

LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

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



Disusun oleh:

Kayla Namira Mariadi K-02 13522050

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023-2024

DAFTAR ISI

DAFTAR ISI	2
BAB 1	3
BAB 2	4
BAB 3	6
BAB 4	16
BAB 5	21
DAFTAR REFERENSI	23

BAB 1

DESKRIPSI MASALAH

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.

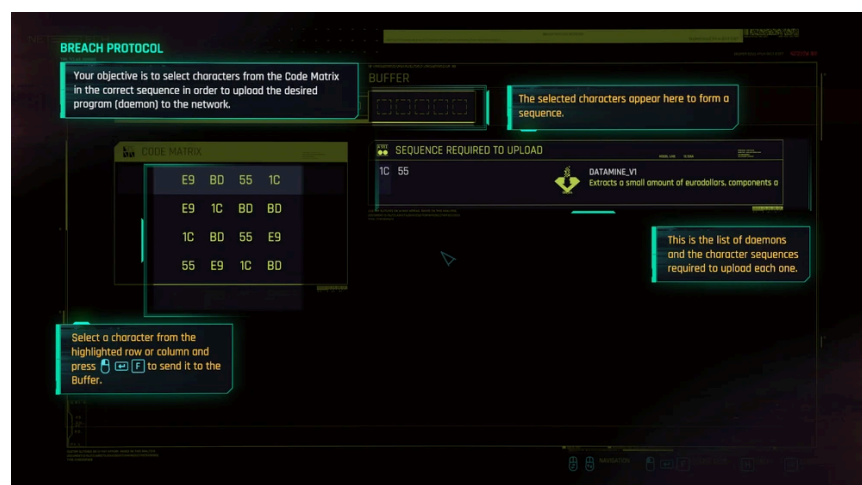
Aturan permainan Breach Protocol antara lain:

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.

(Paragraf di atas dikutip dari link

berikut: https://docs.google.com/document/d/1cezu5NJNdBOp4UZWnOob3Q_t0pw5VLBv/edit)

Laporan ini membahas program yang digunakan untuk menemukan solusi dari permainan Breach Protocol yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma *Brute Force*.



Gambar 1 Permainan Breach Protocol

BAB 2

ALGORITMA BRUTE FORCE

2.1 Algoritma Brute Force

Algoritma yang digunakan untuk menemukan solusi dari permainan Breach Protocol yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer adalah *Brute Force*. Program akan memeriksa semua jenis kemungkinan solusi dengan memilih token selang-seling secara horizontal dan vertikal selama buffer belum penuh atau semua sekuens belum berhasil dicocokkan, dimulai dari token pada koordinat (1,1). Program menggunakan struktur data pohon *n*-ary (*n*-ary Tree) untuk merepresentasikan segala kemungkinan susunan token pada buffer. Akar dari pohon merupakan token dan daun/anaknya berupa token-token yang mungkin ditempuh dari akarnya. Kedalaman dari pohon sebanding dengan ukuran buffer.

Langkah-langkah algoritma yang digunakan dalam menemukan solusi adalah sebagai berikut:

- 1. Memulai pencarian dengan token dari baris pertama matriks**

Sebuah token pada baris pertama matriks dipilih untuk menjadi akar utama dari pohon. Inisiasi pohon diawali dari token pada koordinat (1,1). Pada kondisi awal, pohon belum memiliki anak.

- 2. Pencarian semua kemungkinan solusi dari akar hingga buffer penuh atau setiap sekuens berhasil dicocokkan**

Dilakukan ekspansi dari akar pohon (yang awalnya tidak memiliki anak). Ekspansi akar untuk mencari solusi hanya dilakukan jika rute yang ditempuh belum memenuhi semua sekuens atau buffer belum penuh. Setiap ekspansi, juga dicatat buffer yang bersesuaian dengan rute dari akar hingga anak yang bersangkutan, beserta *reward* sementara yang diperoleh.

Pencarian solusi bergantung pada kondisi arah pencarian, jika sebelumnya bergerak secara horizontal, selanjutnya bergerak secara vertikal. Jika akan bergerak secara vertikal, dipilih token-token pada baris yang sama dengan akar dan belum pernah dilalui. Token-token tersebut ditambahkan sebagai anak. Begitu pula jika akan bergerak secara horizontal, dipilih token-token yang belum pernah dilalui dan berada pada kolom yang sama dengan akar.

Kemudian, setiap anak juga melakukan ekspansi dengan mekanisme yang sama hingga buffer penuh atau berhasil memenuhi semua sekuens.

- 3. Iterasi pencarian rute untuk token selanjutnya dari baris pertama**

Ulangi langkah 1 dan langkah 2 untuk setiap token pada baris pertama matriks. Program membuat pohon sebanyak ukuran kolom matriks, kumpulan pohon tersebut ditampung oleh suatu array *possibilities*.

