# Laporan Tugas Kecil 1

# IF2211 Strategi Algoritma


**Disusun oleh:**

Ahmad Naufal Ramadan (13522005)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG**

**2024**

# Daftar Isi

## 1. Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan algoritma brute force dibuat dengan menggunakan bahasa Python. Proses input dan output dari program diimplementasikan dengan menggunakan GUI dari library customtkinter. Program lalu di-compile menjadi executable yang dapat dijalankan pada sistem operasi linux dan windows dengan menggunakan pyinstaller.

Algoritma brute force program yang dibuat akan melakukan brute force pada seluruh path yang mungkin pada buffer_size yang diberikan. Algoritma akan memulai pengecekan pada tiap token pada baris paling atas. Pada tiap pemrosesan, token baris paling atas akan disimpan dalam array yang bersifat sebagai stack yang berisi posisi token saat ini dan array of path yang telah ditelusuri. Pertama, pop isi stack lalu destructure isi stack tersebut menjadi posisi token saat ini dan array of path. Selanjutnya dilakukan pengecekan terhadap ukuran dari array of path. Jika array of path kurang dari buffer_size, dilakukan proses untuk melakukan brute force path dari posisi saat ini. Syarat untuk melakukan brute force path adalah arah gerak harus bolak balik dari vertikal ke horizontal ataupun sebaliknya dan token yang sudah dilalui tidak boleh dilalui kembali. Path yang memenuhi syarat tersebut akan dipush ke dalam stack. Jika array of path sama dengan buffer_size, simpan path pada array baru. Program ini membatasi total path hasil brute force sebanyak 10 juta path karena jika total path terlalu banyak, program akan langsung berhenti karena memakan terlalu banyak memory. Proses ini dilakukan hingga stack kosong. Setelah stack kosong, dilanjutkan pengecekan pada token baris paling atas selanjutnya.

Setelah didapat seluruh path yang mungkin, dilakukan brute force untuk menemukan path yang menghasilkan bobot maksimum. Pengecekan dilakukan dengan mengecek apakah sequence terdapat pada path yang dicek. Jika sequence terdapat pada path, total bobot akan ditambah dengan bobot sequence tersebut. Jika total bobot lebih besar dari bobot maksimum, maka bobot maksimum yang baru adalah total bobot sekarang. Array of path baru akan dibuat jika total bobot sekarang lebih besar dari bobot maksimum. Jika total bobot sama dengan bobot maksimum, path akan ditambahkan ke dalam array of path. Hasil akhir dari tahap ini adalah array of path yang menghasilkan bobot maksimum serta bobot tersebut.

Setelah didapat path yang menghasilkan bobot maksimum, dilakukan optimalisasi dengan memotong token yang tidak berguna. Proses ini mengecek apakah ada sequence yang terdapat pada akhir path. Jika tidak ada sequence yang terdapat pada akhir path, hilangkan token terakhir dari path. Lakukan pengulangan hingga terdapat sequence pada akhir path. Setelah proses tersebut, cari satu path yang paling pendek. Proses ini menjadi proses terakhir dalam algoritma brute force yang menghasilkan path yang memiliki bobot maksimum dengan panjang path terpendek.

## 2. Source Code

brute_solver.py

```python
def find_paths(grid, buffer_size):
    rows, cols = len(grid), len(grid[0])
    paths = []

    for c in range(cols):
        stack = [((0, c), [(0, c)])]
        while stack:
            (r, c), path = stack.pop()
            if len(path) < buffer_size:
                last_r, last_c = path[-1]

                if len(path) % 2 == 0:
                    for new_c in range(cols):
                        if new_c != last_c and (last_r, new_c) not in path:
                            stack.append(((last_r, new_c), path + [(last_r, new_c)]))
                else:
                    for new_r in range(rows):
                        if new_r != last_r and (new_r, last_c) not in path:
                            stack.append(((new_r, last_c), path + [(new_r, last_c)]))

            elif len(path) == buffer_size:
                paths.append(path)
                # Batesin biar array of pathsnya ga terlalu besar,
                # artinya kemungkinannya sangat banyak sampe bikin laptop mokad
                if len(paths) >= 10_000_000:
                    return None

    return paths
```

Snipped

brute_solver.py

```python
def find_possible_paths(m, paths, sequences, weights):
    max_path = []
    max_weight = 0
    for path in paths:
        seq = ""
        for r, c in path:
            seq += m[r][c]

        weight = 0
        for s in sequences:
            if "".join(s) in seq:
                weight += weights[sequences.index(s)]

        if weight > max_weight:
            max_weight = weight
            max_path = [path]
        elif weight == max_weight:
            max_path.append(path)

    return max_path, max_weight
```

Snipped

brute_solver.py

```python
def find_shortest_path(m, max_path, sequences):
    short_path = []
    for i, path in enumerate(max_path):
        seq = ""
        for r, c in path:
            seq += m[r][c]

        found = False
        while not found:
            for s in sequences:
                s_path = "".join(s)
                try:
                    if seq[-len(s_path) :] == s_path:
                        found = True
                        break
                except IndexError:
                    continue
            else:
                seq = seq[:-2]
                max_path[i] = max_path[i][:-1]

        short_path.append(seq)

    shortest_path = min(short_path, key=len)
    shortest_coordinate = max_path[short_path.index(shortest_path)]
    shortest_path = " ".join(
        shortest_path[i : i + 2] for i in range(0, len(shortest_path), 2)
    )

    return shortest_path, shortest_coordinate
```

Snipped

brute_solver.py

```python
def brute_solve(data):
    start = time.time()

    paths = find_paths(data["m"], data["buffer_size"])
    if paths is None:
        return None

    max_path, max_weight = find_possible_paths(
        data["m"], paths, data["sequences"], data["weights"]
    )
    shortest_path, shortest_coordinate = find_shortest_path(
        data["m"], max_path, data["sequences"]
    )

    end = time.time()

    return {
        "shortest_path": shortest_path,
        "shortest_coordinate": shortest_coordinate,
        "max_weight": max_weight,
        "time": end - start,
    }
```

Snipped

component.py

```python
class TitleFrame(customtkinter.CTkFrame):
    def __init__(self, master, title):
        super().__init__(master)
        self.grid_columnconfigure(0, weight=1)
        self.title = title
        self.title_label = customtkinter.CTkLabel(
            self,
            text=self.title,
            fg_color="gray30",
            corner_radius=6,
        )
        self.title_label.grid(row=0, column=0, padx=10, pady=10, sticky="ew")
```

Snipped

component.py

```python
class InputFrame(customtkinter.CTkFrame):
    def __init__(self, master, title, values):
        super().__init__(master)
        self.grid_columnconfigure((0, 1), weight=1)
        self.values = values
        self.title = title
        self.entries = []
        self.default_values = ["5", "BD 1C 7A 55 E9", "7", "6", "6", "3", "4"]

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(row=0, column=0, padx=10, pady=10, sticky="ew", columnspan=2)

        for i, value in enumerate(self.values):
            label = customtkinter.CTkLabel(self, text=value)
            label.grid(row=i + 1, column=0, padx=10, pady=10, sticky="ew")

            entry = customtkinter.CTkEntry(self)
            entry.insert(0, self.default_values[i])
            entry.grid(row=i + 1, column=1, padx=10, pady=10, sticky="ew")
            self.entries.append(entry)

    def get(self):
        if not all(entry.get() for entry in self.entries):
            return {}

        values = [
            "unique_token",
            "token",
            "buffer_size",
            "m_width",
            "m_height",
            "sequences_count",
            "sequences_max_length",
        ]
        return {key: entry.get() for key, entry in zip(values, self.entries)}
```

Snipped

component.py

```python
class FileUploadFrame(customtkinter.CTkFrame):
    def __init__(self, master, title):
        super().__init__(master)
        self.grid_columnconfigure(0, weight=1)
        self.title = title
        self.data = {}

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(row=0, column=0, padx=10, pady=10, sticky="ew")

        self.button = customtkinter.CTkButton(self, text="Upload File")
        self.button.grid(row=1, column=0, padx=10, pady=10, sticky="ew")
        self.button.configure(command=self.open_file)

        self.text = customtkinter.CTkLabel(self, text="No file uploaded")
        self.text.grid(row=2, column=0, padx=10, pady=10, sticky="nsew")

    def open_file(self):
        filetypes = [("text files", "*.txt")]
        f = customtkinter.filedialog.askopenfilename(
            filetypes=filetypes, title="Select file"
        )

        if f:
            with open(f, "r") as file:
                buffer_size = int(file.readline().strip())
                m_width, m_height = map(int, file.readline().strip().split())
                m = [
                    list(map(str, file.readline().strip().split()))
                    for _ in range(m_height)
                ]
                n = int(file.readline().strip())
                sequences = []
                weights = []
                for _ in range(n):
                    sequence = list(file.readline().strip().split())
                    weight = int(file.readline().strip())
                    sequences.append(sequence)
                    weights.append(weight)

                self.data = {
                    "buffer_size": buffer_size,
                    "m_width": m_width,
                    "m_height": m_height,
                    "m": m,
                    "n": n,
                    "sequences": sequences,
                    "weights": weights,
                }
                self.text.configure(text=f)

    def get(self):
        return self.data
```

Snipped

component.py

```python
class RadioButtonFrame(customtkinter.CTkFrame):
    def __init__(self, master, title, values):
        super().__init__(master)
        self.grid_columnconfigure(0, weight=1)
        self.values = values
        self.title = title
        self.radiobuttons = []
        self.variable = customtkinter.StringVar(value="Manual")

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(row=0, column=0, padx=10, pady=10, sticky="ew")

        for i, value in enumerate(self.values):
            radiobutton = customtkinter.CTkRadioButton(
                self, text=value, value=value, variable=self.variable
            )
            radiobutton.grid(row=i + 1, column=0, padx=10, pady=10, sticky="w")
            self.radiobuttons.append(radiobutton)

    def get(self):
        return self.variable.get()
```

Snipped

component.py

```python
class MatrixFrame(customtkinter.CTkFrame):
    def __init__(self, master, title, m_width, m_height, m):
        super().__init__(master)
        self.grid_columnconfigure((0, m_width - 1), weight=1)
        self.title = title
        self.m_width = m_width
        self.m_height = m_height
        self.entries = []

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(
            row=0, column=0, padx=10, pady=10, sticky="ew", columnspan=m_width
        )

        for i in range(self.m_height):
            for j in range(self.m_width):
                entry = customtkinter.CTkEntry(self, width=30)
                entry.insert(0, m[i][j])
                entry.grid(row=i + 1, column=j, padx=5, pady=5, sticky="ew")
                entry.configure(state="disabled")
                self.entries.append(entry)

    def configure_cell(self, row, column):
        self.entries[row * self.m_width + column].configure(fg_color="#2b719e")
```

Snipped

component.py

```python
class SequenceFrame(customtkinter.CTkFrame):
    def __init__(self, master, title, sequences, weights):
        super().__init__(master)
        self.grid_columnconfigure((0, 3), weight=1)
        self.title = title
        self.sequences = sequences
        self.entries = []

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(row=0, column=0, padx=10, pady=10, sticky="ew", columnspan=4)

        for i, sequence in enumerate(self.sequences):
            label = customtkinter.CTkLabel(self, text=f"Sequence {i + 1}")
            label.grid(row=i + 1, column=0, padx=10, pady=10, sticky="ew")

            entry = customtkinter.CTkEntry(self)
            entry.insert(0, " ".join(sequence))
            entry.grid(row=i + 1, column=1, padx=10, pady=10, sticky="ew")
            entry.configure(state="disabled")

            label1 = customtkinter.CTkLabel(self, text="Weight")
            label1.grid(row=i + 1, column=2, padx=10, pady=10, sticky="ew")

            entry1 = customtkinter.CTkEntry(self)
            entry1.insert(0, weights[i])
            entry1.grid(row=i + 1, column=3, padx=10, pady=10, sticky="ew")
            entry1.configure(state="disabled")

            self.entries.append(entry)
```

