LAPORAN TUGAS KECIL 1
IF2211 Strategi Algoritma

Penyelesaian “Cyberpunk 2077 Breach Protocol” Dengan Algoritma Brute Force

Dipersiapkan oleh:

1. Diero Arga Purnama (13522056)

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung,
Jl. Ganesha 10, Bandung 40132

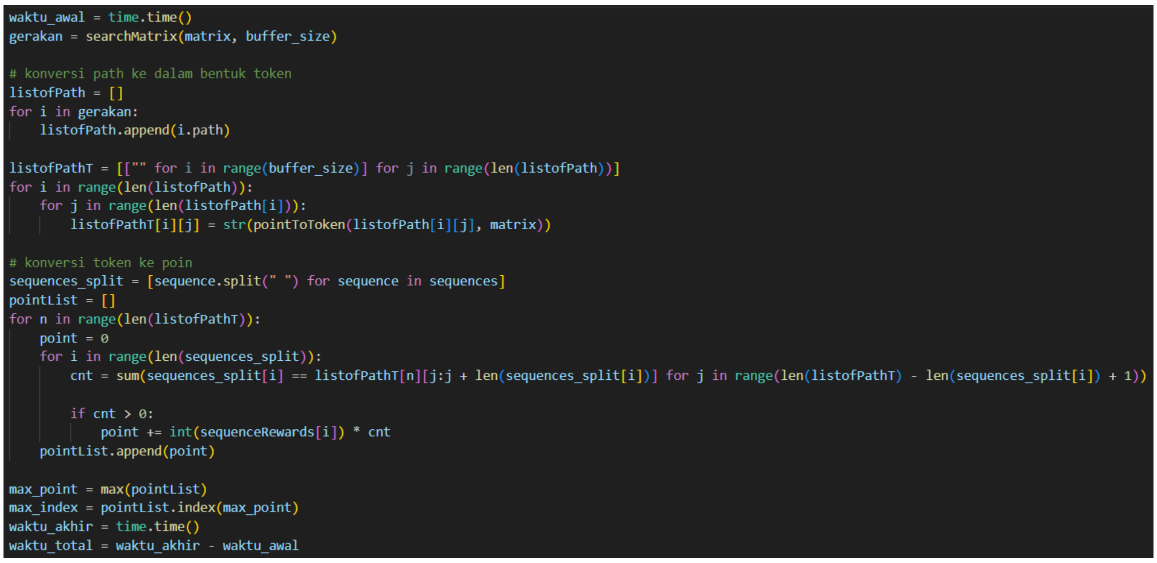
BAB I

1.1 Algoritma Brute Force

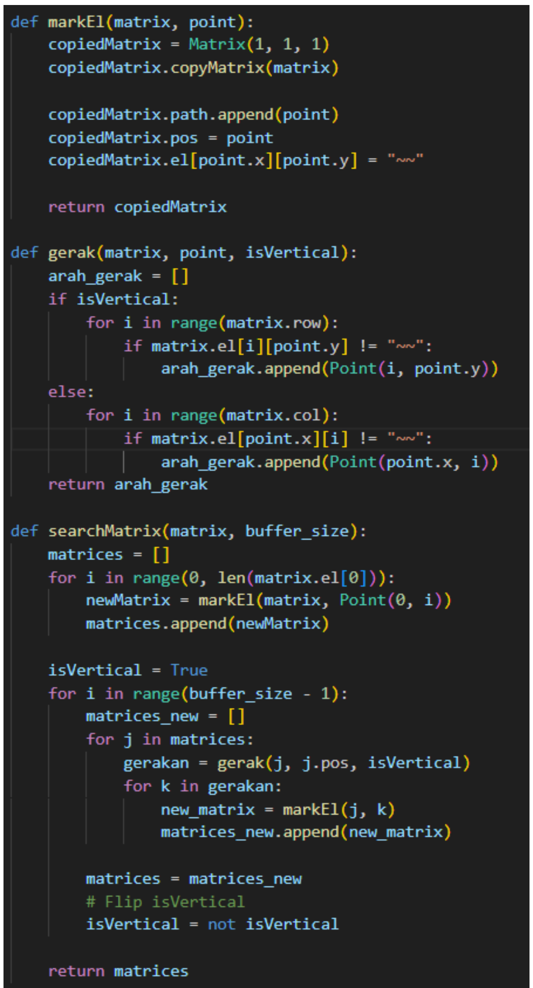
Pertama-tama, program akan mencari semua rangkaian token dengan ukuran sesuai dengan ukuran buffer yang dimasukkan yang ada dalam matriks. Pencarian dimulai dari seluruh elemen baris pertama kemudian bergerak sesuai dengan pola vertikal -> horizontal -> vertikal -> dst.. hingga panjang rangkain sebesar ukuran buffer.

