

Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Semester II Tahun 2023/2024

Penyelesaian Cyberpunk 2077 Breach Protocol dengan
Algoritma Brute Force



Disusun oleh:
Bastian H. Suryapratama
13522034
K-02

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

I. Algoritma Brute Force

Algoritma yang digunakan dalam memecahkan puzzle **Cyberpunk 2077 Breach**

Protocol adalah algoritma brute force. Algoritma tersebut terdiri dari langkah-langkah sebagai berikut:

1. Dipilih sebuah token di baris pertama matriks sebagai titik mulai pencarian, kemudian simpan ke dalam buffer.
2. Pilih salah satu token lain di posisi **kolom** yang sama dengan token terakhir yang dipilih, kemudian simpan ke dalam buffer.
3. Cek reward yang diperoleh dengan kondisi buffer saat ini. Jika reward buffer saat ini lebih besar daripada reward terbesar buffer-buffer sebelumnya, simpan kondisi buffer dan koordinat yang diperoleh saat ini.
4. Pilih salah satu token lain di posisi **baris** yang sama dengan token terakhir yang dipilih, kemudian simpan ke dalam buffer.
5. Ulangi langkah ke-3.
6. Ulangi langkah ke-2 sampai ke-5. Jika pilihan token sudah habis, hapus terlebih dahulu token terakhir yang dipilih, kemudian ulangi langkah ke-2 sampai ke-5.
7. Jika tidak ada lagi kemungkinan buffer yang tersisa, pilih token lain di baris pertama matriks sebagai titik mulai pencarian, kemudian ulangi langkah ke-2 sampai ke-6.
8. Setelah semua kemungkinan buffer sudah ditelusuri, didapat hasil akhir berupa total reward, solusi buffer, serta koordinat-koordinat token di dalam solusi buffer.

II. Source Program

Source program dibuat dengan bahasa Python. Source program tersebut terdiri dari 2 file, yaitu main.py dan functions.py. File main.py merupakan program utama, sedangkan file functions.py berisi fungsi-fungsi yang dipakai di program utama.

1. main.py

```
1 import copy, time, random, os
2 from functions import *
3
4
5 # player choose input from file or randomized
6 while True:
7     input_choice = input("Apakah input dari file atau acak? (file/acak)\n")
8     print()
9     if input_choice in ["file", "acak"]:
10         break
11
12     print("Input tidak valid, coba lagi.\n")
13
14 if input_choice == "file":
15     # read input file
16     while True:
17         input_file_name = input("Masukkan nama file input: (file berada di dalam folder test)\n")
18         print()
19         if os.path.isfile("test/" + input_file_name):
20             break
21
22         print("File tidak ditemukan.\n")
23
24     with open("test/" + input_file_name, "r") as file_input:
25
26         # read buffer size
27         buffer_size = int(file_input.readline())
28
29         # read matrix dimension
30         matrix_dimension = file_input.readline().split()
31         matrix_height = int(matrix_dimension[1])
32         matrix_width = int(matrix_dimension[0])
33
34         # read matrix content
35         matrix = [file_input.readline().split() for i in range(matrix_height)]
36
37         # read number of sequences
38         num_of_sequences = int(file_input.readline())
39
40         # read sequences and its rewards
41         sequences = [[] for i in range(num_of_sequences)]
42         rewards = [0 for i in range(num_of_sequences)]
43         for i in range(num_of_sequences):
44             sequences[i] = file_input.readline().split()
45             rewards[i] = int(file_input.readline())
```

Gambar 2.1.1 Code main.py

```

1  else:
2      while True:
3          # enter the data needed for random generator
4          print("""Masukkan data berikut:
5              1. Banyak token unik
6              2. Token unik (dipisahkan dengan spasi)
7              3. Ukuran buffer
8              4. Ukuran kolom dan baris matriks (dipisahkan dengan spasi)
9              5. Banyak sekuens
10             6. Ukuran maksimal sekuens""")
11         num_of_token = int(input())
12         tokens = list(input().split(sep=" "))
13         buffer_size = int(input())
14         matrix_dimension = input().split(sep=" ")
15         matrix_height = int(matrix_dimension[1])
16         matrix_width = int(matrix_dimension[0])
17         num_of_sequences = int(input())
18         max_size_of_sequences = int(input())
19
20         print()
21
22         # check input validity
23         input_valid = True
24         if num_of_token <= 0:
25             input_valid = False
26             print("Banyak token harus lebih dari 0.")
27
28         if len(tokens) != num_of_token:
29             input_valid = False
30             print("Banyak token tidak sesuai dengan banyaknya token yang dimasukkan.")
31
32         if not all(map(is_token_valid, tokens)):
33             input_valid = False
34             print("Masukan token tidak valid, masing-masing harus 2 karakter alfanumerik.")
35
36         if not is_elements_unique(tokens):
37             input_valid = False
38             print("Masukan token tidak unik.")
39
40         if buffer_size < 2:
41             input_valid = False
42             print("Ukuran buffer minimal adalah 2.")
43
44         if matrix_height < 1 or matrix_width < 1:
45             input_valid = False
46             print("Ukuran baris dan kolom matriks minimal adalah 1.")
47
48         if num_of_sequences < 1:
49             input_valid = False
50             print("Banyak sekuens minimal adalah 1.")
51
52         if max_size_of_sequences < 2:
53             input_valid = False
54             print("Ukuran maksimal sekuens paling kecil adalah 2.")
55
56         if input_valid:
57             break
58
59         print()
60

