

**LAPORAN TUGAS KECIL 1 IF2211  
STRATEGI ALGORITMA  
SEMESTER II TAHUN 2023/2024**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



**Disusun oleh :**

**Andhita Naura Hariyanto - 13522060**

**K2**

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2023**

## **DAFTAR ISI**

DAFTAR ISI	2
BAB 1	3
BAB 2	6
BAB 3	8
BAB 4	16
LAMPIRAN	27

## BAB I

### DESKRIPSI MASALAH



Gambar 1 Permainan Breach Protocol

(Sumber: <https://cyberpunk.fandom.com/wiki/Quickhacking>)

**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

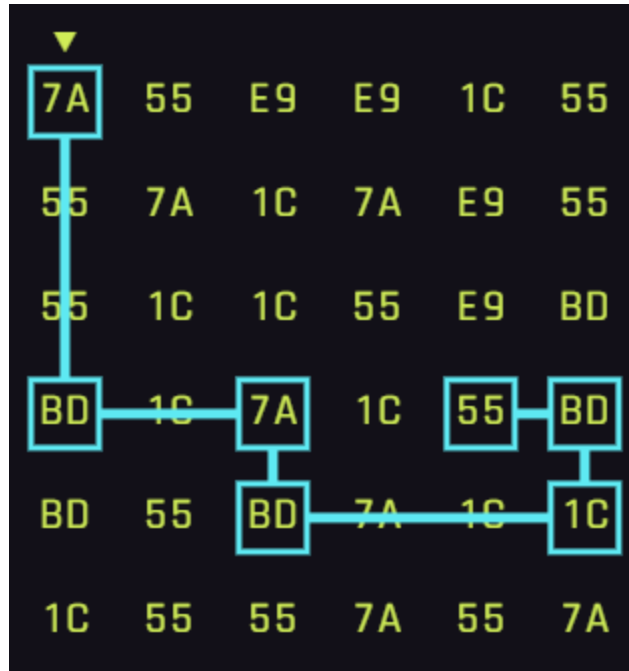
7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

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 2 Contoh Solusi

(Sumber: <https://cyberpunk-hacker.com/>)

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

## BAB II

### ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN CYBERPUNK 2077 BEACH PROTOCOL

Breach Protocol Cyberpunk 2077 dapat diselesaikan menggunakan Algoritma Brute Force dengan mencoba semua kemungkinan pola token yang valid berdasarkan masukan matriks token dan sekuens dari pengguna. Pola token yang dapat memberikan nilai reward maksimal merupakan solusi dari Breach Protocol pada permainan Cyberpunk 2077.

Langkah penyelesaian Breach Protocol Cyberpunk 2077 menggunakan Algoritma Brute Force :

1. Memetakan seluruh kemungkinan pola token dengan struktur data *tree*, spesifiknya *n-ary tree*, secara rekursif. Penggunaan *n-ary tree* didasarkan pada kebutuhan pemetaan secara menyeluruh dan terstruktur dari pola-pola token yang akan terbentuk berdasarkan matriks masukan dan ukuran maksimal buffer yang ditentukan. Kedalaman dari pohon yang terbentuk akan sama dengan ukuran maksimal buffer.

Basis : Jika level pohon sudah sama besar dengan ukuran buffer, simpan pohon

Rekursens : Menggabungkan tiap token dengan child sesuai dengan alur pergerakan matriks token, yaitu bergantian secara vertikal dan horizontal. Apabila pergerakan yang sedang dilakukan adalah pergerakan vertikal, token-token yang akan menjadi child dari token saat ini adalah token yang berada dalam satu kolom yang sama dengan urutan penggabungan dimulai dari token posisi teratas. Apabila pergerakan yang sedang dilakukan adalah pergerakan horizontal, token-token yang akan menjadi child dari token saat ini adalah token yang berada dalam satu baris yang sama dengan urutan penggabungan dimulai dari token posisi ter kiri.

2. Menyimpan seluruh kemungkinan pola token dari *n-ary tree* yang terbentuk dengan metode rekursif dimulai dari pola token dengan panjang 2 token hingga sepanjang ukuran buffer. Pada penyimpanan kemungkinan pola ini, token-token yang dalam penelusuran pola ini sudah pernah terpilih, akan berubah value atribut `isSelected`-nya dari `False` menjadi `True`. Hal ini berakibat token tidak akan dapat dipilih kembali dalam satu jalan

penelusuran yang sama. Ketika jalan penelusuran sudah selesai dilakukan, nilai atribut `isSelected` token akan kembali diubah menjadi `False`.

