Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun Akademik 2023/2024

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma Brute Force

Disusun oleh :

Vanson Kurnialim – 13522049

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

Table of Contents

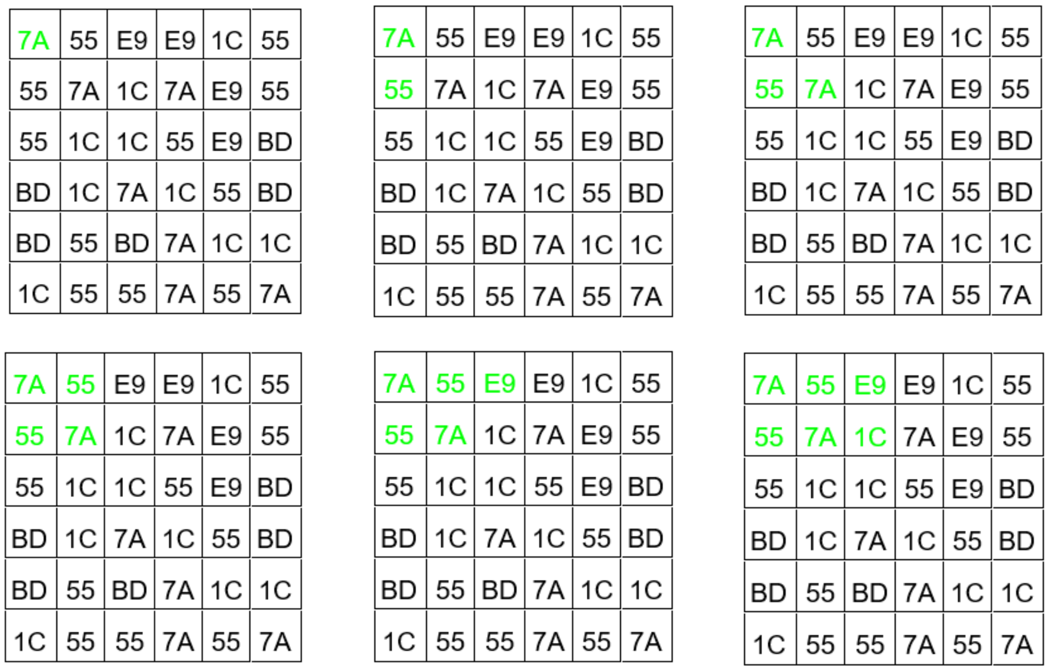
I. Algoritma Brute Force

Brute force memiliki arti ‘kasar’ atau ‘memaksa’. Berdasarkan pengertian tersebut, algoritma *brute force* dapat diartikan sebagai algoritma yang bersifat kasar atau memaksa. Algoritma *brute force* menyelesaikan suatu permasalahan dengan cara pencarian lengkap atau menghabiskan segala kemungkinan yang mungkin dari suatu persoalan sehingga didapat hasil dari persoalan tersebut. Algoritma ini merupakan algoritma yang simpel dan konsisten namun merupakan algoritma yang memakan banyak waktu atau tidak efisien dikarenakan model sistem pencariannya. Kompleksitas waktu dari algoritma ini sering kali berhubungan linear dengan besar input atau sampel.

Berikut adalah langkah-langkah algoritma *brute force* yang saya gunakan untuk menyelesaikan permasalan *Cyberpunk 2077 Breach Protocol :*

