

Tugas Kecil 1 IF2211 Strategi Algoritma

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

Oleh:

Debrina Veisha Rashika W

K01 / 13522025



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024**

DAFTAR ISI

DAFTAR ISI	1
A. Algoritma Brute Force	2
B. Source Program	4
C. Input dan Output	12
D. Link Repository	20
E. Lampiran	20

A. Algoritma Brute Force

Algoritma *Brute force* adalah salah satu algoritma yang digunakan untuk menyelesaikan masalah pemrograman. Pendekatan dalam algoritma ini dilakukan dengan lempang (*straight forward*) dalam memecahkan suatu persoalan. Algoritma Brute force didasarkan pada pernyataan pada persoalan atau definisi yang dilibatkan pada persoalan.

Terdapat kelebihan dari algoritma *brute force* antara lain, sangat sederhana, langsung, caranya sangat jelas, dan mudah untuk diaplikasikan. Oleh sebab itu, algoritma ini dapat digunakan untuk menyelesaikan permainan Cyberpunk 2077 Breach Protocol. Berikut langkah-langkah algoritma yang dibuat.

1. Pertama, program membaca input dari pengguna yaitu melalui file txt atau pengguna dapat memilih untuk memasukkan input secara manual. Jika pengguna memasukkan input secara manual, maka matriks, sekuens beserta bobotnya akan disusun secara acak berdasarkan token yang dimasukkan oleh pengguna. Input dari pengguna akan divalidasi, sehingga jika ada input yang kurang tepat pengguna diminta untuk memperbaiki input yang diberikan. Sekuens-sekuens yang dimiliki akan digunakan untuk mencari kombinasi sekuens yang memiliki bobot tertinggi dan optimal yang terdapat pada matriks.
2. Setelah memiliki sekuens dan bobotnya, dilakukan permutasi pada sekuens-sekuens untuk mencari kemungkinan kombinasi dari sekuens yang dimiliki. Pencarian ini dilakukan dengan cara rekursif. Karena sekuens-sekuens tersimpan dalam sebuah list, permutasi diambil dengan mengambil elemen-elemen dari list tersebut. Pada awalnya, dibuat kombinasi 1 elemen terlebih dahulu, selanjutnya tiap elemen digabungkan dengan elemen lainnya dengan cara memanggil kembali fungsi rekursif tersebut. Hal ini dilakukan hingga membuat kombinasi dengan semua elemennya. Namun, jika terdapat kombinasi yang melebihi batas buffer atau kembar maka kombinasi tersebut tidak akan ditambahkan pada list yang akan digunakan untuk dilakukan pencocokan. Jumlah bobot dari kombinasi sekuens juga ditambahkan di akhir list agar lebih mudah untuk melakukan perbandingan nantinya.

3. Kombinasi sekuens yang didapat selanjutnya digunakan untuk dicocokkan per tokennya pada matriks. Hal ini dilakukan pada semua kombinasi yang didapat. Pada permainan pergerakan harus dimulai dari baris pertama. Oleh karena itu, akan dilakukan pencarian apakah terdapat token pada baris pertama sama dengan token pertama pada kombinasi sekuens yang dimiliki. Jika ya, maka titik koordinat dicatat dan proses pencarian dapat dilakukan pada token selanjutnya. Jika tidak, maka harus dilakukan pencarian pada kolom dari token yang ada pada baris pertama. Jika terdapat suatu elemen pada kolom yang memiliki token yang sama, maka token yang ada baris pertama kolom tersebut akan ditambahkan di bagian depan sekuens dan titik koordinatnya juga dicatat dan bisa melakukan pengecekan ke token selanjutnya.
4. Proses pergerakan dilakukan secara bergantian horizontal dan vertikal. Oleh karena itu, saya menghitung jumlah pergerakan yang akan digunakan untuk menentukan proses pencarian selanjutnya dilakukan secara vertikal atau horizontal. Misalnya, jika total pergerakannya genap maka akan dilakukan pengecekan secara horizontal dan jika pergerakannya ganjil maka akan dilakukan secara vertikal. Selain melihat apakah tokennya sama juga dilakukan pengecekan apakah pada titik tersebut sudah pernah digunakan atau belum. Jika sudah pernah dipakai, maka akan dilakukan pengecekan pada elemen selanjutnya.
5. Jika berhasil, melakukan pencocokan hingga token terakhir pada kombinasi sekuens. Maka akan dilakukan perbandingan bobot sekuens tersebut dengan total bobot maksimum saat ini. Jika bobotnya lebih besar dan panjang kombinasi lebih pendek maka nilai bobot maksimum, sekuens, dan koordinat maksimum akan diubah dengan yang terbaru.
6. Setelah semua solusi didapat, akan ditampilkan bobot maksimum, sekuens, koordinat yang dilewatinya, dan waktu eksekusinya. Namun, jika ternyata tidak ada solusi yang sesuai (tidak ada sekuens yang cocok) maka program akan menampilkan *no solution*.
7. Pengguna juga dapat menyimpan solusi dari program dalam format txt.