Basis : Pola token akan disimpan jika node yang terakhir dipilih sudah tidak memiliki child lagi

Rekursensi : Melakukan penggabungan (meng-*append*) child dari setiap tree dengan konsep *n-ary search tree*

3. Menginisialisasi sebuah matriks yang seukuran dengan matriks pola token yang mungkin dengan nilai 0 untuk menyimpan penghitungan reward dari setiap pola token. Kemudian, melakukan pattern matching antara seluruh pola yang terdapat pada matriks yang berisi pola-pola yang mungkin dari token dengan sekuens-sekuens yang ada. Apabila pola token mengandung sekuens (sekuens merupakan subset pola token secara terurut), matriks penghitungan reward akan bertambah nilainya sesuai dengan nilai reward pada sekuens yang cocok di indeks yang sesuai dengan indeks pola token pada matriks pola token.
4. Membandingkan seluruh elemen pada matriks penghitung nilai reward. Kemudian, akan didapatkan pola matriks paling optimal, yaitu pola matriks dengan panjang token sesedikit mungkin dan reward sebesar mungkin. Apabila tidak ada pola token yang cocok dengan sekuens, akan ditampilkan *output* pernyataan tidak adanya kecocokan matriks token dengan sekuens.

## **BAB III**

### **IMPLEMENTASI PROGRAM DENGAN PYTHON**

Implementasi program dilakukan dengan bahasa python pada file main.py yang terdapat di folder src. Library yang digunakan dalam pengimplementasi program ini adalah adalah time dan random.

1. Main.py



```

1 import time
2 from tokens import *
3 from sekuens import *
4 from tree import *
5 from randomize import *
6
7 def main() :
8     choice = input("Input matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) : ")
9
10    while choice.lower() != "cli" and choice.lower() != "txt" :
11        choice = input("Masukan tidak sesuai!\nInput matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) : ")
12
13    if choice.lower() == "cli" :
14        while True:
15            try:
16                unique_token = int(input("Masukkan jumlah token unik: "))
17                if unique_token <= 0:
18                    print("Jumlah token unik harus berupa integer positif.")
19                    continue
20                break
21            except ValueError:
22                print("Input harus berupa bilangan bulat. Coba lagi.")
23
24        tokens_input = []
25        print("Masukan token-token (dipisahkan oleh spasi): ")
26        tokens_string = input().strip()
27        tokens_list = tokens_string.split()
28
29        while len(tokens_list) != unique_token:
30            print("Jumlah token yang dimasukkan tidak sesuai dengan yang diminta.")
31            tokens_string = input("Masukan ulang token-token (dipisahkan oleh spasi): ").strip()
32            tokens_list = tokens_string.split()
33
34        for a in range(len(tokens_list)):
35            while len(tokens_list[a]) != 2 or not is_alphanumeric(tokens_list[a][0]) or not is_alphanumeric(tokens_list[a][1]) :
36                tokens_list[a] = input("Token harus berupa 2 karakter alfanumerik! Input ulang token ke-{:a}: ")
37            tokens_input.append(tokens_list[a])
38
39        while True:
40            try:
41                buffer_size = int(input("Masukkan ukuran buffer: "))
42                if buffer_size <= 0:
43                    print("Ukuran buffer harus berupa integer positif.")
44                    continue
45                break
46            except ValueError:
47                print("Input harus berupa bilangan bulat. Coba lagi.")
48
49        while True:
50            try:
51                matrix_width = int(input("Masukkan lebar matriks: "))
52                if matrix_width <= 0:
53                    print("Lebar matriks harus berupa integer positif.")
54                    continue
55                break
56            except ValueError:
57                print("Input harus berupa bilangan bulat. Coba lagi.")
58
59        while True:
60            try:
61                matrix_height = int(input("Masukkan tinggi matriks: "))
62                if matrix_height <= 0:
63                    print("Tinggi matriks harus berupa integer positif.")
64                    continue
65                break
66            except ValueError:
67                print("Input harus berupa bilangan bulat. Coba lagi.")
68
69        while True:
70            try:
71                sekuens_count = int(input("Masukkan jumlah sekuens yang ingin dibentuk : "))
72                if sekuens_count <= 0:
73                    print("Jumlah sekuens yang ingin dibentuk harus berupa integer positif.")
74                    continue
75                break
76            except ValueError:
77                print("Input harus berupa bilangan bulat. Coba lagi.")
78
79        while True:
80            try:
81                sekuens_size = int(input("Masukkan ukuran maksimal sekuens: "))
82                if sekuens_size <= 0:
83                    print("Ukuran maksimal sekuens harus berupa integer positif.")
84                    continue
85                break
86            except ValueError:
87                print("Input harus berupa bilangan bulat. Coba lagi.")
88
89        input_matrix = random_matrix(matrix_height, matrix_width, tokens_input)
90        print("Matrix : ")
91        for i in range(len(input_matrix)) :
92            for j in range(len(input_matrix[i])) :
93                print(input_matrix[i][j], end=" ")
94            print()
95
96        sekuens_matrix = random_sekuens(sekuens_count, sekuens_size, tokens_input)
97        print("Sekuens : ")
98        for a in range(len(sekuens_matrix)) :
99            print(a+1, ". ", sekuens_matrix[a])
100
101        reward_array = []
102        print("Reward sekuens : ")
103        for i in range(len(sekuens_matrix)) :
104            random_value = random.randint(-50,50)
105            if i > 0 :
106                while (random_value == reward_array[i-1]) :
107                    random_value = random.randint(-50,50)
108            reward_array.append(random_value)
109        for b in range(len(reward_array)) :
110            print(b+1, ". ", reward_array[b])