Snipped

component.py

```python
class CoordinateFrame(customtkinter.CTkFrame):
    def __init__(self, master, title, coordinates):
        super().__init__(master)
        self.grid_columnconfigure((0, 1), weight=1)
        self.title = title
        self.coordinates = coordinates
        self.entries = []

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(row=0, column=0, pady=10, sticky="ew", columnspan=2)

        for i, coordinate in enumerate(self.coordinates):
            label = customtkinter.CTkLabel(self, text=f"Coordinate {i + 1}")
            label.grid(row=i + 1, column=0, pady=10, sticky="ew")

            entry = customtkinter.CTkEntry(self)
            entry.insert(0, f"({coordinate[1] + 1}, {coordinate[0] + 1})")
            entry.grid(row=i + 1, column=1, pady=10, sticky="ew")
            entry.configure(state="disabled")

            self.entries.append(entry)
```

Snipped

component.py

```python
class ResultFrame(customtkinter.CTkScrollableFrame):
    def __init__(self, master, title, result):
        super().__init__(master)
        self.grid_columnconfigure((0, 1), weight=1)
        self.title = title
        self.result = result

        self.title = customtkinter.CTkLabel(
            self, text=self.title, fg_color="gray30", corner_radius=6
        )
        self.title.grid(row=0, column=0, padx=10, pady=10, sticky="ew", columnspan=4)

        label1 = customtkinter.CTkLabel(self, text="Max Weight")
        label1.grid(row=1, column=0, padx=10, pady=10, sticky="ew")

        entry1 = customtkinter.CTkEntry(self)
        entry1.insert(0, self.result["max_weight"])
        entry1.grid(row=1, column=1, padx=10, pady=10, sticky="ew")
        entry1.configure(state="disabled")

        label2 = customtkinter.CTkLabel(self, text="Path")
        label2.grid(row=2, column=0, padx=10, pady=10, sticky="ew")

        entry2 = customtkinter.CTkEntry(self)
        entry2.insert(0, self.result["shortest_path"])
        entry2.grid(row=2, column=1, padx=10, pady=10, sticky="ew")
        entry2.configure(state="disabled")

        label3 = customtkinter.CTkLabel(self, text="Time Taken")
        label3.grid(row=3, column=0, padx=10, pady=10, sticky="ew")

        entry3 = customtkinter.CTkEntry(self)
        entry3.insert(0, f"{self.result['time']:.3f} seconds")
        entry3.grid(row=3, column=1, padx=10, pady=10, sticky="ew")
        entry3.configure(state="disabled")

        self.coordinate_frame = CoordinateFrame(
            self, "Coordinates", self.result["shortest_coordinate"]
        )
        self.coordinate_frame.grid(
            row=4, column=0, padx=10, pady=10, sticky="nsew", columnspan=2
        )
```

Snipped

app.py

```python
class Result(customtkinter.CTkToplevel):
    def __init__(self, master, data):
        super().__init__(master)
        self.title("Cyberpunk 2077 Breach Protocol Solver Result")
        self.grid_columnconfigure((0, 1), weight=1)

        self.result = None
        self.title_label = customtkinter.CTkLabel(
            self,
            text="Cyberpunk 2077 Breach Protocol Solver Result",
            fg_color="gray30",
            corner_radius=6,
        )
        self.title_label.grid(
            row=0, column=0, padx=10, pady=10, sticky="ew", columnspan=2
        )

        self.matrix_frame = MatrixFrame(
            self,
            "Matrix",
            m_width=int(data["m_width"]),
            m_height=int(data["m_height"]),
            m=data["m"],
        )
        self.matrix_frame.grid(row=1, column=0, padx=10, pady=10, sticky="n")

        self.sequence_frame = SequenceFrame(
            self, "Sequences", data["sequences"], data["weights"]
        )
        self.sequence_frame.grid(row=1, column=1, padx=10, pady=10, sticky="n")

        self.button = customtkinter.CTkButton(self, text="Solve")
        self.button.grid(row=2, column=0, padx=10, pady=10, sticky="n", columnspan=2)
        self.button.configure(command=lambda: self.solve(data))

    def solve(self, data):
        self.result = brute_solve(data)

        if self.result is not None:
            shortest_coordinate = self.result["shortest_coordinate"]
            for coordinate in shortest_coordinate:
                self.matrix_frame.configure_cell(coordinate[0], coordinate[1])

            self.result_frame = ResultFrame(self, "Result", self.result)
            self.result_frame.grid(
                row=3,
                column=0,
                padx=10,
                pady=10,
                sticky="nsew",
                columnspan=2,
                rowspan=2,
            )

            self.button1 = customtkinter.CTkButton(self, text="Export Result")
            self.button1.grid(
                row=5, column=0, padx=10, pady=10, sticky="ew", columnspan=2
            )
            self.button1.configure(command=lambda: self.export_result(self.result))
        else:
            self.error_label = customtkinter.CTkLabel(
                self,
                text="Too many paths for buffer size!",
                fg_color="red",
                corner_radius=6,
            )
            self.error_label.grid(
                row=3, column=0, padx=10, pady=10, sticky="ew", columnspan=2
            )

    def export_result(self, result):
        filetypes = [("text files", "*.txt")]
        file_path = customtkinter.filedialog.asksaveasfilename(
            defaultextension=".txt", filetypes=filetypes
        )

        if file_path:
            try:
                with open(file_path, "w") as f:
                    f.write(f"{result['max_weight']}\n")
                    f.write(f"{result['shortest_path']}\n")
                    for coordinate in result["shortest_coordinate"]:
                        f.write(f"{coordinate[1] + 1},{coordinate[0] + 1}\n")

                    f.write("\n")
                    f.write(f"{result['time']:.3f} seconds\n")
            except Exception as e:
                print(e)
```

Snipped

app.py

```python
class App(customtkinter.CTk):
    def __init__(self):
        super().__init__()

        self.title("Cyberpunk 2077 Breach Protocol Solver")
        self.grid_columnconfigure((0, 1), weight=1)
        self.grid_rowconfigure(0, weight=1)

        self.grid_columnconfigure((0, 1), weight=1)

        self.title_frame = TitleFrame(self, "Cyberpunk 2077 Breach Protocol Solver")
        self.title_frame.grid(
            row=0, column=0, padx=10, pady=(10, 0), sticky="nsew", columnspan=2
        )

        self.values = [
            "Number of Unique Tokens",
            "Tokens",
            "Buffer Size",
            "Matrix Width",
            "Matrix Height",
            "Number of Sequences",
            "Sequences Max Length",
        ]
        self.input_frame = InputFrame(self, "Input", values=self.values)
        self.input_frame.grid(row=1, column=0, padx=10, pady=10, sticky="nsew")

        self.file_upload_frame = FileUploadFrame(self, "File Upload")
        self.file_upload_frame.grid(row=1, column=1, padx=10, pady=10, sticky="new")

        self.radio_button_frame = RadioButtonFrame(
            self, "Select Mode", values=["Manual", "File"]
        )
        self.radio_button_frame.grid(
            row=2, column=0, padx=10, pady=10, sticky="nsew", columnspan=2
        )

        self.button = customtkinter.CTkButton(self, text="Generate")
        self.button.grid(row=3, column=0, padx=10, pady=10, sticky="ew", columnspan=2)
        self.button.configure(command=self.generate)

    def generate(self):
        mode = self.radio_button_frame.get()
        if mode == "Manual":
            values = self.input_frame.get()
            values = self.generate_data(values)
            res = Result(self, values)
        else:
            values = self.file_upload_frame.get()
            res = Result(self, values)

    def generate_data(self, values):
        unique_token = int(values["unique_token"])
        token = values["token"].split()
        buffer_size = int(values["buffer_size"])
        m_width = int(values["m_width"])
        m_height = int(values["m_height"])
        sequences_count = int(values["sequences_count"])
        sequences_max_length = int(values["sequences_max_length"])

        m = self.generate_matrix(m_width, m_height, token)
        sequences = self.generate_sequence(sequences_count, sequences_max_length, token)
        weights = self.generate_weight(sequences_count)

        return {
            "buffer_size": buffer_size,
            "m_width": m_width,
            "m_height": m_height,
            "m": m,
            "n": sequences_count,
            "sequences": sequences,
            "weights": weights,
        }

    def generate_matrix(self, m_width, m_height, token):
        m = []
        for i in range(m_height):
            row = []
            for j in range(m_width):
                row.append(random.choice(token))
            m.append(row)
        return m

    def generate_sequence(self, sequences_count, sequences_max_length, token):
        sequences = []
        for i in range(sequences_count):
            sequence = []
            for j in range(random.randint(2, sequences_max_length)):
                sequence.append(random.choice(token))

            while sequence in sequences:
                sequence = []
                for j in range(random.randint(2, sequences_max_length)):
                    sequence.append(random.choice(token))

            sequences.append(sequence)
        return sequences

    def generate_weight(self, sequences_count):
        weights = []
        for i in range(sequences_count):
            weights.append(random.randint(1, 10) * 5)
        return weights
```