B. Source Program

1. Impor Modul

```
from flask import Flask, render_template, request, redirect, url_for, session, send_file
import os
import time
import json
import random
```

Gambar 2.1 Program Impor Modul

2. Kelas Titik

```
class Titik:
    def __init__(self, x1, y1):
        self.x = x1
        self.y = y1

    def cetaktitik(self):
        print(self.y+1, ",", self.x+1)

    def titiksama(self, t):
        return (t.x == self.x) and (t.y == self.y)

    def to_dict(self):
        return self.__dict__
```

Gambar 2.2 Program Kelas Titik

3. Kelas Matriks

```
class Matrix:
    def setmatrix(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.isi = ["" for i in range(cols)] for i in range(rows)]

    def setIsian(self, z):
        k = 0
        for i in range(self.rows):
            for j in range(self.cols):
                self.isi[i][j] = z[k]
                k += 1

    def setIsianRandom(self, z):
        for i in range(self.rows):
            for j in range(self.cols):
                self.isi[i][j] = random.choice(z)

    def getCols(self):
        return self.cols

    def getRows(self):
        return self.rows

    def printMatrix(self):
        for row in self.isi:
            print(" ".join(row))
```

Gambar 2.3 Program Kelas Matriks

4. Fungsi arraytitiksama, panjangaman, dan cekkembar

```
def arraytitiksama(t1, t2):
    # memeriksa apakah dua titik sama
    for t in t1:
        if t.titiksama(t2):
            return True
    return False

def panjangaman(sequence, buf):
    # memeriksa apakah panjang sequence masih kurang atau sama dengan buffer
    size = 0
    for seq in sequence:
        if size + len(seq) <= buf:
            size += len(seq)
        else:
            return False
    return True

def cekkembar(sequence, buf):
    # memeriksa apakah ada token yang kembar
    for seq in sequence:
        if seq == buf:
            return True
    return False
```

Gambar 2.4 Fungsi untuk Melakukan Suatu Pengecekan

5. Fungsi kombisekuens

```
def kombisequens(sequence, temp, idx, index, index2, seqgab, bobot, bb, k, buf):
    # fungsi untuk melakukan permutasi terhadap sekuens-sekuens
    if k == 0:
        if panjangaman(temp, buf) and not cekkembar(index, idx):
            temp2 = []
            maxi = 0
            for i in range(len(temp)):
                for j in range(len(temp[i])):
                    if i>0 and temp2[-1] == temp[i][j] and j==0:
                        continue
                    temp2.append(temp[i][j])
                    maxi += bb[i]
            temp2.append(str(maxi))
            seqgab.append(temp2)
            index.append(idx)
            return

    for i in range(len(sequence)+1):
        if i not in index2:
            if (i==len(sequence) and len(temp)):
                temp.append(["bebas"])
                idx += str(i)
                index2.append(i)
                bb.append(0)

                kombisequens(sequence, temp, idx, index, index2, seqgab, bobot, bb, k - 1, buf)
                temp.pop()
                idx = idx[:-1]
                bb.pop()
                index2.pop()
            elif(i!=len(sequence)):
                temp.append(sequence[i])
                idx += str(i)
                index2.append(i)
                bb.append(bobot[i])

                kombisequens(sequence, temp, idx, index, index2, seqgab, bobot, bb, k - 1, buf)
                temp.pop()
                idx = idx[:-1]
                bb.pop()
                index2.pop()
```

Gambar 2.5 Program untuk Menghasilkan Kombinasi Sekuens

6. Fungsi cekadatoken

```
def cekadatoken(m, token, bar, kol, pilihan):  
    # fungsi untuk memeriksa apakah terdapat token pada sebuah baris atau kolom  
    if pilihan == 1:  
        for j in range(len(m.isi)):  
            if m.isi[j][kol] == token or token == "bebas": # cek vertikal  
                return True  
    elif pilihan == 2:  
        for j in range(len(m.isi[bar])):  
            if m.isi[bar][j] == token or token == "bebas": # cek horisontal  
                return True  
    else:  
        for j in range(1, len(m.isi)):  
            if m.isi[j][kol] == token or token == "bebas": # cek vertikal dari baris ke 2  
                return True  
    return False
```