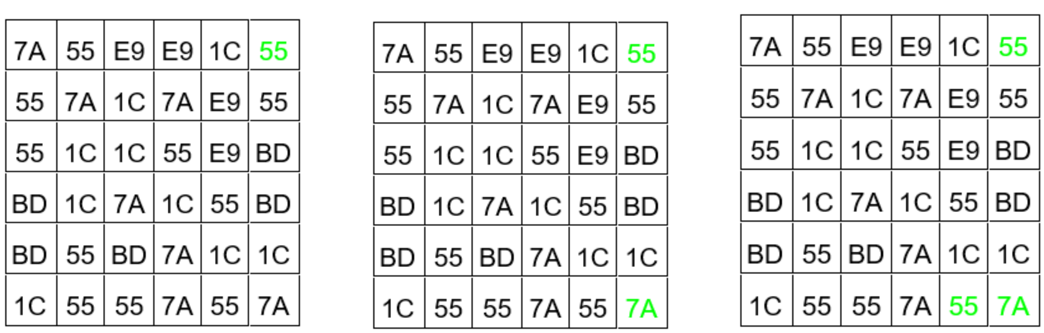
1. Dikarenakan *buffer* harus dimulai dari token di baris paling atas, maka iterasi utama merupakan pergiliran dari token-token paling atas. Selanjutnya, akan dicari semua kombinasi *buffer* yang mungkin dari token pertama tersebut.
2. Langkah berikutnya adalah gerakan vertikal, maka diiterasikan setiap kemungkinan token vertikal yang valid sebagai keberlanjutan dari rangkaian *buffer*. Posisi atau koordinat token terhadapt matrix juga disimpan.
3. Kemudian, untuk setiap kemungkinan token vertikal akan dicari pula setiap kemungkinan token horizontal yang dijadikan sebagai rangkaian *buffer* yang mungkin.
4. Pergerakan vertikal dan horizontal tersebut akan diulangi terus sesuai dengan panjang *buffer*. Setiap token hanya boleh dikunjungi sekali untuk tiap *buffer,* maka jika tidak ada token vertikal atau horizontal yang valid, iterasi pencarian *buffer* akan berhenti. Dengan demikian semua rangkaian *buffer* yang mungkin sudah pernah dilewati.
5. Pada proses pencarian, untuk setiap tambahan token pada *buffer,* akan langsung dicek dengan *sequence* dan akan disimpan kedalam variabel global jika lebih optimal dibanding *buffer* pada variabel global pada saat itu.

Berikut adalah gambaran alur pencarian *buffer-buffer* berukuran 6 yang mungkin :



Gambar 1. Alur iterasi pencarian *buffer* paling pertama

Pada proses pencarian token terakhir pada *buffer*, E9 akan dipilih paling pertama, namun karena sudah pernah dikunjungi, maka akan dilewati. Setelah itu 1C akan dipilih dan karena valid, akan menjadi sebuah *buffer* yang utuh atau selesai. Iterasi tetap akan berjalan dalam penentuan token dengan 1C, 7A, BD, dan 55 sebagai kandidat token terakhir.



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

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

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

Gambar 2. Alur iterasi pencarian *buffer* paling akhir

Proses iterasi akan terus berjalan pada setiap percabangan vertikal dan horizontal sampai proses paling akhir seperti pada gambar di atas.