Snipped

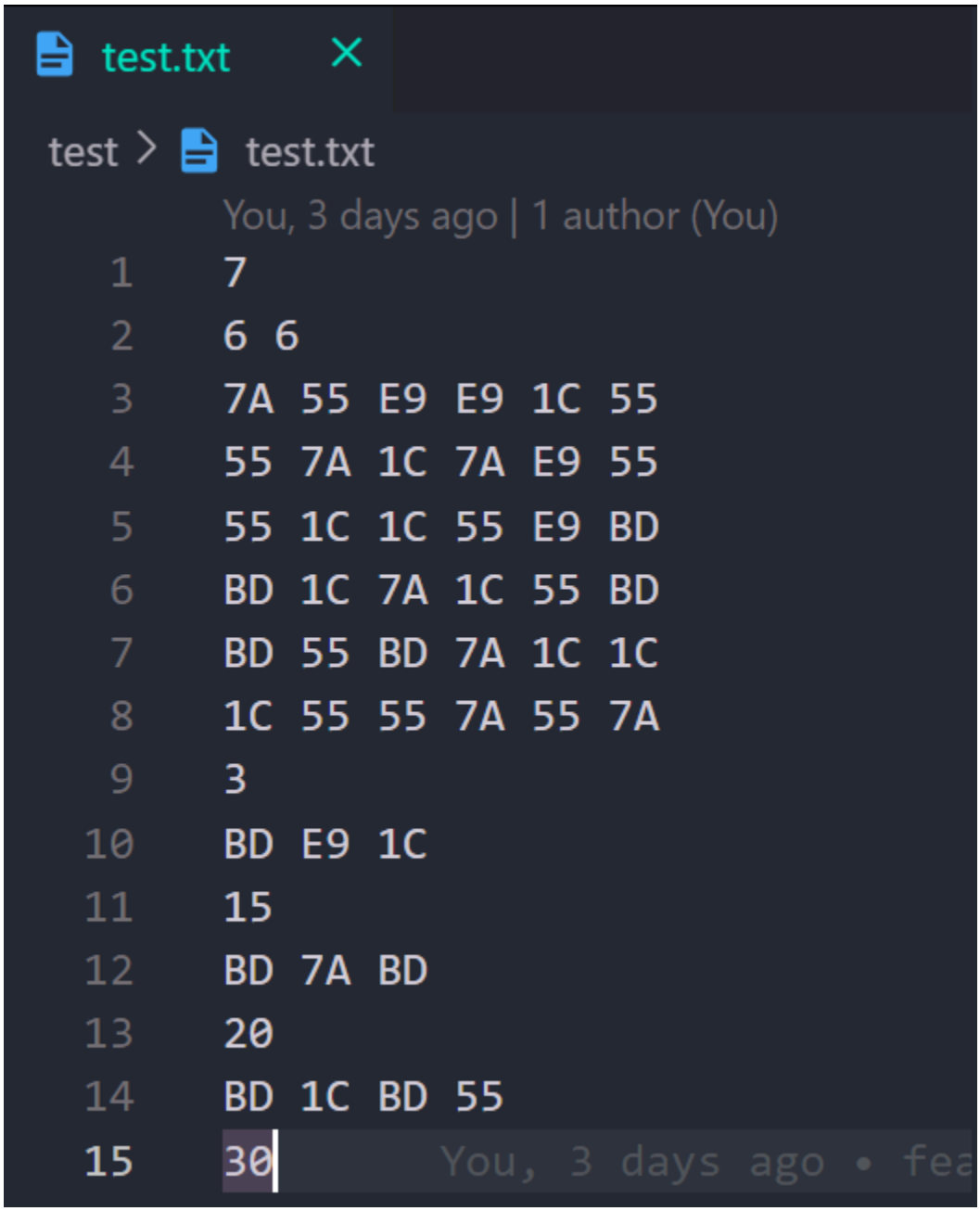
app.py

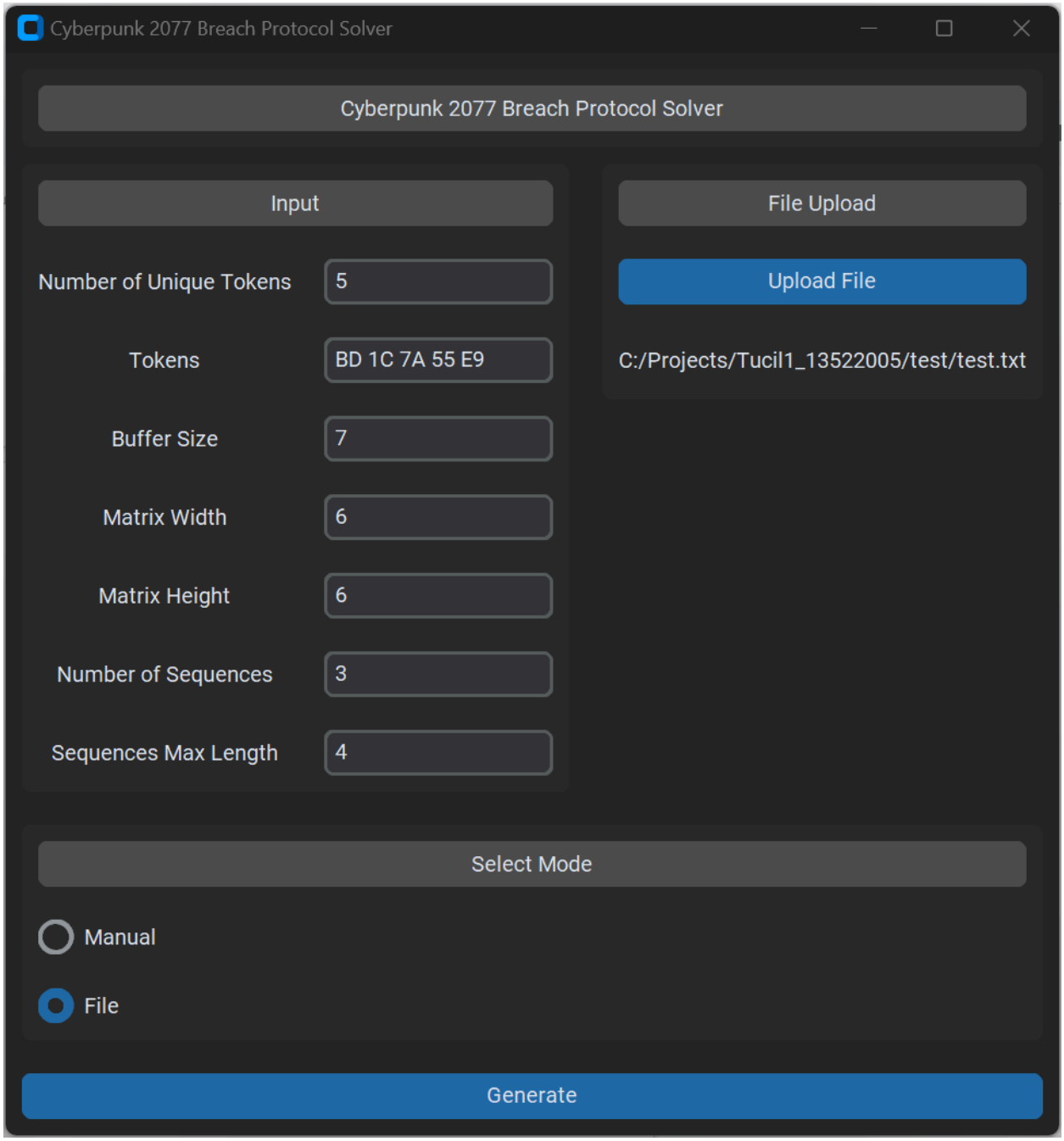
```
202  if __name__ == "__main__":
203      customtkinter.set_appearance_mode("dark")
204      app = App()
205      app.mainloop()
```

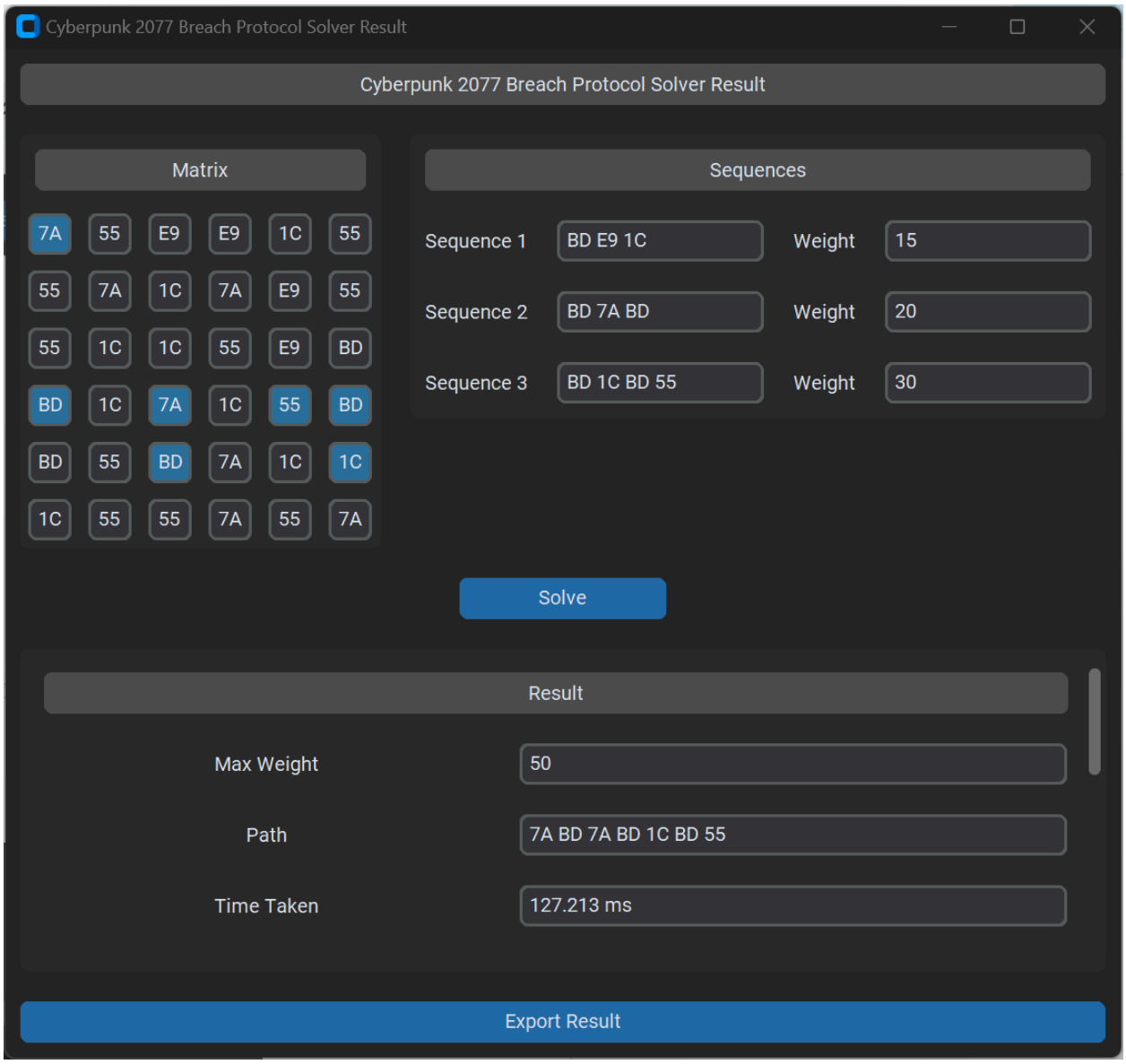
Snipped

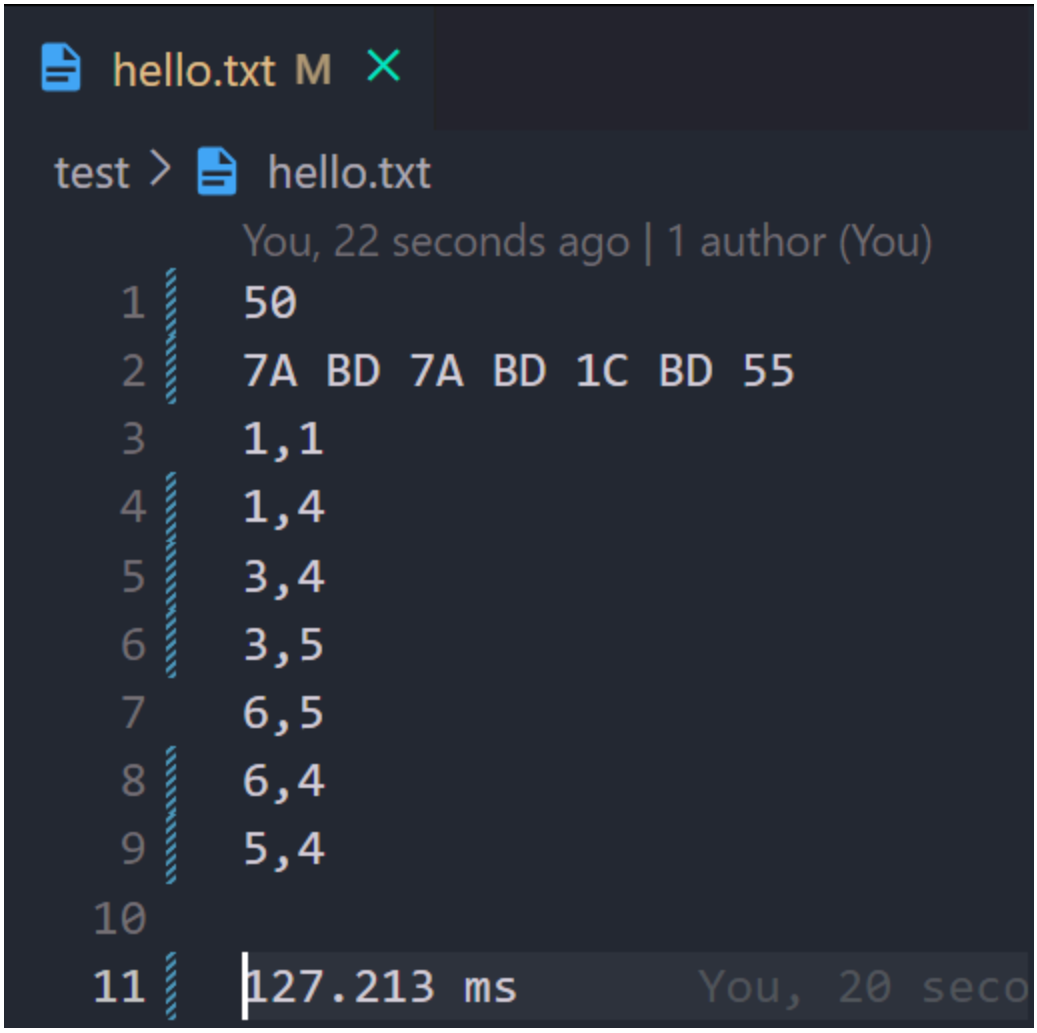
### 3. Uji Kasus

File Upload




Cyberpunk 2077 Breach Protocol Solver
Cyberpunk 2077 Breach Protocol Solver
Input
Number of Unique Tokens
5
Tokens
BD 1C 7A 55 E9
Buffer Size
7
Matrix Width
6
Matrix Height
6
Number of Sequences
3
Sequences Max Length
4
File Upload
Upload File
C:/Projects/Tucil1_13522005/test/test.txt
Select Mode
Manual
File
Generate

