

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

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Albert – 13522081

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

DAFTAR ISI

DAFTAR ISI.....	2
BAB I : DESKRIPSI MASALAH.....	4
BAB II : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL.....	5
BAB III : IMPLEMENTASI PROGRAM DENGAN PYTHON DAN JAVASCRIPT.....	8
1. Framework Backend Flask (Bahasa Pemrograman Python).....	8
a. app.py.....	8
b. algo.py.....	13
2. Framework Frontend ReactJS (Bahasa Pemrograman Javascript).....	20
a. App.jsx.....	20
BAB IV : EKSPERIMEN.....	22
a. Masukan acak oleh program.....	22
i. Contoh 1.....	22
ii. Contoh 2.....	22
iii. Contoh 3.....	23
iv. Contoh 4.....	24
v. Contoh 5.....	25
vi. Contoh 6.....	26
b. Masukan dengan file .txt.....	27
i. Contoh 7.....	27
ii. Contoh 8.....	29
LAMPIRAN.....	31
Github Repository.....	31
Tabel Spesifikasi.....	31

BAB I : DESKRIPSI MASALAH

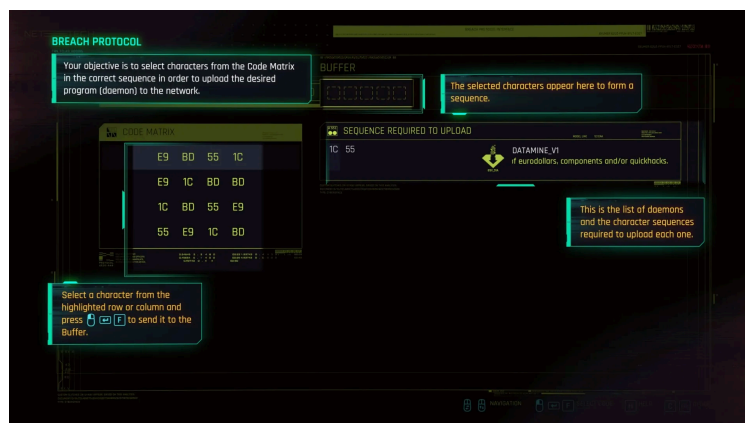
Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video Cyberpunk 2077. Minigame ini merupakan simulasi peretasan jaringan local dari ICE (Intrusion Countermeasures Electronics) pada permainan Cyberpunk 2077. Komponen pada permainan ini antara lain adalah:

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.

Laporan ini akan membahas program yang dapat digunakan untuk mencari solusi optimal untuk menyelesaikan permainan *Breach Protocol* ini dengan algoritma bruteforce.



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

Algoritma Brute Force merupakan strategi pemecahan masalah yang sederhana namun efektif, di mana semua kemungkinan solusi dicoba satu per satu hingga solusi yang tepat ditemukan. Algoritma ini tidak memerlukan pengetahuan khusus tentang domain masalah dan bergantung pada kekuatan komputasi untuk mencapai solusi yang diinginkan. Meskipun tidak selalu efisien dari segi waktu dan sumber daya, Brute Force sering digunakan karena kemudahannya dalam implementasi dan keandalannya dalam menemukan solusi.

Penggunaan algoritma Brute Force dalam penyelesaian Cyberpunk 2077 Breach Protocol didasarkan pada beberapa pertimbangan utama. Pertama, karena sifat permainan yang membutuhkan pencocokan sekuens token dari sebuah matrix, Brute Force memungkinkan eksplorasi setiap kemungkinan kombinasi tanpa melewatkan solusi potensial. Ini menjamin bahwa solusi yang ditemukan adalah solusi optimal, sebab semua kemungkinan telah diuji. Kedua, meskipun Brute Force dikenal karena penggunaan waktu dan sumber daya komputasi yang intensif, kompleksitas permainan ini masih dalam batas yang dapat ditangani efektif dengan pendekatan tersebut. Keterbatasan jumlah token dan sekuens dalam permainan membuat Brute Force menjadi pilihan yang praktis dan dapat diandalkan. Ketiga, kemudahan implementasi algoritma Brute Force memudahkan pengembangan awal dan pengujian solusi, mempercepat iterasi dan peningkatan program. Kesederhanaan, keandalan, dan universalitas Brute Force menjadikannya pilihan yang tepat untuk program ini.

Langkah Penggunaan Algoritma Brute Force pada Program

1. Inisialisasi

Langkah pertama dalam menggunakan algoritma Brute Force adalah menyiapkan struktur data yang diperlukan, termasuk matrix permainan yang berisi token-token dan nilai buffer size yang ditentukan oleh pengguna. Proses inisialisasi ini penting untuk menetapkan ruang lingkup permasalahan yang akan dipecahkan algoritma.

2. Pemilihan Token Awal

Setelah inisialisasi, algoritma memulai pencarian solusi dengan memilih token awal dari baris paling atas matrix. Pemilihan ini dilakukan secara iteratif, dimana setiap token pada baris awal akan dijadikan titik awal dalam mencoba berbagai kemungkinan kombinasi sekuens.

3. Eksplorasi Kemungkinan

Dengan token awal terpilih, algoritma kemudian melanjutkan untuk mengeksplorasi setiap kemungkinan langkah selanjutnya sesuai dengan aturan permainan, yaitu bergerak secara horizontal dan vertikal secara bergantian. Eksplorasi kemungkinan dalam untuk algoritma Brute Force pada Cyberpunk 2077 Breach Protocol memanfaatkan pendekatan rekursif untuk menyelidiki setiap kemungkinan langkah dari token awal yang dipilih. Berikut adalah penjelasan lebih mendalam mengenai basis, rekurens, dan mekanisme backtrack dalam konteks kode tersebut:

a. Basis Rekursif

Basis rekursif dari program akan ditentukan oleh kondisi ketika panjang jalur yang telah dieksplorasi (path) sama dengan ukuran buffer (buffer_size). Pada titik ini, algoritma menghentikan rekursi lebih lanjut karena telah mencapai batas kapasitas buffer, yang berarti tidak ada lagi token yang bisa ditambahkan ke dalam jalur tanpa melebihi batas. Basis ini memastikan bahwa algoritma hanya menghasilkan jalur yang sesuai dengan keterbatasan permainan.

b. Rekurens

Proses rekurens dalam program terjadi melalui eksplorasi kemungkinan langkah selanjutnya dari posisi saat ini, mengikuti aturan gerakan horizontal dan vertikal secara bergantian. Untuk setiap posisi token saat ini, algoritma mengidentifikasi semua posisi token berikutnya yang mungkin, berdasarkan aturan pergerakan, dan memanggil dirinya sendiri (rekursif) untuk setiap posisi tersebut dengan jalur yang diperbarui. Ini memungkinkan algoritma untuk menyelidiki secara mendalam setiap cabang kemungkinan dari pohon pencarian solusi.

c. Backtrack

Backtracking terjadi ketika algoritma kembali dari panggilan rekursif tanpa menemukan solusi yang lebih baik dari yang sudah ditemukan sebelumnya. Dalam konteks ini, backtrack memungkinkan algoritma untuk "mundur" dan mencoba kemungkinan langkah lain dari posisi sebelumnya, efektif membatalkan pilihan terakhir dan menggantinya dengan alternatif lain yang belum diuji. Mekanisme ini esensial untuk memastikan bahwa semua kombinasi jalur diuji, memungkinkan algoritma untuk mencari secara menyeluruh melalui ruang solusi dan menemukan kombinasi yang menghasilkan reward terbesar.

