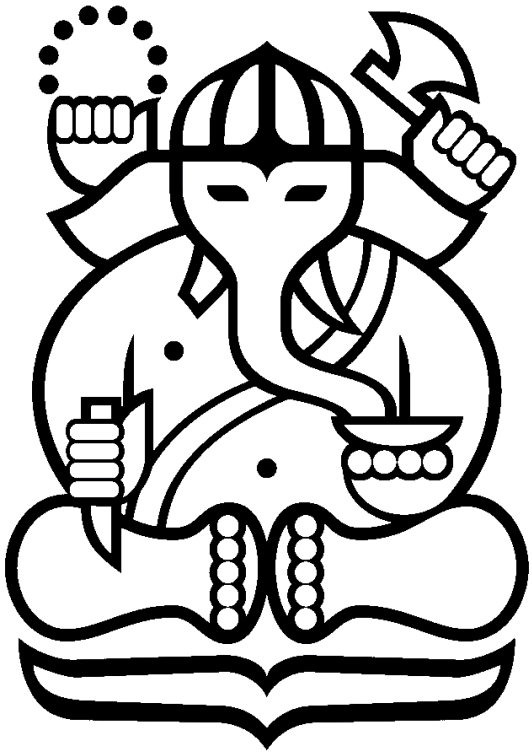
Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

13522069 - Nabila Shikoofa Muida

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

BAB I
DESKRIPSI MASALAH DAN ALGORITMA

1. Algoritma Brute Force

Algoritma Brute force merupakan sebuah pendekatan yang lempang (straightforward) untuk memecahkan suatu persoalan yang didefinisikan. Biasanya algoritma *brute force* didasarkan pada pernyataan pada persoalan (*problem statement*) dan definisi/konsep yang dilibatkan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, serta jelas caranya (*obvious way*).

Pendekatan *brute force* seringkali melibatkan enumerasi semua kemungkinan solusi, sebelum akhirnya menghilangkan jawaban yang tidak memenuhi syarat dan mengambil solusi terbaik (apabila ada). Algoritma dengan pendekatan *brute force* dijamin akan menemukan sebuah solusi apabila solusi tersebut ada. Namun, algoritma dengan pendekatan *brute force* seringkali tidak mangkus atau tidak efektif, dengan O(n) yang lebih buruk dari polinomial.

2. Cyberpunk 2077 Breach Protocol

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 terdiri dari token, matriks, sekuens, dan buffer. Token terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55. Matriks terdiri atas token-token yang akan dipilih untuk menyusun urutan kode. Sekuens adalah sebuah rangkaian token (dua atau lebih) yang harus dicocokkan. Buffer adalah jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol menetapkan bahwa pemain harus bergerak secara bergantian dalam pola horizontal dan vertikal sampai semua sekuens berhasil dicocokkan atau buffer penuh. Permainan dimulai dengan memilih satu token pada posisi baris teratas dari matriks, dan sekuens kemudian dicocokkan pada token-token di buffer. Satu token pada buffer dapat digunakan untuk lebih dari satu sekuens, dan setiap sekuens memiliki bobot hadiah yang berbeda. Panjang minimal sekuens adalah dua token.

3. Algoritma Penyelesaian Cyberpunk 2077 Breach Protocol dengan Pendekatan Brute Force

Dalam penyelesaian Cyberpunk 2077 Breach Protocol menggunakan pendekatan brute force, pemain akan secara sistematis mencoba semua kemungkinan susunan token dalam matriks untuk mencocokkan sekuens yang ada. Pendekatan ini melibatkan enumerasi semua kemungkinan solusi dengan mencoba setiap kemungkinan kombinasi token. Pemain akan mencoba setiap susunan token dalam matriks dengan memasukkan token ke dalam buffer dan mencocokkannya dengan sekuens yang ada, mengulangi proses ini hingga semua sekuens berhasil dicocokkan atau buffer penuh.