```

```

1  elif choice.lower() == "tst" :
2      input_file = input("Beriikan nama file : ")
3
4      sekans_count = 0
5      buffer_size = 0
6
7      file_path = './test/' + input_file
8      with open(file_path, "r") as my_file:
9          # your code to read the file
10         i = 0
11         for line in my_file:
12             i += 1
13             if i == 1:
14                 try:
15                     buffer_size = len(line.strip())
16                     if buffer_size <= 1:
17                         raise ValueError("Input buffer harus berupa bilangan bulat lebih besar dari 1.")
18                     except ValueError:
19                         print("Input Buffer harus berupa bilangan bulat lebih besar dari 1. Coba lagi!")
20                     main()
21                 elif i == 2:
22                     status1 = True
23                     while i < len(line.strip()) and line.strip()[i] != " " and status1:
24                         if len(line.strip()[i]) >= 0:
25                             if i % 2 == 0:
26                                 matrix_width = line.strip()[i]
27                             else:
28                                 matrix_width += line.strip()[i]
29                         else:
30                             status1 = False
31                             j += 1
32                     try:
33                         matrix_width = len(matrix_width)
34                         if matrix_width < 0:
35                             raise ValueError("Input lebar matriks harus berupa bilangan bulat lebih besar dari 1.")
36                         except ValueError:
37                             print("Input lebar matriks harus berupa bilangan bulat. Coba lagi!")
38                             main()
39                         while j < len(line.strip()) and line.strip()[j] == " ":
40                             j += 1
41                         k = j
42                         status2 = True
43                         while j < len(line.strip()) and line.strip()[j] != " " and status2:
44                             if len(line.strip()[j]) >= 0:
45                                 if j % 2 == 0:
46                                     matrix_height = line.strip()[j]
47                                 elif j % 2 == 1:
48                                     matrix_height += line.strip()[j]
49                             else:
50                                 status2 = False
51                                 j += 1
52                     try:
53                         matrix_height = len(matrix_height)
54                         if matrix_height < 0:
55                             raise ValueError("Input tinggi matriks harus berupa bilangan bulat lebih besar dari 1.")
56                         except ValueError:
57                             print("Input tinggi matriks harus berupa bilangan bulat. Coba lagi!")
58                             main()
59                     input_matrix = [[0 for a in range (matrix_width)] for b in range (matrix_height)]
60                     if i % 3 <= 1 < i + 3 + matrix_height:
61                         try:
62                             a = 0
63                             while input_matrix[i - 3][matrix_width - 1] == "" or len(input_matrix[i - 3][matrix_width - 3]) < 2:
64                                 if line.strip()[i] != " ":
65                                     if is_alphanumeric(line.strip()[i]):
66                                         input_matrix[i - 3][a // 3] = line.strip()[a]
67                                         elif a % 3 == 0:
68                                             input_matrix[i - 3][a // 3] = input_matrix[i - 3][a // 3] + line.strip()[a]
69                                         else:
70                                             print("Tidak dalam matriks bukan karakter alfanumerik. Periksa kembali file input")
71                                             main()
72                                         a += 1
73                             except ValueError:
74                                 print("Tanda matriks yang dituliskan dalam file ini tidak dalam keadaan yang ideal (jumlah karakter tidak sesuai ukuran matriks ataupun spasi yang berlebihan antar karakter)")
75                                 print("harus cek format file dan ulangi masukan.")
76                                 main()
77                         elif i % 3 == 3 + matrix_height:
78                             try:
79                                 sekans_count = len(line.strip())
80                                 if sekans_count <= 0:
81                                     raise ValueError("Input banyak sekans harus berupa bilangan bulat.")
82                                 except ValueError:
83                                     print("Input banyak sekans harus berupa bilangan bulat. Coba lagi!")
84                                     main()
85                                 sekans_matrix = []
86                                 sekans_array = []
87                                 reward_array = [0 for c in range (sekans_count)]
88
89                                 elif i + 3 + matrix_height < i <= 3 + matrix_height + sekans_count*2:
90                                     s = i - (4 + matrix_height)
91                                     if s % 2 == 0:
92                                         s = s // 2
93                                         for x in range(len(line.strip()).replace(" ", "")):
94                                             if s % 2 == 1:
95                                                 sekans_matrix.append(line.strip().replace(" ", "")[x-1:line.strip().replace(" ", "")[x]])
96                                                 sekans_matrix.append(sekans_array)
97                                             else:
98                                                 s = s // 2
99                                     try:
100                                         reward_array[s] = len(line.strip())
101                                     except ValueError:
102                                         print("Input nilai reward harus berupa bilangan bulat. Coba lagi!")
103                                     main()
104

