

LAPORAN TUGAS KECIL I
IF2211 STRATEGI ALGORITMA



Disusun oleh :
Christian Justin Hendrawan (13522135)

Program Studi Teknik Informatika
Institut Teknologi Bandung
2024

Daftar Isi

Daftar Isi.....	1
Bab I.....	2
I. Tujuan.....	2
II. Spesifikasi.....	2
Bab II.....	4
2.1. Definisi Algoritma Brute Force.....	4
2.2. Kekuatan dan Kelemahan Algoritma Brute Force.....	4
2.3. Penerapan Algoritma Brute Force Pada Tupil.....	4
Bab III.....	7
Bab IV.....	11
I. File.....	11
II. CLI.....	18
Bab VI.....	24
Bab VII.....	24

Bab I

Deskripsi Masalah

I. Tujuan

Tujuan dari pembuatan tugas besar ini adalah menemukan solusi dari permainan **Breach Protocol** yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan *algoritma brute force*.

II. Spesifikasi

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 di cocokan.
4. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.



Gambar 1. Permainan Breach Protocol

(Sumber : [Breach Protocol \(Hacking Mini Game\) Explained - Cyberpunk 2077 Game Guides \(nerdburglars.net\)](https://nerdburglars.net/breach-protocol-hacking-mini-game-explained-cyberpunk-2077-game-guides/))

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.

Di dalam Tugas Kecil 1 ini, Anda diminta membuat program sederhana dalam bahasa C/C#/C++/Java/Python/JavaScript/Go/PHP yang mengimplementasikan algoritma Brute Force untuk mencari solusi paling optimal permainan **Breach Protocol**.

Bab II

Teori Singkat

2.1. Definisi Algoritma Brute Force

Algoritma brute force adalah suatu pendekatan sederhana namun efektif dalam menyelesaikan masalah. Pendekatan ini melibatkan pengujian setiap kemungkinan solusi secara berurutan dengan menggunakan kekuatan komputasi. Algoritma ini pada dasarnya memeriksa semua kemungkinan secara berurutan dan membandingkan solusi yang dihasilkan hingga ditemukan solusi terbaik. Walaupun seringkali memerlukan waktu yang cukup lama untuk menangani masalah yang kompleks, metode ini tetap relevan dan dapat diterapkan dalam berbagai bidang seperti pemrograman, matematika, dan ilmu komputer.

2.2. Kekuatan dan Kelemahan Algoritma Brute Force

Beberapa kekuatan dari algoritma Brute Force adalah berikut.

1. Algoritma brute force dapat diterapkan untuk memecahkan hampir sebagian besar masalah (wide applicability).
2. Algoritma brute force sederhana dan mudah dimengerti.
3. Algoritma brute force menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, perkalian matriks.
4. Algoritma brute force menghasilkan algoritma baku (standard) untuk tugas-tugas komputasi seperti penjumlahan/perkalian n buah bilangan, menentukan elemen minimum atau maksimum di dalam senarai (larik).

Beberapa kelemahan dari algoritma Brute Force adalah berikut.

1. Algoritma brute force jarang menghasilkan algoritma yang mangkus.
2. Algoritma brute force umumnya lambat untuk masukan berukuran besar sehingga tidak dapat diterima.
3. Tidak sekonstruktif/sekreatif strategi pemecahan masalah lainnya.

2.3. Penerapan Algoritma Brute Force Pada Tupil

Dalam algoritma ini, kita mencoba semua kemungkinan hingga kita menemukan solusi yang optimal. Algoritma ini biasanya digunakan ketika kompleksitas komputasi bukanlah masalah dan kita membutuhkan solusi yang pasti. Berikut adalah langkah-langkah umum dalam algoritma Brute Force:

- 1. Tahap Menghasilkan:** Pertama, kita menghasilkan semua kemungkinan solusi untuk masalah tersebut. Misalnya, jika kita mencoba mencari kata dalam kamus,

kita akan mulai dari kata pertama dan bergerak satu per satu hingga kita menemukan kata yang kita cari.

2. Tahap Testing: Setelah kita memiliki solusi potensial, kita perlu memeriksa apakah solusi tersebut memenuhi semua kriteria yang kita butuhkan. Jika ya, kita telah menemukan solusi. Jika tidak, kita perlu mencoba solusi lain.

3. Tahap Mengembalikan solusi: Setelah kita menemukan solusi yang memenuhi semua kriteria, kita mengembalikan solusi tersebut.