Algoritma brute force memastikan bahwa jika ada solusi yang memenuhi syarat, maka solusi tersebut akan ditemukan. Namun, karena algoritma brute force mencoba semua kemungkinan solusi, ia mungkin tidak efisien dan membutuhkan waktu yang lama terutama untuk matriks yang besar. Oleh karena itu, walaupun brute force menjamin solusi, dalam praktiknya seringkali tidak efektif, terutama karena kompleksitas waktu algoritmanya bisa menjadi buruk, tergantung pada ukuran matriks dan sekuens yang diberikan.

BAB 2
IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON

Dalam pembuatan program ini, penulis menggunakan bahasa pemrograman Java. Struktur dari program ini sendiri terbagi menjadi 3 file, yaitu main.py, matrix.py, dan solver.py.

1. Main.py

File ini berisi *driver* utama dari program sehingga tidak memiliki fungsi di dalamnya.

2. Matrix.py

2.1. generate_random_matrix(width, height):

- 2.1.1. Fungsi ini menghasilkan matriks acak dengan lebar dan tinggi yang ditentukan.
- 2.1.2. Input: lebar dan tinggi matriks.
- 2.1.3. Output: matriks acak.

2.2. generate_random_token():

- 2.2.1. Fungsi ini menghasilkan token acak untuk dimasukkan ke dalam matriks.
- 2.2.2. Input: Tidak ada.
- 2.2.3. Output: token acak.

2.3. read_problem_from_file(filename):

- 2.3.1. Fungsi ini membaca masalah dari file dan mengembalikan informasi matriks, sekuens, dan hadiahnya.
- 2.3.2. Input: nama file.
- 2.3.3. Output: ukuran buffer, matriks, daftar sekuens, dan daftar hadiah.

2.4. write_solution_to_file(output_file_path, buffer_size, matrix, sequences, rewards, solution, score):

- 2.4.1. Fungsi ini menulis solusi ke file.
- 2.4.2. Input: jalur file keluaran, ukuran buffer, matriks, daftar sekuens, daftar hadiah, solusi terbaik, dan skor terbaik.
- 2.4.3. Output: tidak ada (menulis ke file keluaran).

3. Solver.py

3.1. find_solution_brute_force(matrix, sequences):

3.1.1. Fungsi ini mencari solusi untuk masalah pencarian token secara brute force.

3.1.2. Input: matriks dan daftar sekuens.

3.1.3. Output: solusi terbaik, sekuens terbaik, dan skor terbaik.

3.2. find_token_recursive(matrix, sequences, current_sequence, current_index, current_position, visited):

3.2.1. Fungsi ini melakukan pencarian token secara rekursif untuk sekuens saat ini.

3.2.2. Input: matriks, daftar sekuens, sekuens saat ini, indeks saat ini, posisi saat ini, dan titik yang telah dikunjungi.

3.2.3. Output: urutan dari posisi token yang ditemukan.

3.3. find_token(matrix, sequences, current_sequence):

3.3.1. Fungsi ini mencari token awal dari suatu sekuens dalam matriks.

3.3.2. Input: matriks, daftar sekuens, dan sekuens saat ini.

3.3.3. Output: urutan dari posisi token yang ditemukan.

3.4. validate_hexdump(hexdump):

3.4.1. Fungsi ini memvalidasi hexdump untuk memastikan bahwa setiap token terdiri dari dua karakter alfanumerik.

3.4.2. Input: hexdump.

3.4.3. Output: tidak ada (melakukan validasi).

3.5. text_to_sequence(text):

3.5.1. Fungsi ini mengkonversi teks menjadi daftar sekuens berdasarkan karakter alfanumerik.

3.5.2. Input: teks.

3.5.3. Output: daftar sekuens.

BAB 3
SOURCE CODE PROGRAM

1. Repository Program

Repository program dapat diakses melalui:

- Github: https://github.com/nabilashikoofa/Tucil1_13522069

2. Source Code Program

2.1. Main.py

2.1.1. Mengatur alur program utama.

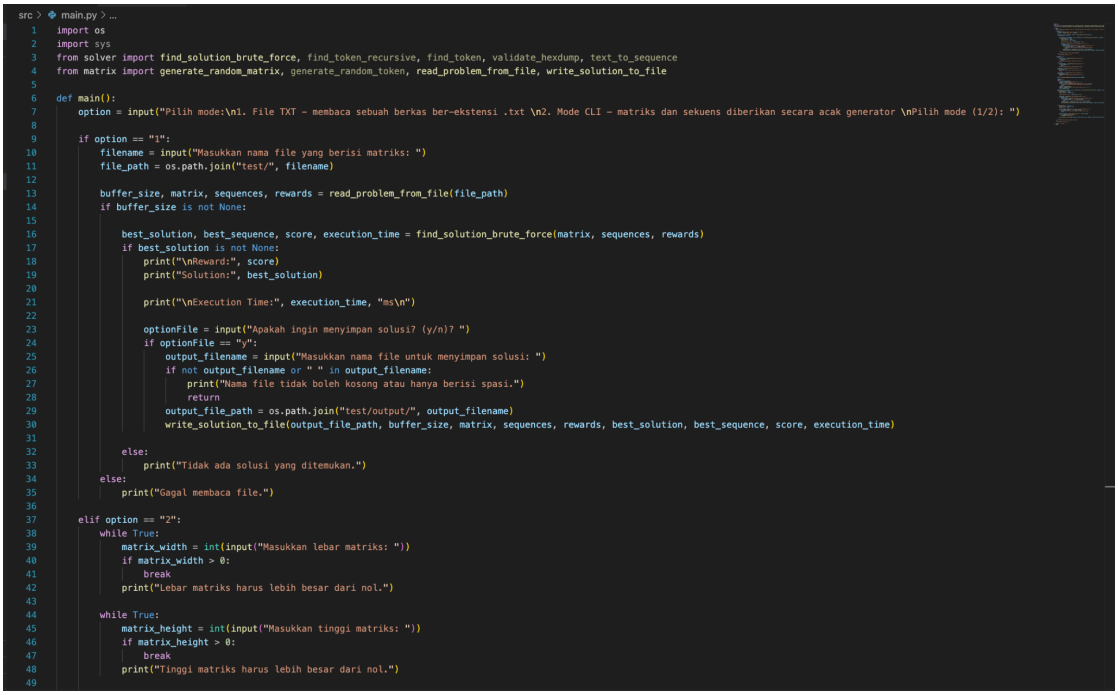
2.1.2. Memanggil fungsi-fungsi dari file lain.