4. Pencocokan Sekuens

Selama proses eksplorasi, algoritma terus memantau isi dari buffer. Setiap kali buffer diisi dengan token, algoritma memeriksa apakah kombinasi token tersebut sesuai dengan salah satu sekuens yang didefinisikan. Jika ada pencocokan, reward dari sekuens tersebut ditambahkan ke total skor. Proses ini membutuhkan pencocokan yang efisien dan akurat untuk memastikan semua peluang reward dapat dimaksimalkan.

5. Optimalisasi Solusi

Algoritma Brute Force mencoba semua kemungkinan perpaduan token hingga seluruh matrix telah dieksplorasi atau buffer telah terisi penuh. Proses ini diulangi untuk setiap token awal yang mungkin. Dari semua kemungkinan solusi yang dihasilkan, algoritma memilih solusi dengan total reward tertinggi sebagai solusi optimal.

6. Evaluasi dan Output

Setelah solusi optimal ditemukan, algoritma kemudian menampilkan jalur solusi tersebut beserta dengan total reward yang diperoleh kepada pengguna. Output ini memberikan solusi kepada pengguna tentang strategi yang paling efektif untuk meraih skor tertinggi dalam permainan Breach Protocol.

Langkah-langkah ini menunjukkan bagaimana algoritma Brute Force dapat diterapkan secara sistematis untuk menyelesaikan Cyberpunk 2077 Breach Protocol. Meskipun membutuhkan waktu komputasi yang signifikan, pendekatan ini memastikan bahwa tidak ada solusi yang terlewat dan solusi terbaik selalu dapat ditemukan.

BAB III : IMPLEMENTASI PROGRAM DENGAN PYTHON DAN JAVASCRIPT

Dalam pengembangan solusi untuk Cyberpunk 2077 Breach Protocol, diterapkan sistem antarmuka pengguna berbasis web (GUI) dan logika pemrosesan backend. Implementasi ini menggabungkan pemanfaatan bahasa pemrograman Python dan JavaScript, serta beberapa framework yang mendukung. Untuk bagian logika pemrosesan backend terdapat program App.py yang ditulis dengan Python. Sedangkan, untuk pemrograman tampilan GUI website dibuat dengan ReactJS sehingga fungsi-fungsi di dalam tampilan website ditulis dengan bahasa pemrograman Javascript.

Selain interaksi melalui GUI website, kami juga menyediakan opsi bagi pengguna untuk menjalankan program melalui Command Line Interface (CLI). Hal ini memungkinkan pengguna yang lebih teknis untuk menginteraksi dengan program tanpa perlu antarmuka grafis, menawarkan fleksibilitas dalam penggunaan. Implementasi CLI terdiri dari berkas algo.py yang berisi logika utama algoritma Brute Force dan main.py yang bertindak sebagai titik masuk program.

Fungsi-Fungsi yang digunakan dalam program akan dideskripsikan sebagai berikut

1. Framework Backend Flask (Bahasa Pemrograman Python)

a. app.py

File app.py dalam konteks backend program ini berperan sebagai pusat pengendali yang mengatur interaksi antara frontend dan logika pemrosesan algoritma Brute Force yang telah dikembangkan. Dengan menggunakan Flask sebagai kerangka kerjanya, app.py mendefinisikan berbagai endpoint API yang memungkinkan pengiriman data dari pengguna melalui frontend web ke server, di mana data tersebut kemudian diproses, dan hasilnya dikembalikan ke pengguna.

API (Application Programming Interface) merupakan sekumpulan definisi dan protokol untuk membangun dan mengintegrasikan aplikasi software. Dalam konteks web, API berfungsi sebagai jembatan komunikasi antara frontend dan backend, memungkinkan pertukaran data secara dinamis. API mendefinisikan

permintaan yang dapat dilakukan, bagaimana membuat permintaan tersebut, format data yang digunakan, dan struktur respons yang diharapkan. Dengan API, aplikasi frontend dapat meminta data dari server, mengirimkan data untuk diproses, atau meminta server untuk melakukan operasi tertentu tanpa perlu tahu detail implementasi di sisi server. Dalam `app.py`, API dibangun dengan mendefinisikan route atau jalur yang sesuai dengan fungsi-fungsi tertentu. Setiap route dikaitkan dengan sebuah fungsi yang akan dieksekusi ketika API dengan URL tertentu dipanggil.

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 konteks program ini, `app.py` mengelola beberapa aspek kritis seperti:

- Menerima data matrix, sequences, dan buffer size dari frontend.
- Memanggil fungsi pemrosesan algoritma Brute Force dengan data yang diterima.
- Mengatur logika untuk menghasilkan matrix dan sequences secara otomatis jika diperlukan.
- Mengembalikan hasil pemrosesan ke frontend, termasuk solusi optimal dan informasi tambahan seperti waktu eksekusi.



```

1  @app.route('/solveHome', methods=['POST'])
2  def solve():
3      data = request.get_json()
4
5      buffer_size = data['bufferSize']
6      matrix = data['matrix']
7      sequences = data['sequences']
8
9      # Convert sequences from the format received to the expected format
10     sequences_formatted = [(seq['sequence'], seq['reward'])
11                             for seq in sequences]
12     start_time = time.time()
13     # Call the solve algorithm function with the processed data
14     best_reward, best_path, best_trimmed_path = solve_algo(
15         matrix, sequences_formatted, buffer_size)
16     end_time = time.time()
17     time_taken = (end_time - start_time) * 1000
18     best_trimmed_path_tokens = [matrix[pos][0]][pos[1]]
19                               for pos in best_trimmed_path]
20
21     # Construct the result to send back to the frontend
22     result = {
23         'bestReward': best_reward,
24         'bestTrimmedPath': best_trimmed_path,
25         'bestTrimmedPathTokens': best_trimmed_path_tokens,
26         'timeTaken': round(time_taken, 2)
27     }
28
29     return jsonify(result)
30

```

Gambar 3.1. API solve()

