TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA
Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Oleh:
Ahmad Mudabbir Arif
13522072

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

BAB I
ALGORITMA

1. Algoritma Brute Force

   Algoritma Brute Force merupakan pendekatan yang lempang (straightforward) untuk memecahkan suatu persoalan. Biasanya algoritma ini didasarkan pada *problem statement* dan *definition/concept*. Program yang dibuat memanfaatkan algoritma ini untuk mencari semua kemungkinan kombinasi atau disebut *path* dengan beberapa batasan yang disesuaikan dengan Algoritma 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 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.

2. Deskripsi Algoritma Program

   Dalam program ini, algoritma brute force dimanfaatkan dengan mencari semua kombinasi dari koordinat matriks (baris dan kolom) secara rekursif dengan langkah:

   a. Langkah awal ialah menerima inputan, berupa jumlah bufffer, matriks, sekuens dan hadiah yang dibuat 1 dictionary
   b. Kombinasi pertama dimulai dengan baris 0 atau baris pertama.
   c. Setiap 2 elemen terurut hanya memiliki baris atau kolom yang sama, contohnya [(1,1), (1,2), (3,2)], dst.

d. Setiap kombinasi dari koordinat akan diubah menjadi kombinasi elemen matriks yang disebut *path* dan ditambahkan pada dictionary dengan hadiah default adalah 0
e. Setiap *path* akan dilakukan pengecekan terhadap sekuens, apakah *path* tersebut memiliki sub-*path* yang sama dengan sekuens, jika iya maka nilai dari hadiah untuk *path* tersebut akan ditambah sesuai hadiah sekuensnya
f. Setelah dilakukan terhadap semua *path*, akan dicari *path* dengan nilai hadiah yang paling besar, jika terdapat *path* dengan nilai hadiah yang sama besar, maka akan diambil *path* dengan elemen yang paling sedikit yang dianggap optimal
g. Menampilkan output berupa nilai hadiah terbesar dan *path* nya.

BAB II
SOURCE CODE

A. Utils

1. Fungsi save_to_file(*buffer_size*, *matrix*, *num_sequences*, *sequences*, *filename*):

   Fungsi ini digunakan untuk menyimpan data matriks dan sekuens ke dalam sebuah file. Data yang disimpan mencakup ukuran buffer, ukuran matriks, matriks itu sendiri, jumlah sekuens, dan sekuens beserta bobot hadiahnya.

2. Fungsi read_file(*filename*):

   Fungsi ini digunakan untuk membaca data dari file yang diberikan. Membaca ukuran buffer, ukuran matriks, matriks itu sendiri, jumlah sekuens, dan sekuens beserta bobot hadiahnya dari file.

3. Fungsi write_file(*matrix*, *max_reward*, *max_path*, *time_ms*, *filename*):

   Fungsi ini digunakan untuk menulis hasil permainan ke dalam file setelah menyelesaikan permainan. Hasil permainan yang ditulis mencakup nilai hadiah maksimal, kombinasi elemen matriks terbaik, dan waktu eksekusi.

B. Main

1. Fungsi input_cli():

   Fungsi ini digunakan untuk mengambil input dari pengguna melalui CLI. Pengguna diminta untuk memasukkan jumlah token, token-token yang tersedia, ukuran buffer, ukuran matriks, jumlah sekuens, dan ukuran maksimal sekuens, jika tidak memasukkan apapun akan diambil nilai defaultnya. Fungsi mengembalikan tuple yang berisi data-data yang digenerate sesuai masukan pengguna.

2. Fungsi generate_matrix(*matrix_size*, *tokens*):

   Fungsi ini digunakan untuk menghasilkan matriks acak berdasarkan ukuran matriks dan token-token yang dimasukkan pengguna. Setiap elemen dalam matriks dipilih secara acak dari token-token yang tersedia.

3. Fungsi generate_sequences(*num_sequences*, *max_sequence_size*, *tokens*):

   Fungsi ini menghasilkan sekuens acak berdasarkan jumlah sekuens dan ukuran maksimal sekuens yang dimasukkan pengguna. Setiap sekuens memiliki panjang acak dan bobot hadiah acak yang ditentukan, dalam program ini randomize diantara -100 sampai 100 saja..

4. Fungsi generate_cli():

   Fungsi ini merupakan inti dari program untuk generate matriks dan sekuens di CLI. Pertama, meminta input dari pengguna menggunakan input_cli(). Selanjutnya, menghasilkan matriks dan sekuens acak. Meminta pengguna apakah ingin menyimpan hasil permainan ke dalam file. Jika pengguna menyetujui, hasil permainan disimpan dalam file dengan format yang sesuai.

