

Tugas Kecil 1

IF2211 Strategi Algoritma

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma Brute Force



Nama : Rici Trisna Putra

NIM : 13522026

Kelas : 02

**PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH
TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2024**

Bab 1

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 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 optimal untuk matriks dan sekuens tersebut adalah sebagai berikut:

- Total bobot langkah : 50 poin
- Total Langkah : 6 langkah



Di dalam Tugas Kecil 1 ini, Saya diminta untuk mengimplementasikan program yang dapat menemukan solusi optimum untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma brute force.

Bab 2

Algoritma Brute Force

Brute force adalah sebuah pendekatan yang lempang (straightforward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Brute Force mencari solusi dengan mengevaluasi seluruh kemungkinan solusi yang mungkin.

Pada permasalahan minigame Cyberpunk 2077 Breach Protocol, algoritma brute force harus dapat mengakomodasi seluruh kemungkinan rangkaian yang dipilih oleh pemain dengan panjang tertentu kemudian mengevaluasi bobot setiap kemungkinan tersebut dan memilih yang paling optimum. Hal ini saya realisasikan dengan memanfaatkan sifat rekursifitas masalah ini untuk menelusuri seluruh kemungkinan.

Apabila terdapat suatu matriks permainan 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 |

Kemudian diberikan koordinat posisi pengambilan token terakhir ((0,0) dalam kasus permainan baru dimulai) dan arah traversal (horizontal atau vertikal) pada langkah tersebut, maka kita dapat membuat suatu sublist (token-token pada kolom atau baris) yang berisikan semua kemungkinan token yang mungkin terpilih.

Dari contoh matriks permainan di atas maka pada awal permainan ((0,0) dengan arah horizontal) sublist yang didapat adalah baris paling atas matriks tersebut.

| | | | | | |
|----|----|----|----|----|----|
| 7A | 55 | E9 | E9 | 1C | 55 |
|----|----|----|----|----|----|

Maka menggunakan for loop untuk panjang sublist tersebut kita dapat menelusuri setiap token yang mungkin diambil kemudian dimasukkan ke dalam buffer. Setelah itu kita ambil posisi token yang terambil dan ubah arah traversal dari horizontal ke vertikal dan sebaliknya. Kemudian lakukan rekursi (langkah yang sama) untuk setiap kemungkinan token pada sublist menggunakan posisi pengambilan token terakhir dan arah traversal yang baru hingga buffer penuh.

Ketika buffer penuh maka hentikan rekursi dan lakukan evaluasi terhadap isi buffer untuk kemudian di simpan rangkaian dan jalurnya apabila memiliki bobot paling maksimum saat ini. Dengan sistem tersebut maka algoritma akan otomatis melakukan backtrack pada kasus-kasus lain yang mungkin.

Perlu diketahui pula bahwa dalam algoritma ini kita butuh suatu mark atau tanda ketika suatu token sudah pernah diambil (di dalam buffer) sehingga ketika algoritma mencoba mengambil kembali token tersebut maka kita dapat mengabaikan langkah tersebut (bukan termasuk kemungkinan solusi).

Selain itu apabila bobot suatu rangkaian di buffer sama dengan bobot maksimum yang mungkin untuk permainan tersebut, maka kita juga dapat membuat suatu sistem untuk menghentikan proses pencarian solusi karena solusi maksimum sudah didapatkan (tidak perlu mengecek sisa kemungkinan solusi).

Bab 3

Source Code Program

Kode program pada tugas kecil ini saya bagi menjadi 5 buah file terpisah untuk memudahkan pembacaan dan organisasi. Adapun kelima file tersebut adalah:

- class_definition.py

merupakan kode program yang berisi definisi kelas-kelas yang digunakan untuk merepresentasikan objek permasalahan. Adapun kelas dalam source code ini antara lain Matrix, Buffer, Sequence, dan Sequences.

```
1 import numpy as np
2 class Matrix:
3     def __init__(self,height,width):
4         self.width = width
5         self.height = height
6         self.matrix = np.full([height,width], '??')
7
8     def print(self):
9         print(self.matrix)
10
11     def get_cell(self,row,column):
12         return(self.matrix[row,column])
13
14     def set_cell(self,row,column,value):
15         self.matrix[row,column] = value
16
17     def copy(self):
18         matrix_cp = Matrix(self.height, self.width)
19         matrix_cp.matrix = np.copy(self.matrix)
20         return matrix_cp
21
22 class Buffer:
23     def __init__(self,size):
24         self.size = size
25         self.list = np.full([size], '??')
26
27     def print(self):
28         print(self.list)
29
30     def get_element(self,idx):
31         return(self.list[idx])
32
33     def set_element(self,idx,value):
34         self.list[idx] = value
```

```
1 class Sequence:
2     def __init__(self,size,weight):
3         self.size = size
4         self.weight = weight
5         self.list = np.full([size], '??')
6
7     def __repr__(self):
8         return np.array2string(self.list)
9
10    def __str__(self):
11        return np.array2string(self.list)
12
13    def print(self):
14        print(np.array2string(self.list))
15
16    def get_element(self,idx):
17        return(self.list[idx])
18
19    def set_element(self,idx,value):
20        self.list[idx] = value
21
22    def match(self,list):
23        length = self.size
24        for i in range(length):
25            temp = list[i:i+length]
26            if np.array_equal(self.list, temp):
27                return self.weight
28        return 0
29
30 class Sequences:
31     def __init__(self,size):
32         self.size = size
33         self.list = np.full([size], Sequence(2,0))
34
35     def print(self):
36         print(self.list)
37         print(f'weight : {list(x.weight for x in self.list)}')
38
39     def get_element(self,idx):
40         return(self.list[idx])
41
42     def set_element(self,idx,value):
43         self.list[idx] = value
44
45     def get_max_weight(self):
46         sum = 0
47         for x in self.list:
48             sum += x.weight
49         return sum
50
51     def match_all(self,list):
52         sum = 0
53         for sequence in self.list:
54             sum += sequence.match(list)
55         return sum
```