Endpoint ini menerima data matrix, sequences, dan buffer size dari pengguna melalui metode POST. Data tersebut kemudian diproses menggunakan fungsi `solve_algo` yang di import dari `algo.py`, yang mengimplementasikan algoritma Brute Force untuk menemukan jalur dengan total reward maksimal. Hasil yang dikembalikan mencakup reward terbaik, jalur terbaik, jalur terpotong, token jalur terpotong, dan waktu eksekusi algoritma.

```

1  @app.route('/upload', methods=['POST'])
2  def upload_file():
3      if 'file' not in request.files:
4          return jsonify({'error': 'No file part'}), 400
5      file = request.files['file']
6      if file.filename == '':
7          return jsonify({'error': 'No selected file'}), 400
8
9      if file and allowed_file(file.filename):
10         content = file.read().decode('utf-8') # Read and decode file content
11
12         # Split content into lines and process
13         lines = content.split('\n')
14         buffer_size = int(lines[0].strip())
15         matrix_size = tuple(map(int, lines[1].strip().split()))
16         matrix = [line.strip().split() for line in lines[2:2+matrix_size[1]]]
17         sequences_count = int(lines[2+matrix_size[1]].strip())
18         sequences = []
19         for i in range(sequences_count):
20             start_line = 3 + matrix_size[1] + i*2
21             sequence = lines[start_line].strip().split()
22             reward = int(lines[start_line + 1].strip())
23             sequences.append((sequence, reward))
24
25         start_time = time.time()
26         best_reward, best_path, best_trimmed_path = solve_algo(
27             matrix, sequences, buffer_size)
28         end_time = time.time()
29         time_taken = (end_time - start_time) * 1000
30
31         best_trimmed_path_tokens = [matrix[pos[0]][pos[1]]
32                                     for pos in best_trimmed_path]
33
34         # Construct the result to send back to the frontend
35         result = {
36             'bestReward': best_reward,
37             'bestPath': best_path,
38             'bestTrimmedPath': best_trimmed_path,
39             'bestTrimmedPathTokens': best_trimmed_path_tokens,
40             'timeTaken': round(time_taken, 2)
41         }
42
43         return jsonify(result)

```

Gambar 3.2.API upload()

Endpoint ini dirancang untuk menangani upload file oleh pengguna. File yang diupload dibaca dan kontennya diolah untuk mengekstrak informasi seperti buffer size, matrix, dan sequences. Informasi ini kemudian diproses serupa dengan endpoint /solve.

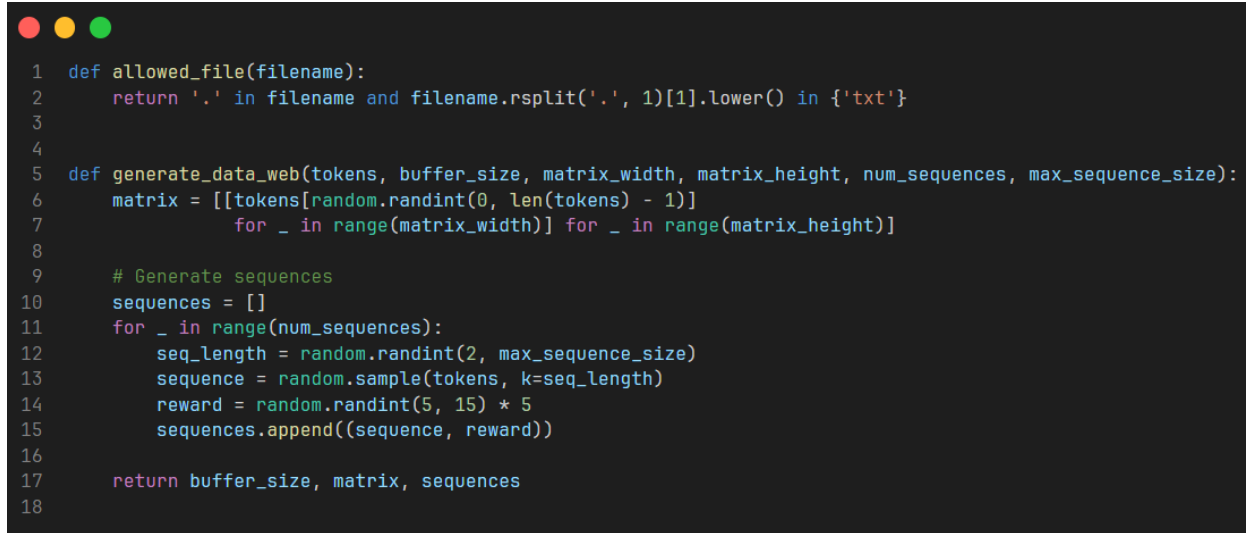
```

1  @app.route('/autoSolve', methods=['POST'])
2  def auto_solve():
3      data = request.get_json()
4      buffer_size, matrix, sequences = generate_data_web(
5          data['uniqueTokens'],
6          data['bufferSize'],
7          data['matrixSize']['width'],
8          data['matrixSize']['height'],
9          data['sequencesAmount'],
10         data['maxSequenceSize']
11     )
12
13     print("success")
14     start_time = time.time()
15     best_reward, best_path, best_trimmed_path = solve_algo(
16         matrix, sequences, buffer_size)
17     end_time = time.time()
18     time_taken = (end_time - start_time) * 1000
19     print("Best Trimmed Path:", best_trimmed_path)
20     print("Matrix Sample:", matrix[0][0])
21     best_trimmed_path_tokens = [
22         ' '.join(matrix[pos[0]][pos[1]] for pos in best_trimmed_path)]
23     result = {
24         'bestReward': best_reward,
25         'bestPath': best_path,
26         'bestTrimmedPath': best_trimmed_path,
27         'bestTrimmedPathTokens': best_trimmed_path_tokens,
28         'timeTaken': round(time_taken, 2)
29     }
30
31     return jsonify(result)

```

Gambar 3.3. API autoSolve()

Endpoint ini memfasilitasi pembangkitan data secara otomatis berdasarkan parameter yang diberikan pengguna, seperti jumlah token unik, ukuran matrix, jumlah sequences, dan ukuran maksimal sequence. Fungsi `generate_data_web` digunakan untuk menghasilkan matrix dan sequences, yang selanjutnya diproses untuk mencari solusi optimal.



```

1 def allowed_file(filename):
2     return '.' in filename and filename.rsplit('.', 1)[1].lower() in {'txt'}
3
4
5 def generate_data_web(tokens, buffer_size, matrix_width, matrix_height, num_sequences, max_sequence_size):
6     matrix = [[tokens[random.randint(0, len(tokens) - 1)]
7                 for _ in range(matrix_width)] for _ in range(matrix_height)]
8
9     # Generate sequences
10    sequences = []
11    for _ in range(num_sequences):
12        seq_length = random.randint(2, max_sequence_size)
13        sequence = random.sample(tokens, k=seq_length)
14        reward = random.randint(5, 15) * 5
15        sequences.append((sequence, reward))
16
17    return buffer_size, matrix, sequences
18