```

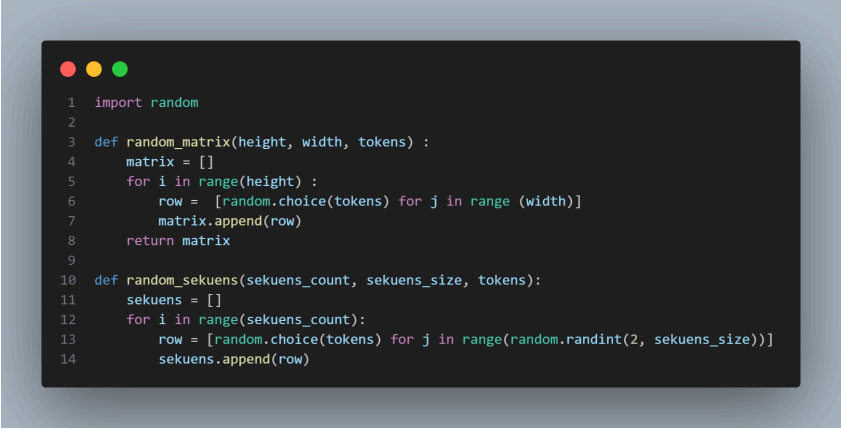
```

1 start_time = time.time()
2 tokens = Tokens.instantiate_from_matrix(input_matrix)
3
4 token_matrix = [[["" for a in range(matrix_width)] for b in range(matrix_height)]]
5 for token in tokens:
6     i = token.rowposition
7     j = token.colposition
8     token_matrix[i][j] = token
9
10 trees = create_n_ary_trees(token_matrix, buffer_size)
11
12 route_matrix = []
13 paths = []
14 for tree in trees :
15     find_all_path(tree, [], paths)
16     route_matrix.append(paths)
17     paths = []
18
19 sekuens = Sekuens.construct_sekuens(sekuens_matrix, reward_array, buffer_size)
20
21 route_reward_array = sequence_matching(sekuens, route_matrix)
22
23 print()
24 output_result = best_route_index(route_reward_array, reward_array, route_matrix)
25 print(output_result)
26
27 end_time = time.time()
28 runtime_seconds = end_time - start_time
29 runtime_ms = runtime_seconds * 1000
30 print(f"(runtime_ms:.2f) ms")
31
32 save_to_file = input("Apakah ingin menyimpan solusi? (y/n) : ")
33 while save_to_file.lower() != "y" and save_to_file.lower() != "n" :
34     save_to_file = input("Masukan tidak sesuai! Apakah ingin menyimpan solusi? (y/n) : ")
35
36 if save_to_file.lower() == "y":
37     file_name = input("Masukkan nama file yang diinginkan : ")
38     if not file_name.endswith('.txt'):
39         file_name += '.txt'
40     try:
41         with open('../test/output/' + file_name, 'w') as f:
42             f.write(output_result)
43             f.write('\n')
44             f.write(f"(runtime_ms:.2f) ms")
45         print("File berhasil disimpan dengan nama:", file_name)
46     except Exception as e:
47         print("Terjadi kesalahan saat menyimpan file:", str(e))
48
49 print()
50 print("Terima kasih!")
51 exit()

```

[illegible]

## 2. Randomize.py



```
1 import random
2
3 def random_matrix(height, width, tokens) :
4     matrix = []
5     for i in range(height) :
6         row = [random.choice(tokens) for j in range (width)]
7         matrix.append(row)
8     return matrix
9
10 def random_sekuens(sekuens_count, sekuens_size, tokens):
11     sekuens = []
12     for i in range(sekuens_count):
13         row = [random.choice(tokens) for j in range(random.randint(2, sekuens_size))]
14         sekuens.append(row)
```

### 3. Sekuens.py

```

1 class Sekuens:
2     def __init__(self, sekuens, reward):
3         self.sekuens = sekuens
4         self.reward = reward
5
6     @classmethod
7     def construct_sekuens(cls, sekuens_matrix, reward_array, buffer_size):
8         result_matrix = []
9         for i in range(len(sekuens_matrix)):
10             if len(sekuens_matrix[i]) <= buffer_size:
11                 result_matrix.append(cls(sekuens_matrix[i], reward_array[i]))
12         return result_matrix
13
14 def is_match (array_sequence, array_token) :
15     i = 0
16     j = 0
17     if len(array_sequence.sekuens) <= len(array_token) :
18         while i < len(array_token) and j < len(array_sequence.sekuens) :
19             if array_token[i].alphanumeric == array_sequence.sekuens[j] :
20                 i += 1
21                 j += 1
22             else :
23                 if i > 0 and j > 0 and array_token[i-1].alphanumeric == array_sequence.sekuens[j-1] :
24                     j = 0
25                     elif j == 0 :
26                         i += 1
27
28             if (j == len(array_sequence.sekuens)) :
29                 return True
30             else :
31                 return False
32     else :
33         return False
34
35 def sequence_matching(sequence_matrix, route_matrix) :
36     j = 0
37     totalreward_array = [[0 for a in range (len(route_matrix[b]))] for b in range (len(route_matrix))]
38     while j < len(sequence_matrix) :
39         i = 0
40         while i < len(route_matrix) :
41             k = 0
42             while k < len(route_matrix[i]) :
43                 if is_match(sequence_matrix[j], route_matrix[i][k]) :
44                     totalreward_array[i][k] += sequence_matrix[j].reward
45                 k += 1
46             i += 1
47         j += 1
48     return totalreward_array
49
50 def all_zero(matrix):
51     for i in range (len(matrix)):
52         for j in range (len(matrix[i])):
53             if matrix[i][j] != 0:
54                 return False
55     return True
56
57 def all_positive(array) :
58     for i in range (len(array)) :
59         if array[i] <= 0 :
60             return False
61     return True
62
63 def all_negative(array) :
64     for i in range (len(array)) :
65         if array[i] >= 0 :
66             return False
67     return True
68
69 def best_route_index(totalreward_array, reward_array, route_matrix):
70     if all_zero(totalreward_array) and all_positive(reward_array) :
71         output_string = "Tidak ditemukan kecocokkan pola dengan sekuens!\n"
72     elif all_zero(totalreward_array) and all_negative(reward_array) :
73         output_string = "Tidak ditemukan kecocokkan pola dengan sekuens!\n"
74     else :
75         i = 0
76         j = 0
77         max_reward = totalreward_array[0][0]
78         a = 0
79         b = 0
80         while i < len(totalreward_array):
81             j = 0
82             while j < len(totalreward_array[i]):
83                 if max_reward < totalreward_array[i][j]:
84                     max_reward = totalreward_array[i][j]
85                     a = i
86                     b = j
87             elif max_reward == totalreward_array[i][j] and len(route_matrix[a][b]) > len(route_matrix[i][j]) :
88                 max_reward = totalreward_array[i][j]
89                 a = i
90                 b = j
91             j += 1
92         i += 1
93
94     output_string = f"{max_reward}\n"
95     for x in range(len(route_matrix[a][b])):
96         output_string += route_matrix[a][b][x].alphanumeric + " "
97     output_string += "\n"
98     for y in range(len(route_matrix[a][b])):
99         output_string += f"({route_matrix[a][b][y].colposition + 1},{route_matrix[a][b][y].rowposition + 1}\n"