- txt_handler.py

Merupakan kode program yang berisi utilitas untuk menangani pembacaan maupun penulisan hasil program ke dalam file .txt. Terdapat dua fungsi yang terdefinisi pada file ini yakni read_txt dan write_txt yang sesuai namanya masing-masing digunakan untuk membaca dan menulis ke dalam file.txt

```
1 from class_definition import *
2 import numpy as np
3 import os
4 import sys
5
6 def read_txt(path):
7     # Membuka file
8     file = open(path, "r")
9
10    # Buffer
11    buffer = Buffer(int(file.readline())) # 1. Panjang buffer
12
13    # Matrix
14    width, height = (int(x) for x in (file.readline().split(" "))) # 2. Dimensi matriks
15    matrix = Matrix(height, width)
16    for i in range(height): # 3. Konten matriks
17        temp = np.array(list(x for x in file.readline().rstrip('\n').split(" ") if x != ""))
18        if len(temp) != width:
19            print("Dimensi matrix input tidak sesuai dengan ukuran")
20            sys.exit()
21        else:
22            matrix.matrix[i] = temp
23
24    # Sequences
25    sequence_amount = int(file.readline()) # 4. Jumlah sekuens
26    sequences = Sequences(sequence_amount)
27    for i in range(sequence_amount): # 5 Token-token sekuens dan bobotnya
28        temp = np.array(list(x for x in file.readline().rstrip('\n').split(" ") if x != ""))
29        weight = int(file.readline())
30        seq = Sequence(len(temp), weight)
31        seq.list = temp
32        sequences.set_element(i, seq)
33
34    # Menutup file
35    file.close
36
37    # Mengembalikan hasil pembacaan matriks, buffer dan sekuens
38    return matrix, buffer, sequences
39
40 def write_txt(max_weight, current_optimum, list_step, duration):
41
42     # Input preferensi simpan atau tidak
43     is_saved = input("Apakah ingin menyimpan solusi? (y/n): ")
44     print("")
45     if is_saved == "y" or is_saved == "Y":
46         # Input nama file .txt
47         file_name = input("Masukkan nama file: ")
48         path = os.path.join("../test", file_name + ".txt")
49         # Open file
50         with open(path, 'w+') as file:
51             file.write(f'{max_weight}\n')
52             for x in current_optimum:
53                 file.write(f'{x} ')
54             file.write('\n')
55             for x in list_step:
56                 file.write(f'{x[0]}, {x[1]}\n')
57             file.write('\n')
58             file.write(f'{duration}s')
59     else:
60         sys.exit()
```

- random_input.py

Merupakan kode program yang berisi utilitas untuk mengenerasi matriks, buffer, dan sekuens secara random. Terdapat beberapa fungsi yang terdefinisi di dalam kode program ini antara lain random yang merupakan program utama untuk melakukan randomisasi, generate_random_matrix untuk membuat matriks acak, generate_random_sequence untuk membuat beberapa sekuens sekaligus secara acak, dan randintskew untuk menghasilkan bilangan bulat yang condong ke arah lower_bound.

```
1 from class_definition import *
2 from random import uniform, randint, randrange
3 import numpy as np
4 import sys
5
6 def random():
7
8     # Token
9     token_count = int(input()) # 1. Jumlah token unik
10    tokens = np.array(list(x for x in input().rstrip('\n').split(" ") if x != "")) # 2. Token unik ex: BD E9 1C 2D 33
11    if len(tokens) != token_count:
12        print("Jumlah token tidak sesuai dengan inputan !!")
13        sys.exit() # Keluar ketika input tidak sesuai
14
15    # Buffer
16    buffer_size = int(input()) # 3. Ukuran buffer
17    buffer = Buffer(buffer_size)
18
19    # Matrix
20    width, height = (int(x) for x in (input().split(" "))) # 4. Width dan height matriks
21    matrix = Matrix(height,width)
22    generate_random_matrix(matrix,tokens)
23
24    # Sequence
25    sequence_count = int(input()) # 5. Jumlah sequence
26    sequences = Sequences(sequence_count)
27    sequence_max_len = int(input()) # 6. Panjang maksimal sequence
28    generate_random_sequences(sequences,sequence_count,sequence_max_len,tokens)
29
30    return matrix, buffer, sequences
31
32 def generate_random_matrix(matrix,tokens):
33     for i in range(matrix.height):
34         for j in range(matrix.width):
35             matrix.set_cell(i,j,tokens[randint(0, len(tokens)-1)]) # Mengambil token random untuk mengisi setiap sel pada matriks
36
37 def generate_random_sequences(sequences,sequence_count,sequence_max_len,tokens):
38     listlen = [] # list untuk menampung beberapa bilangan bulat hasil randomisasi yang digunakan sebagai panjang setiap sekuens
39     listweight = [] # mirip listlen namun digunakan untuk bobot tiap sekuens
40     for i in range(sequence_count):
41         listlen.append(randint(2,sequence_max_len,uniform(1,2))) # mengenerasi bilangan bulat acak yang condong ke arah lower bound
42         listweight.append(randrange(10,100,5)) # mengenerasi bilangan bulat acak keliapatan lima antara 10 dan 100
43     # Menyortir list
44     listlen.sort()
45     listweight.sort()
46
47     for i,x in enumerate(listlen): # Mengenerasi sekuens sesuai panjang dan bobot yang dirandomisasi sebelumnya
48         temp = Sequence(x, listweight[i])
49         for j in range(x):
50             temp.list[j] = tokens[randint(0, len(tokens)-1)]
51         sequences.set_element(i,temp)
52
53 def randintskew(min,max,power): # Fungsi untuk mengenerasi bilangan yang condong ke arah lower bound
54     seed = uniform(0,1)
55     pos = pow(seed,power)
56     return int(round((max - min) * pos + min))
```


