

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

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Filbert – 13522021

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

DAFTAR ISI

DAFTAR ISI.....	2
BAB I : DESKRIPSI MASALAH.....	2
BAB II : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL.....	4
BAB III : IMPLEMENTASI PROGRAM DENGAN PYTHON DAN TYPESCRIPT.....	7
1. Framework Backend Flask (Bahasa Pemrograman Python).....	8
a. app.py.....	8
b. program.py.....	11
c. util.py.....	14
2. Framework Frontend NextJS (Bahasa Pemrograman typescript).....	19
a. page.tsx.....	19
BAB IV : EKSPERIMEN.....	21
a. Masukan acak oleh program.....	21
i. Contoh 1.....	21
ii. Contoh 2.....	23
iii. Contoh 3.....	24
iv. Contoh 4.....	26
v. Contoh 5.....	28
vi. Contoh 6.....	29
b. Masukan dengan file .txt.....	32
i. Contoh 7.....	32
ii. Contoh 8.....	34
LAMPIRAN.....	37
Github Repository.....	37
Tabel Spesifikasi.....	37

BAB I : DESKRIPSI MASALAH

Dalam dunia Cyberpunk 2077, "Breach Protocol" adalah mini-game yang merepresentasikan kemampuan karakter untuk meretas sistem keamanan elektronik. Ini memadukan elemen puzzle dan strategi yang menguji pemain untuk meretas dengan efisien dalam batas waktu yang ditentukan. Mini-game ini sering dihadapi saat pemain ingin membuka pintu yang terkunci, mematikan kamera pengawas, atau mengambil alih sistem pertahanan musuh.

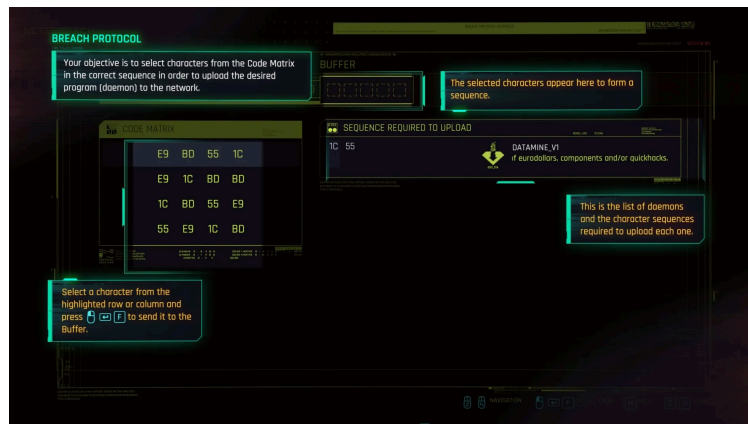
Komponen pada permainan ini antara lain adalah:

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 awalnya memilih token yang ada pada baris pertama, lalu pemain dapat memilih token selanjutnya dari kolom token yang dipilih pertama tadi, kemudian pilih lagi token yang horizontal, vertikal dan seterusnya.
2. Pemain memiliki buffer memori terbatas yang digunakan untuk memasukkan serangkaian kode. Ukuran buffer menentukan berapa banyak urutan yang bisa dimasukkan dan berbeda tergantung pada situasi atau tingkat kemampuan karakter.
3. Sebelum memulai, pemain diberikan satu atau lebih sequence target yang harus diikuti. Ini adalah urutan kode yang harus dipilih pemain dari matriks untuk menyelesaikan meretas.
4. Sekuens selanjutnya akan dicocokkan pada token-token yang berada di buffer.
5. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
6. Setiap sekuens memiliki bobot hadiah atau reward yang variatif.
7. Sekuens memiliki panjang minimal berupa dua token.

Untuk mendapatkan solusi yang paling optimal dalam permainan ini, maka pada laporan ini akan membahas program yang dapat digunakan untuk mencari solusi optimal untuk



menyelesaikan permainan *Breach Protocol* ini dengan algoritma bruteforce.

Gambar 1.1. Permainan *Cyberpunk 2077 Breach Protocol*

BAB II : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL

Algoritma Brute Force merupakan metode sederhana namun efektif dalam menyelesaikan masalah, di mana seluruh kemungkinan solusi diuji satu per satu hingga solusi yang benar ditemukan. Meskipun mungkin tidak selalu optimal dari segi waktu dan sumber daya, Brute Force sering digunakan karena kemudahan implementasinya dan keandalannya dalam menemukan solusi.

Dalam Permainan Cyberpunk 2077 Breach Protocol, penggunaan algoritma Brute Force didasarkan pada beberapa pertimbangan utama. Pertama, permainan ini melibatkan pencocokan sekuens token dari sebuah matrix, dan Brute Force memungkinkan untuk mencoba setiap kemungkinan kombinasi tanpa mengabaikan solusi potensial. Ini menjamin bahwa solusi yang ditemukan adalah yang optimal karena seluruh kemungkinan telah dieksplorasi. Kedua, kompleksitas permainan ini masih dapat ditangani efektif oleh Brute Force, meskipun membutuhkan waktu dan sumber daya komputasi yang intensif. Keterbatasan jumlah token dan sekuens dalam permainan membuat Brute Force menjadi pilihan yang praktis dan dapat diandalkan. Ketiga, kelebihan kemudahan implementasi algoritma Brute Force mempercepat pengembangan awal, pengujian solusi, dan iterasi program.

Dengan sifat yang sederhana, handal, dan universal, Brute Force merupakan pilihan yang cocok untuk menyelesaikan permasalahan pada program ini. Penekanan pada eksplorasi semua kemungkinan solusi, kemampuan menangani kompleksitas permainan, dan kemudahan implementasi menjadikan algoritma ini solusi yang efektif dalam konteks Cyberpunk 2077 Breach Protocol.

Adaptasi Langkah Penggunaan Algoritma Brute Force pada Program

Inisialisasi :

Program dimulai dengan menyiapkan matriks permainan, yang berisi token-token yang tersedia dan menetapkan ukuran buffer sesuai dengan masukan dari pengguna. Tahap ini sangat penting karena menentukan lingkup masalah yang akan diatasi oleh algoritma, termasuk pengaturan awal kondisi permainan yang meliputi token yang tersedia dan batasan langkah yang diperbolehkan berdasarkan ukuran buffer.

Pemilihan Token Awal :

Setelah inisialisasi, algoritma memilih token awal dari baris paling atas matriks sebagai titik awal. Pemilihan ini dilakukan secara iteratif, artinya setiap token pada baris awal akan diuji satu per satu sebagai titik permulaan dalam mencari kombinasi sekuens yang mungkin. Langkah ini mengawali pencarian solusi dengan mengeksplorasi semua kemungkinan yang berasal dari setiap token awal.

Eksplorasi Kemungkinan :

Dengan token awal yang telah dipilih, algoritma kemudian mengeksplorasi setiap kemungkinan langkah berikutnya mengikuti aturan permainan yang memungkinkan gerakan secara bergantian antara horizontal dan vertikal. Pendekatan rekursif digunakan di sini untuk secara sistematis mengeksplorasi setiap kemungkinan jalur dari titik awal yang telah dipilih. Eksplorasi ini melibatkan beberapa konsep kunci:

a. Basis rekursif

Ini adalah kondisi yang menghentikan rekursi lebih lanjut, yaitu ketika panjang jalur yang dieksplorasi sama dengan ukuran buffer. Hal ini memastikan bahwa algoritma tidak menghasilkan jalur yang melebihi batasan permainan, menjaga validitas setiap solusi yang dihasilkan.

b. Rekurens

Proses ini melibatkan eksplorasi setiap langkah potensial dari posisi saat ini, dengan mempertimbangkan semua langkah potensial berikutnya dan melakukan rekursi

dengan jalur yang diperbarui untuk setiap pilihan. Ini memungkinkan penyelidikan menyeluruh atas setiap kemungkinan dan memastikan tidak ada solusi yang terlewat.

c. Backtracking

Ketika algoritma mengembalikan rekursi tanpa menemukan solusi yang optimal, proses backtracking dilakukan. Proses ini memungkinkan algoritma untuk mundur, mencoba alternatif lain yang belum diuji, membatalkan pilihan terakhir, dan menggantinya dengan opsi baru. Mekanisme ini krusial untuk memastikan semua jalur potensial diuji.