4. Cari solusi optimal dari tiap pohon

Setiap pohon pada array *possibilities* yang disebutkan pada langkah 3 menentukan solusi optimal dari anak-anaknya berdasarkan *reward* maksimal yang didapat dan panjang buffer minimal secara rekursif:

Basis: Jika level/kedalaman dari pohon sama dengan ukuran buffer atau sudah memenuhi semua sekuens, fungsi mengembalikan *reward* yang didapat pohon, buffernya (array berisi susunan token dari akar utama hingga pohon yang bersangkutan), dan panjang buffer.

Rekurs: Untuk setiap anak dari pohon, tentukan *reward*, buffer, dan panjang buffer dengan fungsi rekursif. Hasil optimal dari tiap anak kemudian dibandingkan. Jika ada anak dengan *reward* sama, dipilih hasil dengan panjang buffer minimal. Jika panjang buffernya juga sama, dicatat hasil yang pertama ditemukan.

Sebuah hasil optimal dari anak kemudian dibandingkan dengan akar/induknya. Jika *reward*-nya sama, akar/induk dicatat sebagai solusi yang optimal (karena ukuran buffernya lebih pendek)

5. Bandingkan solusi dari tiap pohon dan tentukan solusi optimal

Terakhir, program akan membandingkan solusi optimal untuk setiap pohon pada array *possibilities* yang dilakukan pada langkah 4. Solusi dengan nilai *reward* maksimal dan panjang buffer terpendek akan disimpan sebagai solusi optimal. Jika tidak ada solusi yang ditemukan, *reward* akan bernilai 0. *Reward* juga mungkin bernilai negatif, pada kasus ini nilai yang paling mendekati 0 dianggap sebagai solusi optimal.

BAB 3

IMPLEMENTASI PROGRAM

Program diimplementasikan dengan bahasa python pada file *main.py* di folder *src*. Adapun *library* yang digunakan adalah *os*, *datetime*, dan *random*.

File *main.py* terbagi menjadi 2 bagian, yaitu *Classes* dan *Functions*.

3.1 Classes

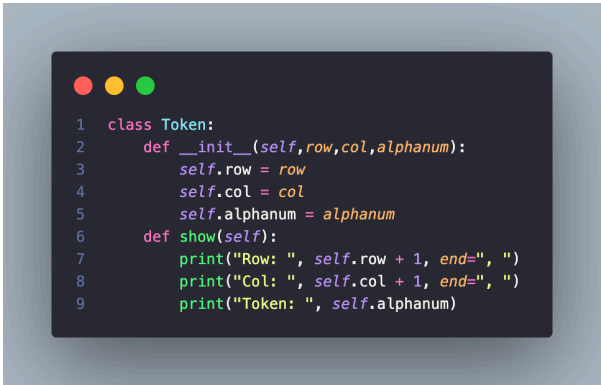
Classes merupakan bagian deklarasi *class* *Token*, *Sequence*, dan *Tree* untuk implementasi program.

1. Token

Token memiliki atribut:

- row* : integer merepresentasikan index baris token pada matriks
- col* : integer merepresentasikan index kolom token pada matriks
- alphanum* : string alfanumerik 2 digit, *case sensitive*

Token memiliki *method* *show()* untuk menampilkan atributnya.



```
1 class Token:
2     def __init__(self, row, col, alphanum):
3         self.row = row
4         self.col = col
5         self.alphanum = alphanum
6     def show(self):
7         print("Row: ", self.row + 1, end=" ")
8         print("Col: ", self.col + 1, end=" ")
9         print("Token: ", self.alphanum)
```

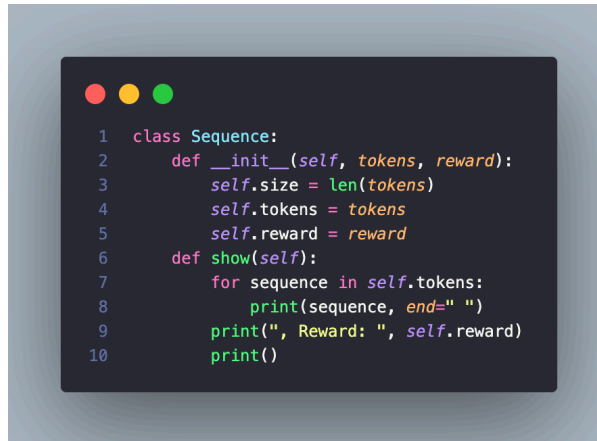
Gambar 2 Deklarasi *Class* *Token*

2. Sequence

Sequence memiliki atribut:

- size* : integer minimal 2, merepresentasikan panjang sekuens
- tokens* : string merepresentasikan rangkaian token
- reward* : bobot/hadiah *Sequence* berupa integer positif, negatif atau 0

Sequence memiliki *method* *show()* untuk menampilkan rangkaian token dan rewardnya.



```

1 class Sequence:
2     def __init__(self, tokens, reward):
3         self.size = len(tokens)
4         self.tokens = tokens
5         self.reward = reward
6     def show(self):
7         for sequence in self.tokens:
8             print(sequence, end=" ")
9         print("\n, Reward: ", self.reward)
10        print()