Setelah didapatkan seluruh rangkaian token yang ada, *point* total dari tiap-tiap rangkain token akan dicek. Setelah didapatkan *point* dari tiap rangkaian, rangkaian token dengan *point* terbesar akan ditampilkan ke layar.

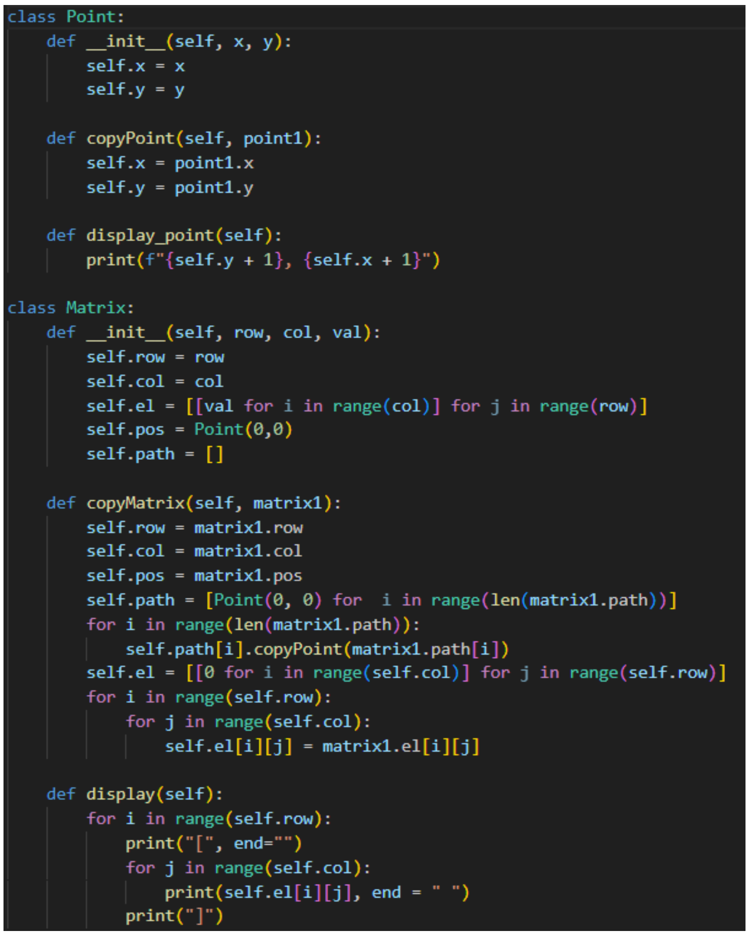
1.2 *Source* Program



```python
waktu_awal = time.time()
gerakan = searchMatrix(matrix, buffer_size)

# konversi path ke dalam bentuk token
listofPath = []
for i in gerakan:
    listofPath.append(i.path)

listofPathT = [["" for i in range(buffer_size)] for j in range(len(listofPath))]
for i in range(len(listofPath)):
    for j in range(len(listofPath[i])):
        listofPathT[i][j] = str(pointToToken(listofPath[i][j], matrix))

# konversi token ke poin
sequences_split = [sequence.split(" ") for sequence in sequences]
pointList = []
for n in range(len(listofPathT)):
    point = 0
    for i in range(len(sequences_split)):
        cnt = sum(sequences_split[i] == listofPathT[n][j:j + len(sequences_split[i])] for j in range(len(listofPathT) - len(sequences_split[i]) + 1))

        if cnt > 0:
            point += int(sequenceRewards[i]) * cnt
    pointList.append(point)

max_point = max(pointList)
max_index = pointList.index(max_point)
waktu_akhir = time.time()
waktu_total = waktu_akhir - waktu_awal
```