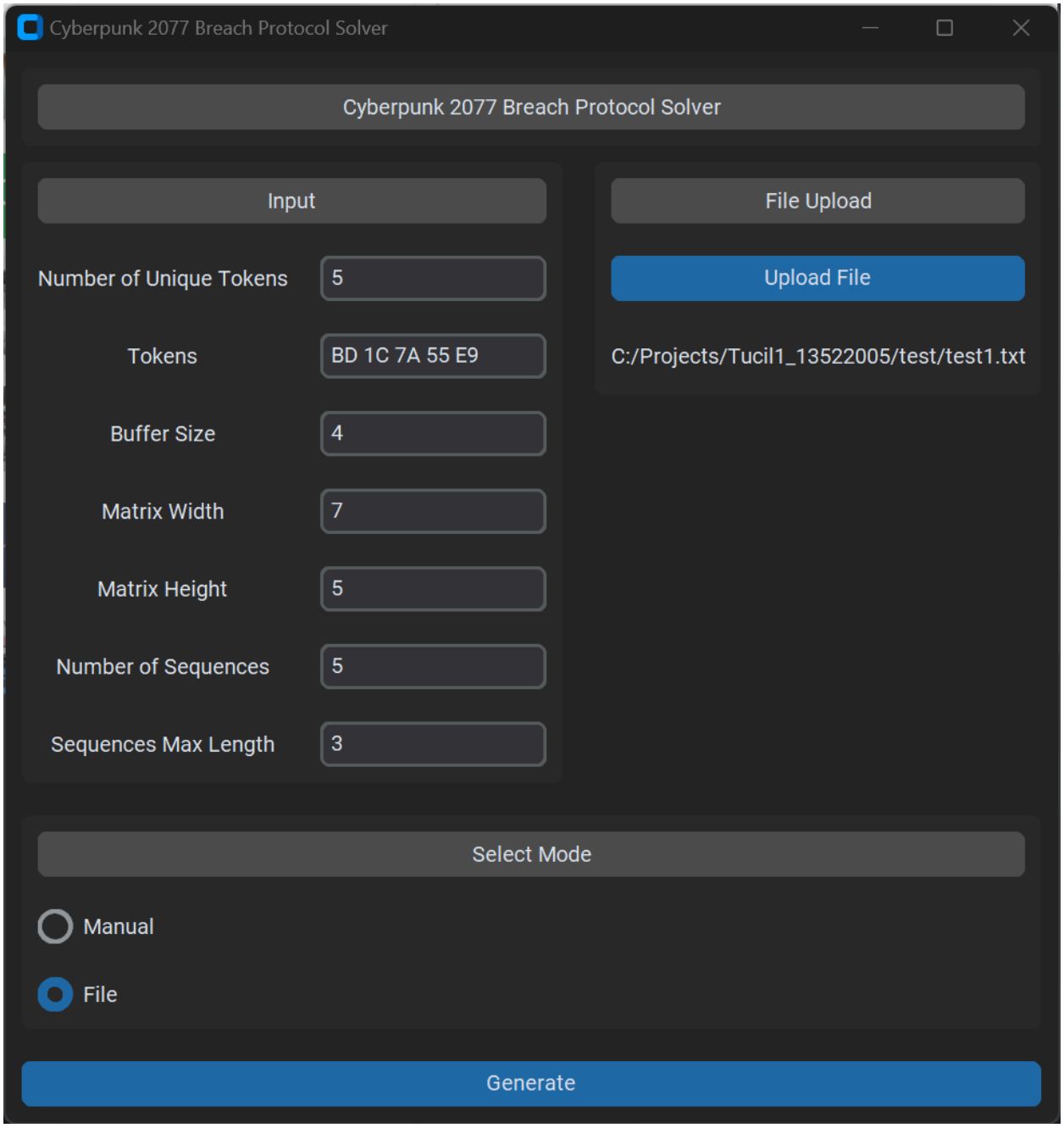

Cyberpunk 2077 Breach Protocol Solver Result
Cyberpunk 2077 Breach Protocol Solver Result
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
Sequences
Sequence 1 BD E9 1C Weight 15
Sequence 2 BD 7A BD Weight 20
Sequence 3 BD 1C BD 55 Weight 30
Solve
Result
Max Weight 50
Path 7A BD 7A BD 1C BD 55
Time Taken 127.213 ms
Export Result

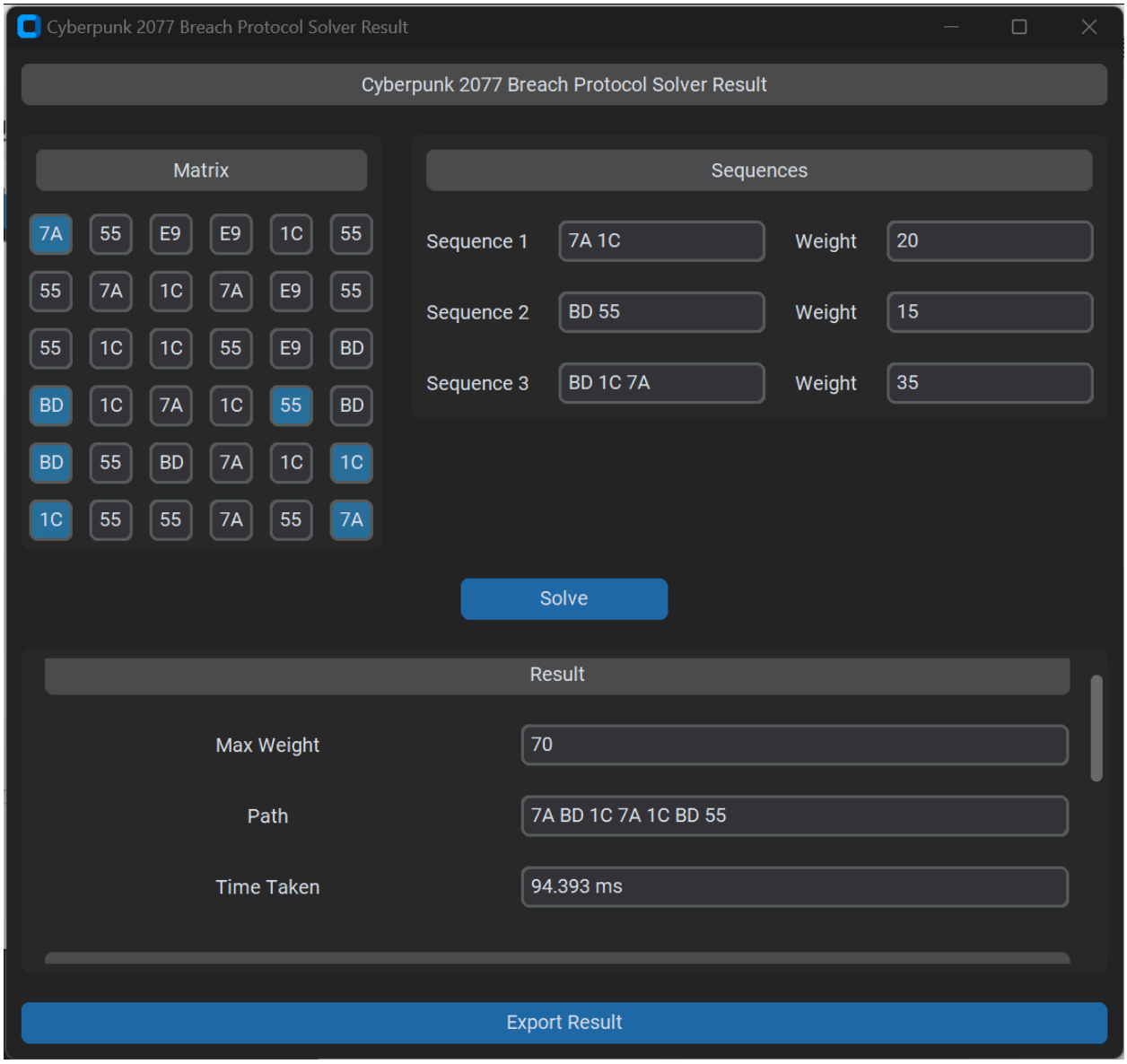

hello.txt M
test > hello.txt
You, 22 seconds ago | 1 author (You)
50
7A BD 7A BD 1C BD 55
1,1
1,4
3,4
3,5
6,5
6,4
5,4
127.213 ms
You, 20 seco