- solution_finder.py

Merupakan kode program realisasi dari metode bruteforce untuk menemukan solusi. Fungsi solution_finder merupakan fungsi rekursi untuk menemukan solusi optimum sedangkan fungsi brute_forced hanya digunakan untuk memanggil solution_finder dan meneruskan hasil optimum ke program di main.py

```
1 from class_definition import *
2 from txt_handler import *
3 import numpy as np
4 import sys
5
6 # Inisiasi
7 max_weight = 0
8 step = 0
9 stop = False
10
11 def solution_finder(matrix, buffer, sequences, direction, start_position, list_step, step):
12     global current_optimum # Variabel global untuk menampung sekuens optimum
13     global step_optimum # Variabel global untuk menampung koordinat setiap token pada sekuens optimum
14     global max_weight # Variabel global untuk menampung bobot maksimum dari sekuens optimum
15     global stop # Untuk menghentikan brute force ketika terdapat sekuens yang memiliki bobot paling maksimum yang mungkin dicapai
16
17     if not stop:
18         if step != buffer.size: # Mengecek apakah buffer belum penuh
19             if direction == "horizontal":
20                 direction = "vertical" # mengubah arah untuk step selanjutnya
21                 step = step + 1 # menambah counter untuk step
22                 for i in range(matrix.width):
23                     new_position = np.copy(start_position) # membuat variabel posisi baru untuk pemanggilan fungsi rekursi selanjutnya
24                     new_position[1] = i # mengubah posisi horizontal dari posisi awal
25                     if matrix.get_cell(new_position[0], new_position[1]) == '##': # Skip apabila buffer mengambil token yang pernah diambil sebelumnya (invalid)
26                         continue
27                     list_step[step-1] = np.array([new_position[0], new_position[1]]) # Mencatat posisi dari token yang akan dimasukkan ke buffer
28                     buffer.set_element(step-1, matrix.get_cell(new_position[0], new_position[1])) # Mencatat token ke dalam buffer
29                     matrix.set_cell(new_position[0], new_position[1], '##') # Mengubah token yang diambil menjadi '##' (mark)
30                     solution_finder(matrix, buffer, sequences, direction, new_position, list_step, step) # Rekursi
31                     matrix.set_cell(list_step[step-1][0], list_step[step-1][1], buffer.get_element(step-1)) # Mengembalikan isi matriks dari mark ke token semula
32                     buffer.set_element(step-1, '??') # Mengembalikan catatan buffer pada kondisi sebelumnya
33                     list_step[step-1] = np.array([-1, -1]) # Menghapus posisi dari token yang dimasukkan ke buffer
34             else:
35                 # Mirip dengan penjelasan diatas namun untuk pergerakan vertikal
36                 direction = "horizontal"
37                 step = step + 1
38                 for i in range(matrix.height):
39                     new_position = np.copy(start_position)
40                     new_position[0] = i
41                     if matrix.get_cell(new_position[0], new_position[1]) == '##':
42                         continue
43                     list_step[step-1] = np.array([new_position[0], new_position[1]])
44                     buffer.set_element(step-1, matrix.get_cell(new_position[0], new_position[1]))
45                     matrix.set_cell(new_position[0], new_position[1], '##')
46                     solution_finder(matrix, buffer, sequences, direction, new_position, list_step, step)
47                     matrix.set_cell(list_step[step-1][0], list_step[step-1][1], buffer.get_element(step-1))
48                     buffer.set_element(step-1, '??')
49                     list_step[step-1] = np.array([-1, -1])
50             else:
51                 weight = sequences.match_all(buffer.list) # Mencocokkan buffer dengan sekuens dan menyimpan hasil evaluasi bobotnya
52                 if weight > max_weight: # Mengecek apakah bobot maksimum dan mencatat step dan sekuens token-tokennya
53                     max_weight = weight
54                     step_optimum = list_step.tolist()
55                     current_optimum = np.copy(buffer.list)
56                     if max_weight == sequences.get_max_weight(): # Menghentikan bruteforce ketika bobot maksimum tercapai
57                         stop = True
58                     for i in range(sequences.get_element(0).size, buffer.size): # Mencari step paling sedikit ketika ditemukan bobot maksimum
59                         temp = current_optimum[:i]
60                         weight = sequences.match_all(temp)
61                         if weight == max_weight:
62                             current_optimum = temp
63                             step_optimum = step_optimum[:i]
64                             break
65
66
67 def brute_forced(matrix, buffer, sequences):
68     global current_optimum
69     global step_optimum
70
71     # Inisiasi
72     direction = "horizontal"
73     start_position = np.array([0, 0])
74     step_optimum = []
75     list_step = np.full((buffer.size, 2), -1)
76     current_optimum = np.full((buffer.size), '??')
77
78     # Brute force solusi
79     solution_finder(matrix, buffer, sequences, direction, start_position, list_step, step)
80
81     # Output hasil
82     print(max_weight)
83     if current_optimum[0] != '??':
84         for x in current_optimum:
85             print(f'x {x}, end="")
86         print("")
87         for x in step_optimum:
88             print(f'x {x[0]}, x {x[1]}")
89     return max_weight, current_optimum, step_optimum
90
91 else:
92     print("Tidak ada solusi")
93     sys.exit()
```