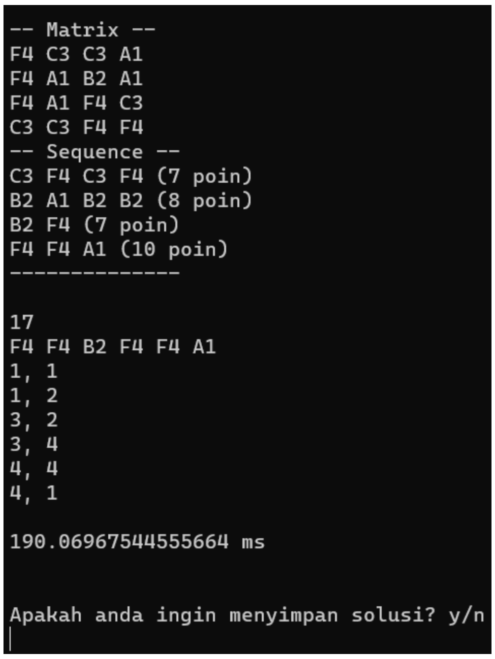


```python
def markEl(matrix, point):
    copiedMatrix = Matrix(1, 1, 1)
    copiedMatrix.copyMatrix(matrix)

    copiedMatrix.path.append(point)
    copiedMatrix.pos = point
    copiedMatrix.el[point.x][point.y] = "~~"

    return copiedMatrix

def gerak(matrix, point, isVertical):
    arah_gerak = []
    if isVertical:
        for i in range(matrix.row):
            if matrix.el[i][point.y] != "~~":
                arah_gerak.append(Point(i, point.y))
    else:
        for i in range(matrix.col):
            if matrix.el[point.x][i] != "~~":
                arah_gerak.append(Point(point.x, i))
    return arah_gerak

def searchMatrix(matrix, buffer_size):
    matrices = []
    for i in range(0, len(matrix.el[0])):
        newMatrix = markEl(matrix, Point(0, i))
        matrices.append(newMatrix)

    isVertical = True
    for i in range(buffer_size - 1):
        matrices_new = []
        for j in matrices:
            gerakan = gerak(j, j.pos, isVertical)
            for k in gerakan:
                new_matrix = markEl(j, k)
                matrices_new.append(new_matrix)

        matrices = matrices_new
        # Flip isVertical
        isVertical = not isVertical

    return matrices
```



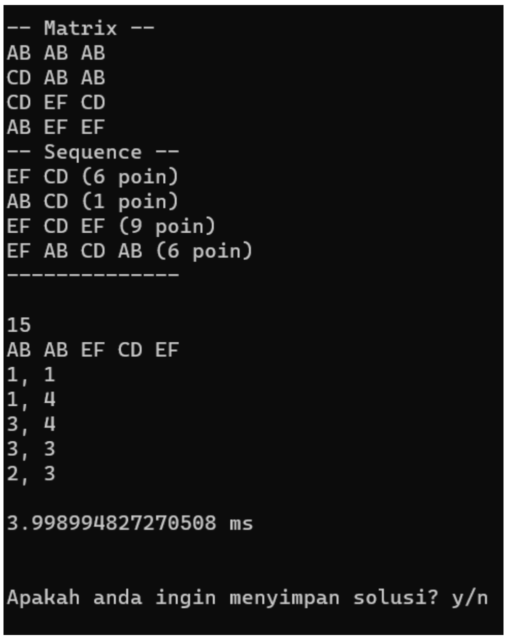
```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def copyPoint(self, point1):
        self.x = point1.x
        self.y = point1.y

    def display_point(self):
        print(f"{self.y + 1}, {self.x + 1}")

class Matrix:
    def __init__(self, row, col, val):
        self.row = row
        self.col = col
        self.el = [[val for i in range(col)] for j in range(row)]
        self.pos = Point(0,0)
        self.path = []

    def copyMatrix(self, matrix1):
        self.row = matrix1.row
        self.col = matrix1.col
        self.pos = matrix1.pos
        self.path = [Point(0, 0) for  i in range(len(matrix1.path))]
        for i in range(len(matrix1.path)):
            self.path[i].copyPoint(matrix1.path[i])
        self.el = [[0 for i in range(self.col)] for j in range(self.row)]
        for i in range(self.row):
            for j in range(self.col):
                self.el[i][j] = matrix1.el[i][j]

    def display(self):
        for i in range(self.row):
            print("[", end="")
            for j in range(self.col):
                print(self.el[i][j], end = " ")
            print("]")
```