```

Gambar 3.4. fungsi `allowed_file` dan `generate_data_web`

Fungsi `allowed_file` mengecek ekstensi file yang diupload untuk memastikan hanya file dengan ekstensi yang diperbolehkan (.txt) yang bisa diproses. Selanjutnya fungsi `generate_data_web` digunakan untuk memfasilitasi pembangkitan data secara otomatis berdasarkan parameter yang diberikan pengguna, seperti jumlah token unik, ukuran matrix, jumlah sequences, dan ukuran maksimal sequence. Fungsi `generate_data_web` digunakan untuk menghasilkan matrix dan sequences, yang selanjutnya diproses untuk mencari solusi optimal.

b. algo.py

File `algo.py` berisi kumpulan fungsi yang dirancang untuk mencari solusi optimal dari permainan dengan eksplorasi semua kemungkinan jalur yang dapat diambil. Fokus utama dari `algo.py` adalah pada pengolahan logika permainan dan pencarian solusi yang efisien melalui pendekatan Brute Force.

```

1  def read_file(file_path):
2      try:
3          with open(file_path, 'r') as file:
4              buffer_size = int(file.readline().strip())
5
6              matrix_width, matrix_height = map(
7                  int, file.readline().strip().split())
8
9              matrix = [file.readline().strip() for _ in range(matrix_height)]
10
11             number_of_sequences = int(file.readline().strip())
12
13             sequences = []
14             for _ in range(number_of_sequences):
15                 sequence = file.readline().strip()
16                 reward = file.readline().strip()
17                 sequences.append((sequence, reward))
18
19             return {
20                 'buffer_size': buffer_size,
21                 'matrix_dimensions': (matrix_width, matrix_height),
22                 'matrix': matrix,
23                 'sequences': sequences
24             }
25     except FileNotFoundError:
26         print(f"Error: The file '{file_path}' was not found.")
27         return None

```

Gambar 3.7. fungsi read_file

Fungsi `read_file` berperan penting dalam memproses data input dari file teks. Fungsi ini membuka file yang ditentukan oleh pengguna, kemudian secara bertahap membaca dan memisahkan informasi yang relevan seperti ukuran buffer, dimensi matriks, isi matriks itu sendiri, dan daftar sequences berserta reward-nya. Setiap bagian data diproses dan dikembalikan dalam bentuk dictionary untuk digunakan dalam tahapan berikutnya dari program. Fungsi ini juga menyediakan penanganan kesalahan jika file tidak ditemukan, memastikan program memberikan respons yang tepat kepada pengguna.



```

1  def find_matching(buffer, sequences):
2      total_reward = 0
3      buffer_str = ' '.join(buffer)
4      for sequence, reward in sequences:
5          sequence_str = ' '.join(sequence)
6          if sequence_str in buffer_str:
7              total_reward += int(reward)
8      return total_reward
9
10
11 def trim_path(matrix, sequences, best_path):
12     max_reward = find_matching([matrix[pos[0]][pos[1]]
13                                for pos in best_path], sequences)
14     for i in range(len(best_path), 0, -1):
15         temp_path = best_path[:i]
16         buffer = [matrix[pos[0]][pos[1]] for pos in temp_path]
17         temp_reward = find_matching(buffer, sequences)
18
19         if temp_reward < max_reward:
20             return best_path[:i+1]
21
22     return best_path

```

Gambar 3.8. fungsi find_matching dan trim_path

Fungsi `find_matching` digunakan untuk mengevaluasi buffer yang dibangun selama proses eksplorasi jalur dan menghitung total reward berdasarkan sequences yang cocok dengan isi buffer. Fungsi ini mengkonversi buffer dan sequences ke dalam string untuk memudahkan pencocokan. Ini memungkinkan program untuk mengidentifikasi dan mengakumulasi reward dari semua sequences yang berhasil dicocokkan, memberikan dasar untuk memilih jalur optimal. Fungsi `trim_path` bertugas mengoptimalkan jalur yang ditemukan dengan memangkas bagian yang tidak berkontribusi terhadap peningkatan total reward. Ini dilakukan dengan membandingkan reward dari buffer yang dipangkas dengan reward maksimal yang diperoleh sejauh ini. Proses ini memastikan bahwa hasil akhir adalah jalur yang lebih ringkas namun tetap mempertahankan reward maksimum.

```

1 def solve_algo(matrix, sequences, buffer_size, path=[], direction='H', best_reward=0, best_path=[], best_trimmed_path=[]):
2     if len(path) == buffer_size:
3         buffer = [matrix[pos[0]][pos[1]] for pos in path]
4         reward = find_matching(buffer, sequences)
5         if reward > best_reward:
6             best_reward = reward
7             best_path = path[:]
8             best_trimmed_path = trim_path(matrix, sequences, best_path)
9         return best_reward, best_path, best_trimmed_path
10
11     if not path:
12         for i in range(len(matrix[0])):
13             temp_reward, temp_path, temp_trimmed_path = solve_algo(
14                 matrix, sequences, buffer_size, [(0, i)], 'V', best_reward, best_path, best_trimmed_path)
15             if temp_reward > best_reward:
16                 best_reward, best_path, best_trimmed_path = temp_reward, temp_path, temp_trimmed_path
17     else:
18         last_pos = path[-1]
19         next_positions = []
20         if direction == 'H':
21             for i in range(len(matrix[0])):
22                 if i != last_pos[1]:
23                     next_positions.append((last_pos[0], i))
24         else:
25             for i in range(len(matrix)):
26                 if i != last_pos[0]:
27                     next_positions.append((i, last_pos[1]))
28
29         for next_pos in next_positions:
30             if next_pos not in path:
31                 new_path = path + [next_pos]
32                 new_direction = 'V' if direction == 'H' else 'H'
33                 temp_reward, temp_path, temp_trimmed_path = solve_algo(
34                     matrix, sequences, buffer_size, new_path, new_direction, best_reward, best_path, best_trimmed_path)
35                 if temp_reward > best_reward:
36                     best_reward, best_path, best_trimmed_path = temp_reward, temp_path, temp_trimmed_path
37
38     return best_reward, best_path, best_trimmed_path

```

Gambar 3.9. Fungsi solve_algo

Fungsi `solve_algo` merupakan inti dari pencarian solusi menggunakan metode Brute Force. Fungsi ini menerima matriks, sequences, dan ukuran buffer sebagai input dan memulai proses eksplorasi semua kemungkinan jalur menggunakan pendekatan rekursif. Pada awalnya, jika belum ada jalur yang terbentuk (path kosong), fungsi akan mencoba setiap token di baris pertama matriks sebagai titik awal. Setelah itu, algoritma bergerak maju dengan memilih langkah selanjutnya berdasarkan aturan pergerakan yang bergantian antara horizontal dan vertikal. Untuk setiap jalur yang dieksplorasi, fungsi menghitung reward dengan membandingkan isi buffer saat ini terhadap semua sequences yang diberikan. Jika kombinasi token dalam buffer cocok dengan salah satu sequences, reward dari sequences tersebut ditambahkan ke total reward. Fungsi `find_matching` digunakan untuk proses penilaian ini. Apabila total reward untuk jalur saat ini melebihi reward terbaik yang telah ditemukan sebelumnya, fungsi