```

#### 4. Tokens.py

```
1 class Tokens:
2     def __init__(self, alphanumeric: str, colposition: int, rowposition : int, isSelected: bool = False):
3         self.alphanumeric = alphanumeric
4         self.colposition = colposition
5         self.rowposition = rowposition
6         self.isSelected = isSelected
7
8     @classmethod
9     def instantiate_from_matrix(cls, matrix):
10         tokens_list = []
11         for i in range(len(matrix)):
12             for j in range(len(matrix[i])):
13                 alphanumeric = matrix[i][j]
14                 colposition = j
15                 rowposition = i
16                 token = cls(alphanumeric, colposition, rowposition)
17                 tokens_list.append(token)
18         return tokens_list
19
20 def is_alphanumeric(char):
21     return (48 <= ord(char) <= 57) or (65 <= ord(char) <= 90) or (97 <= ord(char) <= 122)
22
```

#### 5. Tree.py

```

1 class TreeNode:
2     def __init__(self, info):
3         self.info = info
4         self.children = []
5
6 def create_n_ary_trees(matrix, depth):
7     trees = []
8     for root_info in matrix[0]:
9         root = TreeNode(root_info)
10        construct_tree(matrix, root, depth, 1)
11        trees.append(root)
12    return trees
13
14 def construct_tree(matrix, node, depth, level):
15     if level >= depth:
16         return
17     if level % 2 == 1: # Pergerakan vertikal
18         col_idx = None
19
20         for i, row in enumerate(matrix):
21             if node.info in row:
22                 col_idx = row.index(node.info)
23                 break
24
25         if col_idx is not None:
26             for row_idx, row in enumerate(matrix):
27                 if row_idx != i:
28                     child_info = row[col_idx]
29                     child_node = TreeNode(child_info)
30                     node.children.append(child_node)
31                     construct_tree(matrix, child_node, depth, level + 1)
32
33     else: # Pergerakan horizontal
34         col_idx_found = None
35         for i, row_found in enumerate(matrix):
36             if node.info in row_found:
37                 col_idx_found = row_found.index(node.info)
38                 break
39
40         for row_idx, row in enumerate(matrix):
41             if node.info in row:
42                 for col_idx, info in enumerate(row):
43                     if col_idx != col_idx_found:
44                         child_info = matrix[row_idx][col_idx]
45                         child_node = TreeNode(child_info)
46                         node.children.append(child_node)
47                         construct_tree(matrix, child_node, depth, level + 1)
48
49 # Mencari seluruh kemungkinan lintasan token dari tree yang telah tersusun
50 def find_all_path(root, path=[], result=[]):
51     # Jika sudah mencapai leaf, return
52     if root is None:
53         return
54
55     # Hanya menambahkan node yang belum masuk ke dalam susunan token sebagai lintasan
56     if root.info.isSelected == False:
57         path.append(root.info)
58         root.info.isSelected = True
59
60     # Menambahkan semua kemungkinan jalur dengan ketentuan sebuah lintasan terdiri dari minimal 2 karakter alfanumerik sesuai spesifikasi
61     if len(path) > 1:
62         result.append(path[:])
63
64     # Melakukan rekursi untuk semua child dari node saat ini
65     for child in root.children:
66         find_all_path(child, path, result)
67
68     # Backtracking dengan popping elemen terakhir
69     path[len(path)-1].isSelected = False
70     path.pop()
71
72     else:
73         return

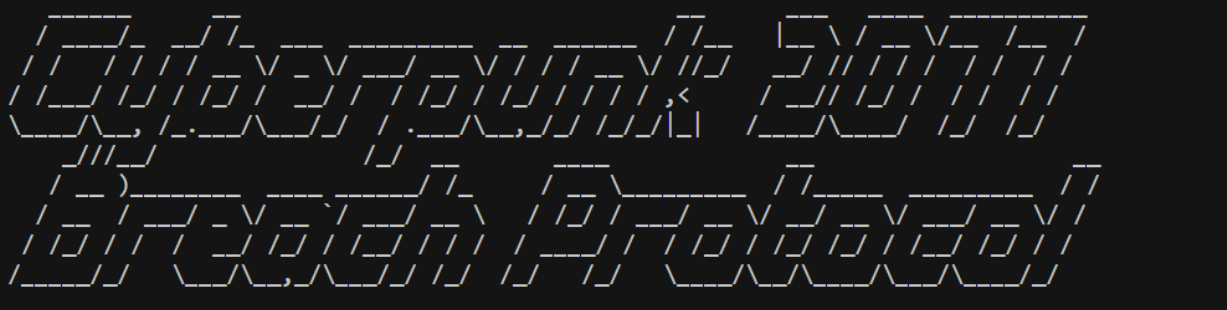
```