```

Gambar 2.1.2 Code main.py

```

1  # random matrix, sequence, and reward generator
2  print("Generating matrix ...")
3  matrix = [random.choices(tokens, k=matrix_width) for i in range(matrix_height)]
4  print("Generating sequences and rewards ...")
5  sequences = [[] for i in range(num_of_sequences)]
6  rewards = [0 for i in range(num_of_sequences)]
7  for i in range(num_of_sequences):
8      sequence_size = random.randrange(2, max_size_of_sequences + 1)
9      sequence = random.choices(tokens, k=sequence_size)
10     if not sequence in sequences:
11         sequences[i] = sequence
12         rewards[i] = random.randrange(-50, 51, 10)
13
14     print()
15
16 # display matrix, sequences, and rewards
17 print("Matriks yang dihasilkan:")
18 for i in range(matrix_height):
19     for j in range(matrix_width):
20         print(matrix[i][j], end="")
21         if j < matrix_width - 1:
22             print(end=" ")
23         else:
24             print()
25
26     print()
27
28     print("Sequence dan reward yang dihasilkan:")
29     for i in range(num_of_sequences):
30         for j in range(len(sequences[i])):
31             print(sequences[i][j], end="")
32             if j < len(sequences[i]) - 1:
33                 print(end=" ")
34             else:
35                 print()
36
37         print(rewards[i])
38
39     print()
40

```

Gambar 2.1.3 Code main.py

```

1  # find the solution
2
3  start_time = time.time()
4
5  # solution variables
6  total_reward = 0
7  buffer_solution = []
8  coordinates = []
9
10 # temporary variables
11 temp_reward = 0
12 temp_buffer = []
13 temp_coordinates = []
14
15 # process
16 MIN_BUFFER_SIZE = 2
17 for start_position in range(0, matrix_width):
18     temp_buffer.append(matrix[0][start_position])
19     temp_coordinates.append([0, start_position])
20     i = 2
21     j = 0
22     while MIN_BUFFER_SIZE <= i <= buffer_size:
23         if is_even(i):
24             go_longer = False
25             while j < matrix_height and not go_longer:
26                 last_x_coordinate = temp_coordinates[len(temp_coordinates) - 1][1]
27                 if not [j, last_x_coordinate] in temp_coordinates:
28                     temp_buffer.append(matrix[j][last_x_coordinate])
29                     temp_coordinates.append([j, last_x_coordinate])
30                     temp_reward = calculate_reward(temp_buffer, sequences, rewards)
31                     if temp_reward > total_reward:
32                         total_reward = temp_reward
33                         buffer_solution = copy.deepcopy(temp_buffer)
34                         coordinates = copy.deepcopy(temp_coordinates)
35
36                 if i < buffer_size:
37                     i += 1
38                     j = 0
39                     go_longer = True
40             else:
41                 temp_buffer.pop()
42                 temp_coordinates.pop()
43                 j += 1
44
45         else:
46             j += 1
47
48     if not go_longer:
49         i -= 1
50         j = temp_coordinates[len(temp_coordinates) - 1][1] + 1
51         temp_buffer.pop()
52         temp_coordinates.pop()
53

```

Gambar 2.1.4 Code main.py

```

1         else:
2             go_longer = False
3             while j < matrix_width and not go_longer:
4                 last_y_coordinate = temp_coordinates[len(temp_coordinates) - 1][0]
5                 if not [last_y_coordinate, j] in temp_coordinates:
6                     temp_buffer.append(matrix[last_y_coordinate][j])
7                     temp_coordinates.append([last_y_coordinate, j])
8                     temp_reward = calculate_reward(temp_buffer, sequences, rewards)
9                     if temp_reward > total_reward:
10                         total_reward = temp_reward
11                         buffer_solution = copy.deepcopy(temp_buffer)
12                         coordinates = copy.deepcopy(temp_coordinates)
13
14                     if i < buffer_size:
15                         i += 1
16                         j = 0
17                         go_longer = True
18                     else:
19                         temp_buffer.pop()
20                         temp_coordinates.pop()
21                         j += 1
22
23             else:
24                 j += 1
25
26             if not go_longer:
27                 i -= 1
28                 j = temp_coordinates[len(temp_coordinates) - 1][0] + 1
29                 temp_buffer.pop()
30                 temp_coordinates.pop()
31
32 # coordinates formatting
33 coordinates = [[b + 1, a + 1] for [a, b] in coordinates]
34
35 end_time = time.time()
36