Dengan langkah-langkah yang sudah disebutkan, maka kita dapat menerapkan algoritma brute force pada masalah pencarian jalur dengan *rewards* maksimum dalam permainan Cyberpunk 2077 Breach Protocol. Berikut adalah penjelasan mengenai penerapannya.

1. Pembuatan semua jalur yang mungkin

Fungsi `generate_paths` digunakan untuk menghasilkan semua jalur yang mungkin dari matriks kode. Fungsi ini menggunakan pendekatan rekursif untuk berjalan melalui semua jalur yang mungkin. Jalur dibuat dengan cara berjalan melalui matriks kode secara vertikal dan horizontal, dimulai dari koordinat (1,1). Fungsi ini memulai dengan jalur kosong dan pada setiap langkah, mencoba menambahkan setiap koordinat yang mungkin ke jalur. Jika jalur baru memiliki panjang yang sama dengan `buffer_size`, jalur tersebut ditambahkan ke daftar `completed_paths`. Jika tidak, jalur baru ditambahkan ke `partial_paths_stack` dan fungsi dipanggil lagi dengan jalur baru dan kandidat berikutnya.

2. Penghitungan skor untuk setiap jalur

Setelah semua jalur yang mungkin dihasilkan, terjadi proses menghitung skor untuk setiap jalur. Skor untuk setiap jalur dihitung dengan cara membandingkan setiap elemen dalam jalur dengan setiap urutan dalam daftar urutan. Jika elemen dalam jalur cocok dengan elemen dalam urutan, skor urutan tersebut ditingkatkan. Jika tidak, skor urutan tersebut dikurangi. Skor total untuk jalur adalah jumlah skor semua urutan.

3. Pencarian jalur dengan skor tertinggi

Setelah skor dihitung untuk setiap jalur, terjadi proses mencari jalur dengan skor tertinggi. Ini dilakukan dengan mencari nilai maksimum dari semua skor dan kemudian mencari jalur yang memiliki skor tersebut.

4. Mengembalikan solusi dengan skor tertinggi

Akhirnya, jalur dengan skor tertinggi yang dimana juga memenuhi semua kriteria dikembalikan dan dicetak. Skor dari jalur tersebut juga dikembalikan dan ditampilkan.

Bab III

Source Code

```
1 import numpy as np
2 import time
3 import random
4
5 class DuplicatedCoordinateException(Exception):
6     pass # Exception yang muncul jika ada koordinat yang sama di dua Path.
7
8 class Path():
9     def __init__(self, coords=[]):
10         # Inisialisasi objek Path. Parameter yang diperlukan adalah coords (koordinat),
11         # yang merupakan list dari koordinat path.
12         self.coords = coords
13
14     def __add__(self, other):
15         # Menambahkan dua objek Path.
16         # Jika ada koordinat yang sama di kedua Path, maka akan muncul DuplicatedCoordinateException.
17         new_coords = self.coords + other.coords
18         if any(map(lambda coord: coord in self.coords, other.coords)):
19             raise DuplicatedCoordinateException()
20         return Path(new_coords)
21
22     def __repr__(self):
23         # Merepresentasi string dari objek Path.
24         # Ini akan mengembalikan string dari list koordinat path.
25         return '\n'.join([f"{col}, {row}" for row, col in self.coords])
```

```
1 class SequenceScore():
2     def __init__(self, sequence, buffer_size, sequence_score, reward_level=0):
3         # Inisialisasi objek SequenceScore. Parameter yang diperlukan adalah sequence (urutan angka),
4         # buffer_size (ukuran maksimum sequence), sequence_score (skor untuk sequence), dan reward_level (level reward).
5         self.max_progress = len(sequence)
6         self.sequence = sequence
7         self.score = 0
8         self.reward_level = reward_level
9         self.buffer_size = buffer_size
10        self.sequence_score = sequence_score
11
12    def compute(self, compare):
13        # Menghitung skor sequence. Parameter yang diperlukan adalah compare (angka untuk dibandingkan
14        # dengan angka saat ini dalam sequence). Jika angka cocok, metode __increase dipanggil. Jika tidak, metode __decrease dipanggil.
15        if not self.__completed():
16            if self.sequence[self.score] == compare:
17                self.__increase()
18            else:
19                self.__decrease()
20
21    def __increase(self):
22        # Meningkatkan skor jika angka dalam sequence cocok. Jika sequence selesai,
23        # skor diatur ke sequence_score.
24        self.buffer_size -= 1
25        self.score += 1
26        if self.__completed():
27            self.score = self.sequence_score
28
29    def __decrease(self):
30        # Mengurangi skor jika angka dalam sequence tidak cocok. Jika sequence selesai,
31        # skor diatur ke 0.
32        self.buffer_size -= 1
33        if self.score > 0:
34            self.score -= 1
35        if self.__completed():
36            self.score = 0
37
38    def __completed(self):
39        # Memeriksa apakah sequence selesai. Sequence dianggap selesai jika skor kurang dari 0,
40        # skor lebih besar atau sama dengan progress maksimum, atau buffer_size kurang dari progress maksimum dikurangi skor.
41        return self.score < 0 or self.score >= self.max_progress or self.buffer_size < self.max_progress - self.score
42
```



```

1 class PathScore():
2     def __init__(self, path, sequences, buffer_size, code_matrix):
3         # Inisialisasi objek PathScore. Parameter yang diperlukan adalah path (jalur), sequences (urutan sequence),
4         # buffer_size (ukuran maksimum sequence), dan code_matrix (matriks kode).
5         self.score = None
6         self.path = path
7         self.code_matrix = code_matrix
8         self.sequence_scores = [SequenceScore(sequence, buffer_size, score, reward_level) for reward_level, (sequence, score) in enumerate(sequences)]
9
10    def compute(self):
11        # Menghitung skor total untuk path. Skor total adalah jumlah skor semua sequence.
12        if self.score != None:
13            return self.score
14        for row, column in self.path.coords:
15            for seq_score in self.sequence_scores:
16                seq_score.compute(self.code_matrix[row-1][column-1])
17        self.score = sum(map(lambda seq_score: seq_score.score, self.sequence_scores))
18        return self.score

