vertikal. Seperti langkah sebelumnya, akan dilakukan perulangan untuk mengiterasi setiap kemungkinan token yang ada pada kolom yang sama dengan token pada *buffer* di posisi pertama (karena pergerakan vertikal). Pada tahap ini, *buffer* pada posisi kedua telah terisi. Adapun jumlah perulangan dilakukan sebanyak (jumlah baris – 1), karena tidak mengikutkan token yang sudah mengisi *buffer* pada posisi tepat sebelumnya dari posisi saat ini. Selain itu, untuk sel matriks yang telah diambil, tidak bisa lagi digunakan, sehingga ketika bertemu dengan sel matriks yang telah diambil, proses perulangan langsung berlanjut untuk sel berikutnya. Hal ini dilakukan dengan memberikan tanda pada setiap sel matriks, dengan ‘0’ belum diambil dan ‘1’ telah diambil.
4. Selanjutnya, untuk setiap perulangan pada langkah ketiga, akan kembali dilakukan perulangan untuk mengisi *buffer* pada posisi ketiga, dengan pergerakan secara horisontal. Perulangan ini dilakukan untuk mengiterasi setiap kemungkinan token yang ada pada baris yang sama dengan token pada posisi kedua. jumlah perulangan dilakukan sebanyak (jumlah kolom – 1), karena tidak mengikutkan token yang sudah mengisi *buffer* pada posisi tepat sebelumnya dari posisi saat ini. Sama seperti sebelumnya, sel matriks yang telah diambil, tidak bisa dipakai kembali pada *buffer* yang sama.
5. Langkah 3 dan 4 selanjutnya diulangi secara terus-menerus secara bergantian hingga *buffer* telah terisi penuh (telah mencapai posisi terakhir) dengan menggunakan pendekatan secara rekursif. Terdapat dua fungsi yang digunakan, yaitu fungsi untuk melakukan iterasi secara vertikal (*function* vertical) dan fungsi untuk melakukan iterasi secara horizontal (*function* horizontal). Basis dari kedua fungsi rekursif ini adalah

ketika *buffer* telah terisi penuh, dimana akan langsung dilakukan proses perhitungan *reward* yang didapatkan dan keluar dari proses rekursif. Jika seandainya *reward* saat ini lebih tinggi dari *reward* sebelumnya, maka *reward* tertinggi akan ditukar dan isi *buffer*-nya akan disimpan. Adapun proses rekursifnya dilakukan dengan cara sebagai berikut:

- a. Pada setiap iterasi pada program utama (langkah 2), dilakukan pemanggilan fungsi vertikal (*function* vertical).
 - b. Selanjutnya, di dalam fungsi vertikal, dilakukan pengecekan apakah kondisi *buffer* telah memenuhi basis atau belum. Jika belum, maka akan dilakukan perulangan kembali secara vertikal (langkah 3). Pada setiap perulangan ini akan dilakukan pemanggilan fungsi horizontal (*function* horizontal).
 - c. Selanjutnya, di dalam fungsi horizontal, juga dilakukan pengecekan apakah kondisi *buffer* telah memenuhi basis atau belum. Jika belum, maka akan dilakukan perulangan kembali secara horizontal (langkah 4). Pada setiap perulangan ini akan dilakukan kembali pemanggilan fungsi vertikal (*function* vertical) dan proses diulangi kembali.
 - d. Proses ini terus berlanjut hingga *buffer* telah terisi penuh.
6. Untuk meningkatkan efisiensi program, pada fungsi rekursif ditambahkan satu basis tambahan, dimana jika isi *buffer* saat ini telah mencapai *reward* maksimum yang mungkin bisa didapatkan (hasil paling optimal), maka akan segera keluar dari proses rekursif.
 7. Setelah proses pencarian selesai, program kemudian menampilkan pada terminal *reward* maksimum yang bisa diperoleh dari matriks yang ada, urutan isi token pada *buffer* yang diperoleh, posisi masing-masing token pada *buffer* di dalam matriks (dengan format: kolom, baris), waktu yang diperlukan untuk melakukan proses pencarian (dalam milisecond/ms) dan *prompt* atau pertanyaan apakah hasil ini ingin disimpan dalam file berekstensi txt (dengan format yang sama). Jika sebelumnya masukan via CLI, maka juga akan ditampilkan matriks yang digunakan, sekuens serta *reward* masing-masing sekuens.

BAB III

IMPLEMENTASI PROGRAM

Program penyelesaian permainan *Cyberpunk 2077 Breach Protocol* ini diimplementasikan menggunakan bahasa pemrograman python (.py). Adapapun kode programnya adalah sebagai berikut:

```
src > main.py > ...
1  import random
2  import time
3  import os
4
5  global current_reward_maksimum, buffer_with_reward_maksimum, max_reward_can_get
6  global list_of_sequence, jumlah_sequence, metode_input
7  global row_col, matriks_of_token, buffer_size, current_buffer
8  # current_reward_maksimum : int
9  # buffer_with_reward_maksimum : List of (string, int, int) -----> (token, baris, kolom)
10 # buffer_size : int
11 # row_col : (int, int)
12 # matriks_of_token : matriks of string
13 # jumlah_sequence : int
14 # list_of_sequence : List of (string, int)
15 # max_reward_can_get : int
16 # current_buffer : List of (string, int, int) -----> (token, baris, kolom)
17 # metode_input : string
18
19 current_reward_maksimum = 0
20 buffer_with_reward_maksimum = []
21
22 ##### FUNGSI/PROSEDUR #####
23 def get_row_col(width_height):
24     # mengubah masukan menjadi tuple baris dan kolom
25     # width_height : string
26     # return : (int, int) -----> (baris, kolom)
27     for i in range(len(width_height)):
28         if width_height[i] == " ":
29             return (int(width_height[i+1:]), int(width_height[:i]))
30
31 def get_row_of_token(string_of_token, column_of_matriks):
32     # mengubah baris masukan menjadi 1 baris elemen matriks token
33     # string_of_token : string
34     # column_of_matriks : int
35     # return : List of string
36     row_of_token = [[' ', 0] for i in range(column_of_matriks)]
37     column = 0
38
39     string_without_space = string_of_token.replace(" ", "")
40
41     if (len(string_without_space) % 2 != 0) or (len(string_without_space)/2 != column_of_matriks):
42         # terdapat token yang panjangnya tidak sama dengan 2
43         print()
44         print("Terdapat token yang tidak valid pada matriks token.")
45         print("Silahkan perbaiki terlebih dahulu file txt-nya!")
46         exit()
47
48     for i in range(0, len(string_of_token), 3):
49         row_of_token[column][0] = string_of_token[i:i+2]
50         column += 1
51     return row_of_token
```