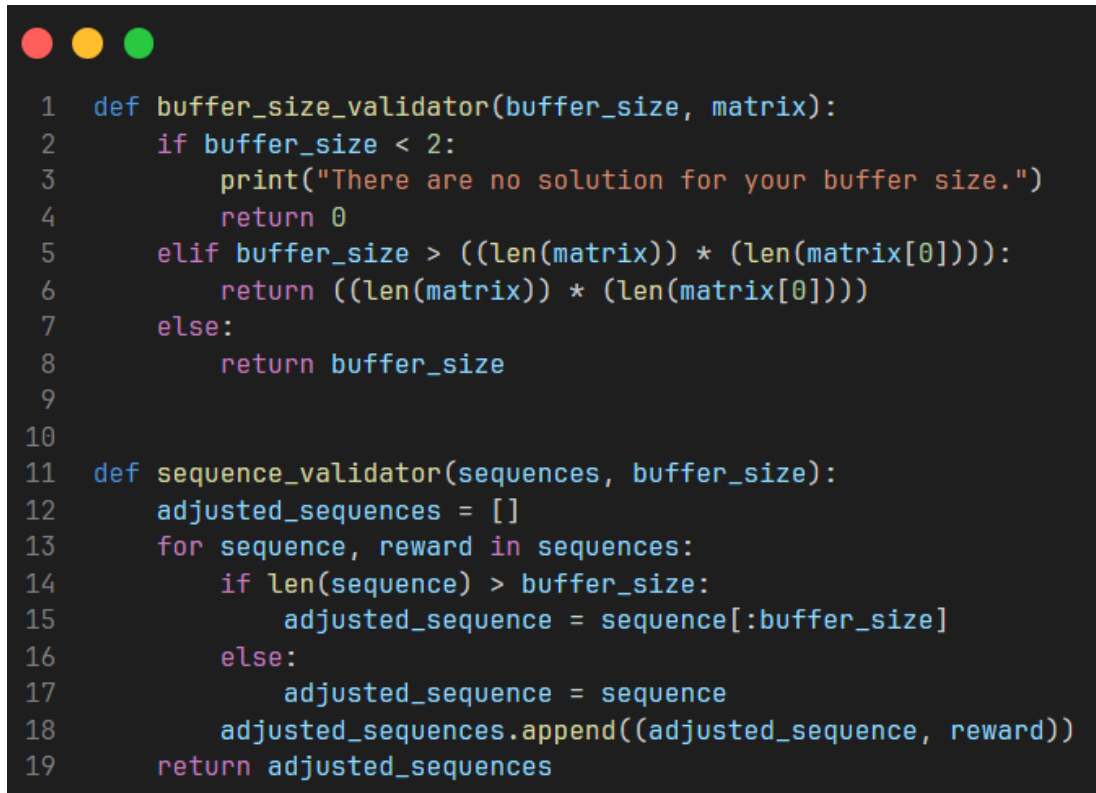
memperbarui nilai `best_reward`, `best_path`, dan melakukan pemangkasan jalur (`best_trimmed_path`) untuk mendapatkan versi jalur yang lebih efisien. Setelah menemukan jalur yang memberikan reward terbaik, fungsi ini mencoba membuang bagian dari jalur yang tidak berkontribusi terhadap peningkatan reward. Ini dilakukan dengan cara menguji jalur dari akhir ke awal, membuang token yang tidak diperlukan sambil memastikan bahwa total reward yang diperoleh tidak berkurang. Proses ini menghasilkan `best_trimmed_path`, yaitu versi jalur yang lebih singkat namun masih mempertahankan nilai reward maksimal.

```
1 def generate_data():
2     while True:
3         try:
4             num_unique_tokens = int(input("Number of unique tokens: "))
5             tokens = input("Tokens (space-separated): ").split()
6             buffer_size = int(input("Buffer size: "))
7             matrix_size = input("Matrix size (width height): ").split()
8             matrix_width, matrix_height = int(
9                 matrix_size[0]), int(matrix_size[1])
10            num_sequences = int(input("Number of sequences: "))
11            max_sequence_size = int(input("Maximum sequence size: "))
12
13            if len(tokens) < num_unique_tokens or len(tokens) > num_unique_tokens:
14                raise ValueError(
15                    "Error: More unique tokens requested than provided.")
16
17            break
18        except ValueError as e:
19            print(f"Invalid input: {e}. Please try again.")
20
21    # Generate Matrix
22    matrix = [[random.choice(tokens) for _ in range(matrix_width)]
23              for _ in range(matrix_height)]
24
25    # Generate sequences
26    sequences = []
27    for _ in range(num_sequences):
28        # Memastikan panjang sequence tidak melebihi batas
29        seq_length = random.randint(2, min(max_sequence_size, len(tokens)))
30        sequence = random.sample(tokens, k=seq_length)
31        # reward antara 5 sampai 25 dan dikali dengan 5
32        reward = random.randint(5, 25) * 5
33        sequences.append(' '.join(sequence), str(reward)))
34
35    return buffer_size, matrix, sequences
```

Gambar 3.10. fungsi generate_data

Fungsi `generate_data` memungkinkan pembangkitan data input secara acak sesuai dengan spesifikasi yang diberikan oleh pengguna. Fungsi ini berguna untuk

menguji algoritma dengan berbagai kondisi input dan memastikan fleksibilitas serta keandalan solusi yang dikembangkan.



```
1 def buffer_size_validator(buffer_size, matrix):
2     if buffer_size < 2:
3         print("There are no solution for your buffer size.")
4         return 0
5     elif buffer_size > ((len(matrix)) * (len(matrix[0]))):
6         return ((len(matrix)) * (len(matrix[0])))
7     else:
8         return buffer_size
9
10
11 def sequence_validator(sequences, buffer_size):
12     adjusted_sequences = []
13     for sequence, reward in sequences:
14         if len(sequence) > buffer_size:
15             adjusted_sequence = sequence[:buffer_size]
16         else:
17             adjusted_sequence = sequence
18         adjusted_sequences.append((adjusted_sequence, reward))
19     return adjusted_sequences
```