II. Source Code

Program dibuat menggunakan bahasa pemrograman python. Berikut adalah keseluruhan kode dari program yang telah dibuat :

```
import time
import random

class Token :
    def __init__(self, content, point, list_of_coordinate=[]):
        self.content = content
        self.list_of_coordinate = list_of_coordinate
        self.point = point

class Coordinate :
    def __init__ (self, row, col) :
        self.row = row
        self.col = col

# variabel global
hasil = Token("Nan", 0, [])
bank = []

# fungsi pendukung
def resetMatrixUsage(matrix) :
    for i in range(len(matrix)) :
        for j in range(len(matrix[0])) :
```

```
            matrix[i][j][1] = False

def fillMatrixUsage(matrix, list_of_coordinate) :
    for i in range(len(list_of_coordinate)) :
        matrix[list_of_coordinate[i].row][list_of_coordinate[i].col][1] = True

def stringCheck(string, sequence) :
    global hasil
    string.point = 0
    for i in range(len(sequence)) :
        if (sequence[i][0] in string.content) :
            string.point += sequence[i][1]

    if (string.point > hasil.point) :
        hasil.content = string.content
        hasil.point = string.point
        hasil.list_of_coordinate = string.list_of_coordinate

    if (string.point == hasil.point) :
        if (len(string.content) < len(hasil.content)) :
            hasil.content = string.content
            hasil.point = string.point
            hasil.list_of_coordinate = string.list_of_coordinate

def createCopy(b) :
    temp = []
    for i in range(len(b)) :
        temp.append(b[i])
    return temp

def displayCoordinate(string) :
    for i in range(len(string.list_of_coordinate)) :
        print(f"{string.list_of_coordinate[i].col+1},{string.list_of_coordinate[i
].row+1}")

def generateBank(matrix, count, arah, answer) : # arah = 0 = ver , arah = 1 = hor
    if (count == 0) : # basis
        global bank
        bank.append(answer)
    else :
        count -= 1
        if (arah == 0) :
            alamat = answer.list_of_coordinate[-1]
            alamatcol = alamat.col
            for i in range(len(matrix)) :
```

```
                resetMatrixUsage(matrix)
                fillMatrixUsage(matrix, answer.list_of_coordinate)
                if (matrix[i][alamatcol][1] == True) :
                    continue
                else :
                    temp = createCopy(answer.list_of_coordinate)
                    answer2 = Token(answer.content, answer.point, temp)
                    matrix[i][alamatcol][1] = True
                    answer2.content += " " + matrix[i][alamatcol][0]
                    answer2.list_of_coordinate.append(Coordinate(i, alamatcol))
                    stringCheck(answer2, sequence)

                    generateBank(matrix, count, 1, answer2)
        else :
            alamat = answer.list_of_coordinate[-1]
            alamatrow = alamat.row
            for i in range(len(matrix[0])) :
                resetMatrixUsage(matrix)
                fillMatrixUsage(matrix,
answer.list_of_coordinate)
                if (matrix[alamatrow][i][1] == True) :
                    continue
                else :
                    temp = createCopy(answer.list_of_coordinate)
                    answer2 = Token(answer.content, answer.point, temp)
                    matrix[alamatrow][i][1] = True
                    answer2.content += " " + matrix[alamatrow][i][0]
                    answer2.list_of_coordinate.append(Coordinate(alamatrow, i))
                    stringCheck(answer2, sequence)

                    generateBank(matrix, count, 0, answer2)

def saveFile(executionTime) :
    while (True) :
            save = str(input("Do you want to save the solution? (y/n) "))
            if (save == "y") :
                nama = str(input("Enter file name : "))
                if (".txt" not in nama) :
                    nama += ".txt"
                file = open("../test/" + nama, "w")
                file.write("")
                file = open("../test/" + nama, "a")
                file.write(str(hasil.point) + "\n")
                if (hasil.point != 0) :
                    file.write(hasil.content + "\n")
```

```
                    for i in range(len(hasil.list_of_coordinate)) :
                        file.write(f"{hasil.list_of_coordinate[i].col+1},{hasil.l
ist_of_coordinate[i].row+1}\n")
                file.write("\n" + str(executionTime) + " ms")

                break
            elif (save == "n") :
                break
            else :
                print("Wrong input! \n")

# main program
print("--------------------------------------")
print("|| BREACH PROTOCOL OPTIMAL SOLUTION ||")
print("--------------------------------------")

load = str(input("\nDo you want to load existing txt file? (y/n) "))
if (load == "y") :
    while(True) :
        nama = str(input("Enter file name : "))
        if (".txt" not in nama) :
            nama += ".txt"
        try :
            file = open("../test/" + nama, "r")
        except FileNotFoundError :
            print("File tidak ditemukan\n")
        else :
            break

    print("\nLoading...\n")
    buffer_size = int(file.readline())
    col = int(file.read(2))
    row = int(file.readline())

    matrix = [[[0 , False] for i in range(col)] for j in range(row)]
    for i in range (row) :
        for j in range(col) :
            matrix[i][j][0] = file.read(2)
            blank = file.read(1)

    number_of_sequence = int(file.readline())

    sequence = [["Nan" , 0] for i in range (number_of_sequence)]
    for i in range(number_of_sequence) :
        sequence[i][0] = file.readline() [:-1]
```

```
        sequence[i][1] = int(file.readline())

    start = time.time()
    for i in range(len(matrix[0])) :
        a = []
        a.append(Coordinate(0,i))
        coba = Token(matrix[0][i][0], 0, a)
        generateBank(matrix, buffer_size-1, 0, coba)
    end = time.time()
    executionTime = round((end - start) * 1000)

    print("--------------")
    print("|| Solution ||")
    print("--------------")
    print(hasil.point)
    if (hasil.point <= 0) :
        print("Tidak ada solusi yang optimal!")
    else :
        print(hasil.content)
        displayCoordinate(hasil)

    print("\n")
    print(executionTime, "ms")

    saveFile(executionTime)


elif (load == "n") :
    print("\nRequesting Information...")
    jumlah_token_unik = int(input("Number of unique token : "))
    token_unik = []
    template = str(input(f"Enter {jumlah_token_unik} unique token separated by
blank: "))
    if (len(template) < ((jumlah_token_unik*3) - 1)) :
        print("\nInput invalid! Abort program... \n")
    else :
        for i in range(jumlah_token_unik) :
            temp = template[i*3] + template[(i*3)+1]
            token_unik.append(temp)

        buffer_size = int(input("\nBuffer size : "))
        print("\n|| Matrix size ||")
        col = int(input("Column size : "))
        row = int(input("Row size : "))
```

```
        matrix = [[[token_unik[random.randint(0,jumlah_token_unik-1)], False] for
i in range(col)] for j in range(row)]

        while(True) :
            number_of_sequence = int(input("\nNumber of Sequence : "))
            sequence_size = int(input("Maximum sequence size : "))
            if ((sequence_size <= 1) or (number_of_sequence < 1)) :
                print("Minimum number of 1 sequence and sequence size of 2!")
            else :
                break

        sequence = [["Nan" , 0] for i in range (number_of_sequence)]
        for i in range(number_of_sequence) :
            temp = ""
            for j in range(0, random.randint(2, sequence_size)) :
                temp += token_unik[random.randint(0, jumlah_token_unik-1)]
                temp += " "
            sequence[i][0] = temp [:-1]
            sequence[i][1] = random.randint(2,6) * 5

        print("\nGenerating...")

        print("\n|| Generated Matrix ||" , end="")
        for i in range(row) :
            print("")
            for j in range(col) :
                print(matrix[i][j][0], end=" ")

        print("\n\n|| Generated Sequences ||")
        for i in range(number_of_sequence) :
            print(sequence[i][0])
            print(sequence[i][1] , "points\n")

        print("Loading...\n")

        start = time.time()
        for i in range(len(matrix[0])) :
            a = []
            a.append(Coordinate(0,i))
            coba = Token(matrix[0][i][0], 0, a)
            generateBank(matrix, buffer_size-1, 0, coba)
        end = time.time()
        executionTime = round((end - start) * 1000)

        print("--------------")
```

```
        print("|| Solution ||")
        print("--------------")
        print(hasil.point)
        if (hasil.point <= 0) :
            print("Tidak ada solusi yang optimal!")
        else :
            print(hasil.content)
            displayCoordinate(hasil)

        print("\n")
        print(executionTime, "ms")

        saveFile(executionTime)

else :
    print("Sorry! Wrong input")
```