Pencocokan Sekuens :

Selama proses eksplorasi, algoritma terus memeriksa isi buffer terhadap sekuens yang didefinisikan. Jika ada kecocokan, reward dari sekuens itu ditambahkan ke skor total, memaksimalkan setiap kesempatan untuk mendapatkan reward.

Optimalisasi Solusi :

Algoritma brute force mencoba semua kombinasi token sampai seluruh matriks terjelajahi atau buffer terisi penuh. Proses ini diulang untuk setiap token awal yang mungkin. Dari semua solusi yang dihasilkan, algoritma memilih solusi dengan total reward tertinggi sebagai solusi optimal.

Evaluasi dan Output :

Setelah menemukan solusi optimal, program kemudian menampilkan jalur dan total reward yang diperoleh kepada pengguna, memberikan strategi efektif untuk meraih skor tertinggi dalam permainan Breach Protocol Cyberpunk 2077.

Melalui langkah-langkah ini, algoritma Brute Force diterapkan dengan sistematis untuk menemukan solusi terbaik dalam permainan Cyberpunk 2077 Breach Protocol. Meskipun pendekatan ini membutuhkan waktu komputasi yang lebih lama, ia memastikan bahwa setiap kemungkinan dijelajahi untuk menemukan jalur dengan solusi dan reward maksimal.

BAB III : IMPLEMENTASI PROGRAM DENGAN PYTHON DAN TYPESCRIPT

Dalam pengembangan solusi untuk permainan Cyberpunk 2077 Breach Protocol, kami menghadirkan sebuah sistem antarmuka pengguna berbasis web (GUI) yang didukung oleh backend yang kuat. Sistem ini dibangun menggunakan kombinasi dari Python dan TypeScript, memanfaatkan kerangka kerja Flask untuk backend dan Next.js untuk frontend. Program backend, yang terdapat dalam file `app.py`, bertanggung jawab atas pemrosesan logika algoritma Brute Force, sedangkan tampilan GUI website dikembangkan dengan TypeScript untuk menyediakan interaksi yang intuitif bagi pengguna.

Selain berinteraksi melalui GUI, sistem ini juga menawarkan kemampuan untuk berinteraksi melalui Command Line Interface (CLI), memberikan alternatif bagi pengguna yang lebih memilih pengalaman berbasis teks. Implementasi CLI ini melibatkan file `program.py`, yang berisi implementasi utama dari algoritma Brute Force, dan `main.py` yang berfungsi sebagai entry point program. File `util.py` juga disertakan untuk menyediakan fungsi-fungsi bantuan dan pelengkap.

Fungsi-Fungsi yang digunakan dalam program akan dideskripsikan sebagai berikut

1. Framework Backend Flask (Bahasa Pemrograman Python)

a. `app.py`

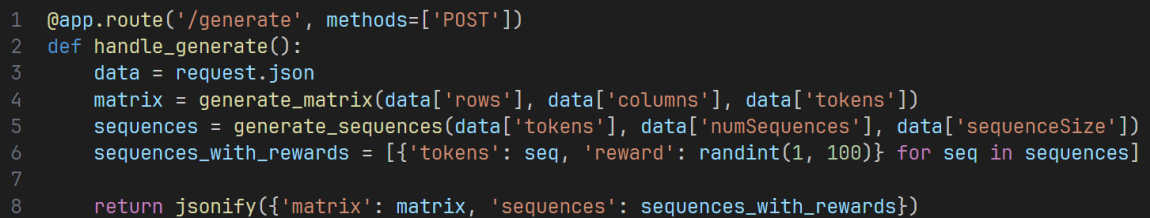
File `app.py` berperan sebagai inti dari backend program, menghubungkan tampilan GUI website dengan logika pemrosesan algoritma Brute Force. Flask digunakan untuk mendefinisikan endpoint-endpoint API yang memfasilitasi komunikasi antara frontend dan backend, memungkinkan pengiriman dan pengolahan data secara efisien. File ini mengelola aspek kritis seperti penerimaan data dari frontend, pemrosesan data dengan algoritma Brute Force, dan pengiriman hasil kembali ke pengguna.

API yang dibangun dalam `app.py` berfungsi sebagai penghubung antara frontend dan backend, memudahkan pertukaran data antara keduanya. Endpoint yang didefinisikan dalam `app.py` menerima data seperti matrix, sequences, dan

buffer size, memprosesnya dengan algoritma Brute Force, dan mengembalikan hasil ke pengguna, termasuk solusi optimal dan waktu eksekusi.

Sebagai backend, app.py berfungsi untuk menerima permintaan dari frontend, memproses permintaan tersebut dengan menggunakan algoritma yang telah ditentukan, dan mengirimkan respons kembali ke pengguna. Dalam program ini, app.py memiliki tanggung jawab untuk:

- Menerima input matrix, sequences, dan buffer size dari GUI.
- Memproses input tersebut dengan memanfaatkan algoritma Brute Force untuk mengidentifikasi jalur dengan reward terbesar.
- Menghasilkan matrix dan sequences secara dinamis bila diperlukan.
- Mengirimkan hasil, termasuk jalur optimal dan detail terkait, kembali ke frontend untuk ditampilkan kepada pengguna.

A screenshot of a code editor with a dark background and light-colored text. The code is written in Python and defines a Flask route for the POST method at the endpoint '/generate'. The function 'handle_generate()' takes a request object and extracts JSON data. It then calls 'generate_matrix()' and 'generate_sequences()' with specific parameters. A list comprehension is used to create 'sequences_with_rewards' by adding a random reward to each sequence. Finally, the function returns a JSON response containing the matrix and the sequences with their rewards.

```
1 @app.route('/generate', methods=['POST'])
2 def handle_generate():
3     data = request.json
4     matrix = generate_matrix(data['rows'], data['columns'], data['tokens'])
5     sequences = generate_sequences(data['tokens'], data['numSequences'], data['sequenceSize'])
6     sequences_with_rewards = [{'tokens': seq, 'reward': randint(1, 100)} for seq in sequences]
7
8     return jsonify({'matrix': matrix, 'sequences': sequences_with_rewards})
```

Gambar 3.1. API generate()

Fungsi 'handle_generate()' berfungsi sebagai pengendali untuk endpoint '/generate', di mana ia mengatur penerimaan data dari frontend yang meliputi ukuran matriks, token, jumlah sekuens, dan ukuran maksimum sekuens. Berdasarkan data ini, fungsi tersebut kemudian membangkitkan matriks permainan dan sekuens yang diisi dengan nilai reward secara acak. Hasilnya dikirim kembali ke frontend dalam format JSON yang berisi matriks dan sekuens beserta reward dari masing-masing sekuens.

```

1 @app.route('/solve', methods=['POST'])
2 def handle_solve():
3     data = request.json
4     sequences_with_rewards = {tuple(seq['tokens']): seq['reward'] for seq in data['sequences']}
5
6     start = time.time()
7     optimal_path, optimal_path_coords, max_reward = find_optimal_path(data['matrix'], sequences_with_rewards, data['bufferSize'])
8     end = time.time()
9     execution_time = (end-start) * 1000
10
11     return jsonify({'maxReward': max_reward,
12                   'optimalPath': optimal_path,
13                   'coordinates': optimal_path_coords,
14                   'executionTime': execution_time})

```

Gambar 3.2. API solve()