Gambar 2.6 Program untuk Memeriksa Validitas Suatu Token

7. Fungsi solve

```
def solve(buffer, bobot, sequence, m):  
    # program untuk mencari solusi optimal dengan mencocokkan sekuens dengan matriks  
    idx = ""  
    max_val = -999  
    max_seq = []  
    max_coords = []  
    koor = []  
  
    seqgab = []  
    for k in range(1, len(sequence) + 1):  
        temp = []  
        kombisequens(sequence, temp, idx, [], [], seqgab, bobot, [], k, buffer)  
  
    cnt = 0  
    while cnt < len(seqgab):  
        for i in range(m.cols):  
            tidakcocok = False  
            lok = 0  
            if m.isi[0][i] == seqgab[cnt][0]: # melakukan pencarian pada baris pertama  
                if (not cekadatoken(m, seqgab[cnt][1], lok, i, 1)):  
                    tidakcocok = True  
            else:  
                koor.append((0,i))  
                ada = True  
                done = False  
                total = 1  
                now = i  
                bebrov=-1  
                bebcov=-1
```

```

while total < len(seqgab[cnt]) - 1 and ada:
    ada = False
    if total % 2 == 1: # melakukan pencarian secara vertikal
        for z in range(m.rows):
            if (m.isi[z][now] == seqgab[cnt][total] or seqgab[cnt][total] == "bebas") and len(koor)>0 and not arraytitiksama(koor, Titik(z, now)):
                if total == len(seqgab[cnt]) - 2 or cekadatoken(m, seqgab[cnt][total + 1], z, z, 2):
                    if (seqgab[cnt][total] == "bebas"):
                        seqgab[cnt][total] = m.isi[z][now]
                        bebrow=cnt
                        bebcot=total
                        koor.append(Titik(z, now))
                        now = z
                        ada = True
                        break
    elif total % 2 == 0: # melakukan pencarian secara horizontal
        for z in range(m.cols):
            if (m.isi[now][z] == seqgab[cnt][total] or seqgab[cnt][total] == "bebas") and not arraytitiksama(koor, Titik(now, z)):
                if total == len(seqgab[cnt]) - 2 or ((total < len(seqgab) - 2) and cekadatoken(m, seqgab[cnt][total + 1], z, z, 1)):
                    if (seqgab[cnt][total] == "bebas"):
                        seqgab[cnt][total] = m.isi[now][z]
                        bebrow=cnt
                        bebcot=total
                        koor.append(Titik(now, z))
                        now = z
                        ada = True
                        break
    if ada and (total == len(seqgab[cnt]) - 2):
        done = True
    if ada:
        total += 1

```

```

if done: # jika terdapat dalam matriks
    total_weight = int(seqgab[cnt][-1])
    if total_weight > max_val:
        if max_val == -999 and len(max_seq) > len(seqgab):
            max_val = total_weight
            max_seq = seqgab[cnt]
            max_coords = koor
            if bebrow != -1:
                seqgab[bebrow][bebcot] == "bebas"
        else:
            max_val = total_weight
            max_seq = seqgab[cnt]
            max_coords = koor
            if bebrow != -1:
                seqgab[bebrow][bebcot] == "bebas"
    koor = []
if (m.isi[0][i] != seqgab[cnt][0] or tidakcocok) and len(seqgab[cnt]) <= buffer: # tidak ada token yang cocok di baris pertama
    if cekadatoken(m, seqgab[cnt][0], 0, i, 3):
        seqgab[cnt].insert(0, m.isi[0][i])
        koor.append(Titik(0, i))
        ada = True
        done = False
        total = 1
        now = i
        bebrow=-1
        bebcot=-1

```

```

while total < len(seqgab[cnt]) - 1 and ada:
    ada = False
    if total % 2 == 1: # melakukan pengecekan secara vertikal
        for z in range(m.rows):
            if (m.isi[z][now] == seqgab[cnt][total] or seqgab[cnt][total] == "bebas") and len(koor)>0 and not arraytitiksama(koor, Titik(z, now)):
                if total == len(seqgab[cnt]) - 2 or cekadatoken(m, seqgab[cnt][total + 1], z, z, 2):
                    if (seqgab[cnt][total] == "bebas"):
                        seqgab[cnt][total] = m.isi[z][now]
                        bebrow=cnt
                        bebcot=total
                        koor.append(Titik(z, now))
                        now = z
                        ada = True
                        break
    elif total % 2 == 0: # melakukan pengecekan secara horizontal
        for z in range(m.cols):
            if (m.isi[now][z] == seqgab[cnt][total] or seqgab[cnt][total] == "bebas") and not arraytitiksama(koor, Titik(now, z)):
                if total == len(seqgab[cnt]) - 2 or ((total < len(seqgab) - 2) and cekadatoken(m, seqgab[cnt][total + 1], z, z, 1)):
                    if (seqgab[cnt][total] == "bebas"):
                        seqgab[cnt][total] = m.isi[now][z]
                        bebrow=cnt
                        bebcot=total
                        koor.append(Titik(now, z))
                        now = z
                        ada = True
                        break
    if ada and total == len(seqgab[cnt]) - 2:
        done = True
    if not ada and bebrow != -1:
        seqgab[bebrow][bebcot] == "bebas"
    if ada:
        total += 1