- main.py

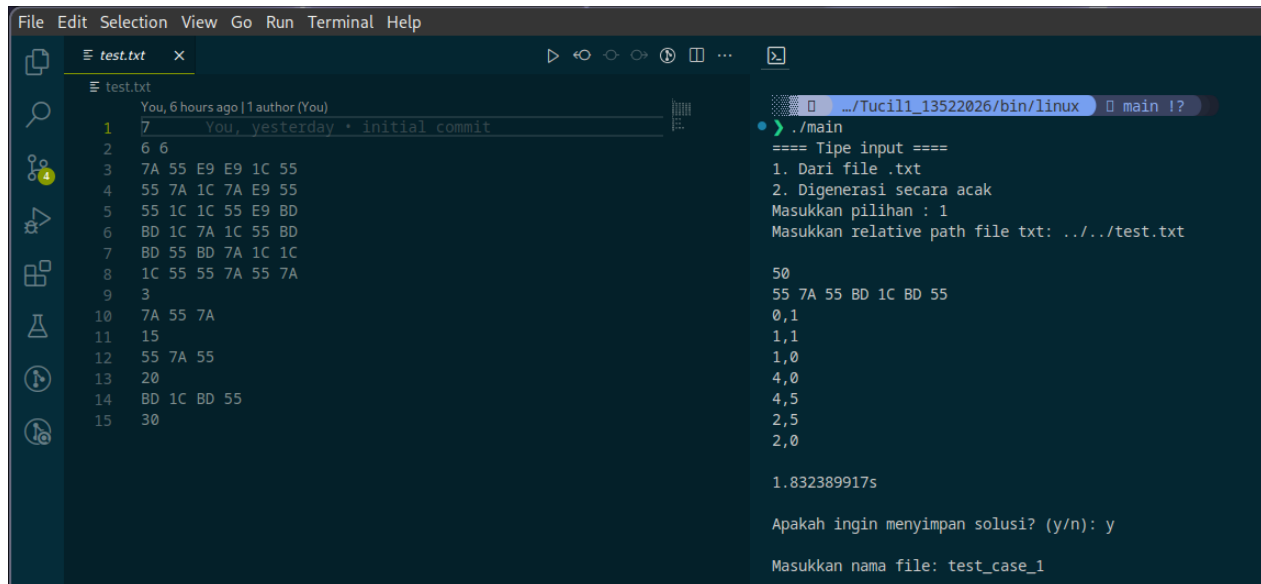
Merupakan kode program utama yang menggabungkan semua kelas dan utilitas pada kode program lain sehingga ketika dijalankan dapat memecahkan masalah dengan interaktif.

```
1 from numpy import matrix
2 from class_definition import *
3 from solution_finder import *
4 from random_input import *
5 from txt_handler import *
6 import sys
7 import os
8 import time
9
10
11 if __name__ == "__main__":
12     # Prompt
13     print("==== Tipe input ====")
14     print("1. Dari file .txt")
15     print("2. Digenerasi secara acak")
16     stop = False
17     option = 0
18     while not stop:
19         try:
20             option = int(input("Masukkan pilihan : "))
21             if option != 1 and option != 2:
22                 raise TypeError
23         except (TypeError, ValueError):
24             print("Masukkan bilangan bulat sesuai opsi")
25         else:
26             stop = True
27     if option == 1:
28         path = os.path.join(input("Masukkan relative path file txt: "))
29         if os.path.isfile(path):
30             matrix,buffer,sequences = read_txt(path)
31             start = time.process_time()
32             print("")
33             max_weight, current_optimum, list_step = brute_forced(matrix,buffer,sequences)
34             duration = time.process_time() - start
35             print("\n" + str(duration) + "s")
36             print("")
37             write_txt(max_weight,current_optimum,list_step,duration)
38         else:
39             print("File tidak ditemukan")
40             sys.exit()
41     else:
42         matrix,buffer,sequences = random()
43         print("\n== Matrix ==")
44         matrix.print()
45         print("\n== Sekuens ==")
46         sequences.print()
47         start = time.process_time()
48         print("")
49         max_weight, current_optimum, list_step = brute_forced(matrix,buffer,sequences)
50         duration = time.process_time() - start
51         print("\n" + str(duration) + "s")
52         print("")
53         write_txt(max_weight,current_optimum,list_step,duration)
```