```

Gambar 3 Deklarasi *Class Sequence*

3. Tree

Tree memiliki atribut:

- root : Token, merepresentasikan akar dari pohon
- child : array of Token, merepresentasikan Token pada baris/kolom yang sama dengan akar dan mungkin ditempuh dari akar sesuai aturan permainan
- level : Integer minimal 1, merepresentasikan level pohon
- buffer : array of Token, merepresentasikan susunan token dari akar hingga Tree yang bersangkutan
- reward : integer, berupa akumulasi sementara *reward* sesuai isi buffer



```

1 class Tree:
2     def __init__(self, root, level, buffer, sequences):
3         self.root = root
4         self.child = []
5         self.level = level
6         temp = []
7         for item in buffer:
8             temp.append(item)
9         temp.append(root)
10        self.buffer = temp #buffer so far
11
12        reward = 0
13        for sequence in sequences:
14            reward += self.calculateReward(sequence)
15        self.rewardgained = reward
16

```

Gambar 4 Deklarasi *Class Tree* (1)

Tree memiliki method:

- findRoute(self, isVertical, matrix, level, buffersize, sequences, maxreward)

Mengimplementasikan pencarian solusi dengan algoritma Brute force yang disebutkan pada langkah 2 bab 2. isVertical adalah status arah pencarian

(vertikal/horizontal), maxreward adalah total reward dari sequences, level adalah status level/kedalaman pencarian (langkah beberapa pada buffer)

b. calculateReward(self, sequence)

Mengembalikan *reward* yang didapatkan Tree dengan melakukan *pattern matching* antara sekuens pada self.buffer dengan sequence. Tiap Token pada sekuens dicocokkan atribut alphanumnya dengan tokens pada sequence.

c. isTokenSelected(self, rowidx, colidx)

Menentukan apakah token dengan posisi indeks rowidx, colidx sudah pernah dilalui/terpilih pada self.buffer. Fungsi ini dipanggil pada implementasi findRoute

```
def findRoute(self, isVertical, matrix, level, buffersize, sequences, maxreward):
    if (level == self.level and self.rewardgained != maxreward):
        if isVertical : #search vertically
            colidx = self.root.col
            for rowidx in range(len(matrix)):
                if not self.isTokenSelected(rowidx, colidx):
                    if (rowidx != self.root.row and self.level <= buffersize-1 and self.rewardgained != maxreward):
                        self.child.append(Tree(matrix[rowidx][colidx], self.level+1, self.buffer, sequences))
        else:
            rowidx = self.root.row
            for colidx in range(len(matrix[0])):
                if not self.isTokenSelected(rowidx, colidx):
                    if (colidx != self.root.col and self.level <= buffersize - 1 and self.rewardgained != maxreward):
                        self.child.append(Tree(matrix[rowidx][colidx], self.level+1, self.buffer, sequences))
    for token in self.child:
        token.findRoute(not isVertical, matrix, level+1, buffersize, sequences, maxreward)

def calculateReward(self, sequence):
    sequences = sequence.tokens
    reward = 0
    bufferlength = len(self.buffer)
    bufferidx = 0
    #pattern matching
    while (bufferidx <= (bufferlength - sequence.size)):
        match = True
        sequenceidx = 0
        while (match and sequenceidx < sequence.size):
            if (self.buffer[bufferidx].alphanum != sequences[sequenceidx]):
                match = False
            else:
                sequenceidx += 1
                bufferidx += 1
        if (match):
            reward += sequence.reward
            break
        else:
            bufferidx += 1
    return reward
```

Gambar 5 Deklarasi Class Tree (2)

```
1 def isTokenSelected(self, rowidx, colidx):
2     for i in range(len(self.buffer)):
3         if (self.buffer[i]):
4             if (rowidx == self.buffer[i].row and colidx == self.buffer[i].col):
5                 return True
6         else:
7             return False
8     return False
```

Gambar 6 Deklarasi Class Tree (3)

3.2 Functions

Function terbagi menjadi 2 kategori, yaitu *Algorithm Functions* yang berkaitan algoritma pencarian solusi, dan *I/O Functions* berkaitan interaksi i/o pengguna.

a. *Algorithm Functions*

1. `createTree(bufferSize, matrixCol, matrix, sequences)`

Program mula-mula menginisiasi array *possibilities* kosong dan *maxreward* berupa total reward dari *sequences*. Lalu, objek *Tree* dengan *Token* pada baris pertama sebagai *root*, *level* = 1, dan *isVertical* = *True* diinstansiasi dan ditambahkan ke array. *Tree* kemudian mengekskpan anaknya dengan memanggil method *findRoute*, sesuai langkah 1-2 algoritma. Proses instansiasi objek dan penambahannya ke array diulang untuk setiap *Token* pada baris pertama sesuai langkah 3.