Endpoint `/solve` dikelola oleh fungsi `handle_solve()`, yang menerima data matriks, sekuens berserta reward, dan buffer size dari frontend. Fungsi ini bertanggung jawab untuk memproses data tersebut menggunakan algoritma untuk menemukan jalur optimal dalam permainan. Selain itu, fungsi ini juga menghitung waktu eksekusi algoritma sebagai bagian dari respons ke pengguna, memberikan insight tambahan mengenai performa solusi yang dihasilkan. Output dari proses ini berupa jalur optimal, koordinat jalur, reward maksimal, dan waktu eksekusi dikirim kembali ke frontend dalam bentuk JSON.

```

1 UPLOAD_FOLDER = os.path.dirname(os.path.abspath(__file__))
2 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
3
4 @app.route('/upload', methods=['POST'])
5 def upload_file():
6     file = request.files.get('file')
7     if file:
8         if not file.filename.endswith('.txt'):
9             return jsonify(message="Please upload a .txt file."), 400
10
11         filename = secure_filename(file.filename)
12         filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
13         file.save(filepath)
14
15         try:
16             buffer_size, matrix, number_of_sequences, sequences, rewards = read_input_file(filepath)
17             sequences_with_rewards = [{'tokens': list(seq), 'reward': reward} for seq, reward in sequences.items()]
18
19             os.remove(filepath)
20
21             return jsonify({
22                 'bufferSize': buffer_size,
23                 'matrix': matrix,
24                 'sequences': sequences_with_rewards
25             }), 200
26         except (ValueError, IOError) as e:
27             os.remove(filepath)
28             return jsonify(message=str(e)), 400
29     else:
30         return jsonify(message="No file uploaded."), 400
31

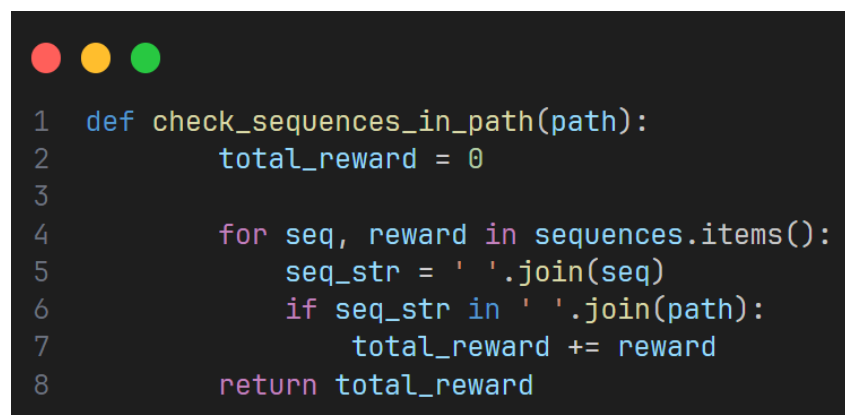
```

Gambar 3.3. API upload()

Sementara itu, endpoint `/upload` diatur oleh fungsi `upload_file()`, yang memungkinkan pengguna untuk mengupload file teks yang berisi data permainan. Fungsi ini memastikan bahwa file yang diupload memenuhi kriteria yang ditentukan (misalnya, harus berformat .txt) sebelum memprosesnya untuk mengambil informasi seperti buffer size, matriks, dan sekuens beserta reward. Setelah data diperoleh, file tersebut dihapus dari server untuk menjaga kebersihan direktori kerja. Kesalahan dalam proses upload atau pemrosesan file di handle dengan mengembalikan pesan error yang sesuai, sedangkan hasil pemrosesan yang berhasil dikembalikan dalam format JSON yang siap dikonsumsi oleh frontend.

b. program.py

File program.py berisi kumpulan fungsi yang dirancang untuk mencari solusi optimal dari permainan dengan eksplorasi semua kemungkinan jalur yang dapat diambil. Fokus utama dari program.py adalah pada pengolahan logika permainan dan pencarian solusi yang efisien melalui pendekatan Brute Force. Inti dari program ini terdiri dari tiga fungsi utama: `check_sequences_in_path`, `generate_path`, dan fungsi `find_optimal_path` itu sendiri, yang bekerja bersama untuk mengeksplorasi semua kemungkinan kombinasi dan mengidentifikasi solusi terbaik.



```
1 def check_sequences_in_path(path):
2     total_reward = 0
3
4     for seq, reward in sequences.items():
5         seq_str = ' '.join(seq)
6         if seq_str in ' '.join(path):
7             total_reward += reward
8     return total_reward
```

Gambar 3.4. fungsi check_sequences_in_path

Fungsi `check_sequences_in_path` bertanggung jawab untuk menghitung total reward dari jalur yang sedang dieksplorasi berdasarkan sekuens token yang ditemukan dalam jalur tersebut. Fungsi ini menerima daftar token yang sedang diperiksa dan membandingkannya dengan setiap sekuens yang telah didefinisikan, mengakumulasi reward untuk setiap pencocokan sekuens yang ditemukan. Hal ini memungkinkan penilaian nilai reward dari sebuah jalur berdasarkan seberapa banyak sekuens yang berhasil dipenuhi.

```
1 def generate_path(row, col, path, coords, step, current_reward, direction):
2     nonlocal max_reward, optimal_path, optimal_path_coords
3
4     potential_reward = current_reward + (buffer_size - step) * max_sequence_reward
5     if potential_reward < max_reward:
6         return
7
8     if step == buffer_size or row >= rows or row < 0 or col >= cols or col < 0:
9         return
10
11     new_path = path + [matrix[row][col]]
12     new_coords = coords + [(row+1, col+1)]
13     new_reward = check_sequences_in_path(new_path)
14
15     if new_reward > max_reward:
16         max_reward = new_reward
17         optimal_path = new_path
18         optimal_path_coords = new_coords
19
20     if direction == 'row':
21         for next_col in range(cols):
22             if next_col != col:
23                 generate_path(row, next_col, new_path, new_coords, step + 1, new_reward, 'col')
24     else:
25         for next_row in range(rows):
26             if next_row != row:
27                 generate_path(next_row, col, new_path, new_coords, step + 1, new_reward, 'row')
```

Gambar 3.5. fungsi generate_path

Fungsi `generate_path` adalah inti dari eksplorasi rekursif, yang secara sistematis menelusuri matriks permainan untuk mencari kombinasi token yang optimal. Dengan menerima posisi saat ini, jalur yang telah diambil, koordinat jalur, langkah saat ini, reward yang telah dikumpulkan, dan arah gerakan selanjutnya, fungsi ini berekursi untuk memperluas jalur dengan mengeksplorasi setiap langkah potensial selanjutnya. Untuk setiap token yang ditambahkan ke jalur, fungsi memeriksa apakah penambahan token tersebut meningkatkan total reward yang dikumpulkan. Jika ya, jalur tersebut dianggap sebagai kandidat solusi optimal. Fungsi ini juga memanfaatkan pemangkasan (pruning) untuk

meningkatkan efisiensi dengan menghentikan eksplorasi jalur yang tidak mungkin melebihi reward maksimal yang telah ditemukan.

```
1 def find_optimal_path(matrix, sequences, buffer_size):
2     rows, cols = len(matrix), len(matrix[0])
3     max_reward = 0
4     optimal_path = []
5     optimal_path_coords = []
6     max_sequence_reward = max(sequences.values())
7
8     def check_sequences_in_path(path):
9         total_reward = 0
10
11         for seq, reward in sequences.items():
12             seq_str = ' '.join(seq)
13             if seq_str in ' '.join(path):
14                 total_reward += reward
15         return total_reward
16
17 def generate_path(row, col, path, coords, step, current_reward, direction):
18     nonlocal max_reward, optimal_path, optimal_path_coords
19
20     potential_reward = current_reward + (buffer_size - step) * max_sequence_reward
21     if potential_reward < max_reward:
22         return
23
24     if step == buffer_size or row >= rows or row < 0 or col >= cols or col < 0:
25         return
26
27     new_path = path + [matrix[row][col]]
28     new_coords = coords + [(row+1, col+1)]
29     new_reward = check_sequences_in_path(new_path)
30
31     if new_reward > max_reward:
32         max_reward = new_reward
33         optimal_path = new_path
34         optimal_path_coords = new_coords
35
36     if direction == 'row':
37         for next_col in range(cols):
38             if next_col != col:
39                 generate_path(row, next_col, new_path, new_coords, step + 1, new_reward, 'col')
40     else:
41         for next_row in range(rows):
42             if next_row != row:
43                 generate_path(next_row, col, new_path, new_coords, step + 1, new_reward, 'row')
44
45     for col_start in range(cols):
46         generate_path(0, col_start, [], [], 0, 0, 'col')
47
48     return optimal_path, optimal_path_coords, max_reward
```

