Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun Akademik 2023/2024

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma *Brute Force*

Disusun oleh :

Muhammad Yusuf Rafi
13522009
K-01

Program Studi S1 Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

BAB 1. Algoritma *Brute force* untuk menyelesaikan *Cyberpunk 2077 Breach Protocol*

Cyberpunk 2077 Breach Protocol adalah permainan *hacking* di dalam permainan video Cyberpunk 2077 yang mensimulasikan serangan peretasan pada jaringan lokal menggunakan ICE (Intrusion Countermeasures Electronics) untuk mencuri sumber daya dari terminal, mengakses informasi penting, atau meretas musuh untuk melemahkan pertahanan mereka. Komponen utama dalam mini-game ini meliputi:

1. Token: Terdiri dari kombinasi dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks: Sebuah grid yang berisi token-token yang akan dipilih oleh pemain untuk membentuk urutan kode.
3. Sekuens: Rangkaian token (minimal dua) yang harus dicocokkan oleh pemain.
4. Buffer: Batas maksimum jumlah token yang dapat disusun secara berurutan.

Dengan aturan dalam permainan ini adalah sebagai berikut:

1. Pemain bergerak secara bergantian horizontal dan vertikal, memilih token dari matriks hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token dari baris paling atas pada matriks.
3. Sekuens kemudian dicocokkan dengan token-token yang ada di dalam buffer.
4. Satu token di dalam buffer dapat digunakan untuk lebih dari satu sekuens.
5. Setiap sekuens memiliki nilai hadiah atau reward yang berbeda-beda.
6. Panjang minimal dari sebuah sekuens adalah dua token.

Adapun berikut langkah-langkah penyelesaian permainan tersebut agar mendapatkan solusi teroptimal dengan algoritma *Brute Force*, yaitu :

1. Tinjau setiap kemungkinan rute yang dapat ditempuh dengan tetap mematuhi aturan permainan.
2. Lakukan iterasi per kolom, mulai dengan mengambil token dan koordinat elemen teratas pada kolom terlebih dahulu, lalu mulai bergerak vertikal dan mencocokkan dengan tiap token dan koordinat di kolom tersebut secara bergantian, setelah melakukan semua kemungkinan pergerakan vertikal, kita simpan terlebih dahulu semua rute dengan panjang masing-masing 2.
3. Lakukan pergerakan horizontal, mulai dengan melanjutkan dari rute yang telah kita simpan sebelumnya, lalu kita cocokkan dengan token dan koordinat pada baris tersebut dengan catatan kita tidak boleh mencocokkan kembali elemen yang koordinatnya sudah kita tempuh sebelumnya, setelah melakukan semua kemungkinan pergerakan horizontal kita simpan terlebih dahulu semua rute dengan panjang masing-masing 3, lakukan langkah 2 dan 3 secara berulang sehingga panjang rute sudah sesuai batas *buffer.*
4. Setelah mendapatkan semua kemungkinan rute dengan panjang maksimal sesuai batas *buffer*, kita lakukan perbandingan semua rute tersebut dengan sekuens melalui *looping*.

5. Lakukan iterasi per rute, lalu lakukan kembali iterasi per sekuen dan jika terdapat sekuen dalam rute tersebut, maka rute tersebut mendapatkan *reward* sesuai sekuensnya jadi satu rute diiterasi sebanyak banyak sekuens untuk perhitungan total *reward*.
6. Setelah semua rute telah diperiksa, maka rute uang memiliki reward tertinggi akan dipilih dan menjadi solusi permainan

BAB 2. Source Program dalam bahasa Python

```python
def get_routes(self, matrix, buffer_size):
        routes = []
        route = []
        coords = []
        coord = []
        num_rows = len(matrix)
        num_columns = len(matrix[0])
        curr_seq = -1

        # Loop melalui setiap kolom untuk memulai rute
        for col in range(num_columns):
            route.clear()
            coord.clear()
            # Memilih satu token pada posisi baris paling atas
            route.append(matrix[0][col])
            coord.append([0, col])
            # Inisialisasi rute baru
            first = True
            vertical = True  # True untuk gerakan horizontal, False untuk gerakan vertikal
            # Loop untuk membangun rute dengan pola horizontal, vertikal, horizontal, vertikal
            while len(route) < buffer_size + 1:
```