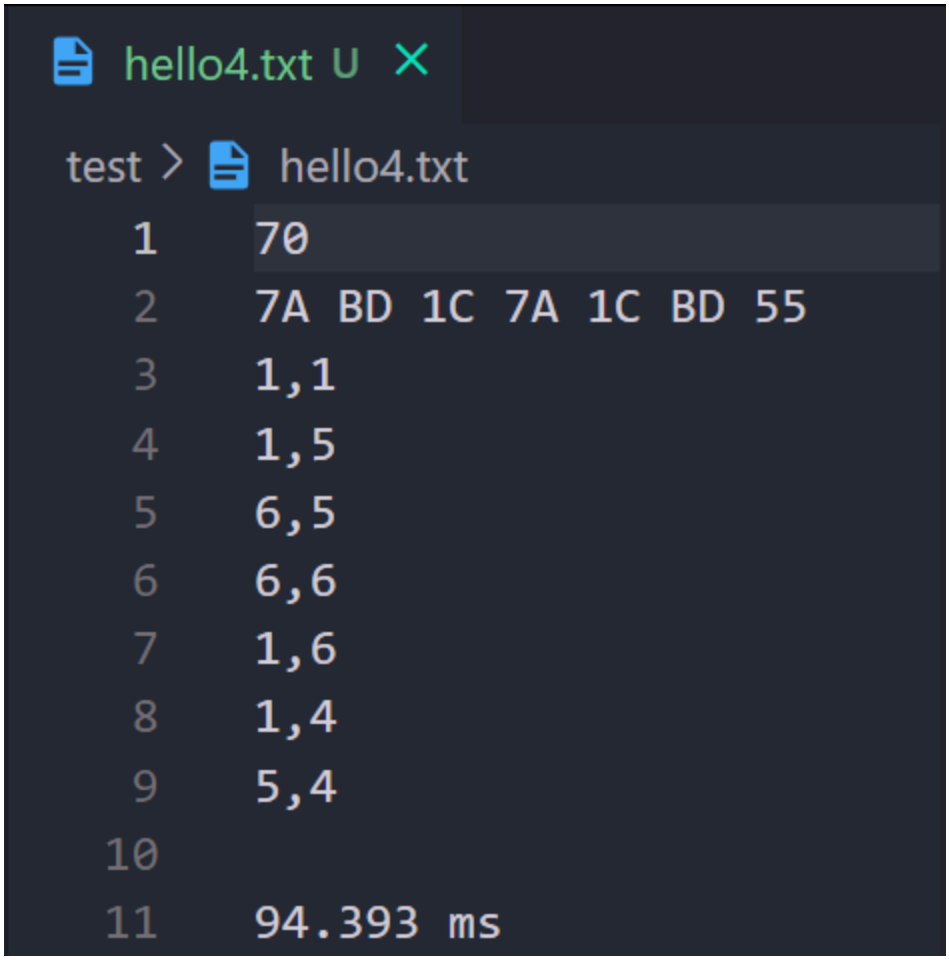
test1.txt U

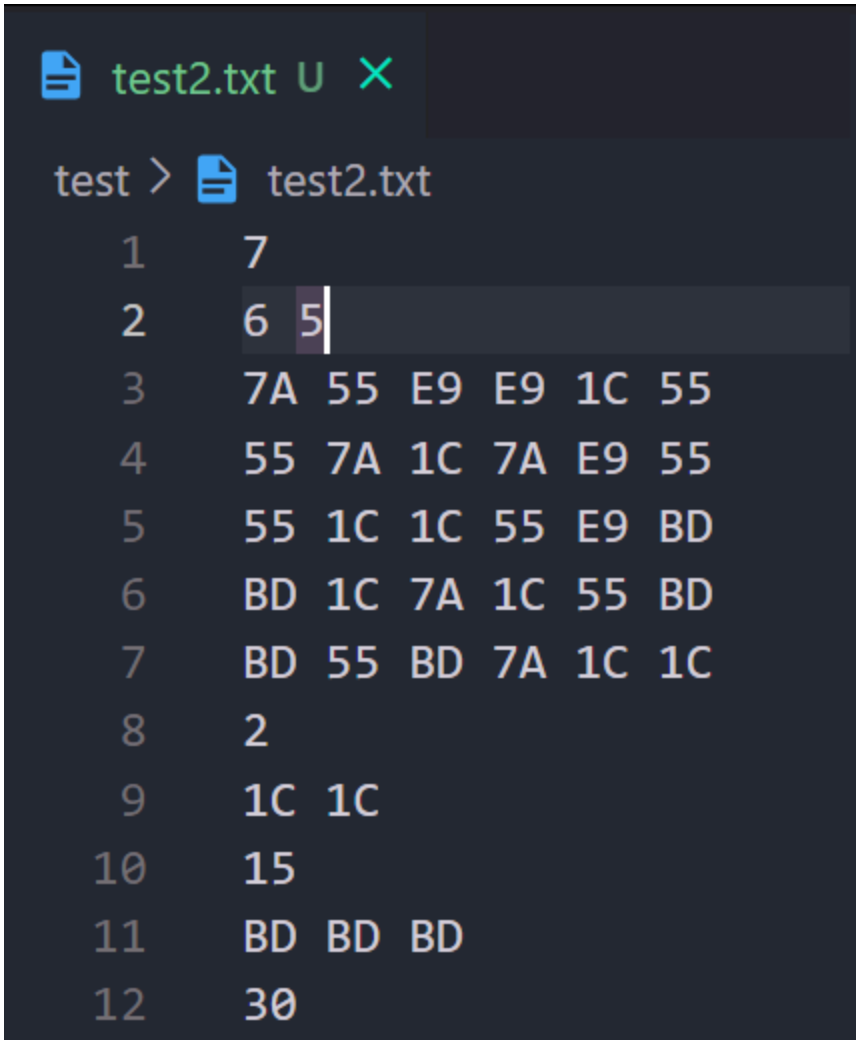
test > test1.txt

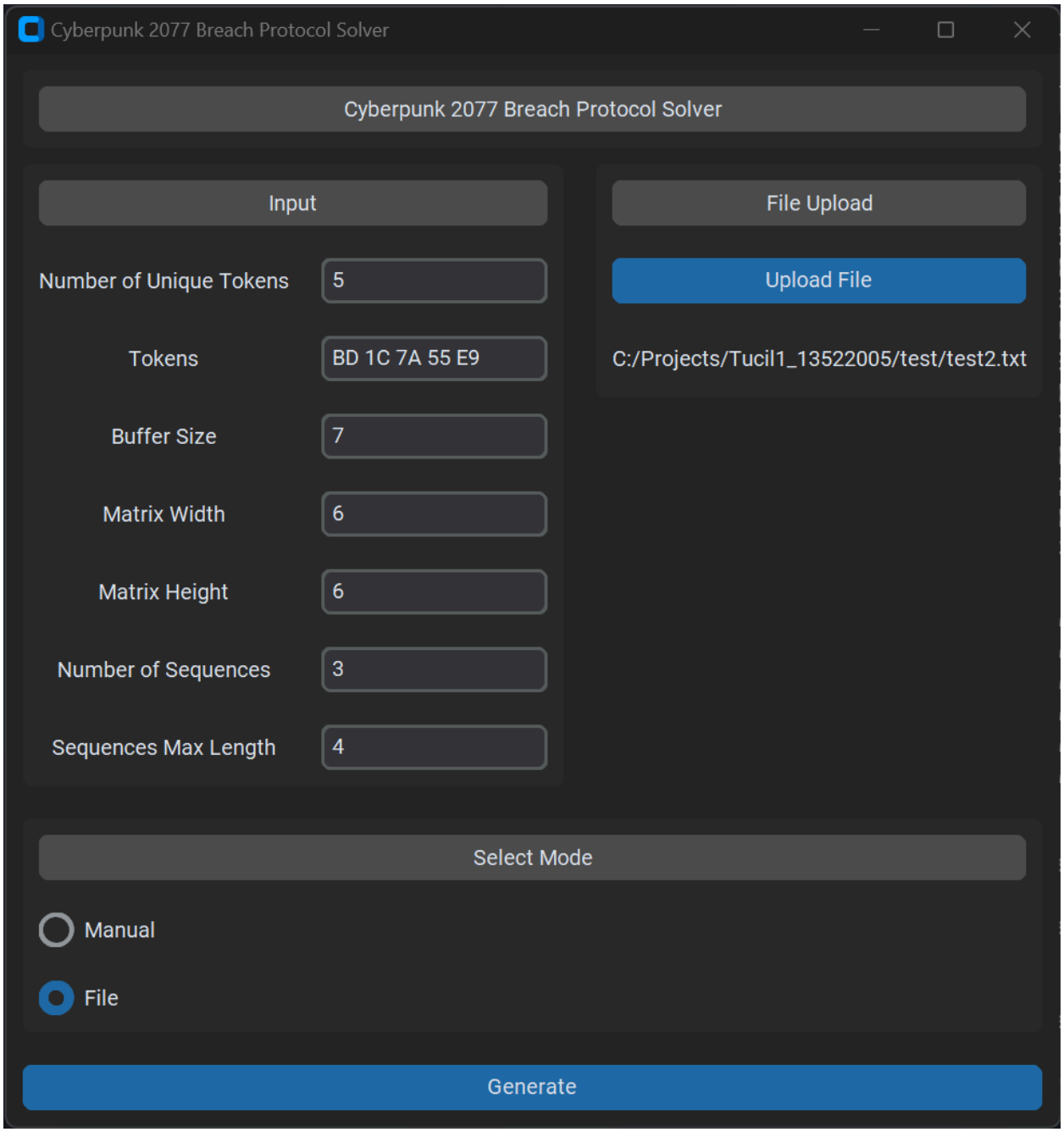
```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
7A 1C
20
BD 55
15
BD 1C 7A
35
```

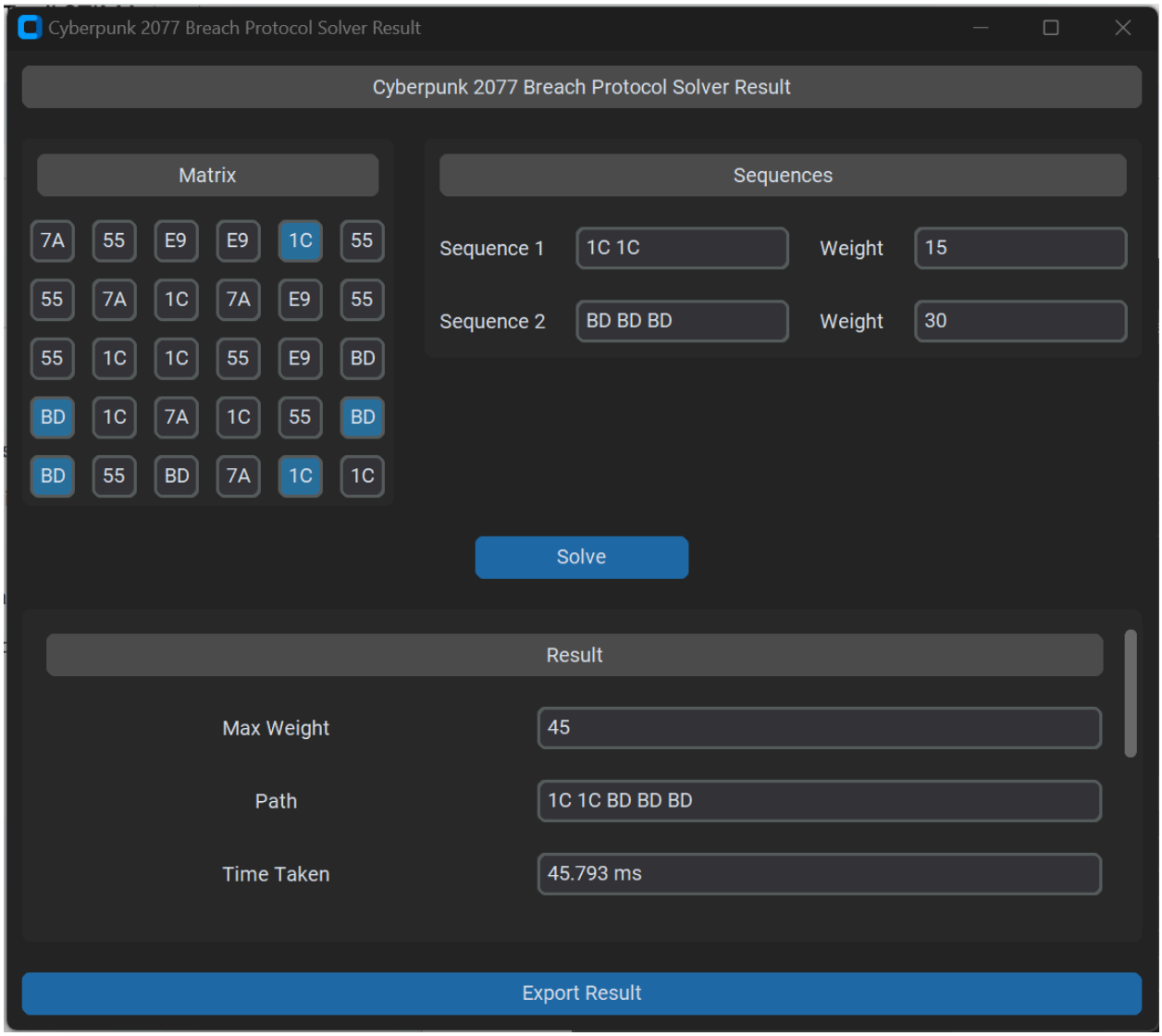

Cyberpunk 2077 Breach Protocol Solver
Cyberpunk 2077 Breach Protocol Solver
Input
Number of Unique Tokens
5
Tokens
BD 1C 7A 55 E9
Buffer Size
4
Matrix Width
7
Matrix Height
5
Number of Sequences
5
Sequences Max Length
3
File Upload
Upload File
C:/Projects/Tucil1_13522005/test/test1.txt
Select Mode
Manual
File
Generate