Gambar 3.6. Fungsi find_optimal_path

Fungsi `'find_optimal_path'` mengkoordinasikan proses pencarian jalur optimal. Fungsi ini menginisialisasi variabel yang diperlukan, termasuk dimensi matriks, reward maksimal yang ditemukan, jalur optimal, dan koordinat jalur optimal. Kemudian, untuk setiap kolom di baris pertama matriks, fungsi memulai eksplorasi jalur menggunakan `'generate_path'`. Ini memastikan bahwa semua kemungkinan titik awal dijelajahi. Setelah semua jalur dieksplorasi, fungsi mengembalikan jalur dengan reward terbesar, beserta koordinatnya, sebagai solusi optimal.

Keseluruhan proses ini menggabungkan strategi pencarian rekursif dan pemangkasan untuk menemukan jalur yang tidak hanya memenuhi sekuens yang didefinisikan tetapi juga mengoptimalkan reward yang diperoleh. Dengan cara ini, program mampu menyelesaikan tantangan permainan Breach Protocol dengan mencari strategi terbaik yang memungkinkan pemain meraih skor maksimal.

c. util.py

File `util.py` berisi semua fungsi-fungsi tambahan atau pembantu untuk melengkapi fungsi-fungsi utama pada program ini.



```

1  def read_input_file(file_path):
2      try:
3          with open(file_path, 'r') as file:
4              buffer_size = int(file.readline().strip())
5              if not file_path.endswith('.txt'):
6                  raise ValueError("Only .txt files are allowed.")
7              if buffer_size <= 0:
8                  raise ValueError("Buffer size must be positive.")
9
10             rows, columns = map(int, file.readline().split())
11             if rows <= 0 or columns <= 0:
12                 raise ValueError("Rows and columns must be positive.")
13
14             matrix = [file.readline().strip().split() for _ in range(rows)]
15             if any(len(row) != columns for row in matrix):
16                 raise ValueError("Matrix dimensions mismatch.")
17
18             number_of_sequences = int(file.readline().strip())
19             if number_of_sequences <= 0:
20                 raise ValueError("Number of sequences must be positive.")
21
22             sequences = {}
23             rewards = []
24             for _ in range(number_of_sequences):
25                 sequence = tuple(file.readline().strip().split())
26                 if not sequence:
27                     raise ValueError("Empty sequence found.")
28                 reward = int(file.readline().strip())
29                 sequences[sequence] = reward
30                 rewards.append(reward)
31
32             return buffer_size, matrix, number_of_sequences, sequences, rewards
33     except ValueError as e:
34         raise ValueError(f"Invalid file format: {e}")
35     except Exception as e:
36         raise IOError(f"Error reading file: {e}")

```

Gambar 3.7. Fungsi read_input_file

Fungsi `read_input_file` bertugas membaca dan memproses data dari file teks yang diberikan. Fungsi ini melakukan beberapa pengecekan untuk memastikan bahwa data dalam file sesuai dengan format yang diharapkan. Ini termasuk memeriksa ukuran buffer yang harus positif, dimensi matriks yang harus sesuai dan positif, serta memverifikasi bahwa jumlah urutan dan urutan itu sendiri valid. Jika ada kesalahan dalam format atau isi file, fungsi akan melemparkan kesalahan dengan pesan yang sesuai untuk memberitahu pengguna tentang masalah tersebut.


```

1 def get_valid_filename(prompt, initial_filename=None, directory="output"):
2     while True:
3         filename = initial_filename or input(prompt)
4         if not filename.endswith('.txt'):
5             print("The file must have a .txt extension.")
6             initial_filename = None # Reset initial_filename to ask again
7             continue
8
9         full_path = os.path.join(directory, filename)
10        if not os.path.exists(full_path):
11            return filename
12        else:
13            print(f"File {filename} already exists. Please enter a different name.")
14            initial_filename = None
15
16 def save_output_to_file(reward, sequence, coordinates, execution_time_ms, file_name=None):
17     directory = "output"
18     os.makedirs(directory, exist_ok=True)
19
20     if file_name is None:
21         file_name_prompt = "Enter the desired file name (with .txt extension): "
22         file_name = get_valid_filename(file_name_prompt, directory=directory)
23     elif not file_name.endswith('.txt'):
24         print("The file must have a .txt extension.")
25
26     full_path = os.path.join(directory, file_name)
27
28     with open(full_path, 'w') as file:
29         file.write(f"{reward}\n")
30         sequence_str = ' '.join(sequence) if sequence else "No sequence found"
31         file.write(f"{sequence_str}\n")
32         for coord in coordinates:
33             file.write(f"{coord[0]}, {coord[1]}\n")
34         file.write(f"{execution_time_ms} ms\n")
35
36     print(f"Results saved to {full_path}.")

```

Gambar 3.8. Fungsi `get_valid_filename` dan `save_output_to_file`

Fungsi `get_valid_filename` memastikan bahwa nama file yang akan digunakan untuk menyimpan output belum ada. Jika file dengan nama yang sama sudah ada, pengguna akan diminta untuk memasukkan nama baru. Ini mencegah penulisan ulang file yang tidak disengaja. Fungsi `save_output_to_file` digunakan untuk menyimpan hasil pemrosesan algoritma ke dalam file teks. Ini mencakup menulis reward, urutan token yang optimal, koordinat dari urutan tersebut, dan waktu eksekusi algoritma. Jika file dengan nama yang diberikan sudah ada, akan dipanggil `get_valid_filename` untuk mendapatkan nama file yang valid.




```

1  def generate_matrix(row, column, tokens):
2      matrix = []
3      for _ in range(row):
4          matrix.append([random.choice(tokens) for _ in range(column)])
5      return matrix
6
7  def generate_sequences(tokens, number_of_sequences, max_sequence_size):
8      sequences = []
9      for _ in range(number_of_sequences):
10         sequence_length = random.randint(1, max_sequence_size)
11         sequences.append([random.choice(tokens) for _ in range(sequence_length)])
12     return sequences

```

Gambar 3.9. Fungsi `generate_matrix` dan `generate_sequences`

`generate_matrix` dan `generate_sequences` adalah dua fungsi yang bertugas menghasilkan matriks permainan dan urutan token secara acak berdasarkan parameter yang diberikan. `generate_matrix` membuat matriks berdasarkan jumlah baris dan kolom yang ditentukan, mengisi setiap sel dengan token acak dari daftar token yang tersedia. `generate_sequences`, di sisi lain, menghasilkan daftar urutan token, dengan setiap urutan memiliki panjang acak yang tidak melebihi batas maksimal yang ditentukan.