```



```

        total += 1
    if done: # jika terdapat dalam matriks
        total_weight = int(seqgab[cnt][-1])
        if total_weight > max_val:
            if max!=999 and len(max_seq)>len(seqgab):
                max_val = total_weight
                seq_copy = seqgab[cnt].copy()
                max_seq = seq_copy
                max_coords = koor
            if bebrow!=-1:
                seqgab[bebrow][bebcot] == "bebas"

            else:
                max_val = total_weight
                seq_copy = seqgab[cnt].copy()
                max_seq = seq_copy
                max_coords = koor
            if bebrow!=-1:
                seqgab[bebrow][bebcot] == "bebas"

    koor = []
    seqgab[cnt].pop(0)

    cnt += 1

sek = " ".join(max_seq[:-1])
return max_val,sek,max_coords

```

Gambar 2.7 Program Utama yang Digunakan untuk Mencari Solusi Optimal

8. Fungsi upload

```

@app.route('/upload', methods=['POST'])
def upload():
    if request.method == 'POST':
        buffers = 0
        bobot = []
        sequence = []
        sequencemax = ""
        m = Matrix()
        file = request.files['fileInput']
        if file:
            seqs = []
            lines = file.stream.readlines()
            for i, line in enumerate(lines):
                line = line.decode('utf-8').strip()
                if i == 0:
                    buffer = int(line)
                elif i == 1:
                    a, b = map(int, line.split())
                    m.setmatrix(b, a)
                elif 1 < i < m.getRows() + 2:
                    seqs += line.split()
                    if i == m.getRows() + 1:
                        m.setIsian(seqs)
                elif i == m.getRows() + 2:
                    count = int(line)

```

```

        for j in range(count * 2):
            line = lines[i + j + 1].decode('utf-8').strip()
            lineWords = []
            if j % 2 == 0:
                lineWords = line.split()
                sequence.append(lineWords)
            else:
                bobot.append(int(line))
start_time = time.time()
buffers, sequencemax, coordinate = solve(buffer, bobot, sequence, m)
end_time = time.time()
elapsed_time = (end_time - start_time)*1000

if(buffers<0):
    buffers=0
    sequencemax = "No Solution"
    coordinate = []

koordinat = []
for i in coordinate:
    koor = []
    koor.append(i.x)
    koor.append(i.y)
    koordinat.append(koor)

session['buffers'] = buffers
session['sekuense'] = sequence
session['bobot'] = bobot
session['sekuenses'] = sequencemax
session['elapsed_time'] = elapsed_time
session['matrix'] = m.isi
session['coordinate'] = koordinat
session['mode'] = "file"

return redirect(url_for('result'))

```

Gambar 2.8 Program untuk Mengolah Input dari File

9. Fungsi manualupload

```
@app.route('/manual_upload', methods=['POST'])
def manual_upload():
    buffers = 0
    bobot = []
    sequence = []
    sequencemax = ""
    m = Matrix()

    jumlah = int(request.form['jumlah'])
    sekuens = request.form['sekuens'].split()
    buffer = int(request.form['buffer'])
    row = int(request.form['row'])
    cols = int(request.form['cols'])
    jumlah_sekuens = int(request.form['jumlah_sekuens'])
    maks_sekuens = int(request.form['maks_sekuens'])

    m.setmatrix(row, cols)
    m.setIsianRandom(sekuens)

    for _ in range(jumlah_sekuens):
        temp = []
        length = random.randint(2, maks_sekuens)
        for _ in range(length):
            temp.append(random.choice(sekuens))
        while temp in sequence:
            for _ in range(length):
                temp.append(random.choice(sekuens))
        sequence.append(temp)
        bobot.append(random.randint(1, 60))

    start_time = time.time()
    buffers, sequencemax, coordinate = solve(buffer, bobot, sequence, m)
    end_time = time.time()
    elapsed_time = (end_time - start_time) * 1000

    if (buffers < 0):
        buffers = 0
        sequencemax = "No Solution"
```

```

coordinate = []

koordinat = []
for i in coordinate:
    koor = []
    koor.append(i.x)
    koor.append(i.y)
    koordinat.append(koor)

session['sekuense'] = sequence
session['bobot'] = bobot
session['buffers'] = buffers
session['sekuenses'] = sequencemax
session['elapsed_time'] = elapsed_time
session['matrix'] = m.isi
session['coordinate'] = koordinat
session['mode'] = "manual"

return redirect(url_for('result'))