Gambar 3.11. fungsi buffer_size_validator dan sequence_validator

Fungsi `buffer_size_validator` dan `sequence_validator` adalah fungsi pembantu yang memastikan input yang diberikan oleh pengguna sesuai dengan kriteria yang diperlukan untuk menjalankan algoritma secara efektif. Validator ini mengatur ukuran buffer dan panjang sequences agar sesuai dengan batasan yang ada, menghindari kasus di mana input tidak valid dapat mengganggu proses pencarian solusi. Fungsi `sequence_validator` akan mengubah sequence yang panjangnya melebihi ukuran buffer, jika panjang dari sequence melebihi ukuran buffer maka sequence tersebut akan dipotong sesuai dengan ukuran dari buffer tersebut.



```

1 def save_to_file(best_reward=0, best_path=[], best_trimmed_path=[], time_taken=0, file_name='result.txt'):
2     # Convert each position tuple into a string
3     path_tokens_str = [
4         '' + ''.join(map(str, pos)) + '' for pos in best_path]
5     trimmed_path_tokens_str = [
6         '(' + ' '.join(map(str, pos)) + ')' for pos in best_trimmed_path]
7
8     with open(file_name, 'w') as file:
9         file.write(f"Best Reward: {best_reward}\n")
10        file.write("Best Path Tokens: " + ' '.join(path_tokens_str) + "\n")
11        file.write("Best Trimmed Path Tokens: " +
12            ' '.join(trimmed_path_tokens_str) + "\n")
13        file.write(f"Time taken: {time_taken:.2f} ms\n")
14    print(f"Results saved to {file_name}.")
15

```

Gambar 3.12. fungsi save_to_file

Fungsi `save_to_file` memberikan kemampuan untuk menyimpan hasil dari proses pencarian solusi ke dalam file teks. Fungsi ini menyusun informasi tentang reward, jalur terbaik, jalur yang dipangkas, dan waktu yang diperlukan untuk proses tersebut, kemudian menuliskannya ke dalam file yang dapat diakses oleh pengguna untuk review atau analisis lebih lanjut.

2. Framework Frontend ReactJS (Bahasa Pemrograman Javascript)

a. App.jsx

```
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

Bagian frontend dari program ini dibangun menggunakan framework React JS, yang memungkinkan pembuatan antarmuka pengguna yang dinamis dan responsif. Salah satu aspek kunci dari frontend adalah fungsi handleSubmit, yang bertugas mengirim data input dari pengguna ke backend untuk diproses. Fungsi ini dipicu ketika pengguna menekan tombol "Solve" pada form. Dalam handleSubmit, pertama-tama mencegah perilaku default submit form untuk menghindari reload halaman. Kemudian, data yang diinput pengguna

dikumpulkan dan disusun dalam format yang sesuai untuk dikirim ke API backend. Ini termasuk ukuran buffer, token unik, jumlah sekuens, ukuran maksimal sekuens, dan ukuran matriks. Setelah itu, data dikirimkan menggunakan fetch API dengan metode POST ke endpoint /autoSolve yang disediakan backend.

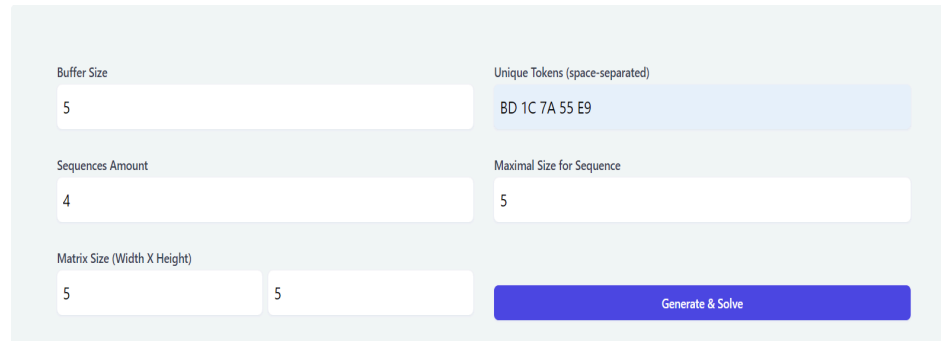
Jika respons dari backend berhasil diterima, data hasil proses akan disimpan dan ditampilkan pada antarmuka pengguna. Namun, jika terjadi kesalahan (misalnya, backend tidak dapat memproses data atau terjadi masalah jaringan), sistem akan menampilkan pesan kesalahan.

Folder frontend tidak hanya mengandung fungsi handleSubmit ini saja, tetapi juga berbagai komponen dan styling lain yang bekerja bersama untuk membentuk tampilan keseluruhan website. Desain dan struktur halaman, pengelolaan state, dan interaksi pengguna semuanya ditangani dalam kode frontend ini. Penggunaan Tailwind CSS sebagai utility-first CSS framework memudahkan pembuatan desain yang konsisten dan responsif di berbagai perangkat.

BAB IV : EKSPERIMEN

a. Masukan acak oleh program

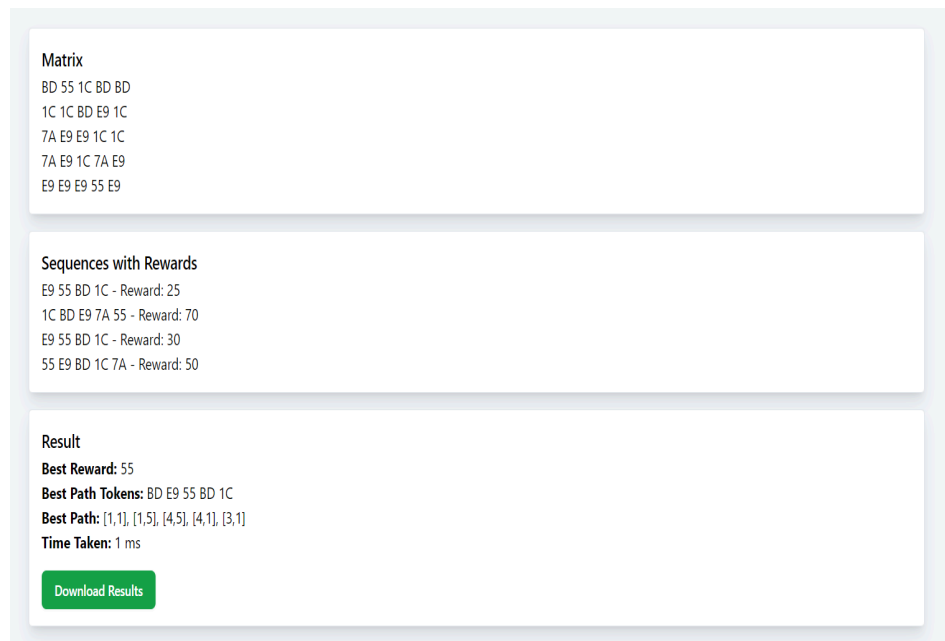
i. Contoh 1



The input form contains the following fields and values:

Field	Value
Buffer Size	5
Unique Tokens (space-separated)	BD 1C 7A 55 E9
Sequences Amount	4
Maximal Size for Sequence	5
Matrix Size (Width X Height)	5
Matrix Size (Height)	5
Generate & Solve	Button

Gambar 4.1. Masukan acak contoh 1



The output results are displayed in three sections:

Matrix

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

Sequences with Rewards