Gambar 2. Implementasi Program (1)

```

53 def buffer_to_string(buffer):
54 # mengubah isi token pada buffer menjadi string token
55 # buffer : List of (string, int, int) -----> (token, baris, kolom)
56 # return : string
57     string_buffer = ''
58     for i in range(buffer_size):
59         string_buffer += buffer[i][0]
60     return string_buffer
61
62 def hitung_reward(buffer):
63 # menghitung reward dari token-token yang ada pada buffer
64 # buffer : List of (string, int, int) -----> (token, baris, kolom)
65 # return : integer
66     hasil_buffer = buffer_to_string(buffer)
67     reward = 0
68     for i in range(jumlah_sequence):
69         if list_of_sequence[i][0] in hasil_buffer:
70             reward += list_of_sequence[i][1]
71     return reward
72
73 def copy_buffer(buffer):
74 # mengcopy isi buffer ke buffer lain
75 # buffer : List of (string, int, int) -----> (token, baris, kolom)
76 # return : List of (string, int, int) -----> (token, baris, kolom)
77     buffer_copy = [['', -999, -999] for i in range(buffer_size)]
78     for i in range(buffer_size):
79         buffer_copy[i] = buffer[i]
80     return buffer_copy
81
82 def generate_random_sequence(jumlah_token_unik, list_token, ukuran_maksimum_sequence):
83 # menghasilkan sequence secara random
84 # jumlah_token_unik : int
85 # list_token : List of string -----> daftar token unik
86 # ukuran_maksimum_sequence : int
87     global max_reward_can_get, list_of_sequence, jumlah_sequence
88     max_reward_can_get = 0
89
90     list_of_sequence = [' ' for i in range(jumlah_sequence)]
91     for i in range(jumlah_sequence):
92         panjang_sequence = random.randint(2, ukuran_maksimum_sequence)
93         string_sequence = ''
94         for j in range(panjang_sequence):
95             string_sequence += list_token[random.randint(0, jumlah_token_unik-1)]
96             list_of_sequence[i] = (string_sequence, random.randint(-10, 10)*5)
97
98         if list_of_sequence[i][1] > 0:
99             max_reward_can_get += list_of_sequence[i][1]
100
101 def is_sequence_unik(list_of_sequence):
102 # mengecek apakah sequence unik
103 # list_of_sequence : List of (string, int) -----> (sekuens, reward)
104 # return : boolean
105     global jumlah_sequence
106     for i in range(jumlah_sequence):
107         for j in range(0,i):
108             if list_of_sequence[j][0] == list_of_sequence[i][0]:
109                 return False
110     return True

```

Gambar 3. Implementasi Program (2)

```

112 def membaca_input(cara_input):
113     # prosedur membaca masukan dari pengguna
114     # cara_input : string
115     global buffer_size, row_col, matriks_of_token, jumlah_sequence, list_of_sequence
116     global max_reward_can_get, current_buffer, metode_input
117
118     metode_input = cara_input
119     max_reward_can_get = 0
120
121     if cara_input == "1":                                     # membaca input dari file txt
122         print()
123         print("===== INPUT =====")
124
125         print("Pastikan file txt berada pada folder test")
126         nama_file = input("Masukkan nama file (ex: test.txt): ")
127
128         while not os.path.exists("test/" + nama_file):
129             print("File tidak ditemukan! Silahkan masukkan nama file lainnya.")
130             nama_file = input("Masukkan nama file (ex: test.txt): ")
131
132         f = open("test/" + nama_file, "r")
133
134         buffer_size = int(f.readline().replace("\n", ""))
135         current_buffer = [['', -999, -999] for i in range(buffer_size)]
136
137         row_col = get_row_col(f.readline().replace("\n", ""))
138         matriks_of_token = [[' ' for i in range(row_col[0])]
139                             for i in range(row_col[1])]
140         for i in range(row_col[0]):
141             matriks_of_token[i] = get_row_of_token(f.readline().replace("\n", ""), row_col[1])
142
143         jumlah_sequence = int(f.readline().replace("\n", ""))
144         list_of_sequence = [[' ' for i in range(jumlah_sequence)]
145                             for i in range(jumlah_sequence)]
146         for i in range(jumlah_sequence):
147             sequence = f.readline().replace("\n", "")
148             jumlah_spasi = sequence.count(" ")
149             seq_without_space = sequence.replace(" ", "")
150
151             if (len(seq_without_space) % 2 != 0) or (len(seq_without_space)/2 != jumlah_spasi+1):
152                 # terdapat token yang panjangnya tidak sama dengan 2
153                 print()
154                 print("Terdapat token yang tidak valid pada sequence.")
155                 print("Silahkan perbaiki terlebih dahulu file txt-nya!")
156                 exit()
157
158             if i >= 1:
159                 for j in range(0,i):
160                     if list_of_sequence[j][0] == seq_without_space:
161                         print()
162                         print("Terdapat 2 sequence yang sama pada file txt.")
163                         print("Silahkan perbaiki terlebih dahulu file txt-nya!")
164                         exit()
165
166             list_of_sequence[i] = (seq_without_space, int(f.readline().replace("\n", "")))
167             if list_of_sequence[i][1] > 0:
168                 max_reward_can_get += list_of_sequence[i][1]
169         f.close()
170
171     elif cara_input == "2":                                     # membaca input via cli
172         print()
173         print("===== INPUT =====")
174
175         jumlah_token_unik = int(input("Masukkan jumlah token unik : "))
176         while (jumlah_token_unik < 1):
177             print("Jumlah token unik tidak valid!")
178             jumlah_token_unik = int(input("Masukkan jumlah token unik : "))
179
180         list_token = [[' ' for i in range(jumlah_token_unik)]
181                     unik = False
182         while (unik == False):
183             token = input("Masukkan token unik : ")
184             total_space = token.count(" ")
185             token_without_space = token.replace(" ", "")
186
187             if (len(token_without_space) % 2 == 0) and (len(token_without_space)/2 == jumlah_token_unik) and (total_space == jumlah_token_unik-1):
188                 for i in range(0, len(token_without_space), 2):
189                     list_token[i//2] = token_without_space[i:(i+2)]
190                 if len(list_token) == len(set(list_token)):
191                     unik = True
192
193             if (unik == False):
194                 print("Token yang dimasukkan tidak valid/tidak unik.")
195                 print("Silahkan masukkan token unik kembali!")