2.1.3. Menyediakan antarmuka pengguna (CLI atau pembacaan dari file teks).

2.1.4. Fungsi untuk menampilkan solusi terbaik di layar.

2.1.5. Fungsi untuk menyimpan solusi terbaik ke dalam file teks.

2.1.6. Dengan file dan fungsi yang terorganisir seperti ini, Anda dapat memisahkan logika program ke dalam unit-unit terpisah yang mudah dikelola dan diuji secara terpisah.



```python
import os
import sys
from solver import find_solution_brute_force, find_token_recursive, find_token, validate_hexdump, text_to_sequence
from matrix import generate_random_matrix, generate_random_token, read_problem_from_file, write_solution_to_file

def main():
    option = input("Pilih mode:\n1. File TXT - membaca sebuah berkas ber-ekstensi .txt \n2. Mode CLI - matriks dan sekuens diberikan secara acak generator \nPilih mode (1/2): ")

    if option == "1":
        filename = input("Masukkan nama file yang berisi matriks: ")
        file_path = os.path.join("test/", filename)

        buffer_size, matrix, sequences, rewards = read_problem_from_file(file_path)
        if buffer_size is not None:

            best_solution, best_sequence, score, execution_time = find_solution_brute_force(matrix, sequences, rewards)
            if best_solution is not None:
                print("\nReward:", score)
                print("Solution:", best_solution)

                print("\nExecution Time:", execution_time, "ms\n")

                optionFile = input("Apakah ingin menyimpan solusi? (y/n)? ")
                if optionFile == "y":
                    output_filename = input("Masukkan nama file untuk menyimpan solusi: ")
                    if not output_filename or " " in output_filename:
                        print("Nama file tidak boleh kosong atau hanya berisi spasi.")
                        return
                    output_file_path = os.path.join("test/output/", output_filename)
                    write_solution_to_file(output_file_path, buffer_size, matrix, sequences, rewards, best_solution, best_sequence, score, execution_time)

            else:
                print("Tidak ada solusi yang ditemukan.")
        else:
            print("Gagal membaca file.")

    elif option == "2":
        while True:
            matrix_width = int(input("Masukkan lebar matriks: "))
            if matrix_width > 0:
                break
            print("Lebar matriks harus lebih besar dari nol.")

        while True:
            matrix_height = int(input("Masukkan tinggi matriks: "))
            if matrix_height > 0:
                break
            print("Tinggi matriks harus lebih besar dari nol.")
```

```
        buffer_size = int(input("Masukkan ukuran buffer: "))
        if buffer_size <= 0:
            print("Ukuran buffer harus lebih besar dari nol.")
            return

        num_sequences = int(input("Masukkan jumlah sequences: "))
        if num_sequences <= 0:
            print("Jumlah sequences harus lebih besar dari nol.")
            return

        matrix = generate_random_matrix(matrix_width, matrix_height)

        sequences = []
        rewards = []
        for i in range(num_sequences):
            sequence = input(f"Masukkan sequence ke-{i + 1}: ")
            sequence = sequence.replace(' ', '')
            if not all(c in '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ' for c in sequence):
                print("Sequence hanya boleh berisi angka dan huruf hexadecimal.")
                return
            sequences.append(sequence)
            reward = int(input(f"Masukkan reward untuk sequence ke-{i + 1}: "))
            rewards.append(reward)

        best_solution, best_sequence, score, execution_time = find_solution_brute_force(matrix, sequences, rewards)
        if best_solution is not None:
            print("\nReward:", score)
            print("Solution:", best_solution)

            print("\nExecution Time:", execution_time, "ms\n")

            optionFile = input("Apakah ingin menyimpan solusi? (y/n)? ")
            if optionFile == "y":
                output_filename = input("Masukkan nama file untuk menyimpan solusi: ")
                if not output_filename or " " in output_filename:
                    print("Nama file tidak boleh kosong atau hanya berisi spasi.")
                    return
                output_file_path = os.path.join("test/output/", output_filename)
                write_solution_to_file(output_file_path, buffer_size, matrix, sequences, rewards, best_solution, best_sequence, score, execution_time)


        if best_solution is not None:
            print("Solution:", best_solution)
            print("Score:", score)

            output_filename = input("Masukkan nama file untuk menyimpan solusi: ")
            if not output_filename or " " in output_filename:
                print("Nama file tidak boleh kosong atau hanya berisi spasi.")
```

```
        if best_solution is not None:
            print("Solution:", best_solution)
            print("Score:", score)

            output_filename = input("Masukkan nama file untuk menyimpan solusi: ")
            if not output_filename or " " in output_filename:
                print("Nama file tidak boleh kosong atau hanya berisi spasi.")
                return

            output_file_path = os.path.join("test/output/", output_filename)
            if not os.path.exists("test/output/"):
                os.makedirs("test/output/")

            write_solution_to_file(output_file_path, buffer_size, matrix, sequences, rewards, best_solution, best_sequence, score, execution_time)

        else:
            print("Tidak ada solusi yang ditemukan.")
    else:
        print("Opsi tidak valid.")

if __name__ == "__main__":
    main()
```

2.2. Matrix.py

2.2.1. Fungsi untuk membuat matriks acak.

```
def generate_random_matrix(width, height):
    matrix = []
    for _ in range(height):
        row = [generate_random_token() for _ in range(width)]
        matrix.append(row)
    return matrix
```

```
def generate_random_token():
    characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    return ''.join(random.choices(characters, k=2))
```