## BAB IV

## EKSPERIMEN

## 1. Inisialisasi Program

```
_Andhita Naura Hariyanto\src> python main.py
```



```
Selamat datang!
```

Input matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) :

### 1.a. Input dengan file txt

```

      _____ 
     /            \
    /              \
   /                \
  /                  \
 /                    \
/                      \
\                      /
 \                    /
  \                  /
   \                /
    \              /
     \            /
      _____ 

Selamat datang!

Input matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) : txt
Tuliskan nama file : testcase1.txt
```

### 1.b. Input dengan CLI untuk menggunakan matrix generator





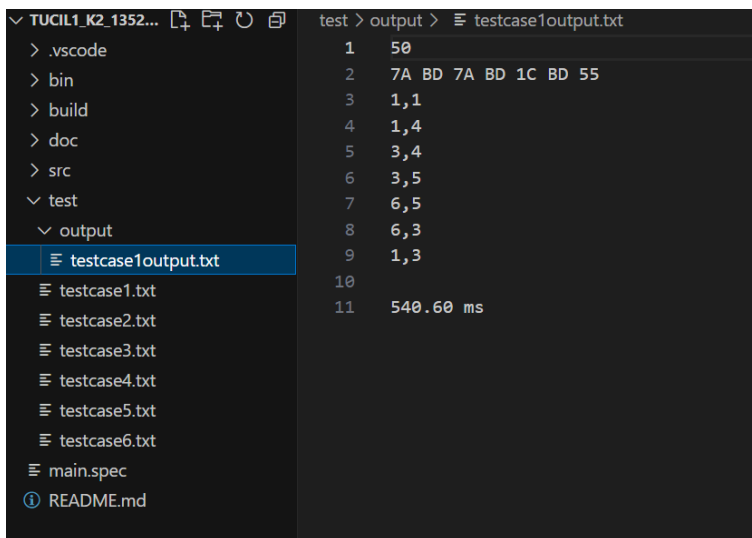
```
50
7A BD 7A BD 1C BD 55
1,1
1,4
3,4
3,5
6,5
6,3
1,3

540.60 ms
Apakah ingin menyimpan solusi? (y/n) : █
```

Save to file txt :

```
Apakah ingin menyimpan solusi? (y/n) : y
Masukkan nama file yang diinginkan : testcase1output.txt
File berhasil disimpan dengan nama: testcase1output.txt

Terima kasih!
```



The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure with folders like .vscode, bin, build, doc, src, test, and output. The file 'testcase1output.txt' is selected under the 'output' folder. The main editor area shows the content of this file, which is a list of test cases and their execution times, matching the output from the first terminal window.

```
test > output > ≡ testcase1output.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
11 540.60 ms
```

3. testcase2.txt

Test case :

```

test > ≡ testcase2.txt
1      | 8
2      5    5
3      1C E9 1C 55 1C
4      E9 55 1C 1C BD
5      55 BD 1C BD 55
6      55 1C 55 55 1C
7      E9 1C 1C 1C 55
8      3
9      55 1C
10     10
11     1C 1C E9
12     20
13     BD E9 55
14     30

```

Solusi program :

```

Tuliskan nama file : testcase2.txt

60
1C BD E9 55 1C 1C E9
5,1
5,2
1,2
1,3
3,3
3,1
2,1

414.82 ms
Apakah ingin menyimpan solusi? (y/n) : y

```

Save to file :

```

Apakah ingin menyimpan solusi? (y/n) : y
Masukkan nama file yang diinginkan : testcase2output
File berhasil disimpan dengan nama: testcase2output.txt

Terima kasih!