```

Gambar 2.9 Program untuk Mengolah Input Manual

10. Fungsi result dan download

```

@app.route('/result')
def result():
    buffers = session.get('buffers')
    sequence = session.get('sekuenses')
    seq = session.get('sekuense')
    bobot = session.get('bobot')
    elapsed_time = session.get('elapsed_time')
    mode = session.get('mode')
    matrix = session.get('matrix')
    matrix = json.dumps(matrix)

    coordinate = session.get('coordinate')
    coordinate = json.dumps(coordinate)

    return render_template('result.html', buffers=buffers, sequence=sequence,
elapsed_time=elapsed_time,matrix=matrix,coordinate=coordinate,seq=seq,bobot=bobot,mode=mode)

```

```

@app.route('/download', methods=['GET'])
def download():
    buffers = session.get('buffers')
    sequencemax = session.get('sekuenses')
    elapsed_time = session.get('elapsed_time')
    matrix = session.get('matrix')
    coordinate = session.get('coordinate')

    content = f"Buffers: {buffers}\n"
    content += f"Sequences: {sequencemax}\n"
    content += f"Elapsed Time: {elapsed_time} seconds\n"
    content += "Coordinates:\n"
    for coord in coordinate:
        content += f" {coord[1]+1}, {coord[0]+1}\n"

    nama = sequencemax + ".txt"
    save_path = os.path.join((os.path.dirname(app.root_path)), 'test', nama)

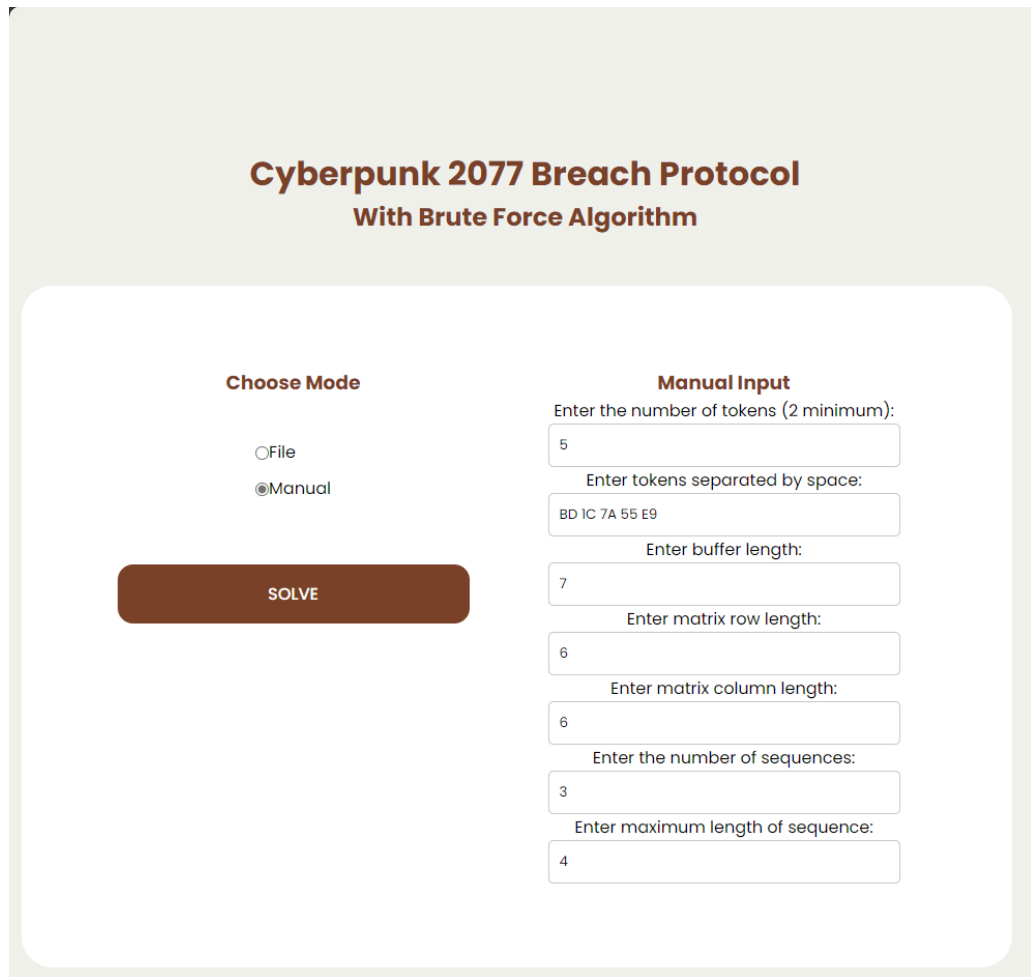
    with open(save_path, "w") as file:
        file.write(content)
    return send_file(save_path, as_attachment=True)