2.2.2. Fungsi untuk membaca persoalan dari file teks.

```python
def read_problem_from_file(filename):
    while True:
        try:
            with open(filename, 'r') as file:
                lines = file.readlines()

            buffer_size = int(lines[0].strip())

            matrix_info = lines[1].strip().split()
            matrix_width = int(matrix_info[0])
            matrix_height = int(matrix_info[1])

            if matrix_width <= 0 or matrix_height <= 0:
                raise ValueError("Ukuran matriks tidak valid.")

            matrix = [line.strip().split() for line in lines[2:2 + matrix_height]]
            matrix = [[str(cell) for cell in row] for row in matrix]

            sequences = []
            rewards = []

            current_line = 2 + matrix_height

            while current_line < len(lines):
                if lines[current_line].strip().isdigit():
                    num_sequences = int(lines[current_line].strip())
                    current_line += 1
                    for _ in range(num_sequences):
                        sequence_line = lines[current_line].strip().split()
                        sequence = [cell for cell in sequence_line]
                        sequences.append(sequence)

                        current_line += 1
                        reward = int(lines[current_line].strip())
                        rewards.append(reward)
                        current_line += 1
                else:
                    current_line += 1

            sorted_index = np.argsort(rewards)[::-1]
            sequences = [sequences[i] for i in sorted_index]
            rewards = [rewards[i] for i in sorted_index]

            return buffer_size, matrix, sequences, rewards

        except FileNotFoundError:
            print("File tidak ditemukan.")
            filename = input("Masukkan nama file yang berisi matriks: ")
            continue
```

2.2.3. Fungsi untuk menyimpan persoalan dan jawaban ke file teks.

```python
def write_solution_to_file(filename, buffer_size, matrix, sequences, rewards, solution, best_sequence, score, execution_time):
    try:
        with open(filename, 'w') as file:
            file.write("Buffer Size:\n")
            file.write(f"{buffer_size}\n\n")

            file.write("Matrix:\n")
            for row in matrix:
                file.write(' '.join(row) + '\n')
            file.write("\n")

            file.write("Sequences:\n")
            for seq in sequences:
                file.write(' '.join(seq) + '\n')
            file.write("\n")

            file.write("Rewards:\n")
            for reward in rewards:
                file.write(f"{reward}\n")
            file.write("\n")

            if solution is not None:
                file.write("Solution:\n")
                for pos in solution:
                    file.write(f"{pos}\n")
                file.write("\n")
                file.write(f"Best Sequence:\n")
                file.write(' '.join(best_sequence) + '\n')
                file.write("\n")
                file.write(f"Score: {score}\n")
                file.write("\n")
                file.write(f"Execution Time: {execution_time} ms\n")
            else:
                file.write("Tidak ada solusi yang ditemukan.\n")
        print(f"Solusi disimpan dalam file {filename}")
    except Exception as e:
        print(f"Terjadi kesalahan saat menyimpan solusi: {e}")
```