```
E9 55 BD 1C - Reward: 25
1C BD E9 7A 55 - Reward: 70
E9 55 BD 1C - Reward: 30
55 E9 BD 1C 7A - Reward: 50
```

Result

Best Reward: 55
Best Path Tokens: BD E9 55 BD 1C
Best Path: [1,1], [1,5], [4,5], [4,1], [3,1]
Time Taken: 1 ms

[Download Results](#)

Gambar 4.2. Keluaran acak contoh 1

ii. Contoh 2

Buffer Size	<input type="text" value="4"/>	Unique Tokens (space-separated)	<input type="text" value="BD 1C 7A 55 E9"/>
Sequences Amount	<input type="text" value="4"/>	Maximal Size for Sequence	<input type="text" value="5"/>
Matrix Size (Width X Height)	<input type="text" value="5"/>	<input type="text" value="5"/>	<input type="button" value="Generate & Solve"/>

Gambar 4.3. Masukan acak contoh 2

Matrix BD 7A 1C BD E9 55 7A E9 BD E9 BD E9 BD 7A E9 BD 7A 55 E9 E9 E9 BD 1C 55 E9
Sequences with Rewards 55 BD - Reward: 45 7A 1C E9 55 BD - Reward: 65 BD 55 - Reward: 45 7A BD - Reward: 45
Result Best Reward: 135 Best Path Tokens: 7A BD 55 BD Best Path: [2,1], [2,5], [4,5], [4,1] Time Taken: 0 ms <input type="button" value="Download Results"/>

Gambar 4.4. Keluaran acak contoh 2

iii. Contoh 3

Buffer Size	<input type="text" value="8"/>	Unique Tokens (space-separated)	<input type="text" value="BD 1C 7A 55 E9"/>
Sequences Amount	<input type="text" value="4"/>	Maximal Size for Sequence	<input type="text" value="4"/>
Matrix Size (Width X Height)	<input type="text" value="5"/>	<input type="text" value="5"/>	<input type="button" value="Generate & Solve"/>

Gambar 4.5. Masukan acak contoh 3

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

Sequences with Rewards
55 E9 7A - Reward: 55
55 1C - Reward: 40
BD E9 55 7A - Reward: 30
55 BD - Reward: 25

Result
Best Reward: 95
Best Path Tokens: BD 7A 55 1C 55 E9 7A
Best Path: [1,1], [1,2], [4,2], [4,3], [5,3], [5,1], [3,1]
Time Taken: 173.64 ms
[Download Results](#)

Gambar 4.6. Keluaran acak contoh 3

iv. Contoh 4

Buffer Size

Unique Tokens (space-separated)

Sequences Amount

Maximal Size for Sequence

Matrix Size (Width X Height)

[Generate & Solve](#)

Gambar 4.7. Masukan acak contoh 4

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

Sequences with Rewards
BD 55 7A - Reward: 25
SC 55 BD 1C - Reward: 60
7A IC - Reward: 25
SC 1C 55 - Reward: 25

Result
Best Reward: 25
Best Path Tokens: BD E9 7A IC
Best Path: [1,1], [1,5], [4,5], [4,6]
Time Taken: 0 ms
[Download Results](#)

Gambar 4.8. Keluaran acak contoh 4

v. Contoh 5

Buffer Size

Unique Tokens (space-separated)

Sequences Amount

Maximal Size for Sequence

Matrix Size (Width X Height)

[Generate & Solve](#)

Gambar 4.9. Masukan acak contoh 5

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

Sequences with Rewards
IC SC 7A E9 - Reward: 40
E9 7A 1C - Reward: 65
7A SC 1C - Reward: 55
BD SC E9 - Reward: 75

Result
Best Reward: 75
Best Path Tokens: BD IC BD SC E9
Best Path: [1,1], [1,2], [3,2], [3,1], [4,1]
Time Taken: 6.87 ms
[Download Results](#)

Gambar 4.10. Keluaran acak contoh 5

vi. Contoh 6

Buffer Size

Unique Tokens (space-separated)

Sequences Amount

Maximal Size for Sequence

Matrix Size (Width X Height)

[Generate & Solve](#)

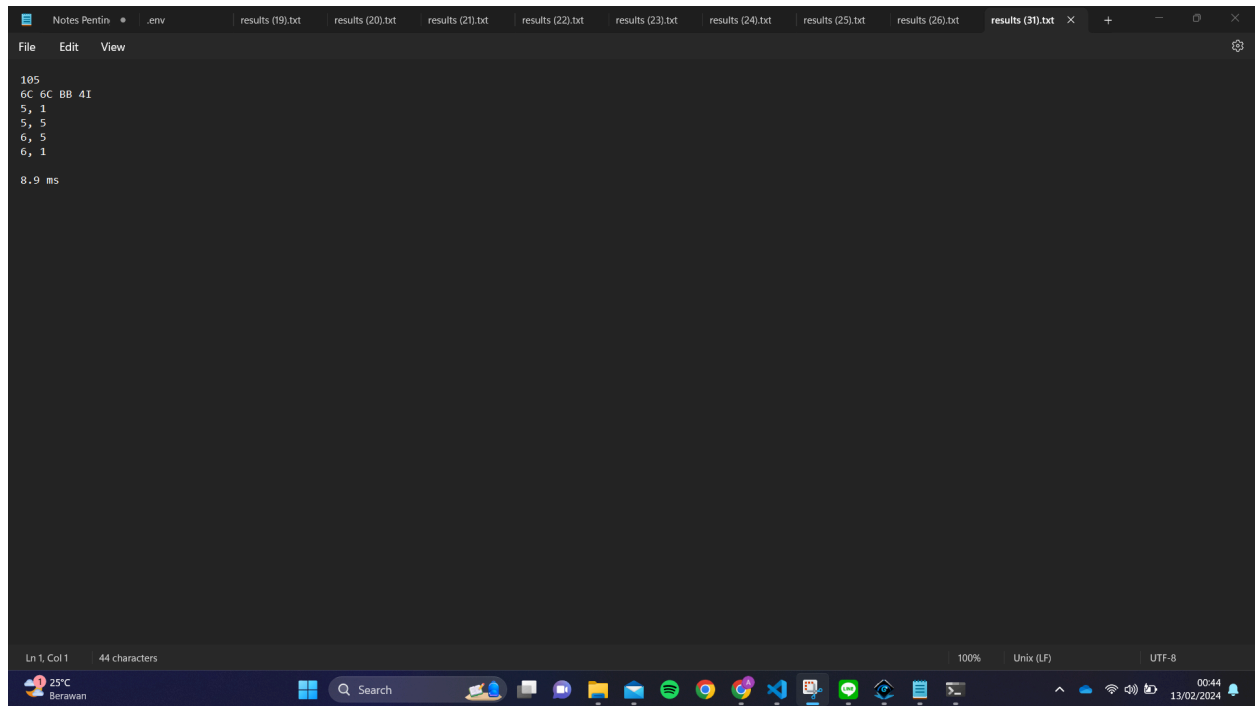
Gambar 4.11. Masukan acak contoh 6

Matrix
4I 6C KC 7A 6C 4I
7A KC KC BB 7A 7A
6C 4I 7A 6C KC 7A
6C 4I BB KC 7A 4I
4I 7A BB 4I 6C BB
KC 6C KC KC 7A 4I

Sequences with Rewards
6C BB 4I - Reward: 60
6C 4I KC - Reward: 50
BB 4I - Reward: 45
KC 7A BB - Reward: 25
4I 7A 6C KC - Reward: 60
4I BB KC - Reward: 30

Result
Best Reward: 105
Best Path Tokens: 6C 6C BB 4I
Best Path: [5,1], [5,5], [6,5], [6,1]
Time Taken: 8.9 ms
[Download Results](#)

Gambar 4.12. Keluaran acak contoh 6

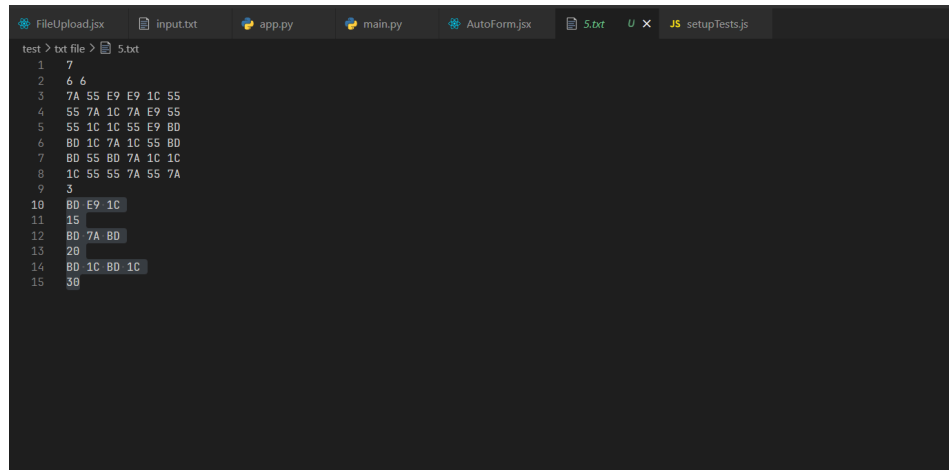