```

```

1 def generate_paths(buffer_size, code_matrix):
2     # Fungsi ini digunakan untuk menghasilkan semua path yang mungkin. Parameter yang diperlukan adalah buffer_size (ukuran maksimum sequence)
3     # dan code_matrix (matriks kode).
4
5     completed_paths = []
6
7     def candidate_coords(code_matrix, turn=0, coordinate=(1,1)):
8         # Fungsi ini digunakan untuk menghasilkan koordinat kandidat untuk langkah selanjutnya. Parameter yang diperlukan adalah code_matrix (matriks kode),
9         # turn (giliran), dan coordinate (koordinat).
10        if turn % 2 == 0:
11            return [(coordinate[0], column) for column in range(1, len(code_matrix)+1)]
12        else:
13            return [(row, coordinate[1]) for row in range(1, len(code_matrix)+1)]
14
15    def _walk_paths(buffer_size, completed_paths, partial_paths_stack = [Path()], turn = 0, candidates = candidate_coords(code_matrix)):
16        # Fungsi ini digunakan untuk berjalan melalui semua path yang mungkin dan menyimpan path yang selesai. Parameter yang diperlukan adalah buffer_size (ukuran maksimum sequence),
17        # completed_paths (path yang selesai), partial_paths_stack (stack path parsial), turn (giliran), dan candidates (kandidat).
18        path = partial_paths_stack.pop()
19        for coord in candidates:
20            try:
21                new_path = path + Path([coord])
22            except DuplicatedCoordinateException:
23                continue
24
25            if len(new_path.coords) == buffer_size:
26                completed_paths.append(new_path)
27            else:
28                partial_paths_stack.append(new_path)
29                _walk_paths(buffer_size, completed_paths, partial_paths_stack, turn + 1, candidate_coords(code_matrix, turn + 1, coord))
30
31    _walk_paths(buffer_size, completed_paths)
32
33    return completed_paths
34
35    # Mengembalikan semua path yang selesai.
36