2. `getTreeReward(tree, bufferSize, maxprize)`

Fungsi rekursif yang disebutkan pada langkah 4, berguna untuk menentukan reward dan solusi optimal yang mungkin didapat dari objek *tree*.

3. `getMaxReward(possibilities, bufferSize, maxprize)`

Menentukan reward dan solusi optimal yang mungkin didapat dari setiap *tree* pada array *possibilities* dengan mengimplementasikan langkah 4 (memanggil *getTreeReward*), lalu mengembalikan solusi optimal final dan *reward* maksimal yang didapatkan.

```
def findRoute(self, isVertical, matrix, level, bufferSize, sequences, maxreward):
    if (level == self.level and self.rewardgained != maxreward):
        if isVertical : #search vertically
            colidx = self.root.col
            for rowidx in range(len(matrix)):
                if not self.isTokenSelected(rowidx, colidx):
                    if (rowidx != self.root.row and self.level <= bufferSize-1 and self.rewardgained != maxreward):
                        self.child.append(Tree(matrix[rowidx][colidx], self.level+1, self.buffer, sequences))
        else:
            rowidx = self.root.row
            for colidx in range(len(matrix[0])):
                if not self.isTokenSelected(rowidx, colidx):
                    if (colidx != self.root.col and self.level <= bufferSize - 1 and self.rewardgained != maxreward):
                        self.child.append(Tree(matrix[rowidx][colidx], self.level+1, self.buffer, sequences))
    for token in self.child:
        token.findRoute(not isVertical, matrix, level+1, bufferSize, sequences, maxreward)

def calculateReward(self, sequence):
    sequences = sequence.tokens
    reward = 0
    bufferlength = len(self.buffer)
    bufferidx = 0
    #pattern matching
    while (bufferidx <= (bufferlength - sequence.size)):
        match = True
        sequenceidx = 0
        while (match and sequenceidx < sequence.size):
            if (self.buffer[bufferidx].alphanum != sequences[sequenceidx]):
                match = False
            else:
                sequenceidx += 1
                bufferidx += 1
        if (match):
            reward += sequence.reward
            break
        else:
            bufferidx += 1
    return reward
```

Gambar 7 Cuplikan Functions (1)

```
def getMaxReward(possibilities, buffersize, maxprize):
    minbufferlength = 1
    maxreward = -500
    solution = []
    for tree in possibilities:
        reward, buffer, bufferlength = getTreeReward(tree, buffersize, maxprize)
        if ((reward > maxreward and reward != 0) or (reward == maxreward and reward != 0 and bufferlength < minbufferlength)):
            maxreward = reward
            solution = buffer
            minbufferlength = bufferlength
    if (len(solution) == 0):
        maxreward = 0
    return maxreward, solution
```

Gambar 8 Cuplikan Functions (2)

b. I/O Functions

1. `is_num(x)` : validasi apakah x integer
2. `validateNumericInput(x, message, minvalue)`: validasi x, mengulang proses input dengan pesan *message* jika nilai x lebih kecil dari minvalue atau x bukan berupa integer
3. `readNumericInput(message, minvalue)` : membaca dan validasi input integer
4. `getMatrixSize()`: Membaca masukan ukuran matrix (format berderet), serta memvalidasi kasus input bukan integer/bukan integer positif/input tidak sesuai format
5. `getToken(numoftoken)`: Membaca dan memvalidasi masukan token (secara berderet). Proses meminta masukan diulang untuk kasus token yang dimasukkan tidak sesuai dengan numoftoken(jumlah token unik)/token bukan berupa 2 karakter alfanumerik/token tidak unik.
6. `getPath()` : Meminta masukan nama file pada folder 'test', dan mengembalikan *path* menuju file
7. `randomizeInput()` : Menghasilkan matriks dan sekuens secara random. Jumlah token unik (numoftoken, minimal 0), ukuran buffer (buffersize, minimal 1), jumlah sekuens (numofsequences, minimal 1), ukuran maksimal sekuens (maxsequencesize, minimal 2) diinput dan divalidasi dengan `readNumericInput`. Token yang unik (tokens) diinput dan divalidasi dengan `getToken(numoftoken)`. Ukuran matrix (matrixcol dan matrixrow) diinput dan divalidasi dengan `getMatrixSize()`. Kemudian, diinstansiasi objek Token dengan atribut row = matrixrow, col = matrixcol, dan alphanum dipilih secara random dari tokens. Objek tersebut ditambahkan menjadi elemen dari matrix. Kemudian, diinstansiasi Sequence sebanyak numofsequences secara random dengan panjang pada range 2 - maxsequencesize, *reward* pada range -500 hingga 500, dan token dipilih dari tokens. Keunikan token juga divalidasi. Fungsi mengembalikan buffersize, matrixcol, matrixrow, matrix, numofsequences, dan sequences.
8. `readFile()` : Membaca data yang dibutuhkan untuk algoritma dari file. Program membaca filepath dengan `getPath` dan memvalidasi kasus file tidak ada. Jika input file invalid, pembacaan file akan diulang. Kemudian, file dibaca, data yang dibutuhkan disimpan pada variabel buffersize, matrixcol, matrixrow,

matrix, numofsequences, sequences. Instansiasi Token sebagai elemen matrix dan instansiasi Sequence dilakukan.