```

Gambar 2.1.5 Code main.py

```

1  # display result to terminal
2  print("Hasil pencarian:")
3
4  print(total_reward)
5  print(*buffer_solution, sep=" ")
6  for i in coordinates:
7      print(f"{i[0]}, {i[1]}")
8
9  print()
10
11 duration = end_time - start_time
12 print(f"{duration * 1000} ms\n")
13
14 # ask user to save the result
15 while True:
16     save_solution = input("Apakah ingin menyimpan solusi? (y/n)\n")
17     print()
18     if save_solution in ["y", "n"]:
19         break
20
21     print("Input tidak valid.\n")
22
23 if save_solution == "y":
24     output_file_name = input("Masukkan nama file output: (file berada di dalam folder test)\n")
25     with open("test/" + output_file_name, "w") as file_output:
26
27         # write total reward
28         file_output.write(str(total_reward) + "\n")
29
30         # write buffer solution
31         for i in range(len(buffer_solution)):
32             file_output.write(buffer_solution[i])
33             if i < len(buffer_solution) - 1:
34                 file_output.write(" ")
35             else:
36                 file_output.write("\n")
37
38         # write coordinates
39         for i in coordinates:
40             file_output.write(f"{i[0]}, {i[1]}\n")
41
42         # write time execution
43         file_output.write(f"\n{duration * 1000} ms\n")
44


```

Gambar 2.1.6 Code main.py

2. functions.py

```
1 # functions for main algorithm
2
3
4 def is_even(x):
5     return x % 2 == 0
6
7
8 def is_sublist(sublist, list):
9     if sublist == list:
10         return True
11     elif sublist == []:
12         return True
13     elif len(sublist) > len(list):
14         return False
15     else:
16         i = 0
17         result = False
18         while i < len(list) and not result:
19             if sublist[0] == list[i]:
20                 j = 1
21                 check = True
22                 while j < len(sublist) and i + j < len(list) and check:
23                     if sublist[j] == list[i + j]:
24                         j += 1
25                     else:
26                         check = False
27
28                 if j == len(sublist):
29                     result = True
30
31             if not result:
32                 i += 1
33
34     return result
35
36
```

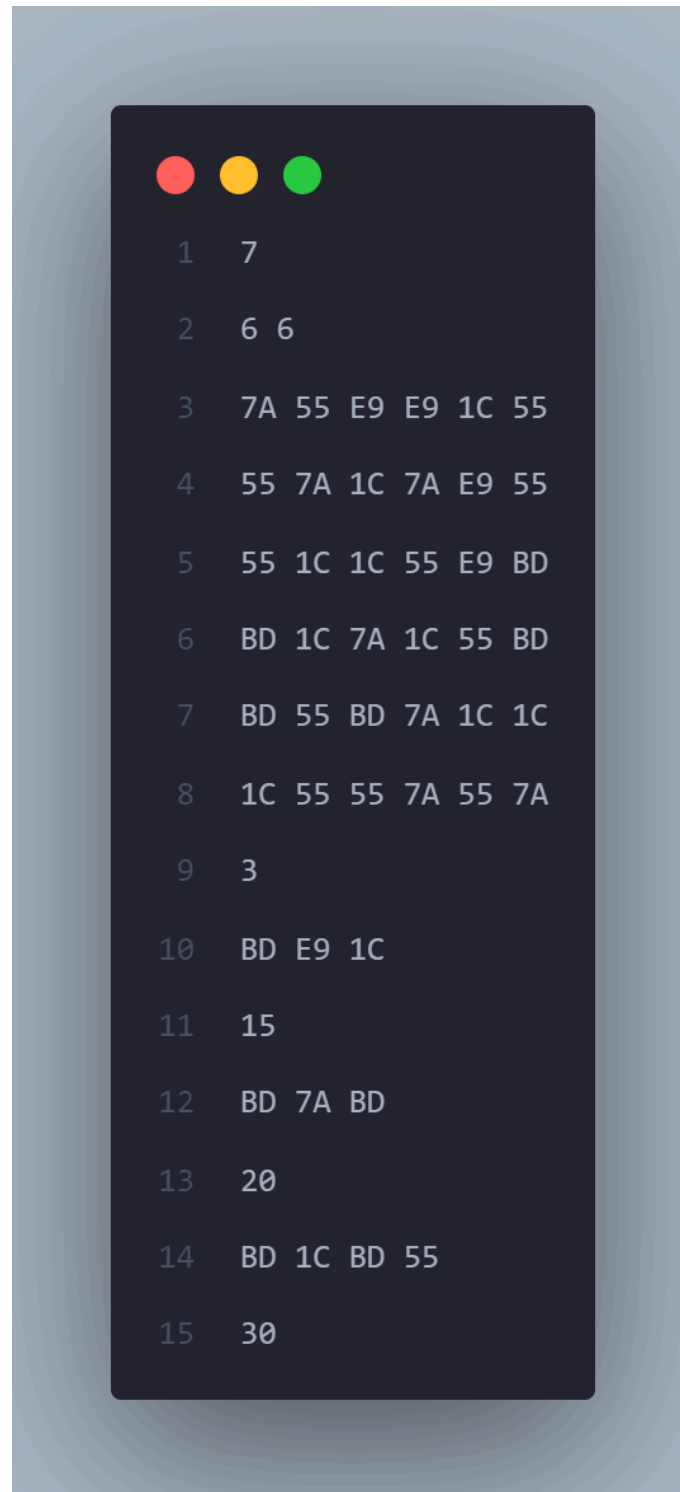
Gambar 2.2.1 Code functions.py



```
1  def calculate_reward(buffer, sequence_list, reward_list):
2      reward = 0
3      for i in range(len(sequence_list)):
4          if is_sublist(sequence_list[i], buffer):
5              reward += reward_list[i]
6
7      return reward
8
9
10 def is_token_valid(token):
11     if len(token) != 2:
12         return False
13     return token.isalnum()
14
15
16 def is_elements_unique(list):
17     stop = False
18     i = 0
19     while i < len(list) - 1 and not stop:
20         j = i + 1
21         while j < len(list) and not stop:
22             if list[i] == list[j]:
23                 stop = True
24             else:
25                 j += 1
26
27         if not stop:
28             i += 1
29
30     return not stop
31
```