```

Gambar 4. Implementasi Program (3)

```

195     buffer_size = int(input("Masukkan ukuran buffer                               : "))
196     while (buffer_size < 1):
197         print("Ukuran buffer tidak valid!")
198         buffer_size = int(input("Masukkan ukuran buffer                               : "))
199
200     row_col = get_row_col(input("Masukkan lebar (kolom) dan tinggi (baris) matriks : "))
201     while (row_col[0] < 1) or (row_col[1] < 1):
202         print("Jumlah baris dan kolom tidak valid!")
203         row_col = get_row_col(input("Masukkan lebar (kolom) dan tinggi (baris) matriks : "))
204
205     jumlah_sequence = int(input("Masukkan jumlah sequence                         : "))
206     while (jumlah_sequence < 1):
207         print("Jumlah sequence tidak valid!")
208         jumlah_sequence = int(input("Masukkan jumlah sequence                       : "))
209
210     ukuran_maksimum_sequence = int(input("Masukkan ukuran maksimum sequence       : "))
211     while (ukuran_maksimum_sequence < 2):
212         print("Ukuran maksimum sequence tidak valid!")
213         ukuran_maksimum_sequence = int(input("Masukkan ukuran maksimum sequence   : "))
214
215     current_buffer = [['', -999, -999] for i in range(buffer_size)]
216
217     matriks_of_token = [[[' ', 0] for i in range(row_col[1])] for j in range(row_col[0])]
218     for i in range(row_col[0]):
219         for j in range(row_col[1]):
220             matriks_of_token[i][j][0] = list_token[random.randint(0, jumlah_token_unik-1)]
221
222     generate_random_sequence(jumlah_token_unik, list_token, ukuran_maksimum_sequence)
223     while not is_sequence_unik(list_of_sequence):
224         generate_random_sequence(jumlah_token_unik, list_token, ukuran_maksimum_sequence)
225
226     else:
227         ulangi = input("Input tidak valid, apakah anda ingin mengulangi input? (y/n) : ")
228         if ulangi == "y" or ulangi == "Y":
229             cara_input = input("Masukkan metode yang diinginkan (pastikan memasukkan 1/2): ")
230             membaca_input(cara_input)
231         elif ulangi == "n" or ulangi == "N":
232             exit()
233         else:
234             membaca_input(3)
235
236     ~~~
237
238     def write_to_file(ingin_menyimpan):
239         # menuliskan hasil ke file txt
240         # ingin_menyimpan : string
241
242         if ingin_menyimpan == "y" or ingin_menyimpan == "Y":
243             nama_file = input("Masukkan nama file (ex: output.txt): ")
244
245             while os.path.exists("test/" + nama_file):
246                 print("File dengan nama tersebut sudah ada! Silahkan masukkan nama file lainnya.")
247                 nama_file = input("Masukkan nama file (ex: output.txt): ")
248
249             f = open("test/" + nama_file, "w")
250
251             f.write(f"{current_reward_maksimum}\n")
252
253             if current_reward_maksimum > 0:
254                 for i in range(buffer_size):
255                     f.write(f"{buffer_with_reward_maksimum[i][0]} ")
256                     f.write("\n")
257                 for i in range(buffer_size):
258                     f.write(f"{buffer_with_reward_maksimum[i][2]+1}, {buffer_with_reward_maksimum[i][1]+1}\n")
259                     f.write("\n")
260             else:
261                 f.write("No Solution\n\n")
262
263             f.write(f"{waktu_eksekusi*1000} ms\n")
264             f.close()
265
266             print("Solusi berhasil disimpan pada file", nama_file)
267
268         elif ingin_menyimpan == "n" or ingin_menyimpan == "N":
269             exit()
270         else:
271             ingin_menyimpan = input("Input tidak valid, apakah anda ingin kembali menyimpan solusi? (y/n): ")
272             write_to_file(ingin_menyimpan)

```

Gambar 5. Implementasi Program (4)


```

236 def vertical(current_buffer_position):
237     # proses brute force
238     # melakukan pencarian token secara vertikal pada matriks token
239     # current_buffer_position : int
240     global current_reward_maksimum, buffer_with_reward_maksimum
241
242     if current_reward_maksimum == max_reward_can_get:
243         return
244     else:
245         if current_buffer_position == buffer_size:
246             reward = hitung_reward(current_buffer)
247             if reward > current_reward_maksimum:
248                 current_reward_maksimum = reward
249                 buffer_with_reward_maksimum = copy_buffer(current_buffer)
250             return
251         else:
252             for i in range(1, row_col[0]):
253                 row = (current_buffer[current_buffer_position-1][1]+i)%row_col[0]
254                 column = current_buffer[current_buffer_position-1][2]
255                 if matriks_of_token[row][column][1] == 0:
256                     current_buffer[current_buffer_position] = (matriks_of_token[row][column][0], row, column)
257                     matriks_of_token[row][column][1] = 1
258                     horizontal(current_buffer_position+1)
259                     matriks_of_token[row][column][1] = 0
260
261 def horizontal(current_buffer_position):
262     # proses brute force
263     # melakukan pencarian token secara horizontal pada matriks token
264     # current_buffer_position : int
265     global current_reward_maksimum, buffer_with_reward_maksimum
266
267     if current_reward_maksimum == max_reward_can_get:
268         return
269     else:
270         if current_buffer_position == buffer_size:
271             reward = hitung_reward(current_buffer)
272             if reward > current_reward_maksimum:
273                 current_reward_maksimum = reward
274                 buffer_with_reward_maksimum = copy_buffer(current_buffer)
275             return
276         else:
277             for i in range(1, row_col[1]):
278                 row = current_buffer[current_buffer_position-1][1]
279                 column = (current_buffer[current_buffer_position-1][2]+i)%row_col[1]
280                 if matriks_of_token[row][column][1] == 0:
281                     current_buffer[current_buffer_position] = (matriks_of_token[row][column][0], row, column)
282                     matriks_of_token[row][column][1] = 1
283                     vertical(current_buffer_position+1)
284                     matriks_of_token[row][column][1] = 0