9. saveToFile(maxreward, solution, runtime): Menyimpan solusi optimal algoritma berupa maxreward, solution, dan runtime ke dalam file berekstensi .txt yang akan disimpan pada folder 'test' dengan format sesuai spek
10. printMatrix(matrix): Mencetak karakter alphanum untuk setiap Token pada matrix
11. printBuffer(buffer) : Mencetak karakter alphanum untuk setiap Token pada buffer
12. printSequence(sequences): Mencetak token untuk setiap tokens pada Sequence di sequences beserta rewardnya
13. printCoordinates(buffer) : Mencetak koordinat (col dan row) untuk setiap Token pada buffer
14. output(maxreward, solution, runtime): Mencetak maxreward (0 jika tak ada solusi), isi buffer solution (jika tidak ada solusi, tidak dicetak), koordinat token pada solution secara terurut (jika tidak ada solusi, tidak dicetak), runtime program(dalam ms) ke layar

Berikut implementasi functions:

```
def is_num(x):
    try:
        int(x)
    except ValueError:
        return False
    else:
        return True

def validateNumericInput(x, message, minvalue):
    while (not is_num(x)):
        print("\nMasukan harus berupa angka!")
        x = input(message)
    while (int(x) <= minvalue):
        print("\nMasukan harus berupa bilangan lebih besar dari", minvalue)
        x = readNumericInput(message, minvalue)
    return int(x)

def readNumericInput(message, minvalue):
    x = input(message)
    return validateNumericInput(x, message, minvalue)
```

Gambar 9 Cuplikan Functions (3)

```
def getMatrixSize():
    matrixsizeinput = input("Masukkan ukuran matriks: ")
    matrixsize = matrixsizeinput.split()
    while (len(matrixsize) != 2):
        print("Masukan harus berupa 'ukuran_kolom ukuran_baris'!")
        matrixsizeinput = input("Masukkan ukuran matriks: ")
        matrixsize = matrixsizeinput.split()
    matrixcol = matrixsize[0]
    matrixrow = matrixsize[1]
    while (not is_num(matrixcol) or not is_num(matrixrow)):
        print("\nMasukan harus berupa angka!")
        matrixcol, matrixrow = getMatrixSize()
    while (int(matrixcol) <= 0 or int(matrixrow) <= 0):
        print("\nMasukan harus berupa bilangan positif")
        matrixcol, matrixrow = getMatrixSize()
    return int(matrixcol), int(matrixrow)
```

Gambar 10 Cuplikan Functions (4)

```

def getToken(numoftoken):
    tokeninput = input("Masukkan token: ")
    tokens = tokeninput.split()
    while (len(tokens) != numoftoken):
        print("Masukan token harus sebanyak ", numoftoken)
        tokeninput = input("Masukkan token: ")
        tokens = tokeninput.split()
    for i in range(numoftoken):
        if (len(tokens[i]) != 2):
            print("\nToken harus terdiri dari 2 karakter!")
            tokens = getToken(numoftoken)
            break
        if (not tokens[i].isalnum()):
            print("\nToken harus berupa karakter alfanumerik!")
            tokens = getToken(numoftoken)
            break
        for j in range (i+1, numoftoken):
            if (tokens[i] == tokens[j]):
                print("\nToken harus unik!")
                tokens = getToken(numoftoken)
                break
    return tokens

def getPath():
    filename = input("\nMasukkan nama file('nama_file.txt') ")
    cwd = os.getcwd()
    directory = os.path.join(cwd, os.pardir, 'test')
    filepath = os.path.join(directory, filename)
    return filepath

```

Gambar 11 Cuplikan Functions (5)

```

def randomizeInput():
    numoftoken = readNumericInput("Masukkan jumlah token unik: ", 0)
    tokens = getToken(numoftoken)
    buffersize = readNumericInput("Masukkan ukuran buffer: ", 1)

    matrix = []
    matrixcol, matrixrow = getMatrixSize()
    for i in range(matrixrow):
        temp = []
        for j in range(matrixcol):
            temp.append(Token(i,j, random.choice(tokens)))
        matrix.append(temp)

    sequences = []
    numofsequences = readNumericInput("Masukkan jumlah sekuens: ", 1)
    maxsequencesize = readNumericInput("Masukkan ukuran maksimal sekuens: ", 2)
    for i in range(numofsequences):
        length = random.randint(2,maxsequencesize)
        token = [random.choice(tokens) for i in range(length)]
        isUnique = True
        j = 0
        while (isUnique and (j < len(sequences))):
            if (sequences[j].tokens == token):
                token = [random.choice(tokens) for i in range(length)]
            else:
                j += 1
        reward = random.randint(-500, 500)
        sequences.append(Sequence(token,reward))

    return buffersize, matrixcol, matrixrow, matrix, numofsequences, sequences