1.3 Testing



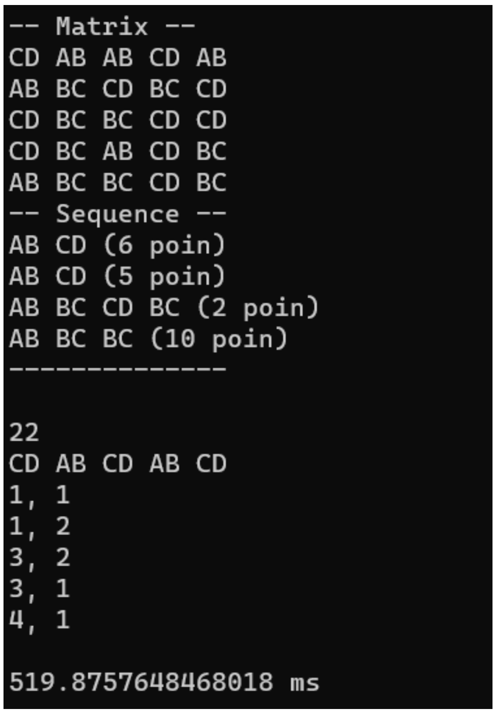
```
-- Matrix --
F4 C3 C3 A1
F4 A1 B2 A1
F4 A1 F4 C3
C3 C3 F4 F4
-- Sequence --
C3 F4 C3 F4 (7 poin)
B2 A1 B2 B2 (8 poin)
B2 F4 (7 poin)
F4 F4 A1 (10 poin)
--------------

17
F4 F4 B2 F4 F4 A1
1, 1
1, 2
3, 2
3, 4
4, 4
4, 1

190.06967544555664 ms


Apakah anda ingin menyimpan solusi? y/n
```

Figure 1. Test 1



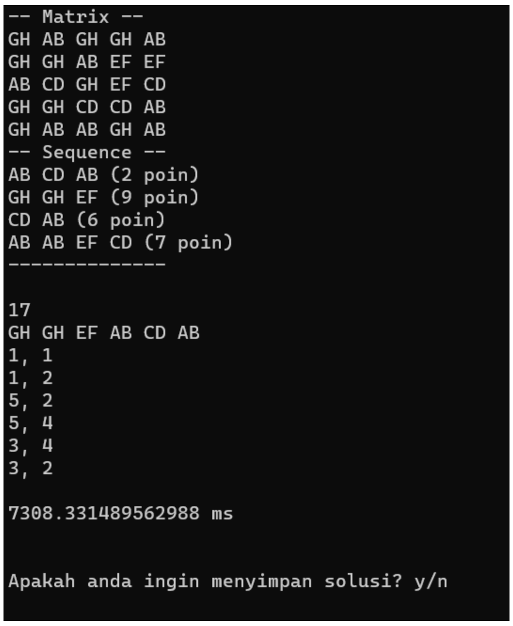
```
-- Matrix --
AB AB AB
CD AB AB
CD EF CD
AB EF EF
-- Sequence --
EF CD (6 poin)
AB CD (1 poin)
EF CD EF (9 poin)
EF AB CD AB (6 poin)
-------------

15
AB AB EF CD EF
1, 1
1, 4
3, 4
3, 3
2, 3

3.998994827270508 ms


Apakah anda ingin menyimpan solusi? y/n
```

Figure 2. Test 2



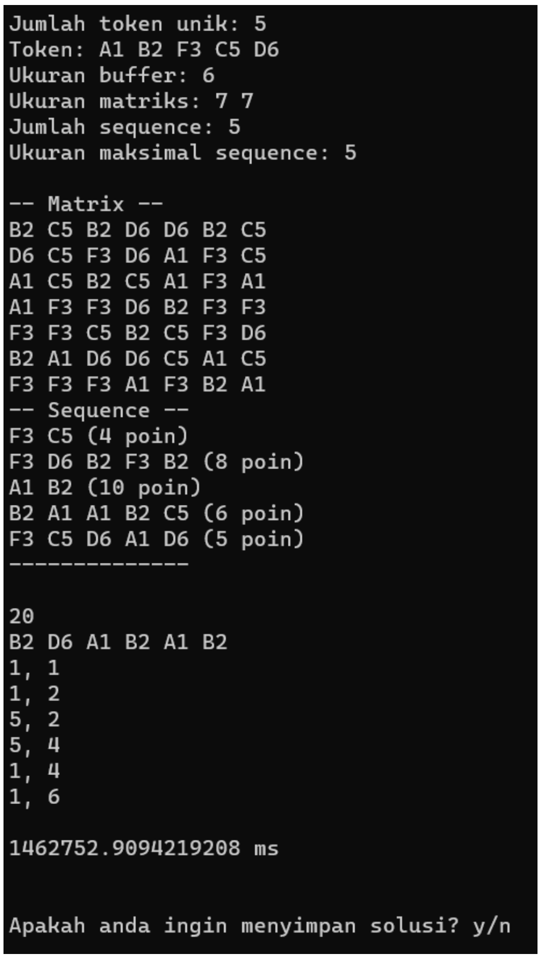
```
-- Matrix --
CD AB AB CD AB
AB BC CD BC CD
CD BC BC CD CD
CD BC AB CD BC
AB BC BC CD BC
-- Sequence --
AB CD (6 poin)
AB CD (5 poin)
AB BC CD BC (2 poin)
AB BC BC (10 poin)
--------------

22
CD AB CD AB CD
1, 1
1, 2
3, 2
3, 1
4, 1

519.8757648468018 ms
```