Gambar 2.2.2 Code functions.py

III. Tangkapan Layar Input/Output



Gambar 3.1.1 Input Kasus Ke-1 dengan .txt

```
$ python src/main.py
Apakah input dari file atau acak? (file/acak)
file

Masukkan nama file input: (file berada di dalam folder test)
input1.txt

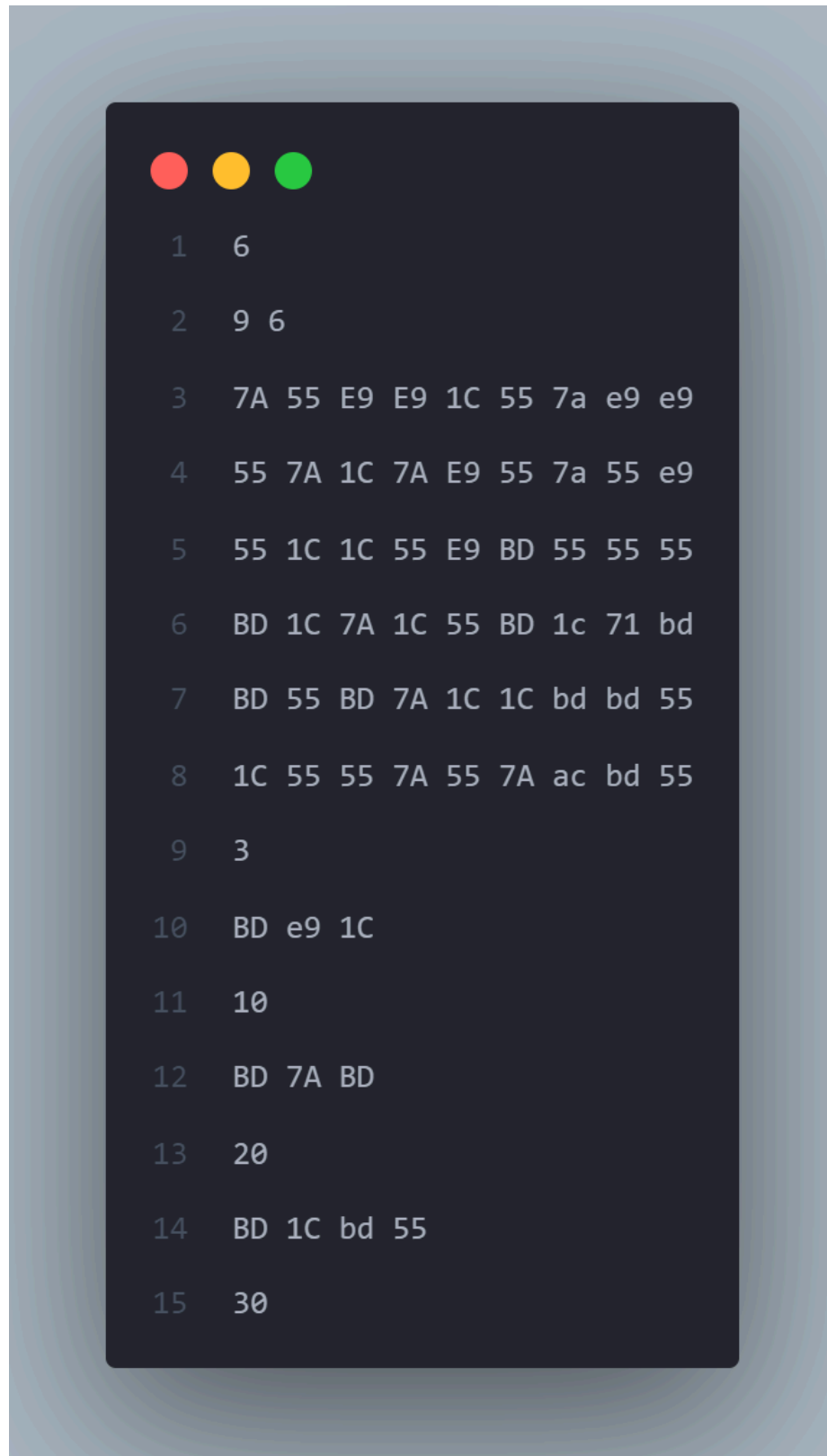
Hasil pencarian:
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

516.0372257232666 ms

Apakah ingin menyimpan solusi? (y/n)
y

Masukkan nama file output: (file berada di dalam folder test)
output1.txt
```