```

Gambar 6. Implementasi Program (5)

```

321 ##### MAIN PROGRAM #####
322 print("=====")
323 print("=== Selamat Datang di Cyberpunk 2077 Breach Protocol ===")
324 print("=====\n")
325
326 print("===== MENU =====")
327 print("Metode input:")
328 print("  1. File txt")
329 print("  2. Input manual")
330 cara_input = input("Masukkan metode yang diinginkan: ")
331 membaca_input(cara_input)
332
333 time_start = time.time()
334 for i in range(row_col[1]):
335     current_buffer[0] = (matriks_of_token[0][i][0], 0, i)
336     matriks_of_token[0][i][1] = 1
337     vertical(1)
338 time_end = time.time()
339
340 waktu_eksekusi = time_end - time_start
341
342 # output
343 print()
344 print("===== OUTPUT =====")
345 if metode_input == "2":
346     # jika input via cli, maka menampilkan matriks token dan sequ
347     print("Matriks of token: ")
348     for i in range(row_col[0]):
349         for j in range(row_col[1]):
350             print(matriks_of_token[i][j][0], end=" ")
351         print()
352
353     print("Sequence: ")
354     for i in range(jumlah_sequence):
355         for j in range(0, len(list_of_sequence[i][0]), 2):
356             print(list_of_sequence[i][0][j:(j+2)], end=" ")
357         print()
358         print(f"Reward: {list_of_sequence[i][1]}")
359     print()
360
361 print(current_reward_maksimum)
362 if current_reward_maksimum > 0:
363     for i in range(buffer_size):
364         print(buffer_with_reward_maksimum[i][0], end=" ")
365     print()
366     for i in range(buffer_size):
367         # format output :      kolom, baris
368         print(f"{{buffer_with_reward_maksimum[i][2]+1}}, {{buffer_with_reward_maksimum[i][1]+1}}")
369 else:
370     print("No Solution")
371
372 print()
373 print(waktu_eksekusi*1000, "ms\n")
374
375 cara_simpan = input("Apakah ingin menyimpan solusi? (y/n): ")
376 write_to_file(cara_simpan)

```

Gambar 7. Implementasi Program (6)

BAB IV

UJI COBA

1. Tampilan Awal Program

```
=====
=== Selamat Datang di Cyberpunk 2077 Breach Protocol ===
=====

===== MENU =====
Metode input:
  1. File txt
  2. Input manual
Masukkan metode yang diinginkan: █
```

Gambar 8. Tampilan Awal Program

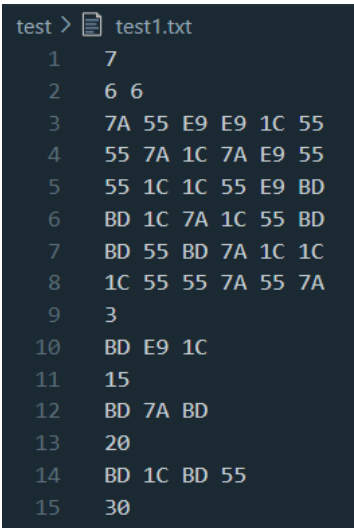
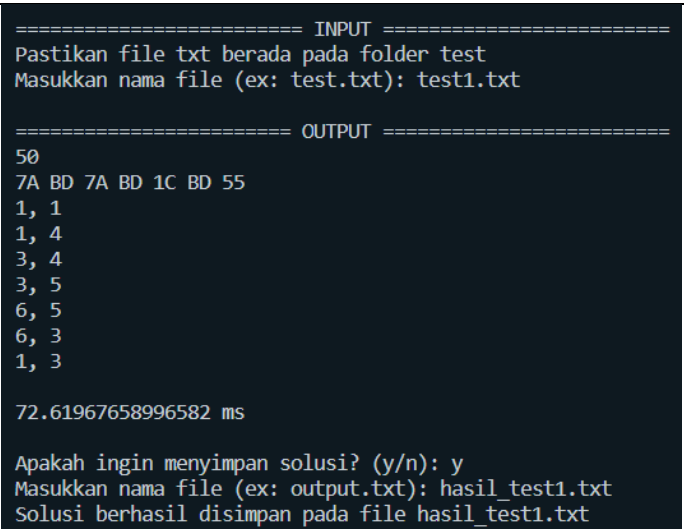
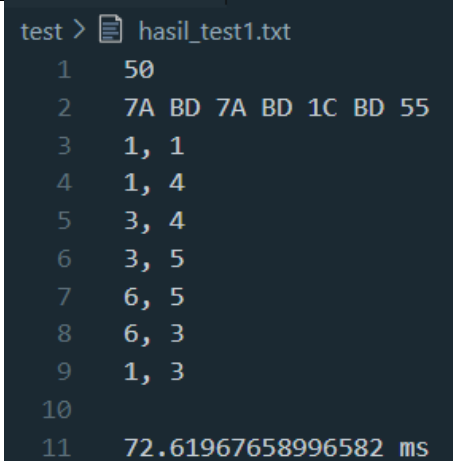
```
=====
=== Selamat Datang di Cyberpunk 2077 Breach Protocol ===
=====

===== MENU =====
Metode input:
  1. File txt
  2. Input manual
Masukkan metode yang diinginkan: 3
Input tidak valid, apakah anda ingin mengulangi input? (y/n) : a
Input tidak valid, apakah anda ingin mengulangi input? (y/n) : y
Masukkan metode yang diinginkan (pastikan memasukkan 1/2): 3
Input tidak valid, apakah anda ingin mengulangi input? (y/n) : n
PS C:\Users\agilf\Documents\MATKUL\SEM 4\STIMA\TUCIL\Tucil1_13522006> █
```