```

```

1  def main():
2      input_method = input("Masukkan input dari file atau dari keyboard? (file/keyboard) ")
3
4      if input_method.lower() == 'file':
5          file_name = input("Masukkan nama file input: ")
6          with open(file_name, 'r') as f:
7              buffer_size = int(f.readline().strip())
8              if buffer_size == 0:
9                  print("No solution : Tidak ada ukuran buffer")
10                 return
11                 rows, cols = map(int, f.readline().split())
12                 code_matrix = []
13                 for _ in range(rows):
14                     row_string = f.readline().strip()
15                     row = [int(hex_string, 16) for hex_string in row_string.split()]
16                     code_matrix.append(row)
17                 code_matrix = np.array(code_matrix)
18
19                 num_sequences = int(f.readline())
20                 sequences = []
21                 for _ in range(num_sequences):
22                     sequence_string = f.readline().strip()
23                     sequence = [int(hex_string, 16) for hex_string in sequence_string.split()]
24                     score = int(f.readline())
25                     sequences.append((sequence, score))
26
27             elif input_method.lower() == 'keyboard':
28                 num_token_types = int(input("Masukkan Jumlah token unik: "))
29                 token_types = input("Masukkan Token: ").split()
30                 if len(token_types) != num_token_types:
31                     print("Banyak masukan tipe token tidak sesuai dengan jumlah tipe token")
32                     return
33
34                 buffer_size = int(input("Masukkan Ukuran buffer: "))
35                 if buffer_size == 0:
36                     print("No solution : Tidak ada ukuran buffer")
37                     return
38                 rows, cols = map(int, input("Masukkan banyak baris dan kolom: ").split())
39
40                 # Generate a random matrix
41                 code_matrix = [[int(random.choice(token_types), 16) for _ in range(cols)] for _ in range(rows)]
42                 code_matrix = np.array(code_matrix)
43
44                 print("Matrix yang terbuat:")
45                 for row in code_matrix:
46                     print(' '.join(format(num, '02X') for num in row))
47
48                 num_sequences = int(input("Masukkan jumlah sekuens: "))
49                 max_sequence_length = int(input("Masukkan ukuran maksimal sekuens: "))
50                 sequences = []
51                 for _ in range(num_sequences):
52                     # Generate a random sequence length up to the maximum
53                     sequence_length = random.randint(1, max_sequence_length)
54                     # Generate a random sequence and a random score
55                     sequence = [int(random.choice(token_types), 16) for _ in range(sequence_length)]
56                     score = random.randint(10, 50)
57                     sequences.append((sequence, score))
58
59                 print("Sekuens beserta skor yang terbuat:")
60                 for sequence, score in sequences:
61                     print(f"Sequence: {' '.join(format(num, '02X') for num in sequence)}, Score: {score}")
62             else:
63                 print("Metode tersebut tidak valid kawan :)")
64                 return

```

```

1  start_time = time.time()
2  paths = [(path, PathScore(path, sequences, buffer_size, code_matrix).compute()) for path in generate_paths(buffer_size, code_matrix)]
3
4  max_score = max(score for _, score in paths) if paths else 0
5
6  if max_score == 0:
7      output = "Skor maksimum: 0\n"
8      output += "Tidak ada token yang dikunjungi\n"
9      output += "Tidak ada jalur yang mencapai skor maksimum\n\n"
10
11  else:
12      max_path = max(paths, key=lambda path: path[1]) # Mengambil path dengan skor tertinggi
13      output = f"Skor maksimum: {max_path[1]}\n"
14
15      elements_visited = [format(code_matrix[row-1][col-1], '02X') for row, col in max_path[0].coords] # Mengambil elemen yang dikunjungi
16      output += f"Token yang dikunjungi: {' '.join(elements_visited)}\n"
17      output += f"Jalur dengan skor maksimum:\n{max_path[0]}\n\n"
18
19  end_time = time.time() # Menghentikan menghitung waktu eksekusi
20  elapsed_time = (end_time - start_time) * 1000 # Kalkulasi waktu eksekusi dalam milidetik
21  output += f"Waktu eksekusi program: {elapsed_time} ms\n" # Menambahkan waktu eksekusi ke output
22  print(output)
23
24  save_output = input("Apakah ingin menyimpan solusi? (y/n) ")
25  if save_output.lower() == 'y':
26      with open('output.txt', 'w') as f:
27          f.write(output)
28
29  if __name__ == '__main__':
30      main()

```

Untuk tambahan konteks, *Source code* yang ditunjukkan di atas adalah *code* program yang tidak memiliki GUI. Hal tersebut dikarenakan *code* dari program ini memiliki logika yang sama dengan program yang mempunyai GUI namun dengan *line of code* yang lebih sedikit.