Gambar 2.1. Fungsi get_routes() bagian 1

```python
        while len(route) < buffer_size + 1:
            if not first:
                route.clear()
                coord.clear()
            if vertical:
                #elemen pertama
                if first:
                    for curr_row in range(num_columns):
                        if curr_row != 0:
                            route.append(matrix[curr_row][col])
                            coord.append([curr_row,col])
                            # Jika sudah mencapai panjang buffer atau panjang minimal rute
                            if 2 <= len(route) <= buffer_size:
                                routes.append(route[:])
                                coords.append(coord[:])
                                route.pop()
                                coord.pop()
                            else:
                                curr_seq = len(routes) - 1
                                break
                    first = False
                    vertical = False
                else:
```

Gambar 2.2. Fungsi get_routes() bagian 2

```
            else:
                route = routes[curr_seq].copy()
                coord = coords[curr_seq].copy()
                for curr_row in range(num_rows):
                    if ([curr_row, coord[-1][1]] not in coord):
                        route.append(matrix[curr_row][coord[-1][1]])
                        coord.append([curr_row,coord[-1][1]])
                        if 2 <= len(route) <= buffer_size:
                            routes.append(route[:])
                            coords.append(coord[:])
                            route.pop()
                            coord.pop()
                        else:
                            curr_seq = len(routes) - 1
                            break
                if (len(routes[curr_seq]) <= buffer_size):
                    if (len(routes[curr_seq]) < len(routes[curr_seq+1])):
                        vertical = False

        else:
```

Gambar 2.3. Fungsi get_routes() bagian 3

```
        else:
            route = routes[curr_seq].copy()
            coord = coords[curr_seq].copy()
            for curr_col in range(num_columns):
                if ([coord[-1][0], curr_col] not in coord):
                    route.append(matrix[coord[-1][0]][curr_col])
                    coord.append([coord[-1][0], curr_col])
                    # Jika sudah mencapai panjang buffer atau panjang minimal rute
                    if 2 <= len(route) <= buffer_size:
                            routes.append(route[:])
                            coords.append(coord[:])
                            route.pop()
                            coord.pop()
                    else:
                        curr_seq = len(routes) - 1
                        break
            if (len(routes[curr_seq]) <= buffer_size):
                if (len(routes[curr_seq]) < len(routes[curr_seq+1])):
                    vertical = True
        curr_seq += 1
    return routes, coords
```

Gambar 2.4. Fungsi get_routes() bagian 4

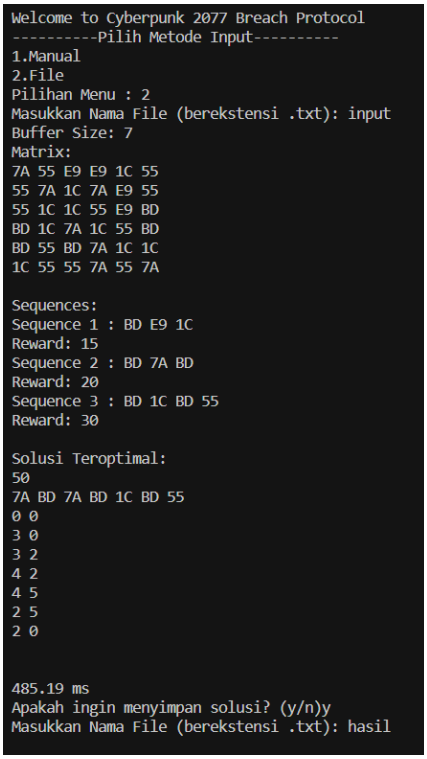
```python
def reward(self, path, sequences, rewards):
    total_reward = 0
    for idx, sequence in enumerate(sequences):
        sequence_length = len(sequence)
        for i in range(len(path) - sequence_length + 1):
            if path[i:i+sequence_length] == sequence:
                total_reward += rewards[idx]
                break
    return total_reward


def best_path(self, possible_paths, sequences, rewards):
    max_reward = 0
    best_path = []
    best_idx = None
    for idx, path in enumerate(possible_paths):
        reward = self.reward(path, sequences, rewards)
        if reward > max_reward:
            max_reward = reward
            best_path = path
            best_idx = idx
    return best_path, max_reward, best_idx
```