Bab 4

Uji Coba Program

- Uji Coba 1



The screenshot shows a code editor with a file named `test.txt` and its execution output. The file contains a commit message and a list of hexadecimal values. The output shows the program's execution flow, including input prompts, file reading, and the generation of a solution.

```
File Edit Selection View Go Run Terminal Help
test.txt x
test.txt
You, 6 hours ago | 1 author (You)
1 7 You, yesterday · initial commit
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 7A 55 7A
11 15
12 55 7A 55
13 20
14 BD 1C BD 55
15 30

./main
==== Tipe input ====
1. Dari file .txt
2. Digenerasi secara acak
Masukkan pilihan : 1
Masukkan relative path file txt: ../../test.txt

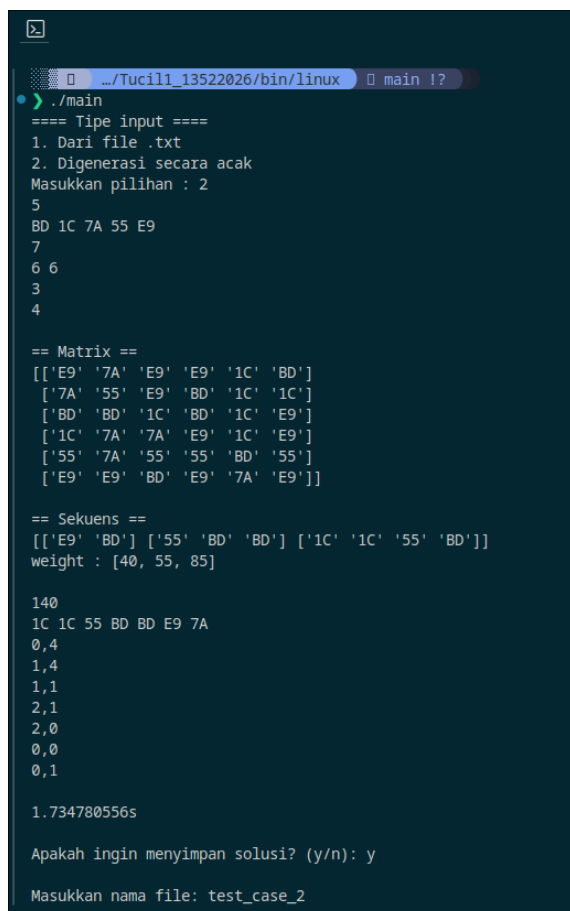
50
55 7A 55 BD 1C BD 55
0,1
1,1
1,0
4,0
4,5
2,5
2,0

1.832389917s

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

Masukkan nama file: test_case_1
```

- Uji Coba 2



The screenshot shows the same program being executed with a different input choice (2). The output displays a matrix of hexadecimal values, a sequence of values, and the resulting solution.

```
./Tucil1_13522026/bin/linux main !?
./main
==== Tipe input ====
1. Dari file .txt
2. Digenerasi secara acak
Masukkan pilihan : 2
5
BD 1C 7A 55 E9
7
6 6
3
4

== Matrix ==
[['E9' '7A' 'E9' '1C' 'BD']
 ['7A' '55' 'E9' 'BD' '1C' '1C']
 ['BD' 'BD' '1C' 'BD' '1C' 'E9']
 ['1C' '7A' '7A' 'E9' '1C' 'E9']
 ['55' '7A' '55' '55' 'BD' '55']
 ['E9' 'E9' 'BD' 'E9' '7A' 'E9']]

== Sekuens ==
[['E9' 'BD'] ['55' 'BD' 'BD'] ['1C' '1C' '55' 'BD']]
weight : [40, 55, 85]

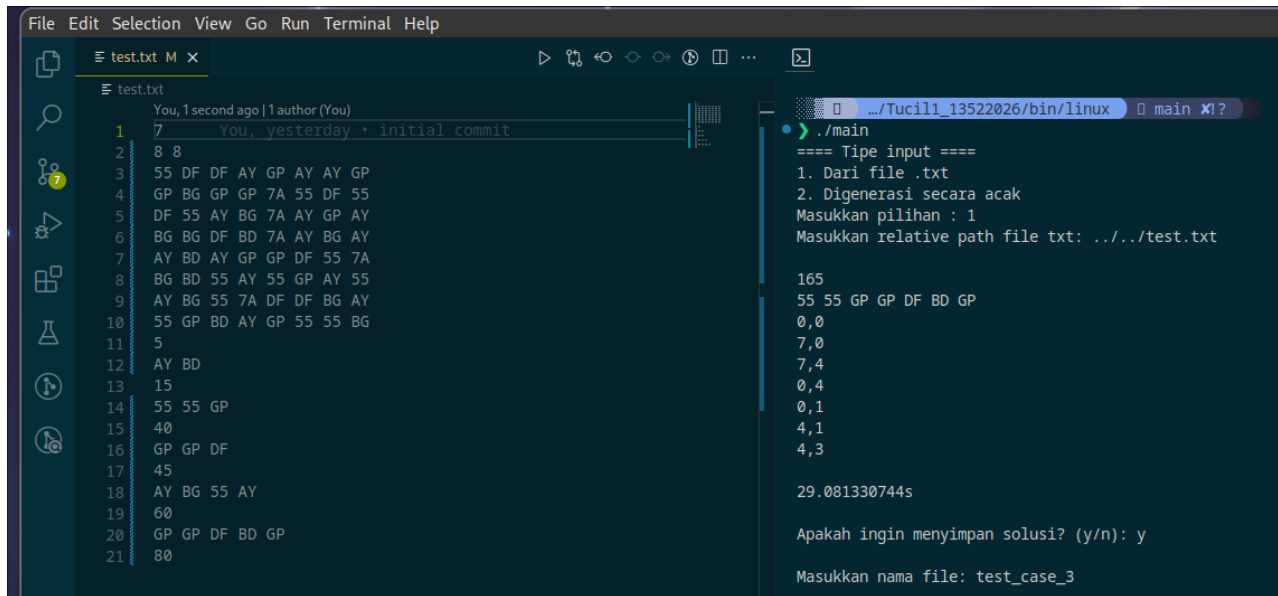
140
1C 1C 55 BD BD E9 7A
0,4
1,4
1,1
2,1
2,0
0,0
0,1

1.734780556s

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

Masukkan nama file: test_case_2
```