Gambar 3.1.2 Output Kasus Ke-1



Gambar 3.2.1 Input Kasus Ke-2 dengan .txt

```
$ python src/main.py
Apakah input dari file atau acak? (file/acak)
file

Masukkan nama file input: (file berada di dalam folder test)
input3.txt

File tidak ditemukan.

Masukkan nama file input: (file berada di dalam folder test)
input2.txt

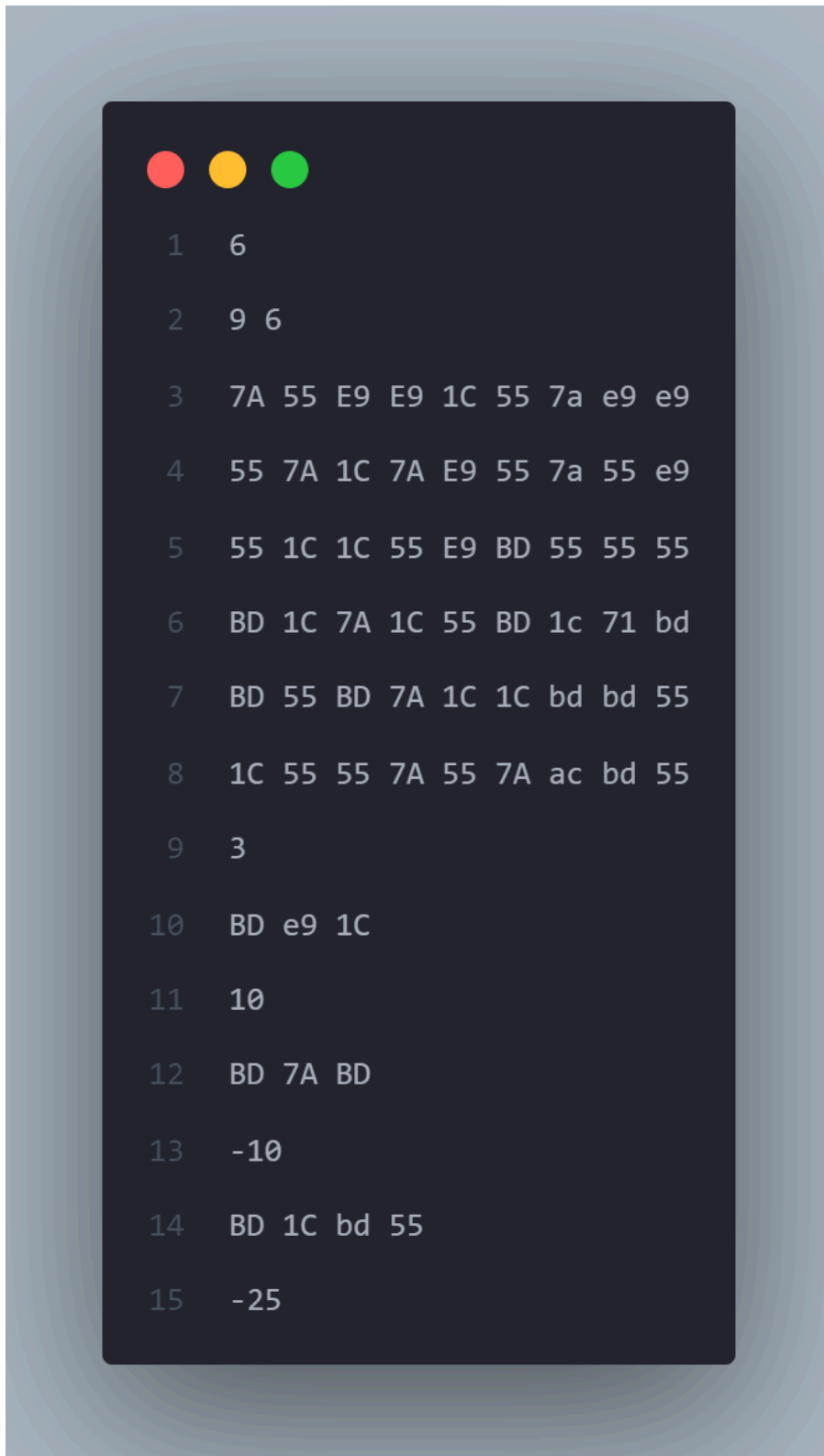
Hasil pencarian:
30
7A 55 BD 1C bd 55
1, 1
1, 3
6, 3
6, 5
7, 5
7, 3

386.03997230529785 ms

Apakah ingin menyimpan solusi? (y/n)
y

Masukkan nama file output: (file berada di dalam folder test)
output2.txt
```