Gambar 2.5. Fungsi best_path() dan reward()

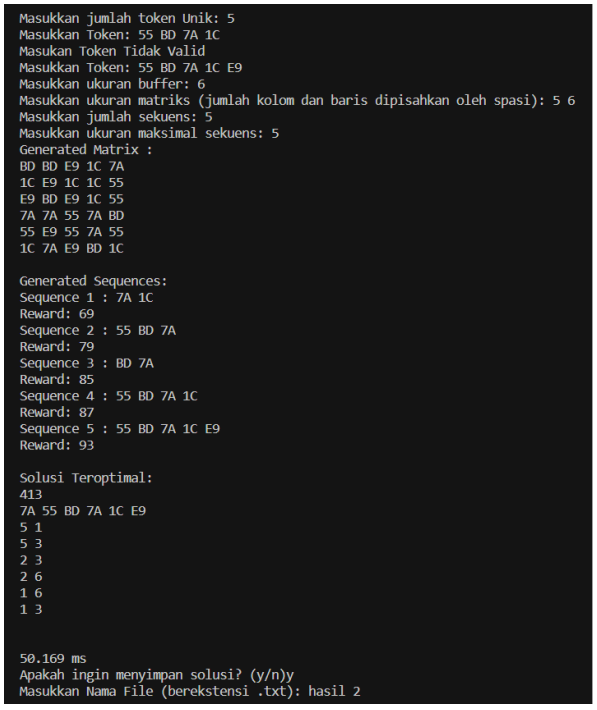
```python
def get_solution(self, matrix, buffer_size, sequences, rewards):
    start = time.time()
    routes, coords = self.get_routes(matrix, buffer_size)
    best_path, max_reward, best_index = self.best_path(routes, sequences, rewards)
    end = time.time()
    execution = round((end - start) * 1000, 3)
    print("\nSolusi Teroptimal:")
    print(max_reward)
    print(" ".join(best_path))
    for coord in coords[best_index]:
        final_coord = [coord[1] + 1, coord[0] + 1]
        print(" ".join(map(str, final_coord)))
    print("\n\n" + str(execution) + " ms")

    return best_path, max_reward, best_index, coords, execution
```

Gambar 2.6. Fungsi get_solution()

BAB 3. Pengujian (Input/Output)



Gambar 3.1 File Test Case 1 Matrix 6 x 6



Gambar 3.2 User Test Case 2 Matrix 5 x 6

```
Masukkan Nama File (berekstensi .txt): input2
Buffer Size: 7
Matrix:
7A 1C 1C E9 BD 1C 55
55 BD 1C KL 1C HI KL
E9 BD 7A KL 7A BD 7A
7A KL KL 1C 7A HI 7A
HI E9 7A E9 55 BD 55
BD HI 55 BD HI E9 7A
E9 BD E9 BD 55 1C KL

Sequences:
Sequence 1 : 1C HI
Reward: 5
Sequence 2 : E9 1C HI
Reward: 34
Sequence 3 : 7A E9 1C HI
Reward: 54
Sequence 4 : 55 BD 7A E9 1C
Reward: 87
Sequence 5 : BD 7A E9 1C HI
Reward: 87

Solusi Teroptimal:
267
1C 55 BD 7A E9 1C HI
3 1
3 6
1 6
1 1
4 1
4 4
6 4


2184.52 ms
Apakah ingin menyimpan solusi? (y/n)y
Masukkan Nama File (berekstensi .txt): hasil2
```