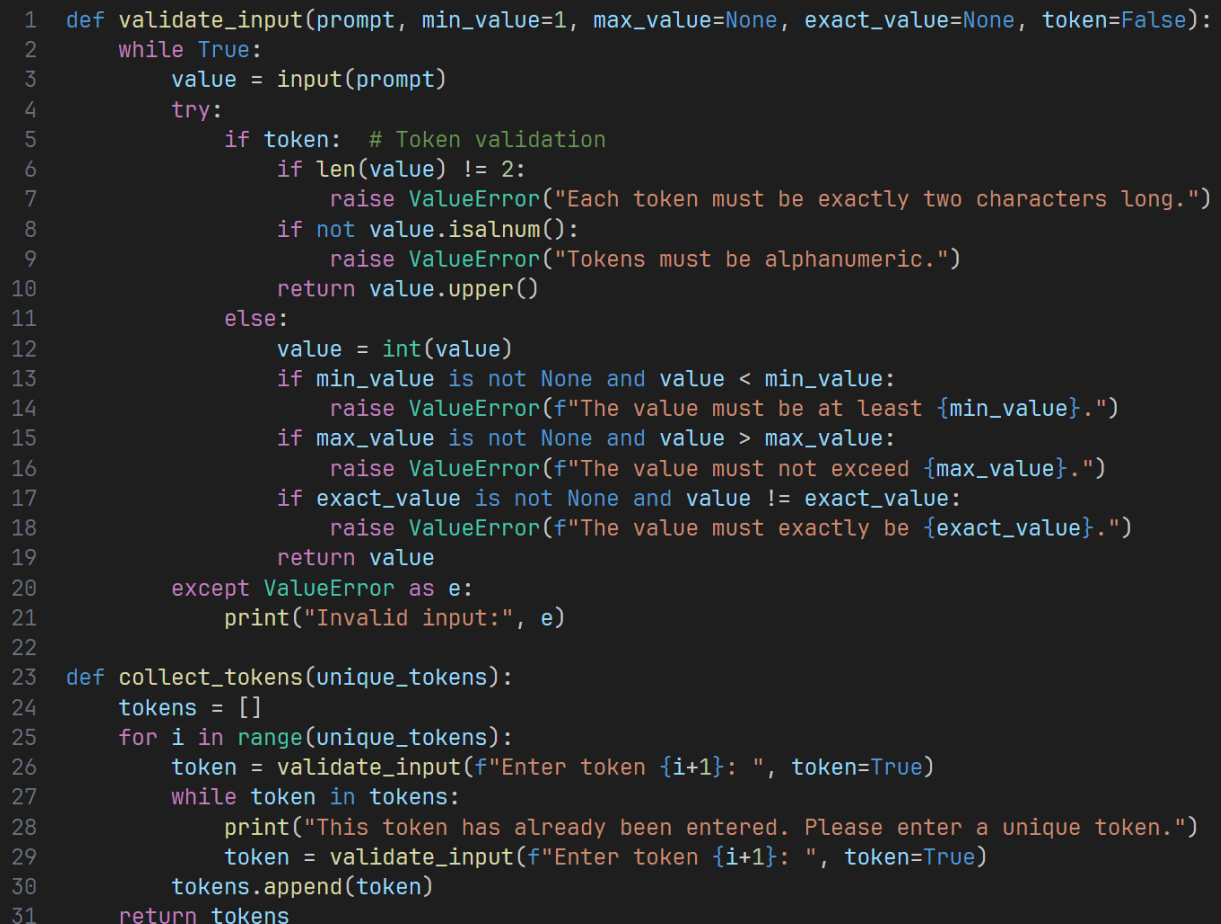
```

1  def show_code_matrix(matrix, sequences, rewards):
2      print("\nCODE MATRIX")
3      for row in matrix:
4          print(" | ".join(row))
5      print("\nSequences:")
6      for sequence in sequences:
7          print(sequence)
8      print("\nRewards:")
9      for reward in rewards:
10         print(reward)
11     print()

```

Gambar 3.10. Fungsi `show_code_matrix`

`show_code_matrix` menyajikan representasi visual dari matriks kode dan urutan token beserta rewardnya kepada pengguna. Ini berguna untuk verifikasi visual oleh pengguna tentang data yang dihasilkan atau diproses oleh program.



```
1 def validate_input(prompt, min_value=1, max_value=None, exact_value=None, token=False):
2     while True:
3         value = input(prompt)
4         try:
5             if token: # Token validation
6                 if len(value) != 2:
7                     raise ValueError("Each token must be exactly two characters long.")
8                 if not value.isalnum():
9                     raise ValueError("Tokens must be alphanumeric.")
10                return value.upper()
11            else:
12                value = int(value)
13                if min_value is not None and value < min_value:
14                    raise ValueError(f"The value must be at least {min_value}.")
15                if max_value is not None and value > max_value:
16                    raise ValueError(f"The value must not exceed {max_value}.")
17                if exact_value is not None and value != exact_value:
18                    raise ValueError(f"The value must exactly be {exact_value}.")
19                return value
20        except ValueError as e:
21            print("Invalid input:", e)
22
23 def collect_tokens(unique_tokens):
24     tokens = []
25     for i in range(unique_tokens):
26         token = validate_input(f"Enter token {i+1}: ", token=True)
27         while token in tokens:
28             print("This token has already been entered. Please enter a unique token.")
29             token = validate_input(f"Enter token {i+1}: ", token=True)
30         tokens.append(token)
31     return tokens
```

Gambar 3.11. Fungsi validate_input dan collect_tokens

`validate_input` merupakan fungsi umum untuk memvalidasi input pengguna. Fungsi ini meminta pengguna untuk memasukkan nilai dan melakukan pengecekan apakah nilai tersebut dalam rentang atau sesuai dengan kondisi yang ditetapkan. Jika input tidak valid, pengguna akan diminta untuk memasukkan ulang sampai input yang valid diterima.

Sedangkan `collect_tokens` memungkinkan pengumpulan token unik dari pengguna. Untuk setiap token, dijamin tidak ada duplikasi karena setiap token baru yang dimasukkan akan dicek terlebih dahulu apakah sudah ada dalam daftar.

2. Framework Frontend NextJS (Bahasa Pemrograman TypeScript)

a. page.tsx

A screenshot of a code editor with a dark background and light-colored text. The code is written in TypeScript and is the `handleSubmit` function for a Next.js page. It includes logic for preventing default behavior, setting result to null, processing matrix and sequence data, making an API call to `http://localhost:5000/solveHome` with a POST method and JSON body, handling errors, and logging the result. The code is numbered from 1 to 38 on the left side.

```
1  const handleSubmit = async (e) => {
2      e.preventDefault();
3      setResult(null);
4      const matrixRows = matrix.trim().split('\n').map(row => row.trim().split(/\s+/));
5      const sequenceLines = sequences.trim().split('\n');
6      const sequencePairs = [];
7      for (let i = 0; i < sequenceLines.length; i += 2) {
8          sequencePairs.push({
9              sequence: sequenceLines[i].trim().split(/\s+/),
10             reward: parseInt(sequenceLines[i + 1], 10)
11         });
12     }
13
14     try {
15         const response = await fetch('http://localhost:5000/solveHome', {
16             method: 'POST',
17             headers: {
18                 'Content-Type': 'application/json',
19             },
20             body: JSON.stringify({
21                 bufferSize: parseInt(bufferSize, 10),
22                 matrix: matrixRows,
23                 sequences: sequencePairs
24             }),
25         });
26
27         if (!response.ok) {
28             throw new Error(`HTTP error! status: ${response.status}`);
29         }
30
31         const resultData = await response.json();
32         setResult(resultData);
33         console.log(resultData);
34     } catch (error) {
35         console.error("Failed to solve the algorithm:", error);
36         setResult(result);
37     }
38 }
```

Gambar 3.12. fungsi handleSubmit pada frontend

Dalam pengembangan frontend program ini, Next JS digunakan sebagai kerangka kerja utama untuk menciptakan antarmuka pengguna yang interaktif dan adaptif.

Salah satu fitur penting dalam antarmuka ini adalah fungsi `handleSolve`, yang berperan penting dalam interaksi antara pengguna dan sistem. Fungsi ini diaktifkan ketika pengguna menekan tombol "Solve" yang tersedia di formulir input.