```

```
TUCIL1_K2_13522060_ANDHITA NA... test > output > testcase2output.txt
> .vscode
> bin
> build
> doc
> src
> test
  > output
    ≡ testcase1output.txt
    ≡ testcase2output.txt
    ≡ testcase1.txt
    ≡ testcase2.txt
    ≡ testcase3.txt
    ≡ testcase4.txt
    ≡ testcase5.txt
    ≡ testcase6.txt
  ≡ main.spec
  ⓘ README.md

1 60
2 1C BD E9 55 1C 1C E9
3 5,1
4 5,2
5 1,2
6 1,3
7 3,3
8 3,1
9 2,1
10
11 414.82 ms
```

#### 4. testcase3.txt

Test case :

```
test > testcase3.txt
1 8
2 5 5
3 1C E9 1C 55 1C
4 E9 +; 1C 1C BD
5 55 BD 1C BD 55
6 55 1C 55 55 1C
7 E9 1C 1C 1C 55
8 3
9 55 1C 10
10 1C 1C E9
11 20
12 BD E9 55
13 30
```

Solusi program :

```
Tuliskan nama file : testcase3.txt
Token dalam matrix bukan karakter alfanumerik! Periksa kembali file input!
Input matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) : █
```

5. testcase4.txt

Test case :

```
test > ≡ testcase4.txt
1      5
2      5 4
3      7A 55 E9 1C FF
4      FF 55 7A 1C 1C
5      BD BD E9 7A BD
6      FF 55 7A 7A 7A
7      3
8      7A 1C 1C 1C
9      10
10     7A FF 55 E9
11     20
12     BD 7A FF 55
13     30
```

Solusi program :

```
Tidak ditemukan kecocokkan pola dengan sekuens!
```

```
5.11 ms
```

```
Apakah ingin menyimpan solusi? (y/n) : y
```

Save to file txt :

```
Apakah ingin menyimpan solusi? (y/n) : y
```

```
Masukkan nama file yang diinginkan : testcase4output.txt
```

```
File berhasil disimpan dengan nama: testcase4output.txt
```

```
Terima kasih!
```

```
▼ TUCIL1_K2_1352... [🔍] [📁] [🔄] [📄]
  > .vscode
  > bin
  > build
  ▼ doc
  > src
  ▼ test
    ▼ output
      ≡ testcase1output.txt
      ≡ testcase2output.txt
      ≡ testcase4output.txt
      ≡ testcase1.txt
      ≡ testcase2.txt
      ≡ testcase3.txt
      ≡ testcase4.txt
      ≡ testcase5.txt
      ≡ testcase6.txt
    ≡ main.spec
    ⓘ README.md

test > output > ≡ testcase4output.txt
1  Tidak ditemukan kecocokkan pola dengan sekuens!
2
3  5.11 ms
```

#### 6. testcase5.txt

Test case :

```
test > ≡ testcase5.txt
1  -5
2  5 4
3  7A 55 E9 1C FF
4  FF 55 7A 1C 1C
5  BD BD E9 7A BD
6  FF 55 7A 7A 7A
7  3
8  7A 1C 1C 1C
9  10
10 7A FF 55 E9
11 20
12 BD 7A FF 55
13 30
```

Solusi program :

```
Tuliskan nama file : testcase5.txt
Input buffer harus berupa bilangan bulat lebih besar dari 1. Coba lagi!
Input matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) : █
```

#### 7. testcase6.txt

Test case :

```

test > ≡ testcase6.txt
1      8
2      5 5
3      1C E9 1C 55 1C
4      E9 55 1C 1C BD
5      55 BD 1C BD 55
6      55 1C 55 55 1C
7      E9 1C 1C 1C 55
8      3
9      55 1C
10     -10
11     1C 1C E9
12     -20
13     BD E9 55
14     -30

```

Solusi program :

```

0
1C E9
1,1
1,2

748.53 ms
Apakah ingin menyimpan solusi? (y/n) : y