```

Gambar 12 Cuplikan Functions (6)

```

def readFile():
    filepath = getPath()
    while (not os.path.isfile(filepath)):
        print("\nFile tidak ada pada folder 'test'")
        filepath = getPath()
    matrix = []
    sequences = []
    with open(filepath, 'r') as file:
        lines = file.readlines()
        buffersize = lines[0][0]
        if (not is_num(buffersize) or int(buffersize) <= 0):
            file.close()
            print("Ukuran buffer harus berupa integer lebih besar dari 1!")
            buffersize, matrixcol, matrixrow, matrix, numofsequences, sequences = readFile()
        matrixcol = lines[1][0]
        matrixrow = lines[1][2]
        if (not is_num(matrixcol) or not is_num(matrixrow) or int(matrixcol) <= 0 or int(matrixrow) <= 0):
            file.close()
            print("Ukuran matrix tidak valid!")
            buffersize, matrixcol, matrixrow, matrix, numofsequences, sequences = readFile()
        matrixcol = int(matrixcol)
        matrixrow = int(matrixrow)
        for i in range(matrixrow):
            tokens = lines[i+2].split()
            temp = []
            j = 0
            for token in tokens:
                temp.append(Token(i,j,token))
                j += 1
            matrix.append(temp)

        numofsequences = lines[matrixrow+2][0]
        if (not is_num(numofsequences) or (int(numofsequences) <= 1)):
            file.close()
            print("Jumlah sekuens harus berupa integer lebih besar dari 1!")
            buffersize, matrixcol, matrixrow, matrix, numofsequences, sequences = readFile()
        for i in range(matrixrow+3, len(lines), 2):
            tokens = lines[i].split()
            for token in tokens:
                if (len(token) != 2 or not token.isalnum()):
                    file.close()
                    print("Token harus berupa 2 karakter alfanumerik!!")
                    buffersize, matrixcol, matrixrow, matrix, numofsequences, sequences = readFile()

    isUnique = True
    j = 0
    if (isUnique and (j < len(sequences))):
        if (sequences[j].tokens == tokens):
            file.close()
            print("Token harus unik!")
            buffersize, matrixcol, matrixrow, matrix, numofsequences, sequences = readFile()
        else:
            j += 1
    reward = int(lines[i+1].rstrip())
    sequences.append(Sequence(tokens, reward))

    return int(buffersize), matrixcol, matrixrow, matrix, int(numofsequences), sequences

```

Gambar 13 Cuplikan Functions (7)

```

def saveToFile(maxreward, solution, runtime):
    filepath = getPath()
    with open(filepath, 'w') as file:
        file.write(str(maxreward) + '\n')
        for item in solution:
            file.write(item.alphanum + ' ')
        if (solution != []):
            file.write('\n')
        for item in solution:
            file.write(str(item.col+1) + ', ')
            file.write(str(item.row+1) + '\n')
        file.write('\n' + str(round(runtime)) + " ms\n")

def printMatrix(matrix):
    for tokens in matrix:
        for token in tokens:
            print(token.alphanum, end = " ")
        print()

def printBuffer(buffer):
    for item in buffer:
        print(item.alphanum, end = " ")
    if (buffer != []):
        print()

def printSequence(sequences):
    for sequence in sequences:
        for token in sequence.tokens:
            print(token, end=" ")
        print("\t\t Reward: ", sequence.reward)
    print()

```

Gambar 14 Cuplikan Functions (8)

```

def printCoordinates(buffer):
    for item in buffer:
        print(item.col+1, end=" ",)
        print(item.row+1)

def output(maxreward, solution, runtime):
    print(maxreward)
    printBuffer(solution)
    printCoordinates(solution)
    print()
    print(round(runtime), "ms")

```

Gambar 15 Cuplikan Functions (9)

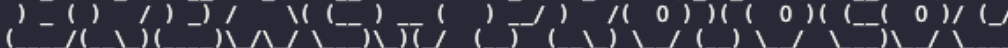
3.3 Main

main() memuat *splash screen* dan program utama aplikasi. Pengguna memilih jenis masukan (1: input dari file, 2: dihasilkan random). Masukan juga akan divalidasi. Jika pengguna memilih masukan dari file, dipanggil readFile(). Jika pengguna memilih opsi masukan secara random, dipanggil randomizeInput(), matrix dan sequences yang dihasilkan tercetak ke layar. Tiap fungsi mengembalikan buffersize, matrixcol, matrixrow, matrix, numofsequences, dan sequences.