Gambar 9. Tampilan Input Tidak Valid

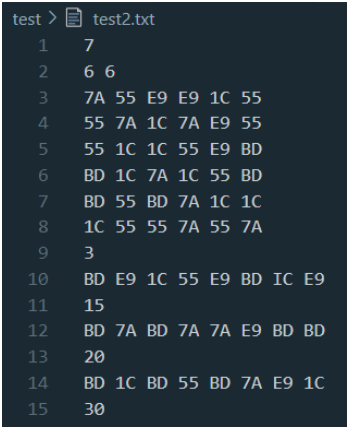
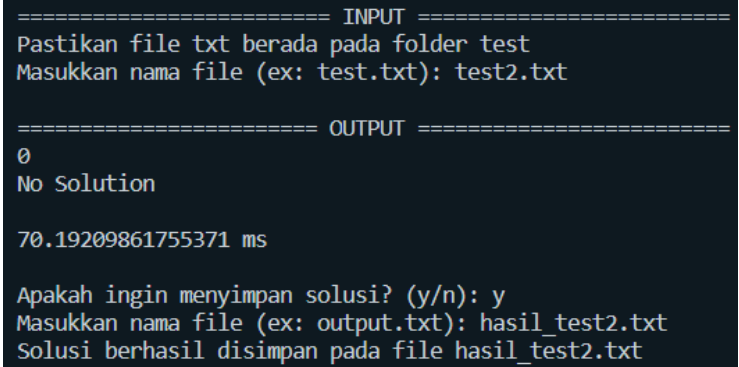
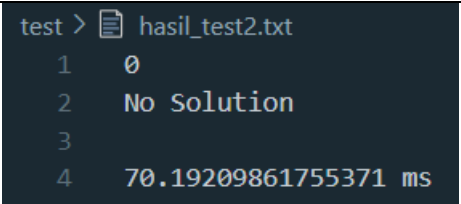
2. Input Melalui File Berekstensi txt

2.1. Contoh 1

Input	Output
 <p>Gambar 10. Contoh Test 1 (test1.txt)</p>	 <p>Gambar 11. Hasil Luaran Test 1</p>
	 <p>Gambar 12. Hasil Luaran Test 1 (<i>txt file</i>)</p>
<p>Keterangan: Masukan dari file test1.txt. <i>Reward</i> maksimal yang diperoleh adalah 50 dengan sekuens 1) BD 7A BD, 20 poin; 2) BD IC BD 55, 30 poin.</p>	

Tabel 2. Contoh Masukan 1 dari File Berekstensi txt

2.2. Contoh 2

Input	Output
 <pre>test > test2.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 55 E9 BD 1C E9 11 15 12 BD 7A BD 7A 7A E9 BD BD 13 20 14 BD 1C BD 55 BD 7A E9 1C 15 30</pre> <p>Gambar 10. Contoh Test 2 (test2.txt)</p>	 <pre>===== INPUT ===== Pastikan file txt berada pada folder test Masukkan nama file (ex: test.txt): test2.txt ===== OUTPUT ===== 0 No Solution 70.19209861755371 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (ex: output.txt): hasil_test2.txt Solusi berhasil disimpan pada file hasil_test2.txt</pre> <p>Gambar 11. Hasil Luaran Test 2</p>  <pre>test > hasil_test2.txt 1 0 2 No Solution 3 4 70.19209861755371 ms</pre> <p>Gambar 12. Hasil Luaran Test 2 (txt file)</p>
<p>Keterangan: Masukan dari file test2.txt. <i>Reward</i> maksimal yang diperoleh adalah 0. Hal ini terjadi karena seluruh sekuens memiliki panjang yang melebihi ukuran <i>buffer</i>.</p>	

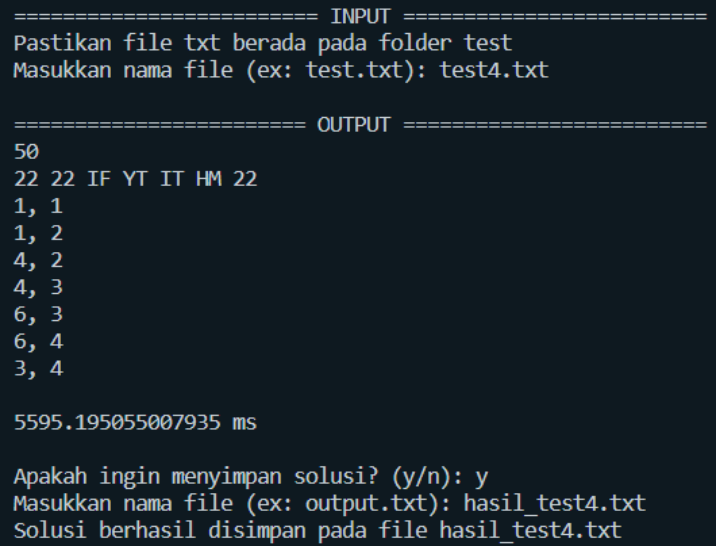
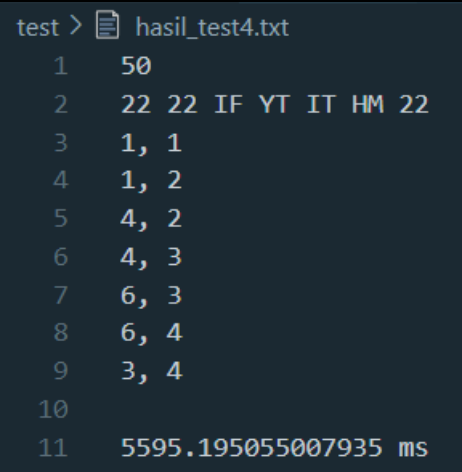
Tabel 3. Contoh Masukan 2 dari File Berekstensi txt