- Uji Coba 3



The screenshot shows a code editor with a file named `test.txt` open. The file contains a commit message and a list of 21 lines of data. The terminal output shows the execution of a program that reads the file and processes the data.

```
File Edit Selection View Go Run Terminal Help
test.txt M x
test.txt
You, 1 second ago | 1 author (You)
7 You, yesterday * initial commit
1 8 8
2 55 DF DF AY GP AY AY GP
3 GP BG GP GP 7A 55 DF 55
4 DF 55 AY BG 7A AY GP AY
5 BG BG DF BD 7A AY BG AY
6 AY BD AY GP GP DF 55 7A
7 BG BD 55 AY 55 GP AY 55
8 AY BG 55 7A DF DF BG AY
9 55 GP BD AY GP 55 55 BG
10 5
11 AY BD
12 15
13 55 55 GP
14 40
15 GP GP DF
16 45
17 AY BG 55 AY
18 60
19 GP GP DF BD GP
20 80
21

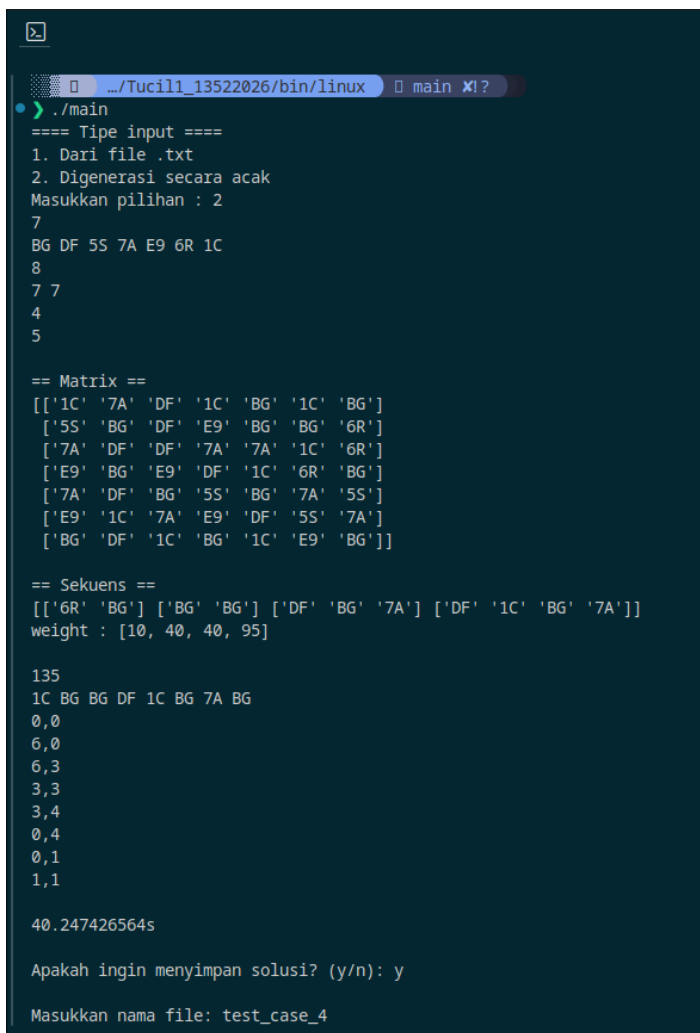
./main
==== Tipe input ====
1. Dari file .txt
2. Digenerasi secara acak
Masukkan pilihan : 1
Masukkan relative path file txt: ../../test.txt

165
55 55 GP GP DF BD GP
0,0
7,0
7,4
0,4
0,1
4,1
4,3

29.081330744s

Apakah ingin menyimpan solusi? (y/n): y
Masukkan nama file: test_case_3
```