Gambar 3.3 File Test Case 3 Matrix 7 x 7

```
Masukkan jumlah token Unik: 7
Masukkan Token: 55 BD 1C 7A E9
Masukan Token Tidak Valid
Masukkan Token: 55 BD 1C E9 7A HI KL
Masukkan ukuran buffer: 8
Masukkan ukuran matriks (jumlah kolom dan baris dipisahkan oleh spasi): 9 9
Masukkan jumlah sekuens: 6
Masukkan ukuran maksimal sekuens: 6
Generated Matrix :
7A E9 1C E9 7A KL 1C 7A HI
7A 7A 55 HI BD 1C BD E9 KL
E9 7A 55 HI HI 1C E9 BD 55
BD 55 E9 7A E9 E9 BD KL 1C
1C BD KL BD 1C HI HI BD KL
E9 E9 55 HI 55 55 7A 55 1C
7A KL HI KL BD HI 55 KL BD
BD E9 BD 55 7A E9 BD KL 55
KL 1C E9 55 1C E9 7A 1C 7A

Generated Sequences:
Sequence 1 : 55 BD 1C E9
Reward: 99
Sequence 2 : 55 BD 1C E9 7A
Reward: 100
Sequence 3 : BD 1C E9 7A HI
Reward: 100
Sequence 4 : BD 1C E9 7A
Reward: 100
Sequence 5 : 1C E9 7A HI
Reward: 100
Sequence 6 : 55 BD 1C
Reward: 100

Solusi Teroptimal:
599
7A BD 55 BD 1C E9 7A HI
1 1
1 4
2 4
2 5
1 5
1 6
7 6
```

Gambar 3.4 Input Test Case 4 Matrix 8 x 8

```
Masukkan Nama File (berekstensi .txt): input3
Buffer Size: 7
Matrix:
YZ KL 7A
7A E9 7A
E9 HI WX
GI WX 55
KL E9 55

Sequences:
Sequence 1 : 1C E9 HI
Reward: 24
Sequence 2 : 7A 1C E9
Reward: 80
Sequence 3 : 7A 1C
Reward: 88
Sequence 4 : HI KL GI WX
Reward: 100

Solusi Teroptimal:
0
Breach ini tidak memeiliki solusi


33.497 ms
Apakah ingin menyimpan solusi? (y/n)y
Masukkan Nama File (berekstensi .txt): hasil4
```

Gambar 3.5 File Test Case 5 Matirks 5 x 3

```
Welcome to Cyberpunk 2077 Breach Protocol
----------Pilih Metode Input----------
1.Manual
2.File
Pilihan Menu : 2
Masukkan Nama File (berekstensi .txt): input4
Buffer Size Tidak boleh lebih besar dari ukuran Matrix

----------Program----------
1.Kembali ke Menu
2.Keluar
Pilihan :
```

Gambar 3.6 File Test Case 6 Matirks 1 x 5

```
----------Pilih Metode Input----------
1.Manual
2.File
Pilihan Menu : 2
Masukkan Nama File (berekstensi .txt): input5
Buffer Size: 3
Matrix:
E9 55
7A 1C

Sequences:
Sequence 1 : 1C E9 HI
Reward: 18
Sequence 2 : 7A 1C E9 HI
Reward: 70
Sequence 3 : 1C E9
Reward: 89

Solusi Teroptimal:
0
Breach ini tidak memeiliki solusi


0.0 ms
Apakah ingin menyimpan solusi? (y/n)y
Masukkan Nama File (berekstensi .txt): hasil6
```

Gambar 3.7 File Test Case 7 Matirks 2 x 2

BAB 4. Lampiran

Lampiran 1

Checklist penilaian :

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 memiliki GUI		✓

Lampiran 2

Link *Repository* github : https://github.com/cupski/Tucil1_13522009