5. Fungsi coordinates_to_elements(*matrix*, *coordinates*):

   Fungsi ini mengonversi koordinat dari matriks menjadi elemen-elemen token.

6. Fungsi breach_protocol(*buffer_size*, *matrix*, *sequences*):

   Fungsi ini merupakan inti dari algoritma permainan. Mencari kombinasi terbaik dari elemen-elemen matriks yang memberikan hadiah maksimal. Menggunakan algoritma brute force untuk mencari kombinasi terbaik. Dalam fungsi ini terdapat beberapa fungsi lagi antara lain.

   - Fungsi is_subarray(*arr*, *subarr*):

     Fungsi ini digunakan untuk mengecek apakah memiliki sub-array atau tidak.

   - Fungsi generate_combinations(*matrix_height*, *matrix_width*, *buffer_size*):

     Fungsi ini digunakan untuk melakukan generate semua kombinasi-kombinasi berdasarkan masukan matriks dan ukurannya

   - Fungsi calculate_reward(*combination*, *sequence*):

     Fungsi ini digunakan untuk menghitung total hadiah tiap kombinasi.

   - Fungsi search_optimal_path(*combinations*, *sequence*, *matrix*):

     Fungsi ini digunakan untuk mencari solusi path dengan total hadiah terbesar dan yang paling optimal.

7. Fungsi main(*buffer_size*, *matrix*, *sequences*):

   Fungsi ini memanggil breach_protocol() untuk menyelesaikan permainan dan mencetak hasilnya. Jika pengguna ingin menyimpan hasil permainan, fungsi ini juga meminta nama file untuk menyimpan hasilnya.

8. Bagian __name__ == "__main__":

   Bagian ini adalah bagian utama dari program yang dipanggil saat program dieksekusi. Program akan menanyakan kepada pengguna apakah ingin menggunakan input dari file atau menghasilkan matriks dan sekuens acak. Setelah itu, program memanggil fungsi sesuai dengan pilihan pengguna.

Berikut potongan program yang digunakan untuk mencari solusi.

```
def breach_protocol(buffer_size, matrix, sequences):
    rows = len(matrix)
    cols = len(matrix[0])
    sequence = {tuple(key): value for key, value in
sequences.items()}

    def is_subarray(arr, subarr):
```

```
        subarr_length = len(subarr)
        for i in range(len(arr) - subarr_length + 1):
            if arr[i:i+subarr_length] == subarr:
                return True
        return False

    def generate_combinations(matrix_height, matrix_width,
buffer_size):
        def backtrack(combination, buffer_size):
            valid_combinations = []
            if len(combination) >= 1 and combination[0][0] == 0:
                valid_combinations.append(combination[:])
            if len(combination) == buffer_size:
                return valid_combinations
            for row in range(matrix_height):
                for col in range(matrix_width):
                    if not any(row == r and col == c for r, c in
combination):
                        if not combination or row ==
combination[-1][0] or col == combination[-1][1]:
                            if len(combination) < 2 or
(len(combination) >= 2 and combination[0][1] == combination[1][1]
and combination[-2][0] != row and combination[-2][1] != col):

valid_combinations.extend(backtrack(combination + [(row, col)],
buffer_size))
            return valid_combinations

        return backtrack([], buffer_size)

    def calculate_reward(combination, sequence):
        reward = 0
        for key in sequence:
            if is_subarray(combination, key):
                reward += sequence[key]
        return reward

    def search_optimal_path(combinations, sequence, matrix):
        paths = {}
```

```
        max_reward = 0
        max_path = []

        for combination in combinations:
            path = tuple(matrix[row][col] for row, col in
combination)
            reward = calculate_reward(path, sequence)
            paths[tuple(matrix[row][col] for row, col in
combination)] = reward
            if reward > max_reward or (reward == max_reward and
len(combination) < len(max_path)):
                max_reward = reward
                max_path = combination

        return paths, max_reward, max_path

    start_time = time.time()
    combinations = generate_combinations(rows, cols, buffer_size)
    time_ms = round((time.time() - start_time) * 1000)

    paths, max_reward, max_path =
search_optimal_path(combinations, sequence, matrix)

    return paths, max_reward, max_path, time_ms
```

Algoritma yang dijelaskan dalam fungsi backtrack adalah bagian dari proses pencarian kombinasi elemen matriks yang sesuai dengan aturan permainan Breach Protocol dengan rincian:

1. Pengecekan awal dengan memeriksa apakah panjang kombinasi saat ini sudah mencapai panjang minimal (minimal 1) dan apakah koordinat awal dari kombinasi adalah (0, 0) atau tidak. Jika ya, maka kombinasi tersebut akan dikembalikan sebagai hasil pencarian.
2. Pengecekan ukuran kombinasi, jika panjang kombinasi mencapai ukuran buffer yang ditentukan, pencarian akan dihentikan untuk kombinasi saat ini.
3. Fungsi melakukan iterasi melalui setiap elemen dalam matriks. Dalam setiap iterasi, fungsi memeriksa apakah elemen yang saat ini dapat ditambahkan ke kombinasi saat ini berdasarkan aturan permainan. Jika elemen tersebut belum ada dalam kombinasi atau jika elemen tersebut berada di baris atau kolom yang

sama dengan elemen sebelumnya, maka elemen tersebut dapat ditambahkan ke kombinasi. Jika penambahan elemen baru memenuhi semua kriteria di atas, maka fungsi akan melakukan rekursi untuk mengecek kombinasi baru yang mungkin dengan elemen baru tersebut.

4. Setiap kali fungsi menemukan elemen yang memenuhi kriteria untuk ditambahkan ke kombinasi, fungsi akan memanggil dirinya sendiri (rekursi) dengan kombinasi yang baru ditambahkan tersebut. Proses ini akan berlanjut hingga kombinasi mencapai ukuran buffer yang ditentukan atau tidak ada lagi elemen yang dapat ditambahkan ke kombinasi.
5. Output: Setiap kali fungsi menemukan kombinasi yang sesuai dengan aturan permainan dan ukuran buffer, kombinasi tersebut akan dikembalikan sebagai bagian dari hasil pencarian.

BAB III
HASIL PENGUJIAN

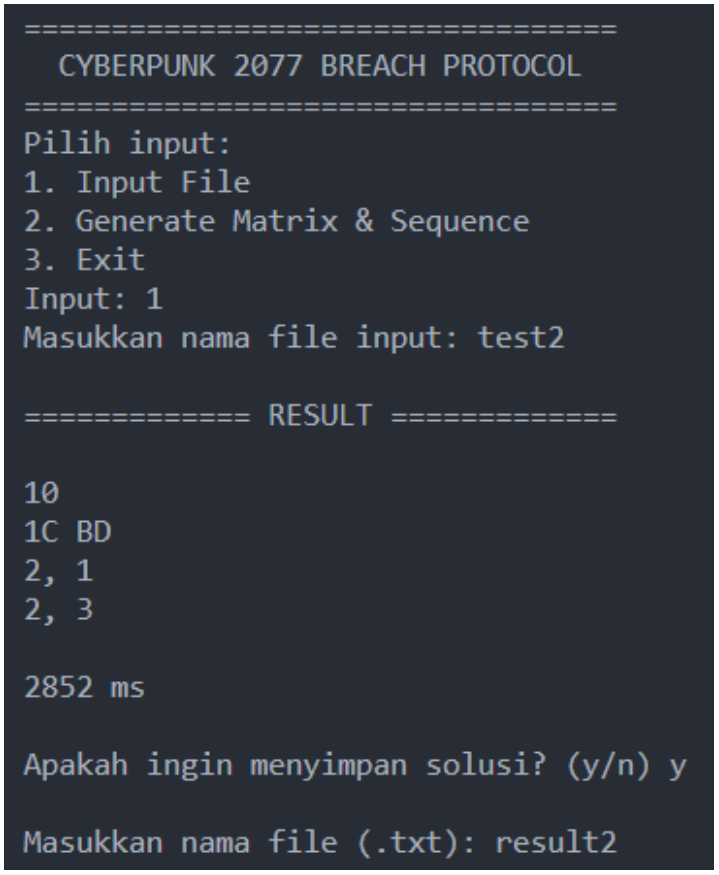
1. Test 1

File: test1.txt

```
7
6 6
55 7A 1C BD E9 E9
E9 1C 55 E9 E9 1C
55 E9 BD 1C E9 1C
7A 1C 7A BD BD 55
E9 55 1C 1C BD 55
E9 E9 BD BD 1C BD
3
BD
-13
7A BD
65
55 7A 1C
85
```

```
==================================
  CYBERPUNK 2077 BREACH PROTOCOL
==================================
Pilih input:
1. Input File
2. Generate Matrix & Sequence
3. Exit
Input: 1

Masukkan nama file input: test1

============ RESULT =============

137
55 7A 1C 7A BD
1, 1
1, 4
2, 4
2, 1
4, 1

2861 ms

Apakah ingin menyimpan solusi? (y/n) y

Masukkan nama file (.txt): result1
```