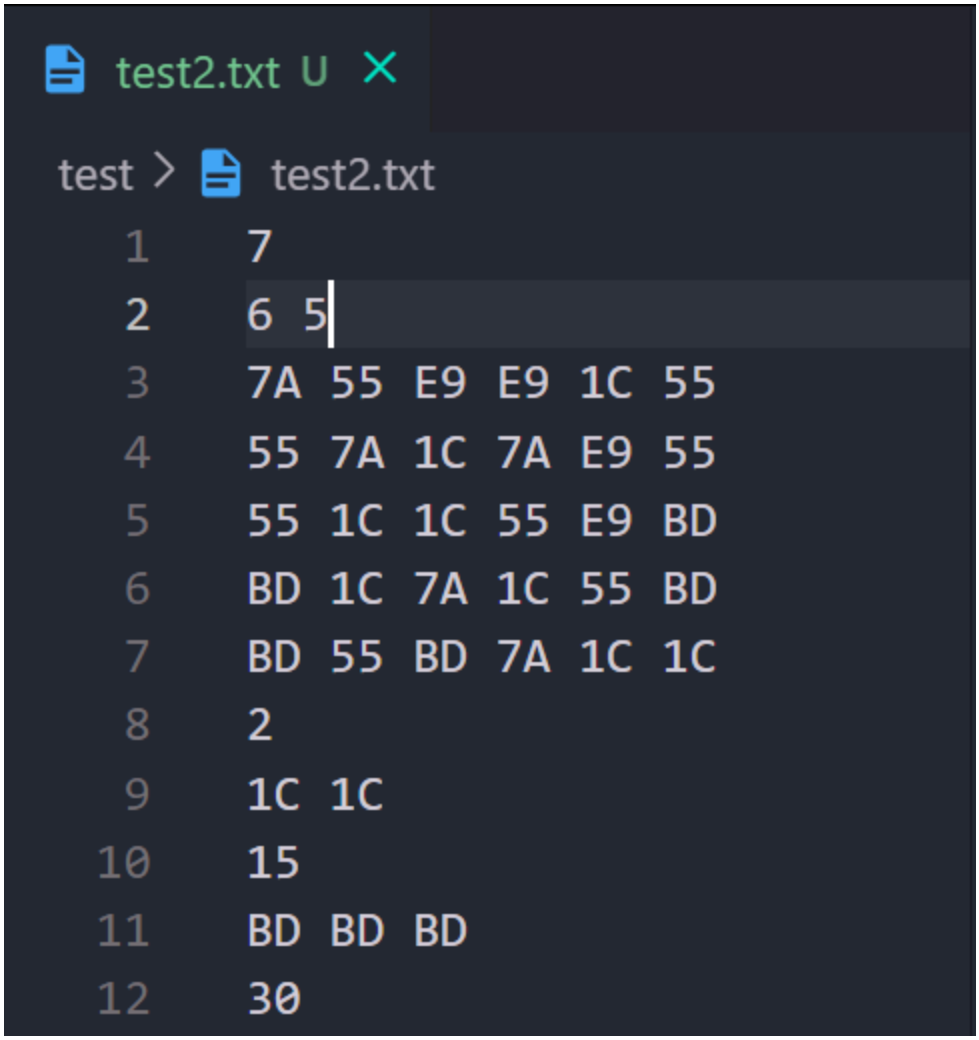

Cyberpunk 2077 Breach Protocol Solver Result
Cyberpunk 2077 Breach Protocol Solver Result
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
Sequences
Sequence 1
7A 1C
Weight
20
Sequence 2
BD 55
Weight
15
Sequence 3
BD 1C 7A
Weight
35
Solve
Result
Max Weight
70
Path
7A BD 1C 7A 1C BD 55
Time Taken
94.393 ms
Export Result


hello4.txt U
test
hello4.txt
70
7A BD 1C 7A 1C BD 55
1,1
1,5
6,5
6,6
1,6
1,4
5,4
94.393 ms

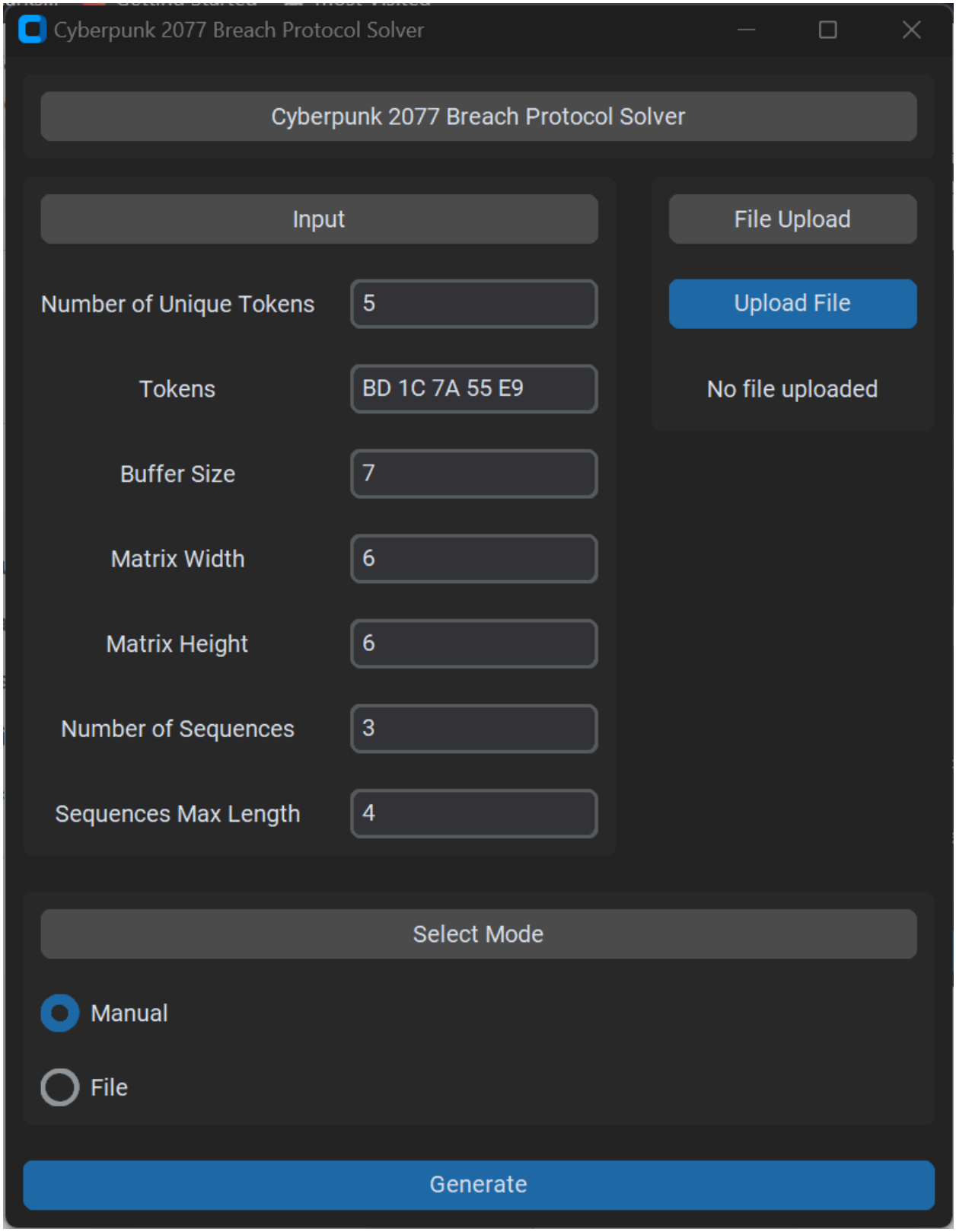

test2.txt U
test > test2.txt
7
6 5
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
2
1C 1C
15
BD BD BD
30

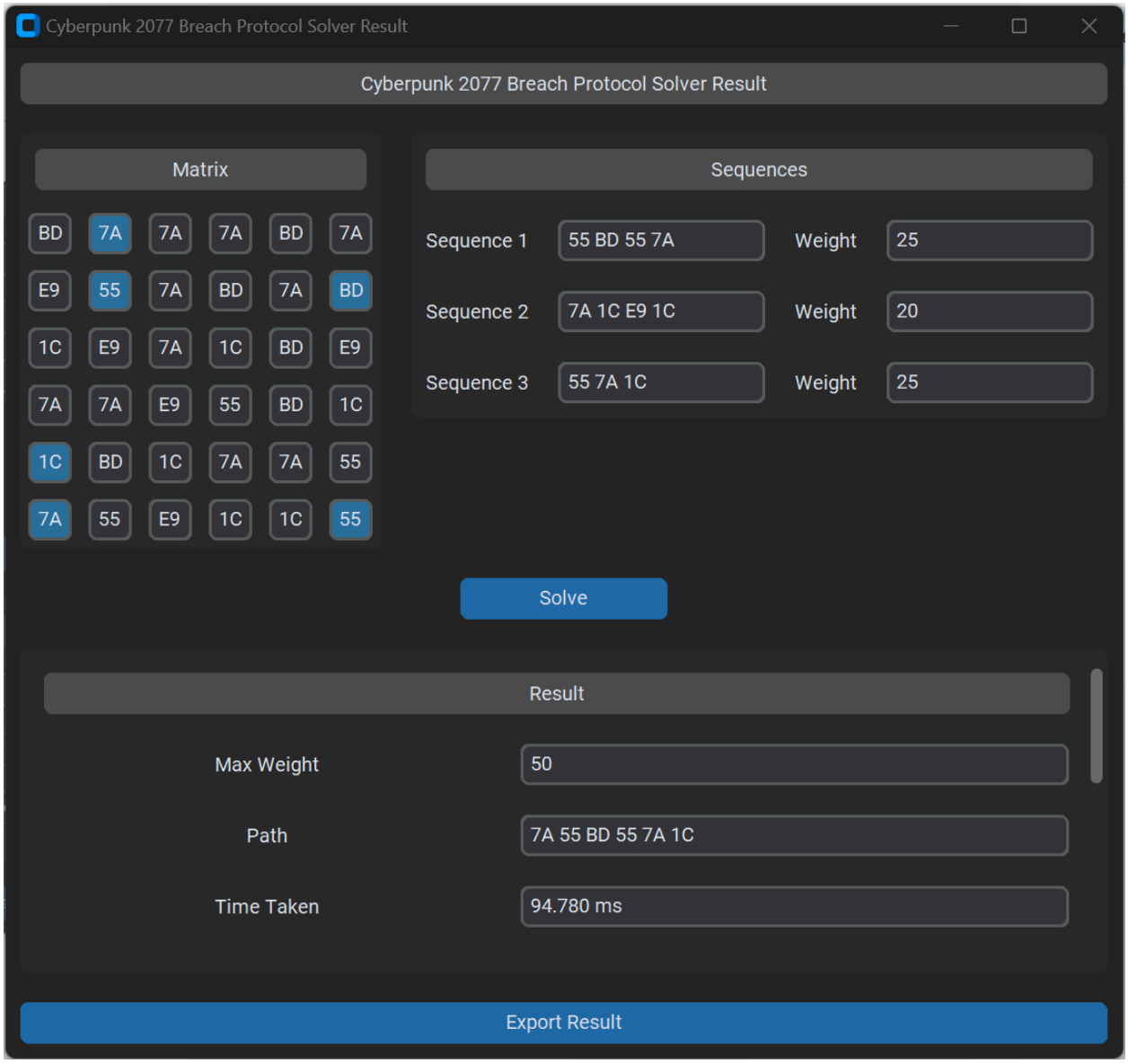

Cyberpunk 2077 Breach Protocol Solver
Cyberpunk 2077 Breach Protocol Solver
Input
Number of Unique Tokens
5
Tokens
BD 1C 7A 55 E9
Buffer Size
7
Matrix Width
6
Matrix Height
6
Number of Sequences
3
Sequences Max Length
4
File Upload
Upload File
C:/Projects/Tucil1_13522005/test/test2.txt
Select Mode
Manual
File
Generate

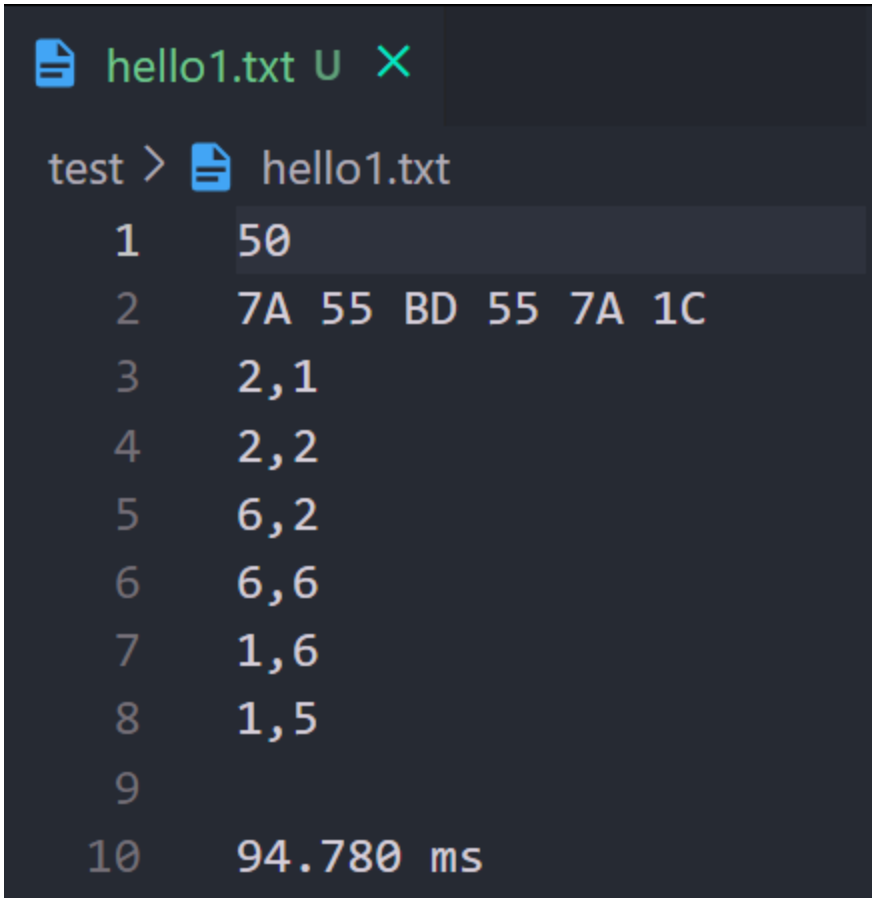

Cyberpunk 2077 Breach Protocol Solver Result
Cyberpunk 2077 Breach Protocol Solver Result
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
Sequences
Sequence 1
1C 1C
Weight
15
Sequence 2
BD BD BD
Weight
30
Solve
Result
Max Weight
45
Path
1C 1C BD BD BD
Time Taken
45.793 ms
Export Result