Proses yang terjadi di dalam fungsi `handleSolve` dimulai dengan aktivasi indikator loading yang memberitahu pengguna bahwa proses pemecahan masalah sedang berlangsung. Ini dilakukan melalui pengaturan state loading menjadi true. Selanjutnya, fungsi mengumpulkan semua data yang telah di input oleh pengguna, termasuk matriks yang dihasilkan, sekuens, dan ukuran buffer. Data ini kemudian diorganisir ke dalam format yang sesuai dan dikirim ke backend melalui API dengan menggunakan metode POST.

Komunikasi dengan backend dilakukan melalui endpoint `/solve`, di mana data dikirimkan dan diproses. Setelah backend selesai memproses data dan mengembalikan hasilnya, frontend akan meng-update state dengan informasi tentang jalur optimal, reward maksimal, koordinat jalur, dan waktu eksekusi. Seluruh proses ini dijalankan secara asinkron untuk memastikan bahwa UI tetap responsif selama operasi berlangsung.

Jika terjadi error selama proses pengiriman atau pemrosesan data, sistem akan menangkap dan menampilkan pesan kesalahan untuk memberitahu pengguna tentang masalah yang terjadi. Setelah proses selesai, baik berhasil maupun gagal, indikator loading akan dinonaktifkan dengan mengatur kembali state loading menjadi false.

Selain fungsi `handleSolve`, untuk frontend juga mengandung berbagai komponen lain yang berkolaborasi untuk membangun pengalaman pengguna yang lengkap. Ini termasuk elemen-elemen untuk input data, tampilan hasil, dan navigasi antar bagian aplikasi. Penggunaan Tailwind CSS sebagai framework CSS memfasilitasi pembuatan tampilan yang estetik dengan pendekatan utility-first,

memungkinkan desain yang responsif dan mudah disesuaikan dengan berbagai ukuran layar.

BAB IV : EKSPERIMEN

a. Masukan acak oleh program

i. Contoh 1

1. SPECIFY BUFFER SIZE	2. UNIQUE TOKENS
<input type="text" value="4"/>	<input type="text" value="7A 55 1C E9"/>
3. ENTER MATRIX SIZE	4. ENTER SEQUENCES
<input type="text" value="4"/> x <input type="text" value="4"/>	Enter Sequence Amount : <input type="text" value="3"/> Enter Maximal Sequences Amount : <input type="text" value="4"/>
<input type="button" value="Generate Matrix and Sequences"/>	

Gambar 4.1. Masukan acak contoh 1

Matrix

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

Sequences and Rewards

Sequence 1 : E9
Reward : 99

Sequence 2 : 7A 7A E9 E9
Reward : 36

Sequence 3 : 7A
Reward : 21

Solve Optimal Path

View Result

RESULT

Partial Solution Found!

Max Reward : 120
Best Path : 55 -> E9 -> 55 -> 7A
Best Path Coordinates : (1, 1) -> (2, 1) -> (2, 3) -> (3, 3)
Execution Time : 1.13 ms

Download Result

Close

Gambar 4.2. Keluaran acak contoh 1

ii. Contoh 2

1. SPECIFY BUFFER SIZE	2. UNIQUE TOKENS
<input type="text" value="5"/>	<input type="text" value="7A 55 1C E9 BD"/>
3. ENTER MATRIX SIZE	4. ENTER SEQUENCES
<div><input type="text" value="5"/> X <input type="text" value="4"/></div>	<div>Enter Sequence Amount : <input type="text" value="4"/></div> <div>Enter Maximal Sequences Amount : <input type="text" value="5"/></div>
<div>Generate Matrix and Sequences</div>	

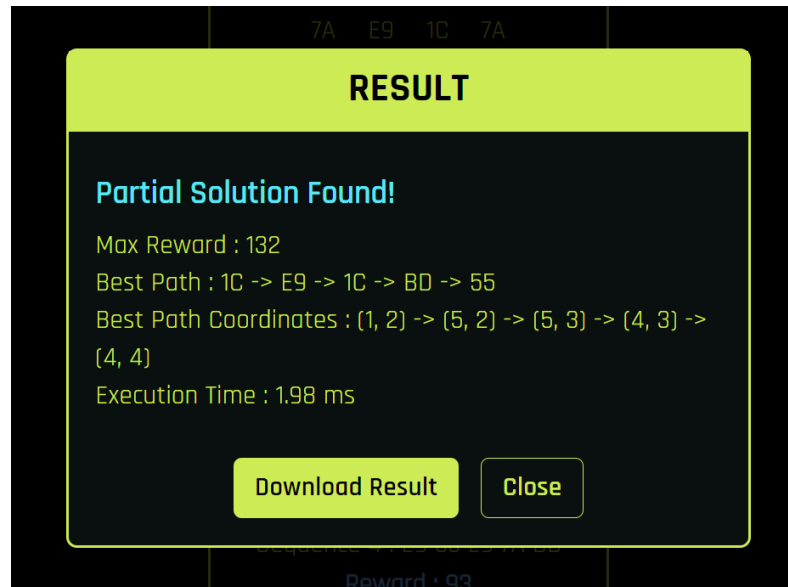
Gambar 4.3. Masukan acak contoh 2

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

Sequences and Rewards
Sequence 1 : E9 1C BD Reward : 48
Sequence 2 : 55 55 E9 Reward : 80
Sequence 3 : BD 55 Reward : 84
Sequence 4 : E9 55 E9 7A BD Reward : 93

Solve Optimal Path

View Result



Gambar 4.4. Keluaran acak contoh 2

iii. Contoh 3

1. SPECIFY BUFFER SIZE <input type="text" value="5"/>	2. UNIQUE TOKENS <input type="text" value="7A 55 1C E9 BD 1D FF"/>
3. ENTER MATRIX SIZE <div> <input type="text" value="6"/> x <input type="text" value="6"/> </div>	4. ENTER SEQUENCES Enter Sequence Amount : <input type="text" value="4"/> <input type="button" value="⬇"/> Enter Maximal Sequences Amount : <input type="text" value="5"/>
<input type="button" value="Generate Matrix and Sequences"/>	

Gambar 4.5. Masukan acak contoh 3

Matrix

1D	1C	1C	BD	1C	E9
1C	E9	7A	BD	1C	1C
BD	E9	1C	E9	1C	1D
FF	7A	1D	7A	55	7A
E9	E9	55	BD	55	1D
1C	55	1D	7A	BD	FF

Sequences and Rewards

Sequence 1 : E9 FF E9 FF
Reward : 14

Sequence 2 : 1C BD
Reward : 45

Sequence 3 : E9 7A 1C
Reward : 28

Sequence 4 : 7A
Reward : 68

Solve Optimal Path

View Result

RESULT

Partial Solution Found!

Max Reward : 141

Best Path : 1C -> E9 -> 7A -> 1C -> BD

Best Path Coordinates : (1, 2) -> (2, 2) -> (2, 3) -> (1, 3) -> (1, 4)

Execution Time : 1.51 ms

Download Result

Close

Gambar 4.6. Keluaran acak contoh 3

iv. Contoh 4

1. SPECIFY BUFFER SIZE <input type="text" value="6"/>	2. ENTER UNIQUE TOKENS <input type="text" value="7A 55 1C E9 BD 1D FF KI AA"/>
3. ENTER MATRIX SIZE <div><input type="text" value="7"/> x <input type="text" value="7"/></div>	4. ENTER SEQUENCES Enter Sequence Amount : <input type="text" value="3"/> Enter Maximal Sequences Amount : <input type="text" value="3"/>

Gambar 4.7. Masukan acak contoh 4

Matrix

E9	1D	BD	55	7A	1C	55
KI	1D	55	BD	KI	AA	7A
7A	7A	1D	1D	AA	55	7A
7A	AA	55	55	1C	BD	55
7A	1C	1C	AA	1D	KI	1D
FF	AA	FF	55	E9	E9	BD
7A	55	BD	7A	1D	7A	1C