- Uji Coba 4



The screenshot shows a terminal window with the following output:

```
./main
==== Tipe input ====
1. Dari file .txt
2. Digenerasi secara acak
Masukkan pilihan : 2
7
BG DF 55 7A E9 6R 1C
8
7 7
4
5

== Matrix ==
[['1C' '7A' 'DF' '1C' 'BG' '1C' 'BG']
['55' 'BG' 'DF' 'E9' 'BG' 'BG' '6R']
['7A' 'DF' 'DF' '7A' '7A' '1C' '6R']
['E9' 'BG' 'E9' 'DF' '1C' '6R' 'BG']
['7A' 'DF' 'BG' '55' 'BG' '7A' '55']
['E9' '1C' '7A' 'E9' 'DF' '55' '7A']
['BG' 'DF' '1C' 'BG' '1C' 'E9' 'BG']]

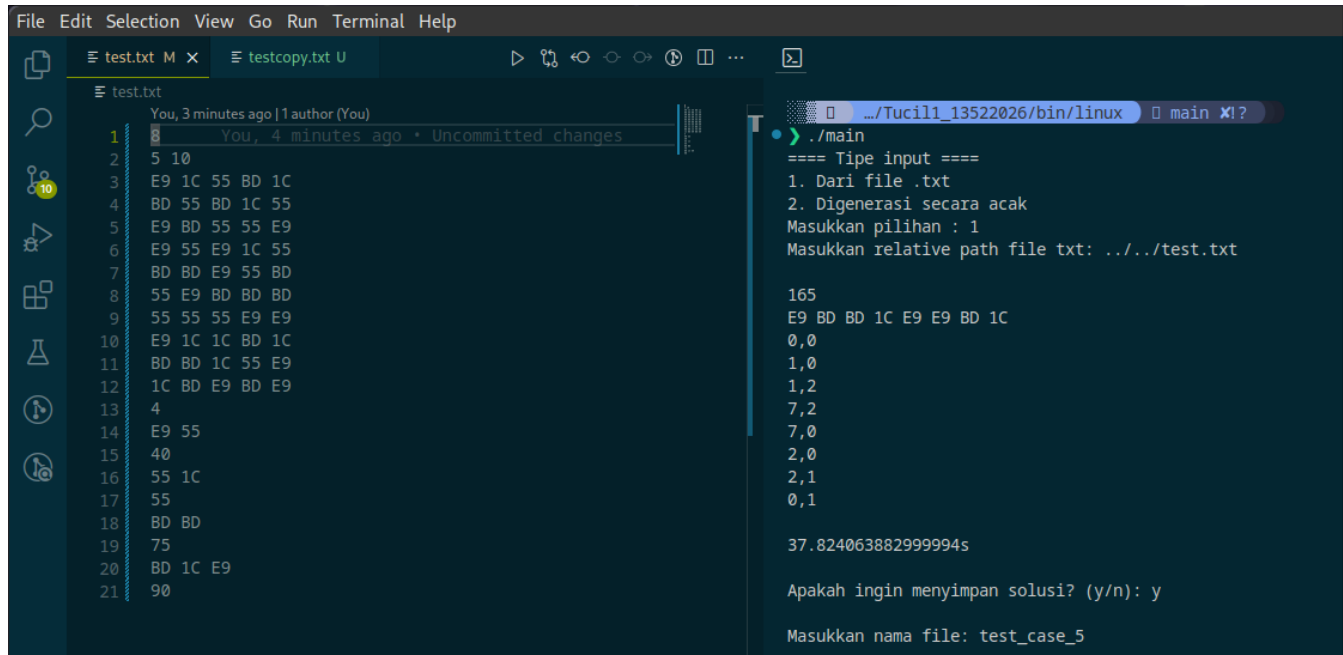
== Sekuens ==
[['6R' 'BG'] ['BG' 'BG'] ['DF' 'BG' '7A'] ['DF' '1C' 'BG' '7A']]
weight : [10, 40, 40, 95]

135
1C BG BG DF 1C BG 7A BG
0,0
6,0
6,3
3,3
3,4
0,4
0,1
1,1

40.247426564s

Apakah ingin menyimpan solusi? (y/n): y
Masukkan nama file: test_case_4
```