```

Gambar 2.10 Program untuk Mendownload Solusi

C. Input dan Output

1. Test case 1 (Manual Input)



Cyberpunk 2077 Breach Protocol
With Brute Force Algorithm

Choose Mode

☐ File

☒ Manual

SOLVE

Manual Input

Enter the number of tokens (2 minimum):
5

Enter tokens separated by space:
BD IC 7A 55 E9

Enter buffer length:
7

Enter matrix row length:
6

Enter matrix column length:
6

Enter the number of sequences:
3

Enter maximum length of sequence:
4

Gambar 3.1 Input Test Case 1

Result

Maximum Weight: 52

Sequence: 55 7A 55 E9 7A 1C

Elapsed Time: 0.0 ms

7A	1C	E9	55	55	1C
BD	55	55	55	7A	7A
1C	BD	BD	E9	E9	BD
7A	BD	1C	E9	7A	55
55	55	BD	BD	BD	55
E9	1C	BD	55	BD	E9

Sequences

7A 55 Bobot: 7

E9 BD 7A 55 Bobot: 19

55 E9 7A 1C Bobot: 45

[Download Result](#)

[BACK](#)

Gambar 3.2 Output Test Case 1

2. Test Case 2 (Manual Input)

Cyberpunk 2077 Breach Protocol With Brute Force Algorithm

Choose Mode

☐ File

☒ Manual

SOLVE

Manual Input

Enter the number of tokens (2 minimum):
6

Enter tokens separated by space:
A1 A2 A3 B1 B2 B3

Enter buffer length:
10

Enter matrix row length:
10

Enter matrix column length:
10

Enter the number of sequences:
9

Enter maximum length of sequence:
5

Gambar 3.1 Input Test Case 2

Result

Maximum Weight: 134

Sequence: B3 B2 A1 A3 B1 A2 A3 A1 B3

Elapsed Time: 22.040367126464844 ms

B2	B1	A3	B1	B1	B3	B1	A3	B2
B3	B1	A3	B1	A3	B1	A1	A3	A2
B2	A1	B1	A1	A2	B1	B3	B3	B1
A2	A1	A1	A1	B1	A1	A1	A1	B2
B2	A2	B3	A1	B1	A2	A3	A3	B3
A1	B3	A2	B1	B2	A2	A3	A1	B1
B1	A3	A1	A1	B2	B2	A3	B2	B1
A3	A1	B1	B2	B2	B3	A2	B3	A3
B3	A3	B1	B2	A1	A3	B1	B3	A3

Sequences

B3 A3 B3 B3 Bobot: 42

B3 B2 Bobot: 36

A1 A3 B1 Bobot: 48

A3 B2 Bobot: 16

A1 A3 B1 A2 B2 Bobot: 3

A2 A3 A1 B3 Bobot: 50

[Download Result](#)

[BACK](#)

Gambar 3.4 Output Test Case 2

3. Test Case 3 (Manual Input)

Cyberpunk 2077 Breach Protocol

With Brute Force Algorithm

Choose Mode

☐ File

☒ Manual

SOLVE

Manual Input

Enter the number of tokens (2 minimum):

3

Enter tokens separated by space:

ab cd ss

Enter buffer length:

7

Enter matrix row length:

2

Enter matrix column length:

2

Enter the number of sequences:

1

Enter maximum length of sequence:

10

Gambar 3.5 Output Test Case 3

Result

No Solution

Elapsed Time: 0.0 ms

ss	cd
cd	cd

Download Result

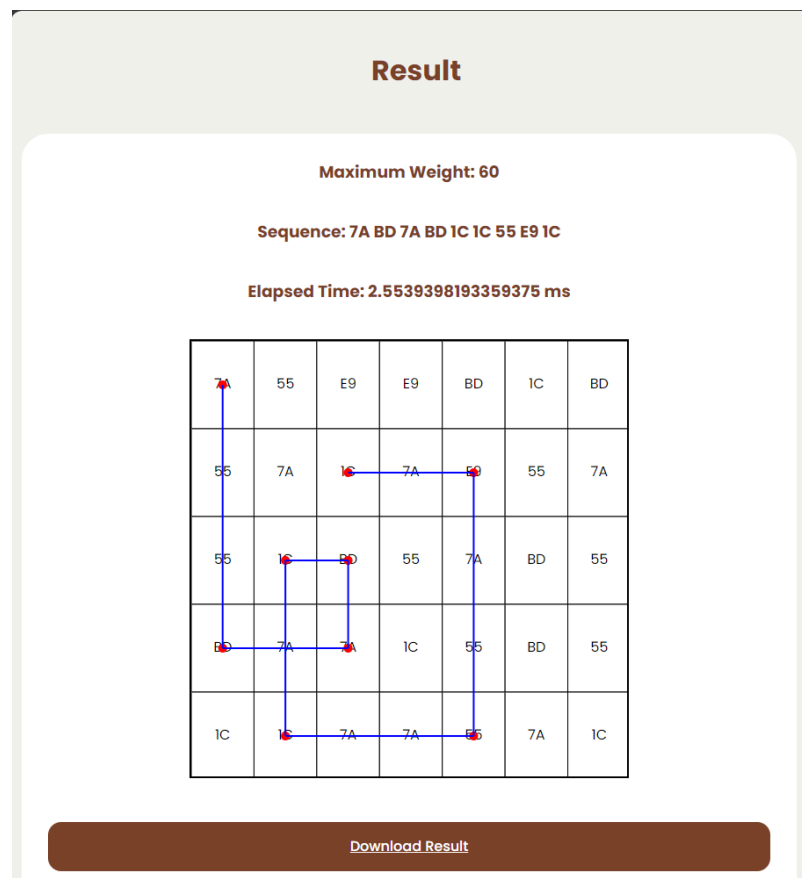
Gambar 3.6 Output Test Case 3

4. Test Case 4 (input 1.txt)

A. Input File

```
9
7 5
7A 55 E9 E9 BD 1C BD
55 7A 1C 7A E9 55 7A
55 1C BD 55 7A BD 55
BD 7A 7A 1C 55 BD 55
1C 1C 7A 7A 55 7A 1C
4
55 E9 1C
10
BD 7A BD
30
1C 55 E9 BD
-10
1C 1C
20
```