Sequences and Rewards

Sequence 1 : BD KI KI

Reward : 35

Sequence 2 : 7A BD KI

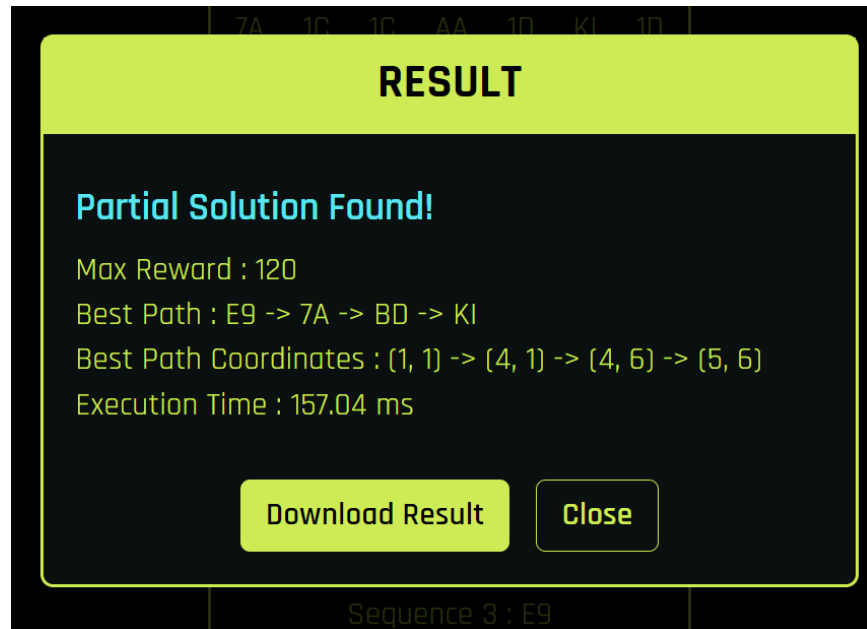
Reward : 35

Sequence 3 : E9

Reward : 85

Solve Optimal Path

View Result



Gambar 4.8. Keluaran acak contoh 4

v. **Contoh 5**

1. SPECIFY BUFFER SIZE <input type="text" value="3"/>	2. UNIQUE TOKENS <input type="text" value="7A 55 1C E9 BD 1D FF KI AA"/>
3. ENTER MATRIX SIZE <div> <input type="text" value="3"/> x <input type="text" value="4"/> </div>	4. ENTER SEQUENCES Enter Sequence Amount : <input type="text" value="3"/> <input type="button" value="↩"/> Enter Maximal Sequences Amount : <input type="text" value="3"/>
<input type="button" value="Generate Matrix and Sequences"/>	

Gambar 4.9. Masukan acak contoh 5

Matrix

AA	E9	E9	E9
E9	1D	1D	BD
AA	AA	FF	1D

Sequences and Rewards

Sequence 1 : 1D 55 1C
Reward : 74

Sequence 2 : E9 1C
Reward : 36

Sequence 3 : 7A
Reward : 64

Solve Optimal Path

View Result

RESULT

There are no sequences.

Close

Gambar 4.10. Keluaran acak contoh 5

vi. Contoh 6

1. SPECIFY BUFFER SIZE <input type="text" value="5"/>	2. UNIQUE TOKENS <input type="text" value="7A 55 1C E9 BD 1D FF KI AA"/>
3. ENTER MATRIX SIZE <div><input type="text" value="7"/> x <input type="text" value="10"/></div>	4. ENTER SEQUENCES Enter Sequence Amount : <input type="text" value="3"/> Enter Maximal Sequences Amount : <input type="text" value="3"/>
<div>Generate Matrix and Sequences</div>	

Gambar 4.11. Masukan acak contoh 6

Matrix

1C	FF	KI	E9	1C	KI	AA	7A	FF	AA
55	KI	1C	1D	55	1D	7A	FF	1C	1D
1D	E9	1D	FF	1D	BD	55	E9	1D	KI
BD	1C	55	7A	1D	1C	KI	FF	1C	KI
7A	E9	AA	55	BD	FF	KI	1C	KI	1C
1D	7A	FF	E9	7A	55	55	7A	1C	1C
E9	AA	55	1D	KI	E9	BD	1D	7A	1D

Sequences and Rewards

Sequence 1 : KI

Reward : 17

Sequence 2 : E9 7A 1C

Reward : 5

Sequence 3 : E9

Reward : 67

Solve Optimal Path

View Result

7A E9 AA 55 BD FF KI 1C KI 1C

RESULT

Full Solution Found!

Max Reward : 89

Best Path : 1C -> E9 -> 7A -> 1C -> KI

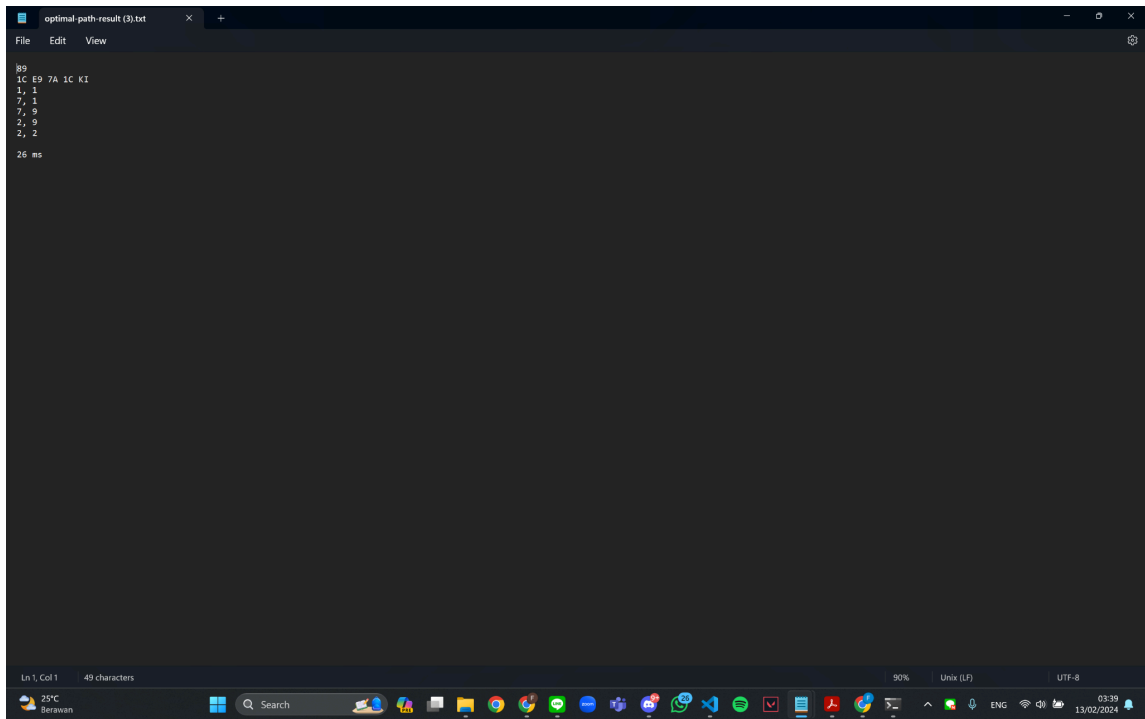
Best Path Coordinates : (1, 1) -> (7, 1) -> (7, 9) -> (2, 9) -> (2, 2)

Execution Time : 25.60 ms

Download Result

Close

Gambar 4.12. Keluaran acak contoh 6



The image shows a text editor window titled "optimal-path-result (3).txt". The editor has a dark theme and a menu bar with "File", "Edit", and "View". The content of the file is as follows:

```
B9
1C E9 7A 1C KI
1, 1
7, 1
7, 9
2, 9
2, 2
26 ms
```

The status bar at the bottom indicates "Ln 1, Col 1" and "49 characters". The Windows taskbar is visible at the bottom, showing the Start button, a search bar, and various application icons. The system tray on the right shows the temperature as 25°C, the location as Berawan, and the date and time as 03:39 on 13/02/2024.