Gambar 3.2.2 Output Kasus Ke-2



Gambar 3.3.1 Input Kasus Ke-3 dengan .txt

```
$ python src/main.py
Apakah input dari file atau acak? (file/acak)
file

Masukkan nama file input: (file berada di dalam folder test)
input3.txt

Hasil pencarian:
0

391.02816581726074 ms

Apakah ingin menyimpan solusi? (y/n)
y

Masukkan nama file output: (file berada di dalam folder test)
output3.txt
```

Gambar 3.3.2 Output Kasus Ke-3


```

$ python src/main.py
Apakah input dari file atau acak? (file/acak)
acak

Masukkan data berikut:
    1. Banyak token unik
    2. Token unik (dipisahkan dengan spasi)
    3. Ukuran buffer
    4. Ukuran kolom dan baris matriks (dipisahkan dengan spasi)
    5. Banyak sekuens
    6. Ukuran maksimal sekuens

4
B3 A1 88 R5
5
18 20
4
4

Generating matrix ...
Generating sequences and rewards ...

Matriks yang dihasilkan:
B3 88 A1 R5 R5 88 B3 A1 88 B3 88 A1 88 B3 R5 B3 R5 B3
A1 88 88 A1 88 A1 B3 R5 88 B3 R5 A1 A1 R5 88 R5 A1 R5
88 B3 88 R5 R5 B3 88 A1 B3 88 A1 R5 88 A1 88 B3 B3 B3
B3 B3 88 88 R5 A1 A1 A1 A1 R5 A1 R5 R5 B3 88 88 88
88 B3 88 B3 A1 A1 R5 B3 A1 A1 B3 A1 88 A1 R5 A1 R5 B3
88 B3 A1 R5 R5 B3 88 A1 R5 A1 R5 B3 A1 R5 A1 88 A1 88
A1 88 A1 R5 R5 A1 A1 88 B3 R5 R5 B3 R5 A1 A1 88 R5 A1
R5 R5 88 88 R5 88 A1 R5 R5 B3 88 R5 88 88 B3 A1 88 A1
B3 B3 B3 R5 R5 88 B3 R5 R5 A1 88 R5 B3 A1 A1 88 A1 B3
R5 A1 R5 B3 R5 R5 B3 R5 A1 B3 A1 R5 88 A1 B3 B3 B3 A1
R5 88 R5 A1 B3 B3 R5 A1 A1 R5 B3 R5 R5 A1 R5 88 B3 R5
A1 A1 R5 88 R5 88 B3 A1 A1 A1 A1 B3 A1 R5 B3 88 A1 A1
R5 A1 A1 A1 A1 A1 B3 B3 B3 B3 R5 A1 A1 R5 B3 B3 88 A1
A1 A1 88 88 R5 A1 B3 R5 88 B3 B3 88 B3 88 B3 A1 88 B3
88 88 R5 R5 A1 A1 88 R5 R5 88 A1 R5 88 A1 A1 R5 B3 A1
A1 88 A1 B3 R5 R5 B3 A1 B3 B3 88 A1 B3 88 R5 R5 B3 B3
B3 B3 R5 A1 B3 A1 B3 A1 A1 R5 88 R5 B3 A1 B3 B3 R5 88
88 A1 R5 B3 A1 R5 B3 88 A1 B3 R5 R5 A1 88 B3 R5 B3 88
A1 B3 B3 B3 B3 A1 R5 B3 A1 B3 88 88 88 R5 88 B3 88 88
R5 B3 B3 A1 A1 B3 R5 B3 A1 B3 A1 A1 B3 B3 B3 R5 B3 B3
Sequence dan reward yang dihasilkan:

```

Gambar 3.4.1 Input Kasus Ke-4 (Random) dan Matriks yang Terbentuk

```
B3 B3 B3 R5 R5 88 B3 R5 R5 A1 88 R5 B3 A1 A1 88 A1 B3
R5 A1 R5 B3 R5 R5 B3 R5 A1 B3 A1 R5 88 A1 B3 B3 B3 A1
R5 88 R5 A1 B3 B3 R5 A1 A1 R5 B3 R5 R5 A1 R5 88 B3 R5
A1 A1 R5 88 R5 88 B3 A1 A1 A1 A1 B3 A1 R5 B3 88 A1 A1
R5 A1 A1 A1 A1 A1 B3 B3 B3 B3 R5 A1 A1 R5 B3 B3 88 A1
A1 A1 88 88 R5 A1 B3 R5 88 B3 B3 88 B3 88 B3 A1 88 B3
88 88 R5 R5 A1 A1 88 R5 R5 88 A1 R5 88 A1 A1 R5 B3 A1
A1 88 A1 B3 R5 R5 B3 A1 B3 B3 88 A1 B3 88 R5 R5 B3 B3
B3 B3 R5 A1 B3 A1 B3 A1 A1 R5 88 R5 B3 A1 B3 B3 R5 88
88 A1 R5 B3 A1 R5 B3 88 A1 B3 R5 R5 A1 88 B3 R5 B3 88
A1 B3 B3 B3 B3 A1 R5 B3 A1 B3 88 88 88 R5 88 B3 88 88
R5 B3 B3 A1 A1 B3 R5 B3 A1 B3 A1 A1 B3 B3 B3 R5 B3 B3
```

Sequence dan reward yang dihasilkan:

A1 B3 A1 B3

0

88 88 A1

40

R5 88 A1

30

A1 A1

40

Hasil pencarian:

80

B3 88 88 A1 A1

1, 1

1, 3

3, 3

3, 1

8, 1

11718.09720993042 ms

Apakah ingin menyimpan solusi? (y/n)

y

Masukkan nama file output: (file berada di dalam folder test)
output4.txt

Gambar 3.4.2 Sequence dan Reward yang Terbentuk serta Output Kasus Ke-4

```

$ python src/main.py
Apakah input dari file atau acak? (file/acak)
acak

Masukkan data berikut:
    1. Banyak token unik
    2. Token unik (dipisahkan dengan spasi)
    3. Ukuran buffer
    4. Ukuran kolom dan baris matriks (dipisahkan dengan spasi)
    5. Banyak sekuens
    6. Ukuran maksimal sekuens

3
aa AA aA
4
20 21
2
3

Generating matrix ...
Generating sequences and rewards ...

Matriks yang dihasilkan:
aA aa AA aa aA aA aA aa aa aA aA aA aa AA AA AA aa aa AA AA
AA aa aa aA AA aA aa aA aA AA aA aA aA aa AA AA aA AA aa aa
AA AA aa AA aA aa aa aa AA AA aA AA aa aA aA aA AA aa aA AA
AA aa aA AA aA aa AA AA AA aA aA aa AA aA aA aa aa AA AA aa
AA aa AA aa aa aA aA aA AA AA aA aa aa aA AA aA AA aA AA aA
aA aA AA aa aA aA AA aA aa aA AA AA aA aa aA AA aa AA aA AA
AA aA aA aa aA aA aa aa aA AA aa aa aa aa aA aa AA AA AA AA
aa AA aA aa aA aa AA AA AA AA aa aA aa aA aa aa aA AA aa aA
AA aa aa aA AA AA AA aA AA aA aa aa aa aA aa aa aa aA aa AA
aA aA AA aa AA aa aa aA AA aa AA aa aA aA aa aA AA aA AA aa
aA AA aA AA aa aa aa aA AA aa AA aa aa aA AA aa AA AA AA aa
aA aa aa AA aA AA aA aa AA AA aA aa aA aA aA aa aA aA aA
aa aa aA aA aA aA aa aa AA aA aA AA aa aa aa aA AA aa AA aA
aA aa aa aa aA AA AA aA aa AA aA aA aa aa aA aA aa aa AA aA
aa aa aA aA AA aa aA aA AA aA aA aa AA aa AA AA aa aA aa aa
aa aa aa aa aa aa aA aa AA aa aA AA AA aa aA AA aa aA aA aA
aA AA aa AA AA AA aA aa AA aA AA aa aa AA aa AA aa aa AA aa
aA aA aA aa aA aA aA aA aa aA aA aa aA aA AA aA aa AA AA
aA AA AA aa aa AA aa aA aa aa AA aA aa aA AA aa AA aa AA AA
aA aA aa AA aa aA aa AA aa AA AA aA AA aA aa AA aA AA aA