EKSPERIMEN

Inisialisasi Program

```
Kaylas-MacBook-Air:src usagitsukino$ python3 main.py
```



```
Welcome to Cyberpunk 2077 Breach Protocol Solver!
```

```
-----MENU-----  
1. Input dari file  
2. Hasilkan input secara random  
-----  
Silakan pilih jenis masukan!(f : file, r : random)
```

Gambar 10 tampilan awal program

```

-----MENU-----
1. Input dari file
2. Hasilkan input secara random

Silakan pilih jenis masukan!(f : file, r : random)e
Silakan pilih jenis masukan!(f : file, r : random)

```

Gambar 11 tampilan input tidak valid

Input dari File (Opsi f)

```
Masukkan nama file('nama_file.txt') input1.txt
50
7A BD 7A BD 1C BD 55
1 , 1
1 , 4
3 , 4
3 , 5
6 , 5
6 , 3
1 , 3

1864 ms

Apakah ingin menyimpan solusi?(y/n)v

Masukan tidak valid. Harap memasukkan huruf y/n
Apakah ingin menyimpan solusi?(y/n)y

Masukkan nama file('nama_file.txt') output1.txt
Kaylas-MacBook-Air:src usagitsukino$
```

Gambar 12 tampilan input dari file


```
Silakan pilih jenis masukan!(f : file, r : random)f
Masukkan nama file('nama_file.txt') input1
File tidak ada pada folder 'test'
```

```
main.py M  invalidfile.txt U  Python src X
test > invalidfile.txt
1 2
2 5 3
3 55 1C 55 BD 7A
4 7A BD BD 55 7A
5 BD E9 1C BD 55
6 3
7 1C 1C 55 1C
8 247
9 1C 1C 55 1C
10 95
11 55 55 BD
12 21
13

Kaylas-MacBook-Air:src usagitsukino$ python3 main.py

Welcome to Cyberpunk 2077 Breach Protocol Solver!

-----MENU-----
1. Input dari file
2. Hasilkan input secara random

Silakan pilih jenis masukan!(f : file, r : random)f
Masukkan nama file('nama_file.txt') invalidfile.txt
Token harus unik!
Masukkan nama file('nama_file.txt') input3.txt
```

Gambar 13 tampilan kasus file yang tidak ditemukan dan format masukan pada file tidak valid (pada kasus ini, sekuens tidak unik misalnya)

Menghasilkan Matriks, Sekuens, dan Hadiah Secara Random (Opsis r)

```
Silakan pilih jenis masukan!(f : file, r : random)r
Masukkan jumlah token unik: 4
Masukkan token: A5 F1 W3 DB
Masukkan ukuran buffer: 8
Masukkan ukuran matriks: 5 4
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 3

Generated Matrix :
F1 F1 A5 DB W3
A5 W3 F1 A5 DB
A5 A5 W3 W3 A5
F1 W3 F1 A5 A5

Generated Sequences :
A5 F1 Reward: 454
W3 A5 A5 Reward: 42
DB DB F1 Reward: 306

496
A5 F1 W3 A5 A5
3 , 1
3 , 2
2 , 2
2 , 3
1 , 3

374 ms

Apakah ingin menyimpan solusi?(y/n)~
Masukan tidak valid. Harap memasukkan huruf y/n
Apakah ingin menyimpan solusi?(y/n)n
Kaylas-MacBook-Air:src usagitsukino$

Silakan pilih jenis masukan!(f : file, r : random)r
Masukkan jumlah token unik: 0

Masukan harus berupa bilangan lebih besar dari 0
Masukkan jumlah token unik: 4
Masukkan token: FF551CBD
Masukkan token harus sebanyak 4
Masukkan token: F# 55 1C BD

Token harus berupa karakter alfanumerik!
Masukkan token: FF 55 1C 1C

Token harus unik!
Masukkan token: FF 55 1C BD
Masukkan ukuran buffer: 6
Masukkan ukuran matriks: 55
Masukan harus berupa 'ukuran_kolom ukuran_baris'!
Masukkan ukuran matriks: 5 0

Masukan harus berupa bilangan positif
Masukkan ukuran matriks: 5 5
Masukkan jumlah sekuens: 1

Masukan harus berupa bilangan lebih besar dari 1
Masukkan jumlah sekuens: #

Masukan harus berupa angka!
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 4

Generated Matrix :
55 1C BD FF 55
BD FF 1C FF BD
1C 55 FF 1C BD
FF 55 FF 1C 55
1C FF 1C 55 55

Generated Sequences :
55 FF FF Reward: -189
BD 55 FF Reward: -490
55 FF BD Reward: -233
```

Gambar 14 tampilan *randomized* input valid (gambar a) dan *invalid input handling* (gambar b)

Menyimpan Hasil Solusi ke Dalam File

```
Masukkan nama file('nama_file.txt') input1.txt
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

1864 ms

Apakah ingin menyimpan solusi?(y/n)v
Masukan tidak valid. Harap memasukkan huruf y/n
Apakah ingin menyimpan solusi?(y/n)y
Masukkan nama file('nama_file.txt') output1.txt
Kaylas-MacBook-Air:src usagitsukino$
```

```
main.py M  output1.txt M X
test > output1.txt
1 50
2 7A BD 7A BD 1C BD 55
3 1, 1
4 1, 4
5 3, 4
6 3, 5
7 6, 5
8 6, 3
9 1, 3
10
11 1864 ms
```