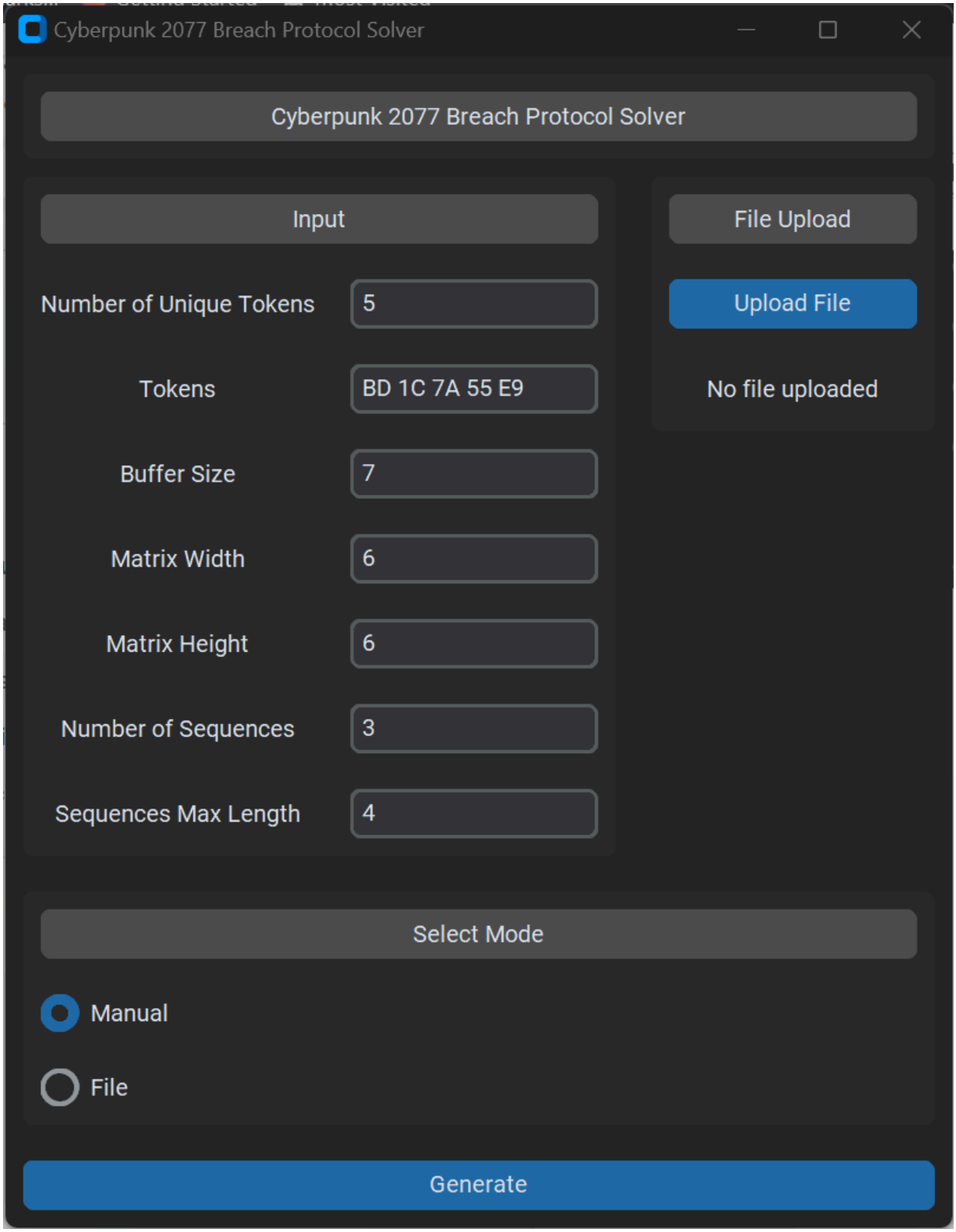


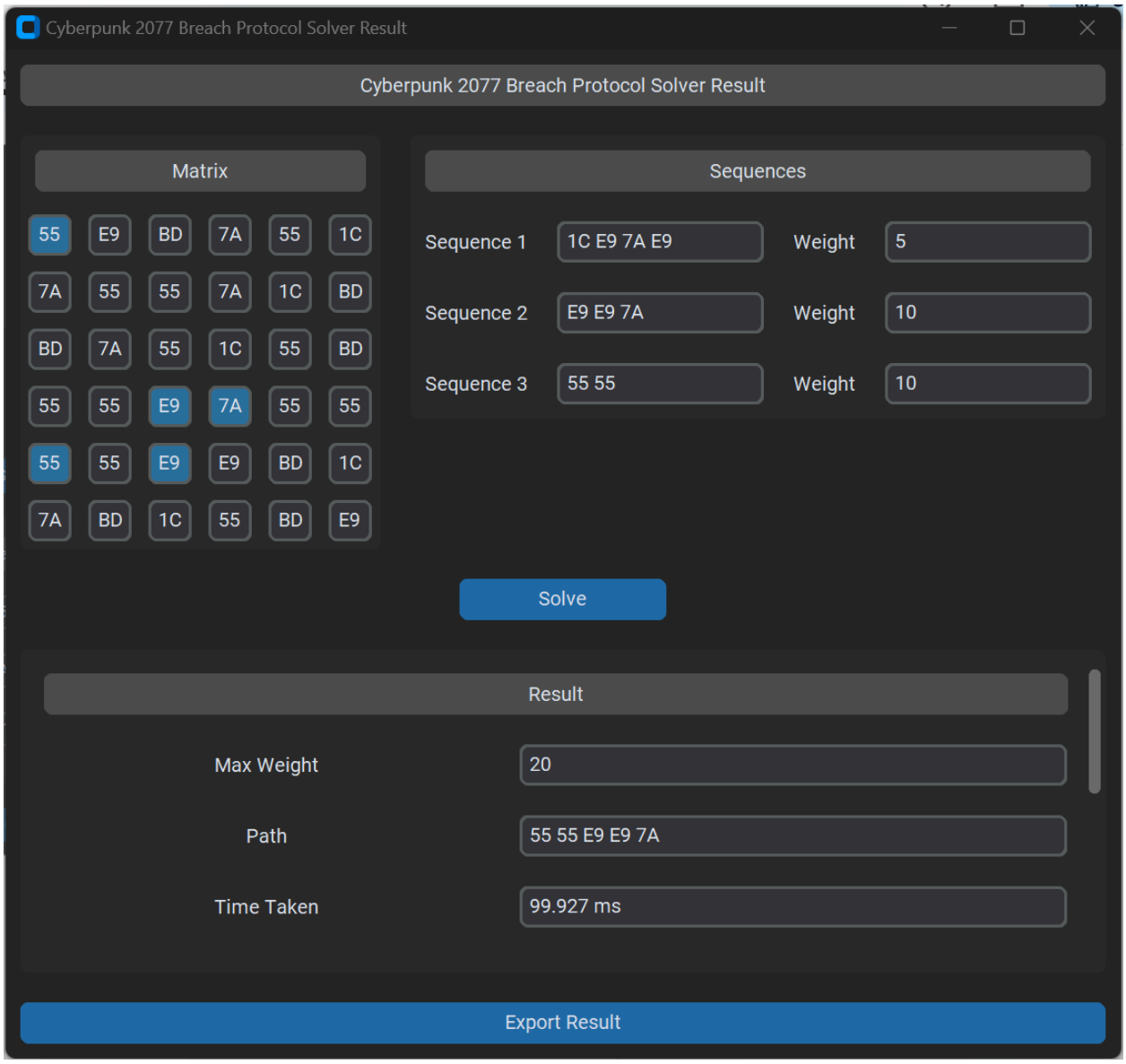

Auto Generated

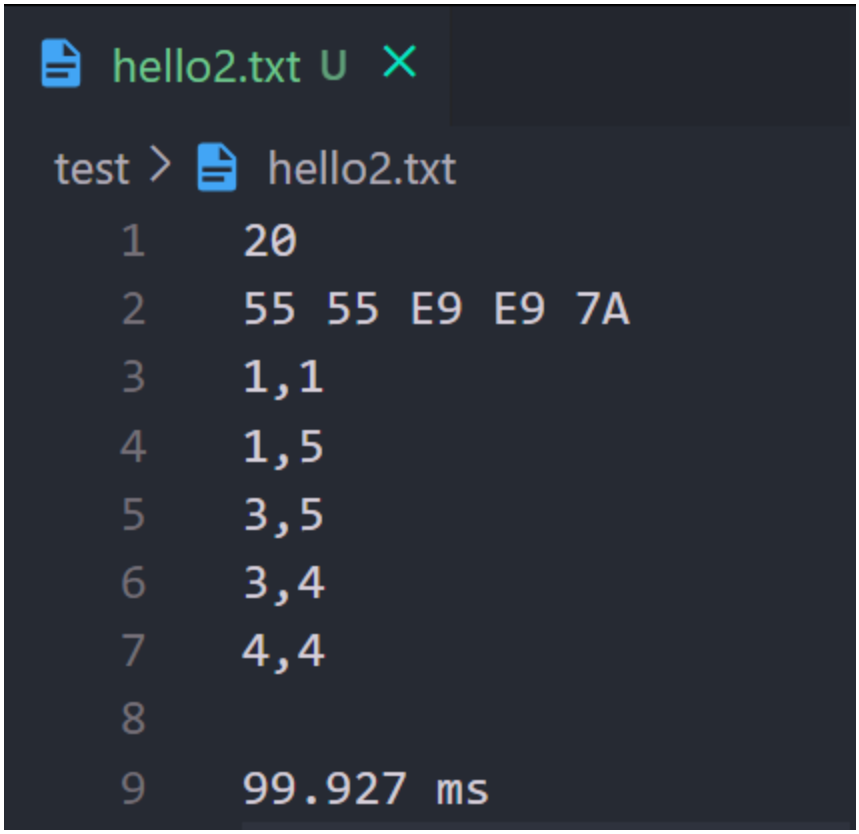

Cyberpunk 2077 Breach Protocol Solver
Cyberpunk 2077 Breach Protocol Solver
Input
Number of Unique Tokens
5
Tokens
BD 1C 7A 55 E9
Buffer Size
7
Matrix Width
6
Matrix Height
6
Number of Sequences
3
Sequences Max Length
4
File Upload
Upload File
No file uploaded
Select Mode
Manual
File
Generate