Bab IV

Tangkapan Layar Input dan Output

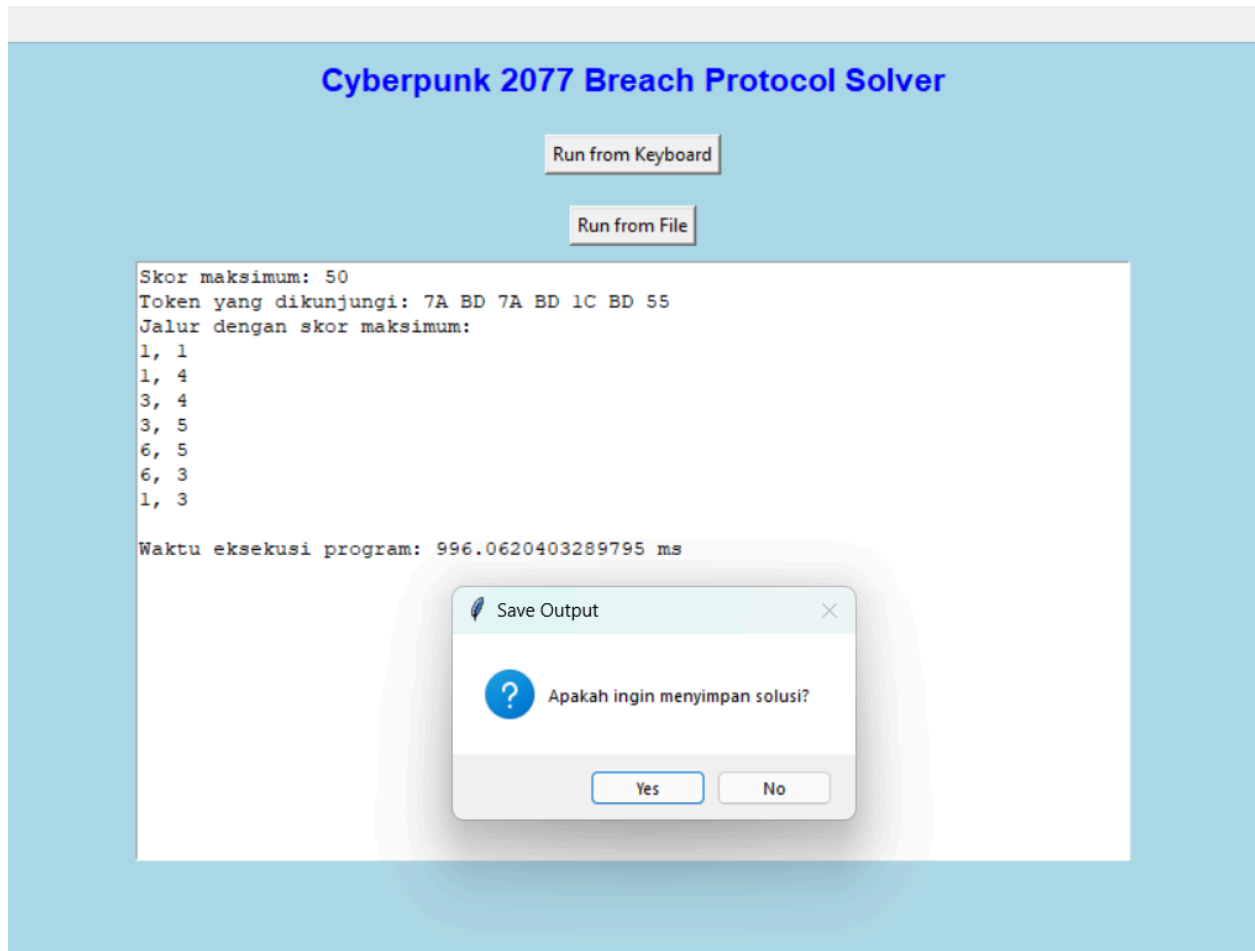
I. File

Bagian ini akan meliputi tangkapan layar input dan output dari pemrosesan program dengan menggunakan metode *file*. Program akan pertama-tama meminta input file yang dapat dipilih dari komputer user. File yang akan dipindai hanya berupa file yang ber-ekstensi **.txt** Berikut merupakan hasilnya.

1. Ketika input yang diberikan sudah sesuai dan ditemukan jalur optimal pada matrix
Input :

```
≡ input1.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 55
15 30
```

Output :



Catatan : Koordinat jalur terdiri dari (Kolom),(Baris)

2. Ketika tidak ada sekuens yang dapat dibentuk menjadi jalur pada matrix

Input :

```
input2.txt
1 6
2 5 6
3 7A 7A 7A 7A 7A 7A
4 7A 7A 7A 7A 7A 7A
5 7A 7A 7A 7A 7A 7A
6 7A 7A 7A 7A 7A 7A
7 7A 7A 7A 7A 7A 7A
8 3
9 BD E9 1C
10 10
11 55 55 55
12 20
13 1C FF E9
14 30
```

Output:



3. Ketika semua skor sekuence adalah 0

Input :

```
≡ input3.txt  
1 7  
2 6 6  
3 E9 E9 E9 1C E9 E9  
4 E9 E9 1C 55 7A 55  
5 55 55 1C BD BD BD  
6 55 55 55 1C E9 E9  
7 55 E9 55 7A BD E9  
8 1C 55 1C 7A 1C 7A  
9 3  
10 1C 7A 7A  
11 0  
12 E9 55 1C 55  
13 0  
14 E9 1C  
15 0
```

Output :