2.3. Contoh 3

Input	Output
 <pre> test > test3.txt 1 10 2 8 6 3 FA AH LL FA FA 5A LL FA 4 BR 5A 8L BR 5A A7 LL 5A 5 8L FA LL A7 4I LL LL 4I 6 4I 8L BR 4I A7 FA FA BR 7 AH 8L LL FA AH LL BR A7 8 AH 5A BR FA 5A A7 A7 5A 9 5 10 4I LL 4I AH 11 35 12 AH A7 BR 5A AH 8L 13 40 14 LL FA 15 15 16 4I AH 4I BR A7 17 50 18 FA LL AH BR 5A 19 15 </pre> <p>Gambar 13. Contoh Test 3 (test3.txt)</p>	 <pre> ===== INPUT ===== Pastikan file txt berada pada folder test Masukkan nama file (ex: test.txt): test3.txt ===== OUTPUT ===== 55 FA AH AH A7 BR 5A AH 8L LL FA 1, 1 1, 5 5, 5 5, 4 8, 4 8, 6 1, 6 1, 3 6, 3 6, 4 33600.28958320618 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (ex: output.txt): hasil_test3.txt Solusi berhasil disimpan pada file hasil_test3.txt </pre> <p>Gambar 14. Hasil Luaran Test 3</p>  <pre> test > hasil_test3.txt 1 55 2 FA AH AH A7 BR 5A AH 8L LL FA 3 1, 1 4 1, 5 5 5, 5 6 5, 4 7 8, 4 8 8, 6 9 1, 6 10 1, 3 11 6, 3 12 6, 4 13 14 33600.28958320618 ms </pre> <p>Gambar 15. Hasil Luaran Test 3 (txt file)</p>
<p>Keterangan: Masukan dari file test3.txt. <i>Reward</i> maksimal yang diperoleh adalah 55 dengan sekuens 1) AH 47 BR 5A AH 8L, 40 poin; 2) LL FA, 15 poin.</p>	

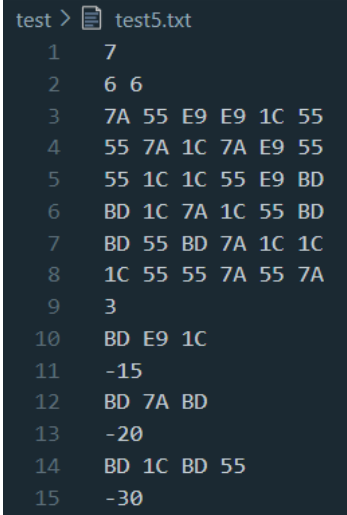
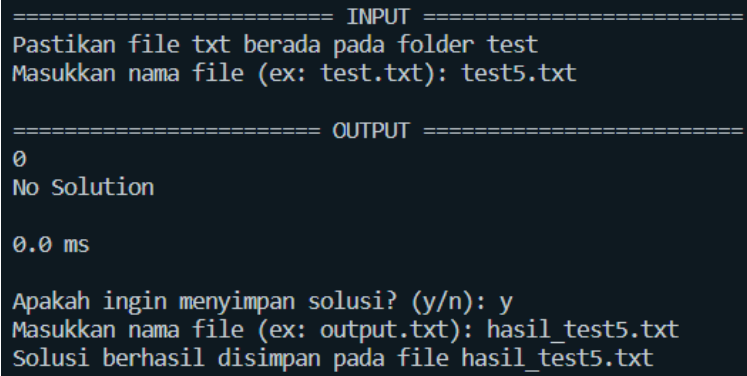
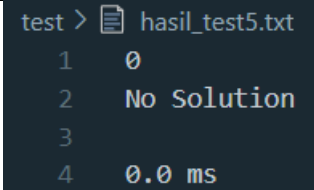
Tabel 4. Contoh Masukan 3 dari File Berektensi txt

2.4. Contoh 4

Input	Output
 <pre> test > test4.txt 1 10 2 10 10 3 22 IF IT HM 22 YT 22 IT YT 20 4 22 BB 20 IF HM BB IF IF IF BB 5 IT IF IF YT IF IT IT YT 22 YT 6 IT IF 22 22 BB HM 20 20 BB BB 7 HM 20 22 IT IF 20 YT 20 22 IF 8 IF 22 IF 22 20 HM E0 IT YT YT 9 YT YT 20 YT IF 20 22 22 22 22 10 E0 YT BB E0 YT 22 IF YT IT E0 11 20 20 YT IF YT YT BB 20 IT YT 12 BB E0 BB 20 BB HM IF IF 22 IT 13 6 14 E0 20 IF IT BB 15 5 16 20 IF 17 -30 18 YT IT HM 22 19 50 20 E0 HM E0 BB E0 E0 21 -50 22 E0 YT 20 E0 23 45 24 IF IT 22 IT 25 40 </pre> <p>Gambar 16. Contoh Test 4 (test4.txt)</p>	 <pre> ===== INPUT ===== Pastikan file txt berada pada folder test Masukkan nama file (ex: test.txt): test4.txt ===== OUTPUT ===== 50 22 22 IF YT IT HM 22 1, 1 1, 2 4, 2 4, 3 6, 3 6, 4 3, 4 5595.195055007935 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (ex: output.txt): hasil_test4.txt Solusi berhasil disimpan pada file hasil_test4.txt </pre> <p>Gambar 17. Hasil Luaran Test 4</p>  <pre> test > hasil_test4.txt 1 50 2 22 22 IF YT IT HM 22 3 1, 1 4 1, 2 5 4, 2 6 4, 3 7 6, 3 8 6, 4 9 3, 4 10 11 5595.195055007935 ms </pre> <p>Gambar 18. Hasil Luaran Test 4 (txt file)</p>
<p>Keterangan: Masukan dari file test4.txt. <i>Reward</i> maksimal yang diperoleh adalah 50 dengan sekuens 1) YT IT HM 22, 50 poin.</p>	