2. Test 2

File: test2.txt 7 6 6 E9 1C 1C 7A 1C 7A 7A E9 55 7A 7A BD E9 BD 1C 7A 1C 1C 55 7A 1C E9 E9 55 1C BD E9 7A 1C E9 BD 1C 55 7A 55 1C 3 1C -65 55 BD 55 -53 1C BD 75	

3. Test 3

File: test3.txt

```
8
7 7
7A E9 55 7A 55 55 E9
E9 BD BD E9 E9 55 1C
E9 7A 55 7A 55 7A 7A
55 7A 1C 1C E9 55 55
BD 55 55 55 55 BD 55
7A E9 55 E9 E9 55 BD
55 7A E9 E9 E9 1C 7A
4
1C 7A 7A BD 55 7A
79
BD
-95
BD BD E9 55 55
4
7A 1C 7A
51
```

```
==================================
  CYBERPUNK 2077 BREACH PROTOCOL
==================================
Pilih input:
1. Input File
2. Generate Matrix & Sequence
3. Exit
Input: 1

Masukkan nama file input: test3

============ RESULT =============

51
7A 1C 7A
4, 1
4, 4
2, 4

78016 ms

Apakah ingin menyimpan solusi? (y/n) y

Masukkan nama file (.txt): result3
```

4. Test 4

```
==================================
  CYBERPUNK 2077 BREACH PROTOCOL
==================================
Pilih input:
1. Input File
2. Generate Matrix & Sequence
3. Exit
Input: 2

Jumlah token [5] :
Token (dipisah spasi) [BD 1C 7A 55 E9] :
Ukuran Buffer [7] :
Ukuran Matriks (dipisah spasi) [6 6] :
Jumlah Sekuens [3] :
Ukuran Maksimal Sekuens [4] :
7
6 6
7A 7A BD 1C 55 55
7A E9 E9 55 1C E9
55 BD E9 7A 1C BD
E9 E9 E9 BD 7A 55
55 E9 1C 7A 7A E9
55 E9 1C 55 BD BD
3
7A 1C 55
59
7A 55 BD E9
6
55 1C
50

============= RESULT =============
```

```
============= RESULT =============

109
7A 7A 1C 55 1C
1, 1
1, 2
5, 2
5, 1
4, 1

2863 ms

Apakah ingin menyimpan solusi? (y/n) y

Masukkan nama file (.txt): result4

Apakah ingin menyimpan Matriks & sekuens permainan (beserta dengan bobot hadiahnya)? (y/n) y
Masukkan nama file untuk disimpan: test4
```

5. Test 5

```
==================================
  CYBERPUNK 2077 BREACH PROTOCOL
==================================
Pilih input:
1. Input File
2. Generate Matrix & Sequence
3. Exit
Input: 2

Jumlah token [5] :
Token (dipisah spasi) [BD 1C 7A 55 E9] :
Ukuran Buffer [7] : 4
Ukuran Matriks (dipisah spasi) [6 6] : 3 3
Jumlah Sekuens [3] : 2
Ukuran Maksimal Sekuens [4] : 4
4
3 3
55 E9 55
7A 55 55
E9 55 E9
2
E9 1C 55 7A
68
1C 55 BD
-66

============ RESULT ============

0


0 ms

Apakah ingin menyimpan solusi? (y/n) y
```

6. Test 6

```
==================================
  CYBERPUNK 2077 BREACH PROTOCOL
==================================
Pilih input:
1. Input File
2. Generate Matrix & Sequence
3. Exit
Input: 2

Jumlah token [5] : 3
Token (dipisah spasi) [BD 1C 7A 55 E9] : AA BB CC
Ukuran Buffer [7] : 6
Ukuran Matriks (dipisah spasi) [6 6] : 5 5
Jumlah Sekuens [3] : 5
Ukuran Maksimal Sekuens [4] : 4
6
5 5
CC BB CC BB AA
BB AA AA CC AA
AA BB CC BB AA
CC CC CC AA CC
AA BB CC BB BB
5
CC CC BB
-75
BB BB BB
1
AA BB AA
17
BB BB CC
-90
AA CC BB BB
-62

============= RESULT =============
```

```
============= RESULT =============

18
BB BB BB AA BB AA
2, 1
2, 5
5, 5
5, 1
4, 1
4, 4

127 ms

Apakah ingin menyimpan solusi? (y/n) y

Masukkan nama file (.txt): result6
```

BAB IV
LAMPIRAN

Link Repositoy:
https://github.com/Dabbir/Tucil1_13522072

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		✅

*discalimer beberapa menggunakan optimized code karena testing di laptop saya lemot