Catatan : Dikarenakan semua nilai sekuens adalah 0, maka penulis memberikan asumsi bahwa tidak terdapat jalur yang memiliki skor paling optimal. Maka dari itu, program tidak mengunjungi elemen mana pun dalam matrix

4. Ketika terdapat nilai sekuens yang negatif

Input :

```
input4.txt
1 5
2 6 7
3 55 1C FF FF FF 7A 55
4 E9 55 FF 1C E9 7A 1C
5 7A FF 55 55 1C FF 7A
6 1C 55 7A FF 1C 7A FF
7 FF 1C 7A FF 55 E9 55
8 1C 7A E9 7A FF FF E9
9 3
10 FF FF FF 7A
11 32
12 55 7A FF 7A
13 -10
14 55 1C
15 26
```

Output:

Cyberpunk 2077 Breach Protocol Solver

Run from Keyboard

Run from File

Skor maksimum: 32
Token yang dikunjungi: 55 FF FF FF 7A
Jalur dengan skor maksimum:
1, 1
1, 5
4, 5
4, 1
6, 1
Waktu eksekusi program: 34.04045104980469 ms

Save Output

Apakah ingin menyimpan solusi?

Yes No

5. Ketika semua nilai sekuens negatif

Catatan : Karena semua nilai sekuens negatif maka asumsi yang diberikan oleh penulis adalah langkah paling optimal adalah untuk tidak melangkah sama sekali. Maka dari itu, solusi yang didapat adalah jarak maksimumnya 0 dan tidak ada elemen yang dikunjungi.

Input :

```
≡ input5.txt
1 7
2 7 7
3 FF 7A FF FF BD BD BD
4 1C BD 55 E9 1C 7A BD
5 BD 55 55 7A BD 1C FF
6 7A E9 1C 55 55 1C BD
7 FF FF 7A 7A 1C 7A FF
8 BD 1C FF BD BD 1C BD
9 1C 1C E9 BD 7A FF 55
10 4
11 E9 55
12 -13
13 55 1C 7A BD 55 FF
14 -12
15 7A 7A FF
16 -1
17 7A 55 55 FF FF BD
18 -4
```

Output:

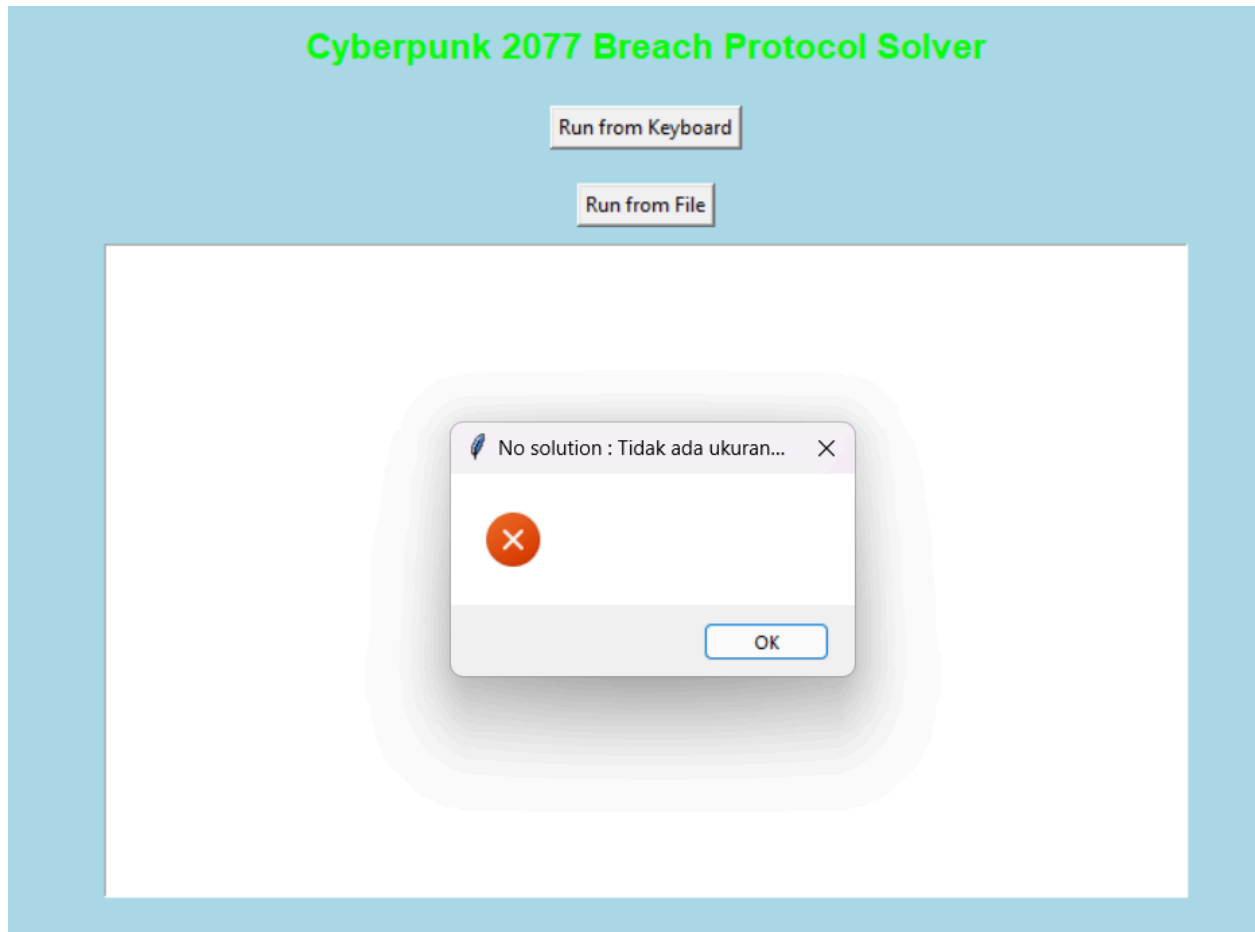


6. Ketika ukuran buffer diinput salah yakni 0

Input:

```
input6.txt
1  0
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 55
15 30
```

Output:



Catatan : Jika ukuran buffer 0, maka langsung terlihat pesan error yang memberitahu bahwa “tidak ada ukuran buffer”.

II. CLI

Bagian ini akan meliputi tangkapan layar input dan output dari pemrosesan program dengan menggunakan metode *CLI*. Pada bagian CLI, user akan diminta untuk melakukan input jumlah_token_unik, token, ukuran_buffer, ukuran_matriks, jumlah_sekuens, dan ukuran_maksimal_sekuens secara terurut. Lalu mengklik tombol submit. Berikut merupakan hasilnya.

1. Input dan Output yang didapat apabila masukan user sesuai

Cyberpunk 2077 Breach Protocol Solver

5
BD 1C 7A 55 E9
7
6 6
3
4

Submit

Matrix yang terbuat:

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

Sekuens beserta skor yang terbuat:

```
Sekuens: BD 55 BD, Score: 36
Sekuens: 1C 1C 7A 1C, Score: 23
Sekuens: 1C 55 1C 1C, Score: 24
Skor maksimum: 60
Token yang dikunjungi: BD 55 BD 1C 55 1C 1C
Jalur dengan skor maksimum:
1, 1
1, 5
2, 5
2, 6
1, 6
1, 3
3, 3
```

Waktu eksekusi program: 977.6670932769775 ms

Save Output

?

Apakah ingin menyimpan solusi?

Yes

No

2. Input dan Output yang didapat apabila masukan user sesuai

Cyberpunk 2077 Breach Protocol Solver

6
BD 1C 7A 55 E9 FF
6
6 6
5
4

Submit

Matrix yang terbuat:

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

Sekuens beserta skor yang terbuat:

Sekuens: 1C, Score: 20
Sekuens: 55 BD, Score: 20
Sekuens: BD 55, Score: 26
Sekuens: 55 1C FF E9, Score: 45
Sekuens: 1C FF BD 7A, Score: 23
Skor maksimum: 91
Token yang dikunjungi: E9 BD 55 1C FF E9
Jalur dengan skor maksimum:

1, 1
1, 5
5, 5
5, 6
6, 6
6, 1

Waktu eksekusi program: 276.37434005737305 ms

Save Output

?

Apakah ingin menyimpan solusi?