- Uji Coba 5



The screenshot shows a code editor with two tabs: 'test.txt' and 'testcopy.txt'. The 'test.txt' tab is active, displaying a list of 21 lines of text. The text is a mix of letters and numbers, some of which are repeated. The 'testcopy.txt' tab is also visible, showing a similar list of text. The editor's interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help) and a toolbar with various icons. The output of the program is shown in a terminal window on the right, displaying the program's execution flow, including input prompts and the resulting output.

```
File Edit Selection View Go Run Terminal Help
test.txt M x testcopy.txt U
test.txt
You, 3 minutes ago | 1 author (You)
1 8 You, 4 minutes ago * Uncommitted changes
2 5 10
3 E9 1C 55 BD 1C
4 BD 55 BD 1C 55
5 E9 BD 55 55 E9
6 E9 55 E9 1C 55
7 BD BD E9 55 BD
8 55 E9 BD BD BD
9 55 55 55 E9 E9
10 E9 1C 1C BD 1C
11 BD BD 1C 55 E9
12 1C BD E9 BD E9
13 4
14 E9 55
15 40
16 55 1C
17 55
18 BD BD
19 75
20 BD 1C E9
21 90

.../Tucil1_13522026/bin/linux main x! ?
./main
==== Tipe input ====
1. Dari file .txt
2. Digenerasi secara acak
Masukkan pilihan : 1
Masukkan relative path file txt: ../../test.txt

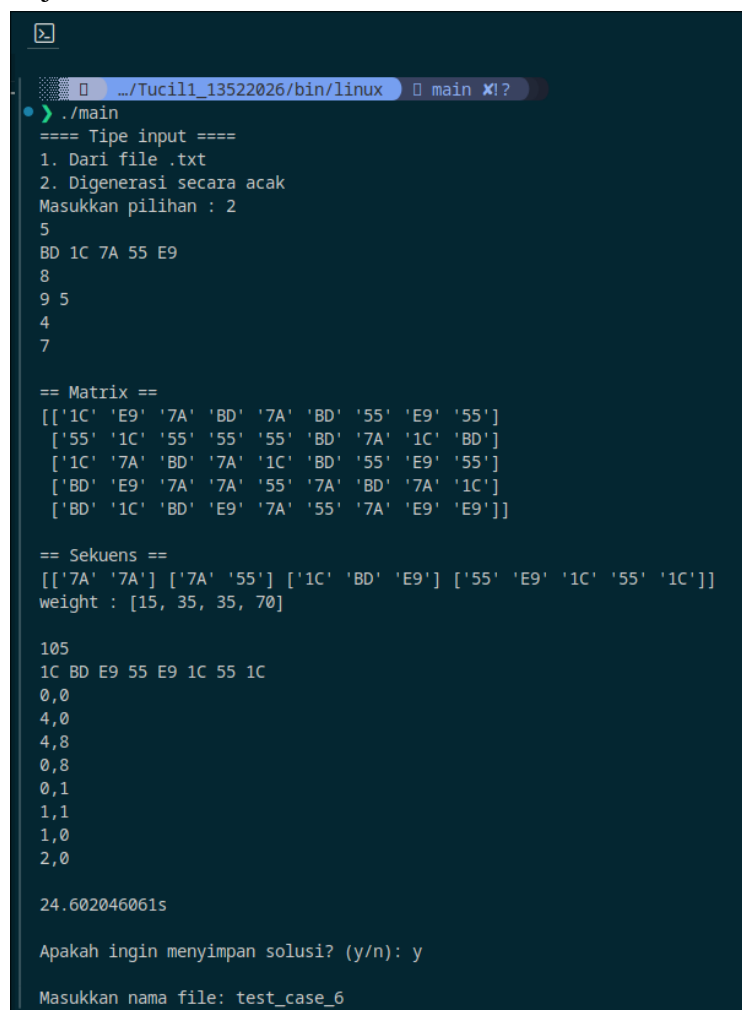
165
E9 BD BD 1C E9 E9 BD 1C
0,0
1,0
1,2
7,2
7,0
2,0
2,1
0,1

37.824063882999994s

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

Masukkan nama file: test_case_5
```

-Uji Coba 6



The screenshot shows a terminal window with the following output:

```
.../Tucil1_13522026/bin/linux main x! ?
./main
==== Tipe input ====
1. Dari file .txt
2. Digenerasi secara acak
Masukkan pilihan : 2
5
BD 1C 7A 55 E9
8
9 5
4
7

== Matrix ==
[['1C' 'E9' '7A' 'BD' '7A' 'BD' '55' 'E9' '55']
['55' '1C' '55' '55' '55' 'BD' '7A' '1C' 'BD']
['1C' '7A' 'BD' '7A' '1C' 'BD' '55' 'E9' '55']
['BD' 'E9' '7A' '7A' '55' '7A' 'BD' '7A' '1C']
['BD' '1C' 'BD' 'E9' '7A' '55' '7A' 'E9' 'E9']]

== Sekuens ==
[['7A' '7A'] ['7A' '55'] ['1C' 'BD' 'E9'] ['55' 'E9' '1C' '55' '1C']]
weight : [15, 35, 35, 70]

105
1C BD E9 55 E9 1C 55 1C
0,0
4,0
4,8
0,8
0,1
1,1
1,0
2,0

24.602046061s

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

Masukkan nama file: test_case_6
```

LAMPIRAN

| Poin | Ya | Tidak |
|---|-------------------------------------|-------------------------------------|
| 1. Program berhasil dikompilasi tanpa kesalahan | <input checked="" type="checkbox"/> | |
| 2. Program berhasil dijalankan | <input checked="" type="checkbox"/> | |
| 3. Program dapat membaca masukan berkas .txt | <input checked="" type="checkbox"/> | |
| 4. Program dapat menghasilkan masukan secara acak | <input checked="" type="checkbox"/> | |
| 5. Solusi yang diberikan program optimal | <input checked="" type="checkbox"/> | |
| 6. Program dapat menyimpan solusi dalam berkas .txt | <input checked="" type="checkbox"/> | |
| 7. Program memiliki GUI | | <input checked="" type="checkbox"/> |

Pranala Github

https://github.com/RiciTrisnaP/Tucil1_13522026