Tabel 5. Contoh Masukan 4 dari File Berekstensi txt

2.5. Contoh 5

Input	Output
 <pre> test > test5.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 </pre> <p>Gambar 19. Contoh Test 5 (test5.txt)</p>	 <pre> ===== INPUT ===== Pastikan file txt berada pada folder test Masukkan nama file (ex: test.txt): test5.txt ===== OUTPUT ===== 0 No Solution 0.0 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (ex: output.txt): hasil_test5.txt Solusi berhasil disimpan pada file hasil_test5.txt </pre> <p>Gambar 20. Hasil Luaran Test 5</p>  <pre> test > hasil_test5.txt 1 0 2 No Solution 3 4 0.0 ms </pre> <p>Gambar 21. Hasil Luaran Test 5 (<i>txt file</i>)</p>
<p>Keterangan: Masukan dari file test5.txt. <i>Reward</i> maksimal yang diperoleh adalah 0. Hal ini terjadi karena seluruh sekuens memiliki <i>reward</i> negatif, sehingga 0 menjadi <i>reward</i> yang paling optimal.</p>	

Tabel 6. Contoh Masukan 5 dari File Berekstensi txt

3. Input Melalui CLI (Terminal)

Pada masukan melalui CLI, matriks, sekuens, serta *reward* masing-masing sekuens diperoleh secara *random* (acak). Adapun rentang *reward* yang digunakan adalah [-50,50] dengan selang 5 (-50, -45, -40, ..., 0, ..., 40, 45, 50).

3.1. Contoh 1

Input	Output
<pre> ===== INPUT ===== Masukkan jumlah token unik : 5 Masukkan token unik : BD 1C 7A 55 E9 Masukkan ukuran buffer : 7 Masukkan lebar (kolom) dan tinggi (baris) matriks : 6 6 Masukkan jumlah sequence : 3 Masukkan ukuran maksimum sequence : 4 </pre>	<pre> ===== OUTPUT ===== Matriks of token: 1C BD E9 7A E9 1C 1C E9 55 BD BD BD 55 55 1C 1C 1C 1C 1C BD 7A E9 7A 7A 55 E9 55 55 BD 7A 55 7A 7A 55 BD 55 Sequence: 1C 55 1C Reward: -25 7A 7A E9 E9 Reward: 30 E9 BD 1C Reward: 15 45 BD 7A 7A E9 E9 BD 1C 2, 1 2, 6 3, 6 3, 1 5, 1 5, 2 1, 2 55.585384368896484 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (ex: output.txt): hasil_manual1.txt Solusi berhasil disimpan pada file hasil_manual1.txt </pre>
<p>Gambar 22. Contoh Test 6 (terminal)</p>	<p>Gambar 23. Hasil Luaran Test 6</p> <pre> test > 📄 hasil_manual1.txt 1 45 2 BD 7A 7A E9 E9 BD 1C 3 2, 1 4 2, 6 5 3, 6 6 3, 1 7 5, 1 8 5, 2 9 1, 2 10 11 55.585384368896484 ms </pre> <p>Gambar 24. Hasil Luaran Test 6 (<i>txt file</i>)</p>
<p>Keterangan: Masukan dari terminal. <i>Reward</i> maksimal yang diperoleh adalah 45 dengan sekuens 1) 7A 7A E9 E9, 30 poin; 2) E9 BD 1C, 15 poin.</p>	

Tabel 7. Contoh Masukan 1 dari Terminal

3.2. Contoh 2

Input	Output
<pre> ===== INPUT ===== Masukkan jumlah token unik : 10 Masukkan token unik : HM 1F 1T B0 BY TE 20 22 AG IL Masukkan ukuran buffer : 10 Masukkan lebar (kolom) dan tinggi (baris) matriks : 9 5 Masukkan jumlah sequence : 6 Masukkan ukuran maksimum sequence : 8 </pre> <p>Gambar 25. Contoh Test 7 (terminal)</p>	<pre> ===== OUTPUT ===== Matriks of token: BY TE B0 AG AG 1T BY B0 AG BY IL AG 22 AG 1F 20 22 TE 1T BY HM TE 20 TE AG 1T BY TE 1T HM TE 20 1F HM 1T 1T BY 1F B0 22 22 1T BY HM 22 Sequence: IL 20 1F Reward: -35 IL B0 BY IL IL IL AG Reward: -45 1T 1F IL 22 Reward: 50 HM B0 Reward: 45 1T HM BY 1T Reward: 20 B0 1T 20 22 TE BY Reward: -50 95 BY BY 1F 1T 1F IL 22 TE HM B0 1, 1 1, 2 6, 2 6, 5 2, 5 2, 2 4, 2 4, 3 3, 3 3, 5 41917.86289215088 ms Apakah ingin menyimpan solusi? (y/n): y Masukkan nama file (ex: output.txt): hasil_manual2.txt Solusi berhasil disimpan pada file hasil_manual2.txt </pre> <p>Gambar 26. Hasil Luaran Test 7</p> <pre> test > hasil_manual2.txt 1 95 2 BY BY 1F 1T 1F IL 22 TE HM B0 3 1, 1 4 1, 2 5 6, 2 6 6, 5 7 2, 5 8 2, 2 9 4, 2 10 4, 3 11 3, 3 12 3, 5 13 14 41917.86289215088 ms </pre> <p>Gambar 27. Hasil Luaran Test 7 (txt file)</p>
<p>Keterangan: Masukan dari terminal. <i>Reward</i> maksimal yang diperoleh adalah 95 dengan sekuens 1) 1T 1F IL 22, 50 poin; 2) HM B0, 45 poin.</p>	