Figure 3. Test 3



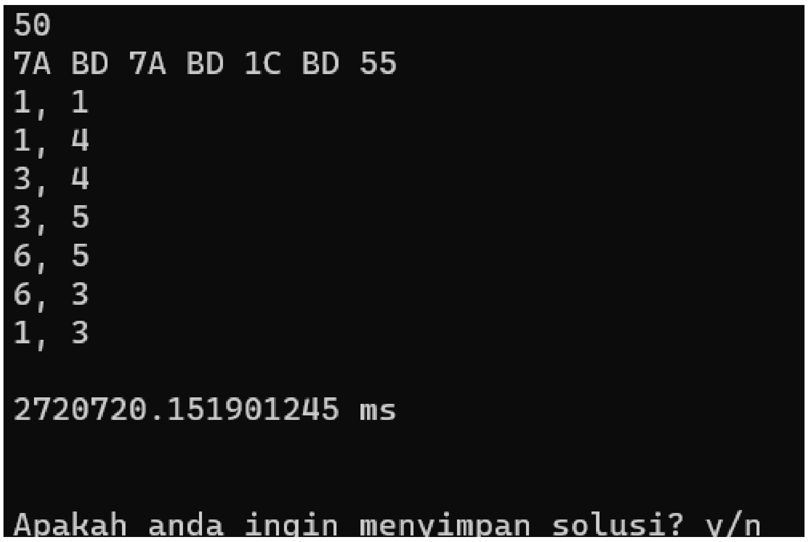
```
-- Matrix --
GH AB GH GH AB
GH GH AB EF EF
AB CD GH EF CD
GH GH CD CD AB
GH AB AB GH AB
-- Sequence --
AB CD AB (2 poin)
GH GH EF (9 poin)
CD AB (6 poin)
AB AB EF CD (7 poin)
--------------

17
GH GH EF AB CD AB
1, 1
1, 2
5, 2
5, 4
3, 4
3, 2

7308.331489562988 ms


Apakah anda ingin menyimpan solusi? y/n
```

Figure 4. Test 4



```
Jumlah token unik: 5
Token: A1 B2 F3 C5 D6
Ukuran buffer: 6
Ukuran matriks: 7 7
Jumlah sequence: 5
Ukuran maksimal sequence: 5

-- Matrix --
B2 C5 B2 D6 D6 B2 C5
D6 C5 F3 D6 A1 F3 C5
A1 C5 B2 C5 A1 F3 A1
A1 F3 F3 D6 B2 F3 F3
F3 F3 C5 B2 C5 F3 D6
B2 A1 D6 D6 C5 A1 C5
F3 F3 F3 A1 F3 B2 A1
-- Sequence --
F3 C5 (4 poin)
F3 D6 B2 F3 B2 (8 poin)
A1 B2 (10 poin)
B2 A1 A1 B2 C5 (6 poin)
F3 C5 D6 A1 D6 (5 poin)
--------------

20
B2 D6 A1 B2 A1 B2
1, 1
1, 2
5, 2
5, 4
1, 4
1, 6

1462752.9094219208 ms


Apakah anda ingin menyimpan solusi? y/n
```

Figure 5. Test 5



```
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

2720720.151901245 ms


Apakah anda ingin menyimpan solusi? y/n
```

Figure 6. Test 6

LAMPIRAN

Link Repository

https://github.com/DieroA/Tucil1_13522056

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca 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		✓