Gambar 4.13. Tampilan file tersimpan

b. Masukkan dengan file .txt

i. Contoh 7

```
7
6 6
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
3
BD 7A BD
15
BD 1C 55
20
BD E9 1C E9
30
```

Choose File

input (2).txt

Upload

Matrix

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

Sequences and Rewards

Sequence 1 : BD 7A BD

Reward : 15

Sequence 2 : BD 1C 55

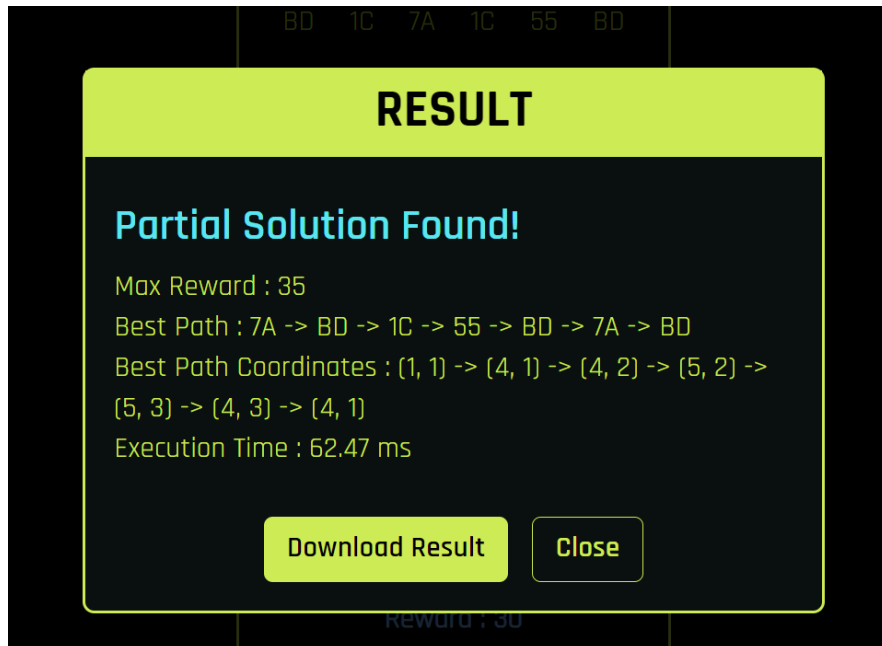
Reward : 20

Sequence 3 : BD E9 1C E9

Reward : 30

Solve Optimal Path

Gambar 4.14. Tampilan masukan dengan upload file contoh 7



Gambar 4.15. Tampilan keluaran dengan upload file contoh 7

ii. Contoh 8

```
7
6 6
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
3
BD 7A BD
10
BD 1C 55
20
BD E9 1C E9
30
```

Choose File

input (2).txt

Upload

Matrix

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

Sequences and Rewards

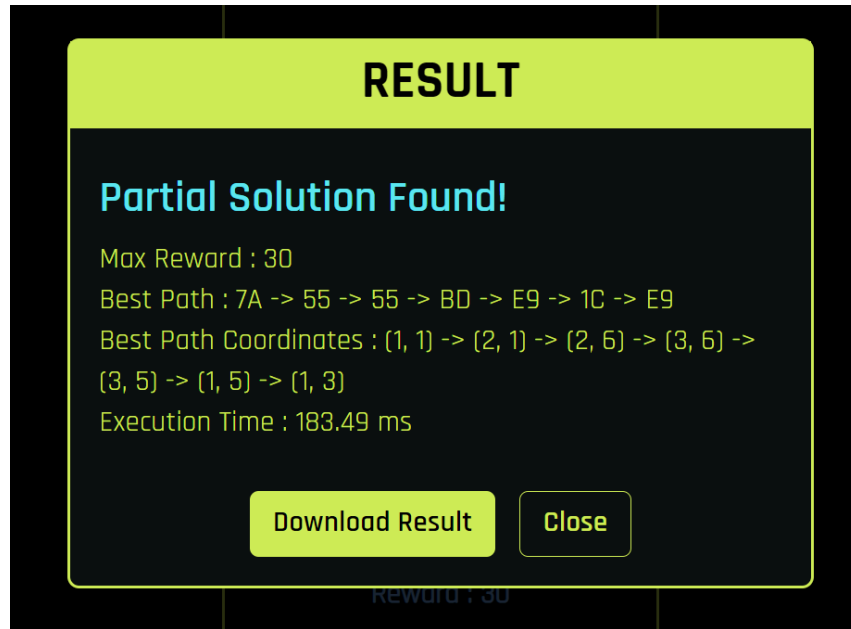
Sequence 1 : BD 7A BD
Reward : 10

Sequence 2 : BD 1C 55
Reward : 20

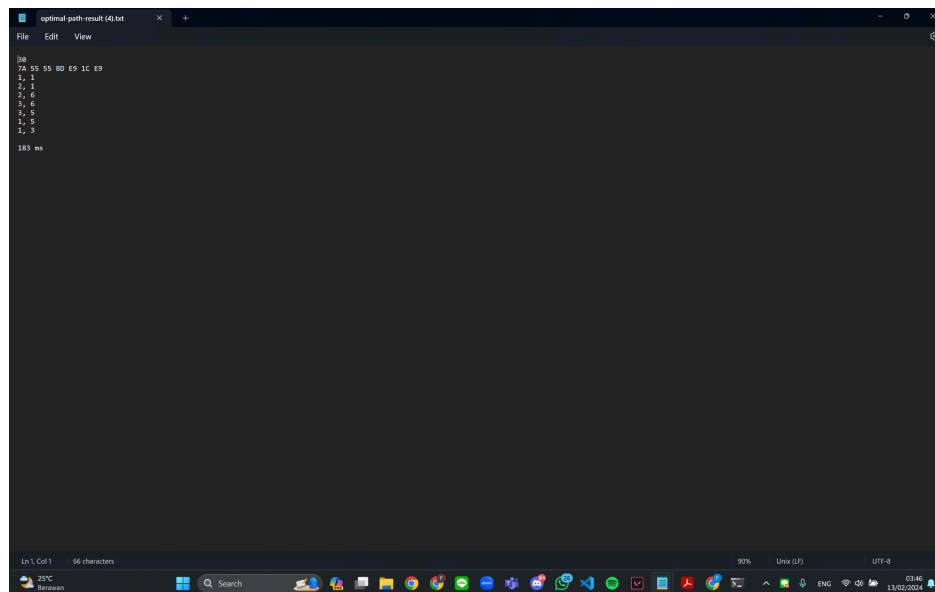
Sequence 3 : BD E9 1C E9
Reward : 30

Solve Optimal Path

Gambar 4.16. Tampilan masukan dengan file contoh 8



Gambar 4.17. Tampilan keluaran dengan file contoh 8



Gambar 4.19. Tampilan hasil penyimpanan file contoh 8

Dengan adanya bab eksperimen, dapat terlihat bahwa program dengan GUI website sudah berhasil menjalankan aktivitas-aktivitas program sesuai dengan spesifikasi. Program terbukti dapat melakukan pembacaan file dengan metode masukan dengan upload file (seperti pada contoh 7 dan contoh 8) dan juga dengan metode masukan acak (seperti pada contoh 1, contoh 2, contoh 3, contoh 4, contoh 5, dan contoh 6). Program juga dapat melakukan algoritma bruteforce

dan menampilkan solusi setelah pengguna melakukan *solve* atas masukan yang diberikan. Program juga dapat melakukan penyimpanan solusi jika pengguna menggunakan fitur *Download Solution*, yang akan dikirimkan ke direktori lokal download/ pengguna.

NOTE : format koordinat yang dipakai adalah (row,col)

LAMPIRAN

Github Repository

https://github.com/Filbert88/Tucil1_13522021

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	✓	