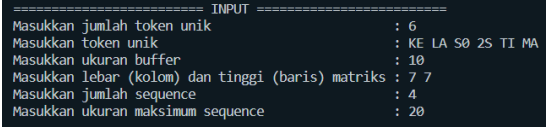
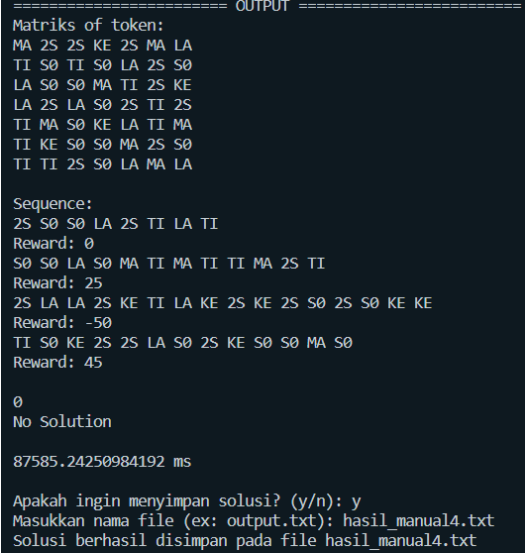
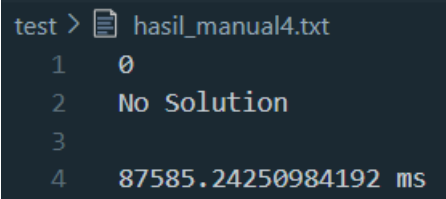
Tabel 8. Contoh Masukan 2 dari Terminal

3.3. Contoh 3

Input	Output
	
<p align="center">Gambar 28. Contoh Test 8 (terminal)</p>	<p align="center">Gambar 29. Hasil Luaran Test 8</p>  <p align="center">Gambar 30. Hasil Luaran Test 8 (<i>txt file</i>)</p>
<p>Keterangan: Masukan dari terminal. <i>Reward</i> maksimal yang diperoleh adalah 35 dengan sekuens 1) L1 L1 L1 GA, 35.</p>	

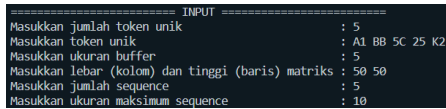
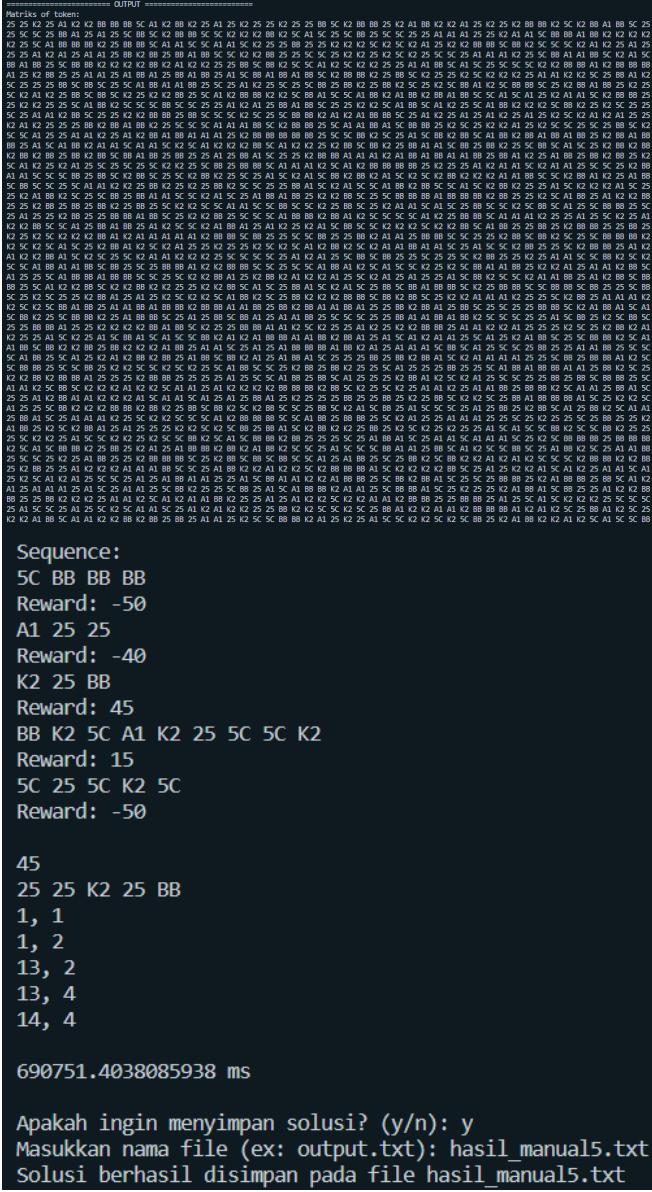
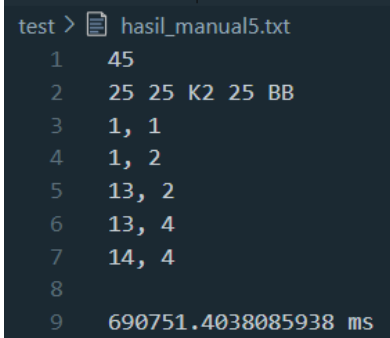
Tabel 9. Contoh Masukan 3 dari Terminal

3.4. Contoh 4

Input	Output
 <p>Gambar 31. Contoh Test 9 (terminal)</p>	 <p>Gambar 32. Hasil Luaran Test 9</p>  <p>Gambar 33. Hasil Luaran Test 9 (<i>txt file</i>)</p>
<p>Keterangan: Masukan dari terminal. <i>Reward</i> maksimal yang diperoleh adalah 0. Hal ini terjadi karena seluruh sekuens memiliki panjang yang melebihi ukuran <i>buffer</i>.</p>	

Tabel 10. Contoh Masukan 4 dari Terminal

3.5. Contoh 5

Input	Output
	
<p>Gambar 34.</p> <p>Contoh Test 10 (terminal)</p>	<p>Gambar 35. Hasil Luaran Test 10</p> 
<p>Keterangan: Masukan dari terminal. <i>Reward</i> maksimal yang diperoleh adalah 45 dengan sekuens 1) K2 25 BB, 45 poin.</p>	<p>Gambar 36. Hasil Luaran Test 10 (<i>txt file</i>)</p>

Tabel 11. Contoh Masukan 5 dari Terminal

DAFTAR PUSTAKA

<https://cyberpunk-hacker.com/> (Diakses pada 8 Februari 2024).

LAMPIRAN

1. Tampilan Awal Program

Link : https://github.com/Agil0975/Tucil1_13522006

2. 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		✓