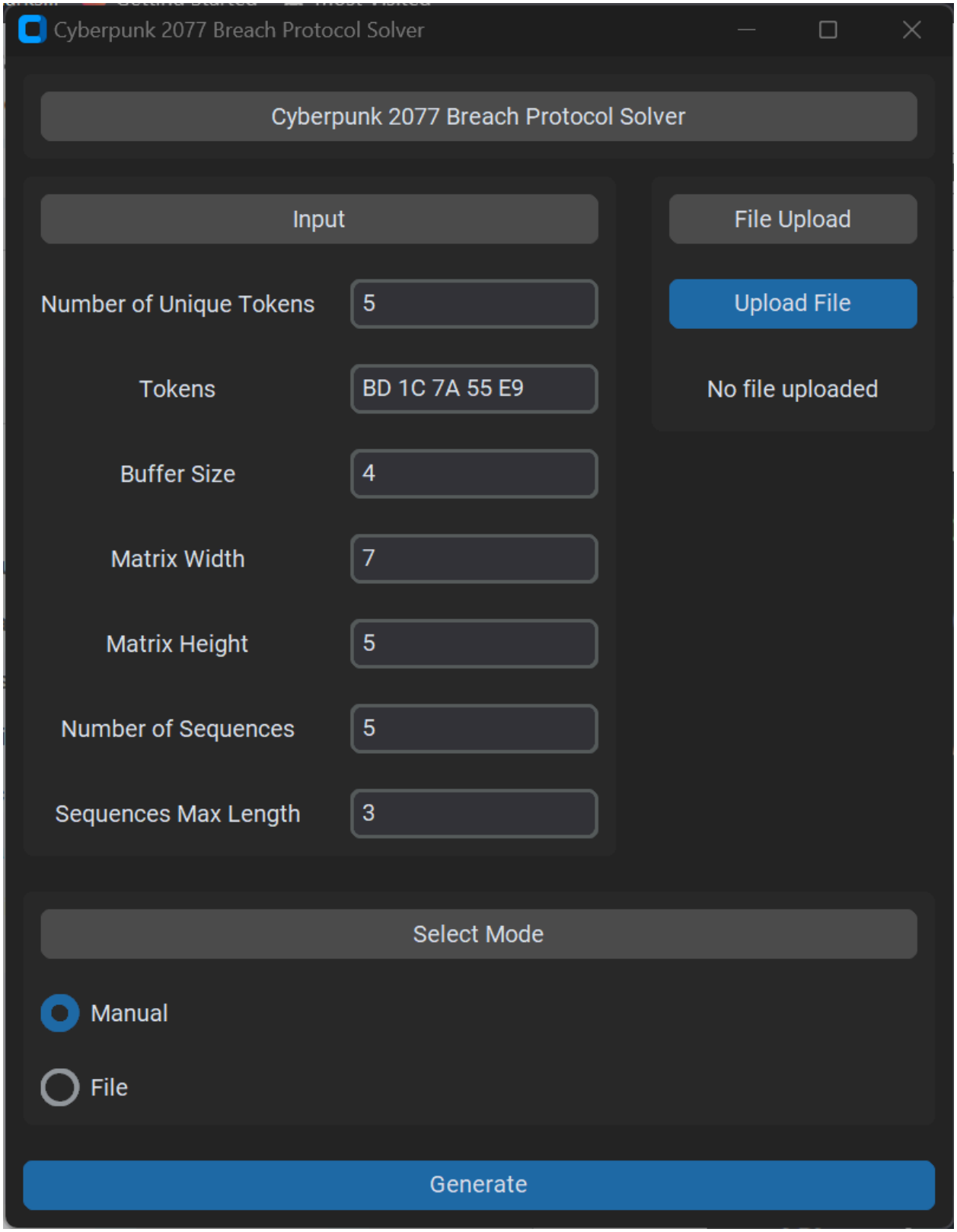

Cyberpunk 2077 Breach Protocol Solver Result
Cyberpunk 2077 Breach Protocol Solver Result
Matrix
BD 7A 7A 7A BD 7A
E9 55 7A BD 7A BD
1C E9 7A 1C BD E9
7A 7A E9 55 BD 1C
1C BD 1C 7A 7A 55
7A 55 E9 1C 1C 55
Sequences
Sequence 1
55 BD 55 7A
Weight
25
Sequence 2
7A 1C E9 1C
Weight
20
Sequence 3
55 7A 1C
Weight
25
Solve
Result
Max Weight
50
Path
7A 55 BD 55 7A 1C
Time Taken
94.780 ms
Export Result

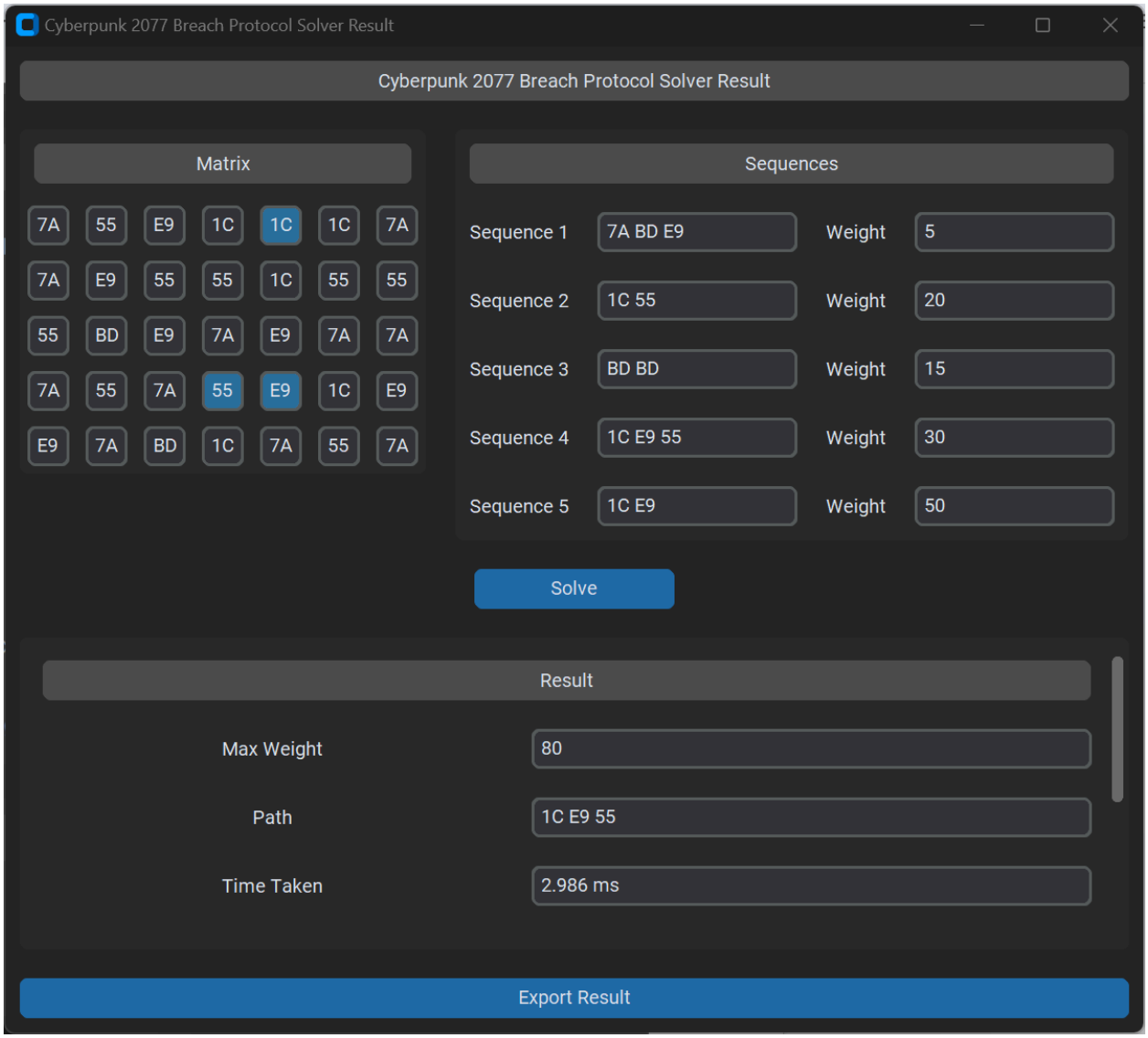

hello1.txt U
test > hello1.txt
1 50
2 7A 55 BD 55 7A 1C
3 2,1
4 2,2
5 6,2
6 6,6
7 1,6
8 1,5
9
10 94.780 ms

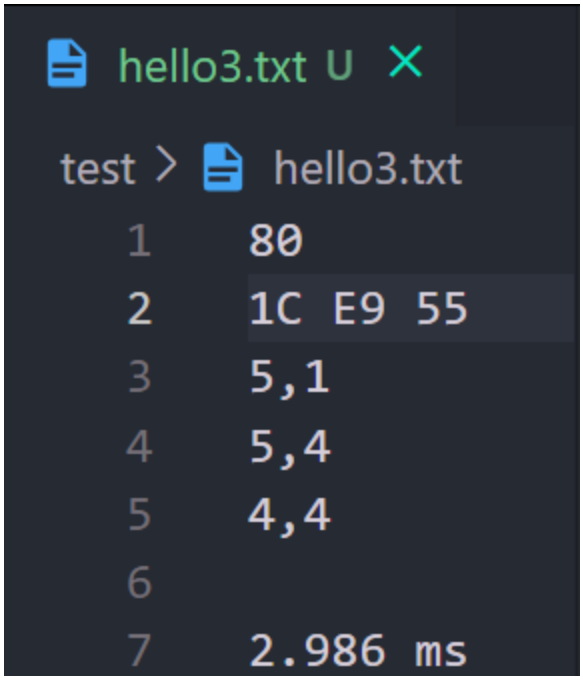

Cyberpunk 2077 Breach Protocol Solver
Cyberpunk 2077 Breach Protocol Solver
Input
Number of Unique Tokens
5
Tokens
BD 1C 7A 55 E9
Buffer Size
7
Matrix Width
6
Matrix Height
6
Number of Sequences
3
Sequences Max Length
4
File Upload
Upload File
No file uploaded
Select Mode
Manual
File
Generate


Cyberpunk 2077 Breach Protocol Solver Result
Cyberpunk 2077 Breach Protocol Solver Result
Matrix
55 E9 BD 7A 55 1C
7A 55 55 7A 1C BD
BD 7A 55 1C 55 BD
55 55 E9 7A 55 55
55 55 E9 E9 BD 1C
7A BD 1C 55 BD E9
Sequences
Sequence 1
1C E9 7A E9
Weight
5
Sequence 2
E9 E9 7A
Weight
10
Sequence 3
55 55
Weight
10
Solve
Result
Max Weight
20
Path
55 55 E9 E9 7A
Time Taken
99.927 ms
Export Result


hello2.txt U
test > hello2.txt
20
55 55 E9 E9 7A
1,1
1,5
3,5
3,4
4,4
99.927 ms


Cyberpunk 2077 Breach Protocol Solver
Cyberpunk 2077 Breach Protocol Solver
Input
Number of Unique Tokens
5
Tokens
BD 1C 7A 55 E9
Buffer Size
4
Matrix Width
7
Matrix Height
5
Number of Sequences
5
Sequences Max Length
3
File Upload
Upload File
No file uploaded
Select Mode
Manual
File
Generate


Cyberpunk 2077 Breach Protocol Solver Result
Cyberpunk 2077 Breach Protocol Solver Result
Matrix
7A 55 E9 1C 1C 1C 7A
7A E9 55 55 1C 55 55
55 BD E9 7A E9 7A 7A
7A 55 7A 55 E9 1C E9
E9 7A BD 1C 7A 55 7A
Sequences
Sequence 1 7A BD E9 Weight 5
Sequence 2 1C 55 Weight 20
Sequence 3 BD BD Weight 15
Sequence 4 1C E9 55 Weight 30
Sequence 5 1C E9 Weight 50
Solve
Result
Max Weight 80
Path 1C E9 55
Time Taken 2.986 ms
Export Result


hello3.txt U
test > hello3.txt
1 80
2 1C E9 55
3 5,1
4 5,4
5 4,4
6
7 2.986 ms

## 4. Lampiran

Link Repository: https://github.com/SandWithCheese/Tucil1_13522005

| 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 | ✓ | |