```

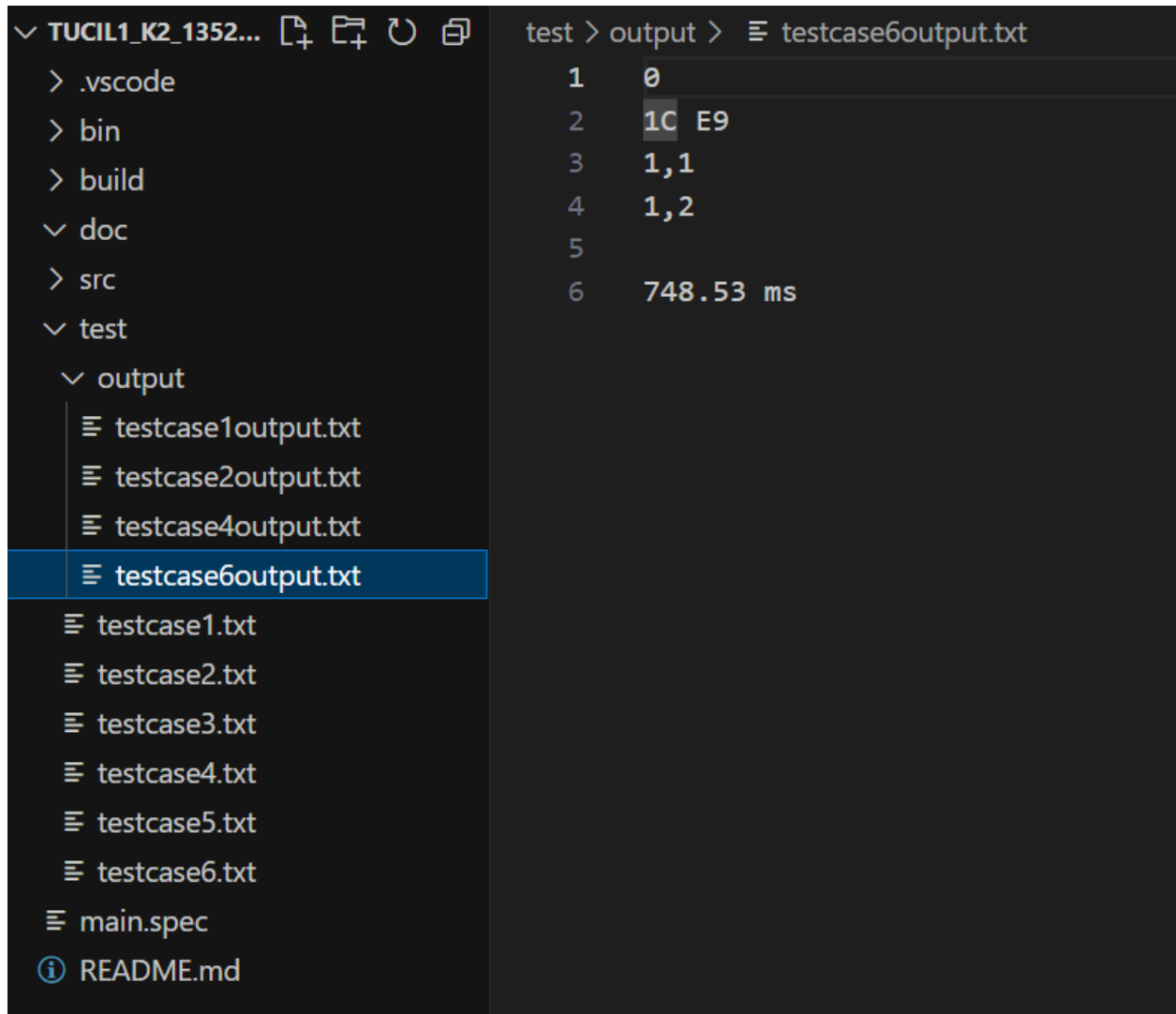
Save to file txt :

```

Apakah ingin menyimpan solusi? (y/n) : y
Masukkan nama file yang diinginkan : testcase6output
File berhasil disimpan dengan nama: testcase6output.txt

Terima kasih!

```



8. testcase7 (melakukan randomisasi matriks token dan sekuens berdasarkan data yang diinput melalui CLI)

Input :

```
Input matrix dan sekuens dengan file txt atau generator melalui input CLI? (txt/cli) : cli
Masukkan jumlah token unik: 5
Masukkan token-token (dipisahkan oleh spasi):
BD 1C 7A 55 E9
Masukkan ukuran buffer: 7
Masukkan lebar matriks: 6
Masukkan tinggi matriks: 6
Masukkan jumlah sekuens yang ingin dibentuk : 3
Masukkan ukuran maksimal sekuens: 4
```

Hasil matrix random generator dan sekuens random generator :



```
Matrix :
55 1C BD BD 7A 55
BD E9 E9 1C BD 55
7A BD 7A E9 7A E9
E9 7A 7A 7A BD BD
E9 E9 55 7A BD BD
7A 1C 55 E9 55 E9
Sekuens :
1 . ['E9', 'E9', '55']
2 . ['BD', '55', '55', 'BD']
3 . ['1C', 'BD', 'E9']
```

Solusi program :

```
77
BD 1C E9 BD 55 E9 55
1,1
1,4
6,4
6,5
4,5
4,2
3,2

914.41 ms
Apakah ingin menyimpan solusi? (y/n) : Y
```

Save to file txt :

```
Apakah ingin menyimpan solusi? (y/n) : Y
Masukkan nama file yang diinginkan : testcase7output.txt
File berhasil disimpan dengan nama: testcase7output.txt

Terima kasih!
```

TUCIL1\_K2\_13522060\_ANDHITA NA...

✓ .vscode

✓ doc

✓ src

> \_\_pycache\_\_

main.py

randomize.py

sekuens.py

tokens.py

tree.py

test

output

testcase1output.txt

testcase2output.txt

testcase4output.txt

testcase6output.txt

testcase7output.txt

testcase1.txt

testcase2.txt

testcase3.txt

testcase4.txt

testcase5.txt

testcase6.txt

README.md

test > output > testcase7output.txt

179

21C BD E9 BD 55 55 BD

32,1

42,3

54,3

64,1

76,1

86,2

91,2

10

11909.15 ms

## LAMPIRAN

GitHub Repository :

[https://github.com/andhitanh/Tucil1\\_K2\\_13522060.git](https://github.com/andhitanh/Tucil1_K2_13522060.git)

Tabel Spesifikasi :

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		✓