Gambar 16 tampilan menyimpan solusi pada file

Contoh 1 (input1.txt)

```
Masukkan nama file('nama_file.txt') input1.txt
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

1900 ms

Apakah ingin menyimpan solusi?(y/n)y
Masukkan nama file('nama_file.txt') output1.txt
```

```
input1.txt X
test > 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
```

Gambar 17 test case 1 (berdasarkan contoh pada spek)

disclaimer: Terdapat perbedaan koordinat yang dihasilkan antara program ini dan pada spek, walaupun *reward* maksimal dan isi buffer yang ditemukan sama. Pada spek, token BD memiliki koordinat (6,4) dan 55 memiliki koordinat (5,4). Sedangkan pada program, token BD memiliki koordinat (6, 3) dan 55 memiliki koordinat (1, 3). Hal ini dikarenakan solusi tersebut yang pertama kali ditemukan pada struktur Tree, mengingat prioritas pencarian dari baris teratas dan kolom terkiri.

Contoh 2 (input2.txt)

Masukkan nama file('nama_file.txt') input2.txt
70
E9 1C FF 7A
3 , 1
3 , 2
2 , 2
2 , 4

9 ms

Apakah ingin menyimpan solusi?(y/n)y
Masukkan nama file('nama_file.txt') output2.txt

input2.txt ×
test > input2.txt
1 5
2 4 4
3 1C BD E9 FF
4 FF FF 1C 1C
5 E9 BD 1C FF
6 E9 7A FF 1C
7 2
8 E9 1C FF
9 40
10 FF 7A
11 30

Gambar 18 *test case 2*

Contoh 3 (input3.txt)

Silakan pilih jenis masukan!(f : file, r : random)f
Masukkan nama file('nama_file.txt') input3.txt
0

0 ms

Apakah ingin menyimpan solusi?(y/n)n

test > input3.txt
1 2
2 5 3
3 55 1C 55 BD 7A
4 7A BD BD 55 7A
5 BD E9 1C BD 55
6 3
7 1C 1C 55 1C
8 247
9 BD E9 7A
10 95
11 55 55 BD
12 21
13

Gambar 19 *test case 3* (kasus ukuran buffer lebih kecil dari ukuran sekuens)

Contoh 4 (input4.txt)

Masukkan nama file('nama_file.txt') input4.txt
937
3E 22 3E 4D BC 22 W3
1 , 1
1 , 3
4 , 3
4 , 5
6 , 5
6 , 1
2 , 1

3821 ms

Apakah ingin menyimpan solusi?(y/n)y

test > input4.txt
1 8
2 6 5
3 3E W3 W3 BC 4D 22
4 4D 3E W3 22 4D 4D
5 22 BC 3E 3E 22 3E
6 22 BC 4D W3 4D W3
7 W3 3E BC 4D 4D BC
8 3
9 22 3E 4D
10 421
11 BC 22
12 292
13 22 W3
14 224

Gambar 20 *test case 4*

Contoh 5 (input5.txt)

<pre> Silakan pilih jenis masukan!(f : file, r : random)f Masukkan nama file('nama_file.txt') input5.txt -15 7A BD 1C BD 55 1 , 1 1 , 5 6 , 5 6 , 3 1 , 3 1953 ms Apakah ingin menyimpan solusi?(y/n)y </pre>	<pre> test > input5.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 -30 12 BD 7A BD 13 -20 14 BD 1C BD 55 15 -15 </pre>
---	---

Gambar 21 *test case* 5 (Kasus *reward* negatif)

Contoh 6 (input6.txt)

<pre> Masukkan nama file('nama_file.txt') input6.txt 145 BD FF 55 55 3 , 1 3 , 2 1 , 2 1 , 3 12 ms Apakah ingin menyimpan solusi?(y/n)y </pre>	<pre> test > input6.txt 1 5 2 4 4 3 FF BD BD 1C 4 55 BD FF BD 5 55 1C BD FF 6 1C 1C 55 BD 7 3 8 55 BD BD 9 33 10 FF BD FF 11 79 12 FF 55 55 13 145 14 </pre>
--	---

Gambar 22 *test case* 6

BAB 5

PENUTUP

5.1 Kesimpulan

Melalui tugas besar ini, saya semakin belajar mengeksplor hal terkait *library* dan bahasa pemrograman python. Selain itu, algoritma *brute force* dapat menyelesaikan hampir segala macam persoalan algoritma, namun kurang efisien, dapat dilihat dari runtime yang cukup lama.

5.2 Saran

- Jangan mengerjakan h-1

LAMPIRAN

Link Repository

Link repository untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut

Link : https://github.com/kaylanamira/Tucil1_13522050

Tabel Checkpoint Program

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		✓

DAFTAR REFERENSI

[Algoritma Brute Force](#)

<https://cyberpunk-hacker.com/>

https://www.w3schools.com/python/ref_random_choice.asp