Yes

No

3. Input dan output apabila user memasukkan ukuran buffer 0

Cyberpunk 2077 Breach Protocol Solver

6
BD 1C 7A 55 E9 FF
0
1 2
5
3

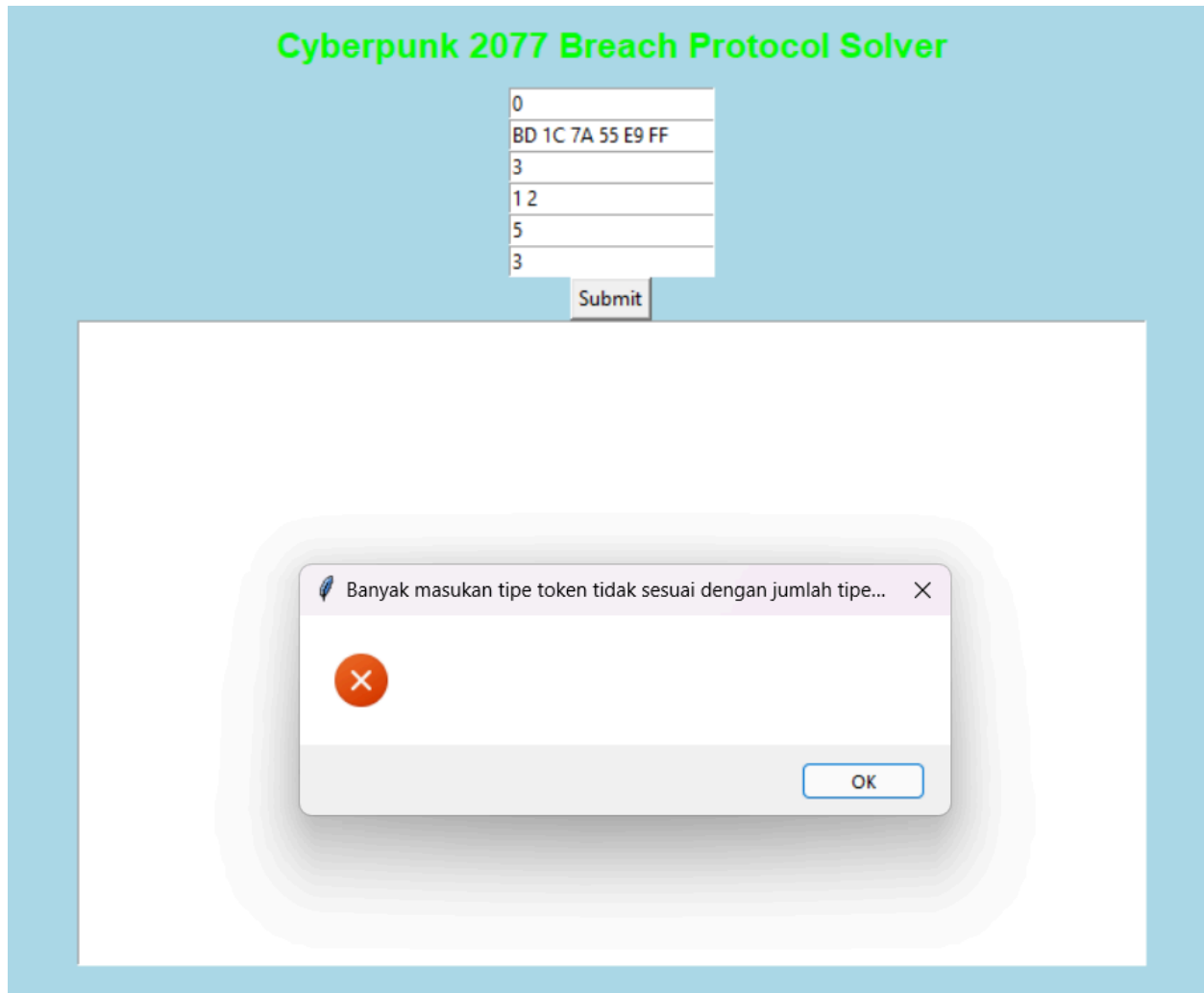
Submit

No solution : Tidak ada ukuran... ✕

✕

OK


4. Input dan output apabila user memasukkan jumlah token unik 0 atau ketika user memasukkan jumlah token unik yang tidak sesuai dengan banyak token unik yang dimasukkan pada kolom token



5. Ketika masukkan jumlah token unik tidak sesuai dengan token yang dimasukkan

Cyberpunk 2077 Breach Protocol Solver

6
BD 1C 7A 55 E9
6
7 7
5
5

 Banyak masukan tipe token tidak sesuai dengan jumlah tipe... ✕



Bab VI

Daftar Pustaka dan Link Repository

Berikut adalah daftar referensi yang dipakai dalam pengerjaan tugas besar ini.

1. <https://lamanit.com/algoritma-brute-force/> (Diakses pada 8 Februari 2024)
2. [PowerPoint Presentation \(itb.ac.id\)](https://www.itb.ac.id/) (Diakses pada 8 Februari 2024)
3. [Cyberpunk 2077 Breach Protocol guide - Polygon](#) (Diakses pada 9 Februari 2024)

Link Repository GitHub : [ChrisCS50X/Tucil1_13522135 \(github.com\)](https://github.com/ChrisCS50X/Tucil1_13522135)

Bab VII

Lampiran

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	✓	