B. Output



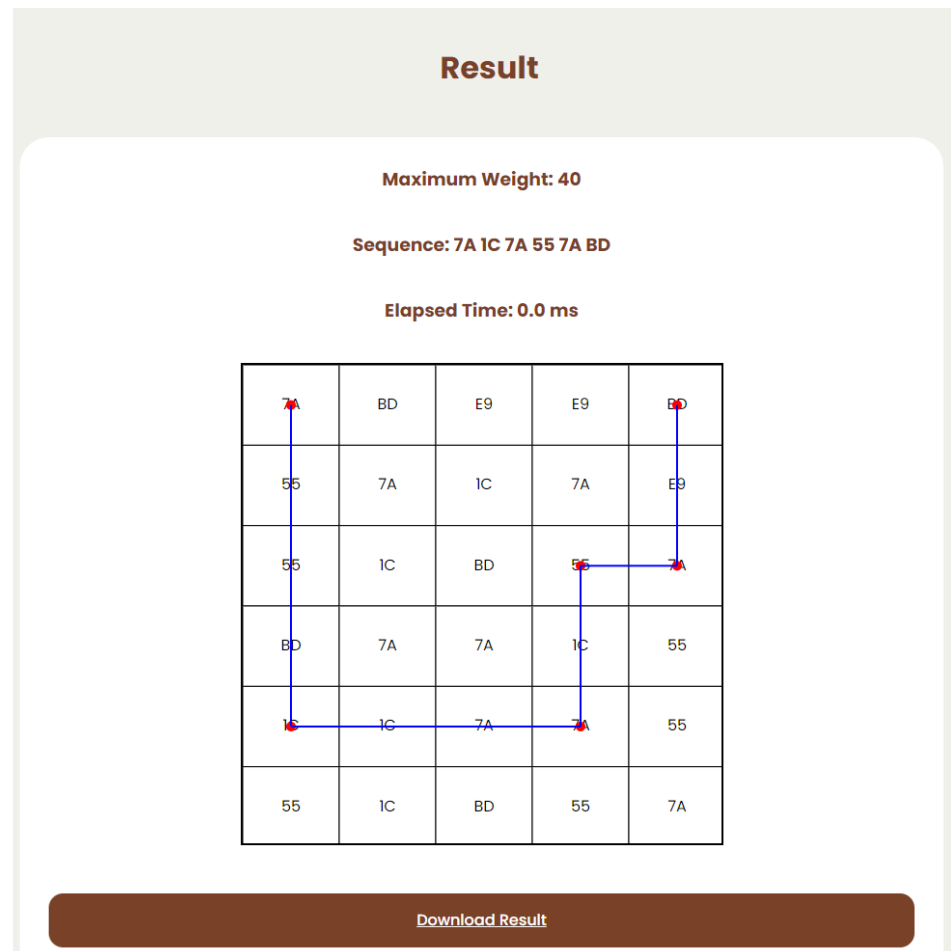
Gambar 3.7 Output Test Case 4

5. Test Case 5 (input 2.txt)

A. Input File

```
9
5 6
7A BD E9 E9 BD
55 7A 1C 7A E9
55 1C BD 55 7A
BD 7A 7A 1C 55
1C 1C 7A 7A 55
55 1C BD 55 7A
3
7A 1C 7A
10
7A 55 7A BD
30
1C 55 E9 BD
5
```

B. Output



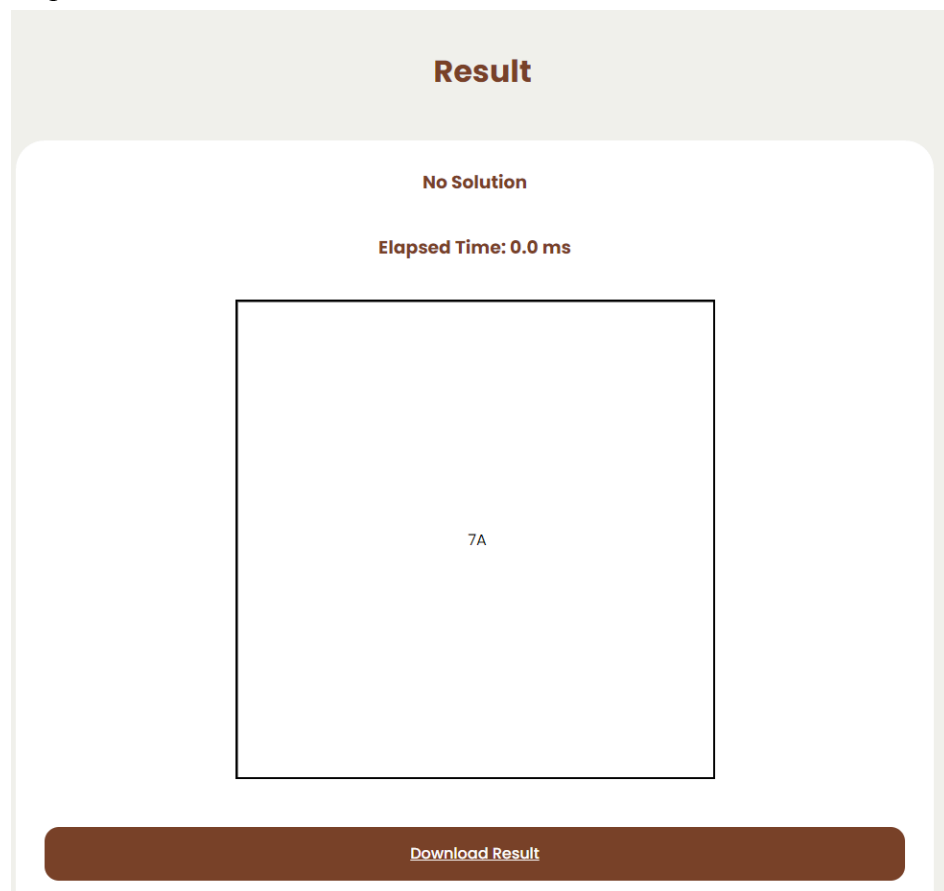
Gambar 3.8 Output Test Case 5

6. Test Case 6 (input 3.txt)

A. Input File

```
9
1 1
7A
3
1C 55
100
7A 55 BD
300
55 E9 BD
50
```

B. Output



Gambar 3.9 Output Test Case 6

D. Link Repository

Berikut Repository Github:

https://github.com/debrinashika/Tucil1_Stima_13522025

E. Lampiran

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	✓	