2.3. Solver.py

2.3.1. Fungsi untuk melakukan pencarian solusi brute force.

```python
def find_solution_brute_force(matrix, sequences, rewards):
    best_solution = None
    best_sequence = None
    best_score = 0
    start_time = time.time()
    execution_time = 0

    for i in range(len(matrix)):
        for j in range(len(matrix[0])):

            for seq_index, sequence in enumerate(sequences):
                if matrix[i][j] == sequence[0]:

                    solution = find_token(matrix, sequences, sequence)
                    if solution:
                        reward = sum(sequences[seq_index].count(matrix[pos[0]][pos[1]]) * rewards[seq_index] for pos in solution)
                        if reward > best_score:
                            best_solution = solution
                            best_sequence = sequence
                            best_score = reward

    end_time = time.time()
    execution_time = end_time - start_time

    return best_solution, best_sequence, best_score, execution_time
```

```python
def find_token_recursive(matrix, sequences, current_sequence, current_index, current_position, visited, current_token_index):
    if current_index == len(current_sequence):
        return [current_position]

    row, col = current_position
    next_token = current_sequence[current_token_index]

    # Implementasi untuk arah atas
    if (row + 1 < len(matrix) and matrix[row + 1][col] == next_token and (row + 1, col) not in visited):
        result = find_token_recursive(matrix, sequences, current_sequence, current_index + 1, (row + 1, col), visited + [(row + 1, col)], current_token_index + 1)
        if result:
            return [(row, col)] + result

    # Implementasi untuk arah kanan
    if (col + 1 < len(matrix[0]) and matrix[row][col + 1] == next_token and (row, col + 1) not in visited):
        result = find_token_recursive(matrix, sequences, current_sequence, current_index + 1, (row, col + 1), visited + [(row, col + 1)], current_token_index + 1)
        if result:
            return [(row, col)] + result

    # Implementasi untuk arah kiri
    if (col - 1 >= 0 and matrix[row][col - 1] == next_token and (row, col - 1) not in visited):
        result = find_token_recursive(matrix, sequences, current_sequence, current_index + 1, (row, col - 1), visited + [(row, col - 1)], current_token_index + 1)
        if result:
            return [(row, col)] + result

    # Implementasi untuk arah atas
    if (row - 1 >= 0 and matrix[row - 1][col] == next_token and (row - 1, col) not in visited):
        result = find_token_recursive(matrix, sequences, current_sequence, current_index + 1, (row - 1, col), visited + [(row - 1, col)], current_token_index + 1)
        if result:
            return [(row, col)] + result

    return None

def find_token(matrix, sequences, current_sequence):

    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == current_sequence[0]:

                result = find_token_recursive(matrix, sequences, current_sequence, 1, (i, j), [(i, j)], 0)
                if result:
                    return result
    return None
```

2.3.2. Fungsi-fungsi bantu seperti validasi hexdump dan konversi teks ke sekuens.

```python
def validate_hexdump(hexdump):
    # Fungsi untuk mengecek apakah semua elemen hexdump adalah string yang valid
    for row in hexdump:
        for token in row:
            if not isinstance(token, str) or len(token) != 2 or not token.isalnum():
                raise ValueError(f"Setiap token harus terdiri dari dua karakter alfanumerik")

def text_to_sequence(text):
    sequences = []
    current_sequence = ""

    for char in text:
        if char.isalnum():
            current_sequence += char

        else:
            if current_sequence:
                sequences.append(current_sequence)
                current_sequence = ""

    # Tambahkan urutan terakhir ke daftar sekuens jika tidak kosong
    if current_sequence:
        sequences.append(current_sequence)

    return sequences
```

BAB 4
MASUKAN DAN LUARAN PROGRAM

case1.txt

test > case1.txt

```
5
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30

```

ansCase1.txt

test > output > ansCase1.txt

```
Buffer Size:
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:
BD 1C BD 55
BD 7A BD
BD E9 1C

Rewards:
30
20
15

Solution:
(1, 1)
(1, 4)
(3, 4)
(3, 5)
(6, 5)
(6, 4)
(5, 4)

Best Sequence:
7A BD 7A BD 1C BD 55

Score: 50
Execution Time: 0.0006222724914550781 ms
```

```
nabilashikoofa@Nabilas-MacBook-Air Tucil1_13522069 % /usr/local/bin/python3
"
Pilih mode:
1. File TXT - membaca sebuah berkas ber-ekstensi .txt
2. Mode CLI - matriks dan sekuens diberikan secara acak generator
Pilih mode (1/2): 1
Masukkan nama file yang berisi matriks: case1.txt

Reward: 50
Sequence: 7A BD 7A BD 1C BD 55: 50
Solution: [(1, 1), (1, 4), (3, 4), (3, 5), (6, 5), (6, 4), (5, 4)

Execution Time: 0.0006070137023925781 ms

Apakah ingin menyimpan solusi? (y/n)? y
Masukkan nama file untuk menyimpan solusi: ansCase1.txt
Solusi disimpan dalam file test/output/ansCase1.txt
nabilashikoofa@Nabilas-MacBook-Air Tucil1_13522069 %
```

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

https://cyberpunk-hacker.com/

LAMPIRAN

Checklist Fitur

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		✓