```

Sequence dan reward yang dihasilkan:

Gambar 3.5.1 Input Kasus Ke-5 (Random) dan Matriks yang Terbentuk

```
aA aa aa aa aA AA AA aA aa AA aA AA aa aa aA aA aa aa AA aA
aa aa aA aA AA aa aA aA AA aA aA aa AA aa AA AA aa aA aa aa
aa aa aa aa aa aa aA aa AA aa aA AA AA aa aA AA aa aA aA aA
aA AA aa AA AA AA aA aa AA aA AA aa aa AA aa AA aa aa AA aa
aA aA aA aa aA aA aA aA aa aA aA aa aA aA AA aA aa AA AA
aA AA AA aa aa AA aa aA aa aa AA aA aa aA AA aa AA aa AA AA
aA aA aa AA aa aA aa AA aa AA AA aA AA aA aa AA aA AA aA
```

Sequence dan reward yang dihasilkan:

aa AA

-30

aA AA AA

-30

Hasil pencarian:

0

555.037260055542 ms

Apakah ingin menyimpan solusi? (y/n)

y

Masukkan nama file output: (file berada di dalam folder test)
output5.txt

Gambar 3.5.2 Sequence dan Reward yang Terbentuk serta Output Kasus Ke-5

```
$ python src/main.py
Apakah input dari file atau acak? (file/acak)
acak

Masukkan data berikut:
    1. Banyak token unik
    2. Token unik (dipisahkan dengan spasi)
    3. Ukuran buffer
    4. Ukuran kolom dan baris matriks (dipisahkan dengan spasi)
    5. Banyak sekuens
    6. Ukuran maksimal sekuens

4
abc de 5t
1
-1 5
-1
-1

Banyak token tidak sesuai dengan banyaknya token yang dimasukkan.
Masukan token tidak valid, masing-masing harus 2 karakter alfanumerik.
Ukuran buffer minimal adalah 2.
Ukuran baris dan kolom matriks minimal adalah 1.
Banyak sekuens minimal adalah 1.
Ukuran maksimal sekuens paling kecil adalah 2.

Masukkan data berikut:
    1. Banyak token unik
    2. Token unik (dipisahkan dengan spasi)
    3. Ukuran buffer
    4. Ukuran kolom dan baris matriks (dipisahkan dengan spasi)
    5. Banyak sekuens
    6. Ukuran maksimal sekuens
```

Gambar 3.6.1 Input Kasus Ke-6 (Random) dan Outputnya

Masukkan data berikut:

1. Banyak token unik
2. Token unik (dipisahkan dengan spasi)
3. Ukuran buffer
4. Ukuran kolom dan baris matriks (dipisahkan dengan spasi)
5. Banyak sekuens
6. Ukuran maksimal sekuens

```
4
dD Dd DD dd
5
10 10
8
3
```

Generating matrix ...
Generating sequences and rewards ...

Matriks yang dihasilkan:

```
dD dd dD Dd dd dD DD dd DD
dd dd dD DD DD Dd DD Dd Dd Dd
dd Dd DD Dd dd DD dd dd Dd Dd
dd dd DD dd dD dd Dd DD dd dd
DD dD DD Dd dd dd dd dd dD dd
dD dd Dd DD DD dD DD DD Dd DD
DD Dd Dd dd DD DD Dd Dd dd dD
dD Dd Dd dd dD dD dd DD DD Dd
DD dd Dd DD Dd Dd dd Dd Dd dd
dD dd Dd Dd Dd dD Dd Dd dd dD
```

Sequence dan reward yang dihasilkan:

```
Dd Dd
-50
dD DD DD
-40
dD dD
40
dD DD
-10
dD dd
-10
Dd dd
```

Gambar 3.7.1 Input Kasus Ke-7 (Random) dan Matriks yang Terbentuk

Sequence dan reward yang dihasilkan:

Dd Dd

-50

dD DD DD

-40

dD dD

40

dD DD

-10

dD dd

-10

Dd dd

10

DD DD dD

-50

dd dd

-10

Hasil pencarian:

50

dD dD dD Dd dd

1, 1

1, 6

6, 6

6, 2

1, 2

745.0542449951172 ms

Apakah ingin menyimpan solusi? (y/n)

y

Masukkan nama file output: (file berada di dalam folder test)
output7.txt

Gambar 3.7.2 Sequence dan Reward yang Terbentuk serta Output Kasus Ke-7

IV. Repository dan Lampiran

Link repository: https://github.com/bastianhs/Tucil1_13522034

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	tidak perlu	
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		✓