```
105
6C 6C BB 4I
5, 1
5, 5
6, 5
6, 1
8.9 ms
```

Gambar 4.13. Tampilan file tersimpan

b. Masukan dengan file .txt

i. Contoh 7



```
test > txt file > 5.txt
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 BD E9 1C
11 15
12 BD 7A BD
13 20
14 BD 1C BD 1C
15 30
```

Gambar 4.14. Tampilan masukan dengan upload file contoh 7

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 with Rewards

BD E9 1C - Reward: 15
BD 7A BD - Reward: 20
BD 1C BD 1C - Reward: 30

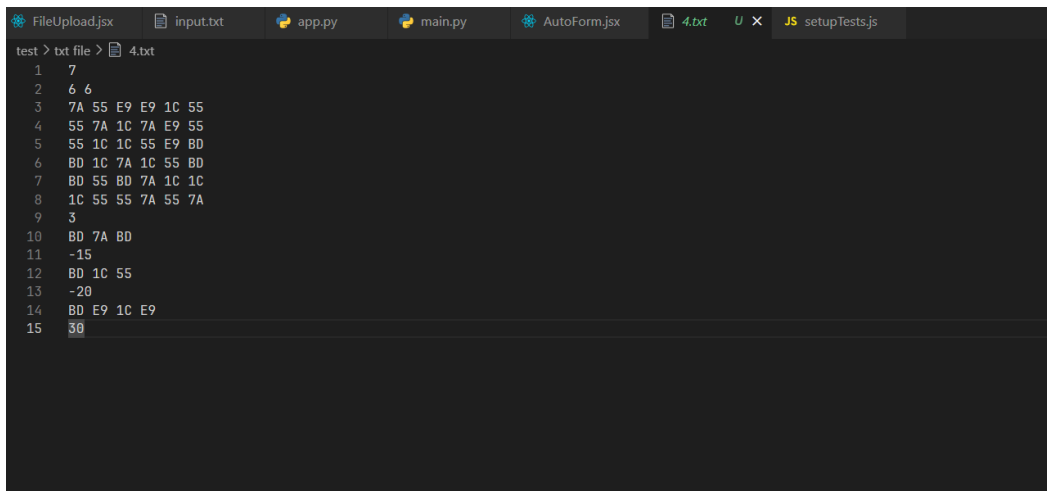
Result

Best Reward: 50
Best Path Tokens: 7A BD 7A BD 1C BD 1C
Best Path: [1,1], [1,4], [3,4], [3,5], [6,5], [6,3], [2,3]
Time Taken: 98.99 ms

[Download Results](#)

Gambar 4.15. Tampilan keluaran dengan upload file contoh 7

ii. Contoh 8



```
test > txt file > 4.txt
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 BD 7A BD
11 -15
12 BD 1C 55
13 -20
14 BD E9 1C E9
15 30
```

Gambar 4.16. Tampilan masukan dengan file contoh 8

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 with Rewards

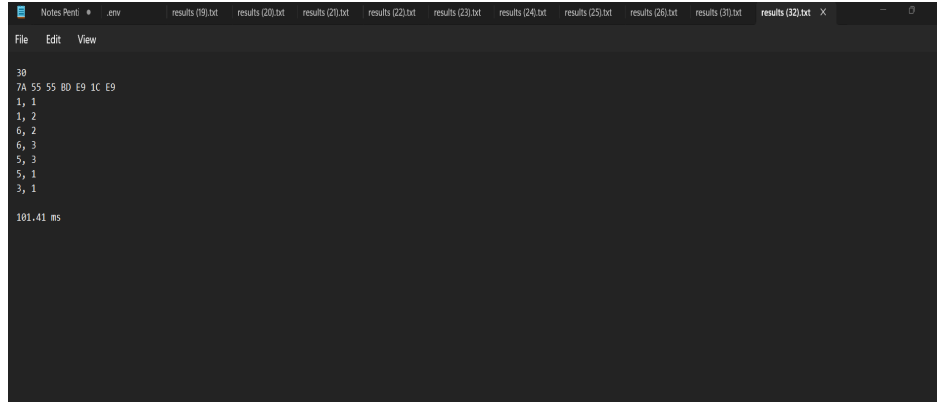
BD 7A BD - Reward: -15
BD 1C 55 - Reward: -20
BD E9 1C E9 - Reward: 30

Result

Best Reward: 30
Best Path Tokens: 7A 55 55 BD E9 1C E9
Best Path: [1,1], [1,2], [6,2], [6,3], [5,3], [5,1], [3,1]
Time Taken: 101.41 ms

[Download Results](#)

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 *Save Solution*, yang akan dikirimkan ke direktori lokal download/ pengguna.

LAMPIRAN

Github Repository

https://github.com/AlbertChoe/Tucil1_13522081

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	✓	