Gambar 3. Source code program

Catatan kecil pada bagian *execution time*, yang ditampilkan di layar dan disimpan dalam file merupakan waktu eksekusi dalam milisekon yang sudah dibulatkan. Maka jika input size cukup kecil, waktu eksekusi dapat berjumlah 0 ms dikarenakan waktu sebelum dibulatkan lebih kecil dari 0,5 ms.

Nilai poin pada *sequence* yang di-*generate* dibatasi dalam range kelipatan 5 dari 5 sampai 30. Hal yang perlu diperhatikan juga adalah variable global bank yang menyimpan seluruh hasil pencarian *buffer* dalam bentuk list. Walaupun sebenarnya tidak digunakan karena setiap hasil pencarian akan langsung diperiksa dengan *sequence* pada saat terbentuk, namun variabel global ini diputuskan disimpan dalam program karena dapat membantu proses *debugging* ataupun pengecekan.

III. Installation and Requirement

Tidak ada kebutuhan khusus untuk menjalankan program ini selain bisa menjalankan program berbahasa python beserta modul-modulnya. Program dapat dijalankan sesuai dengan OS user. Pengguna dapat menggunakan *command* di bawah ini pada *command line*:

Windows : python BruteForce.py

Linux : python3 BruteForce.py

Namun pastikan *path* pada saat menjalankan program berakhiran folder src. Dikarenakan pembacaan file .txt dan penyimpanan solusi pada file menggunakan *relative path directories* pada terminal.

Untuk format input dalam program disesuaikan sama seperti spesifikasi tugas kecil yang diberikan.

IV. Test Case

Berikut adalah 6 contoh pengujian program dengan input yang berbeda-beda :

1) Test Case 1

Program menerima input dari file TC1.txt dan menyimpan solusi pada file Solusi_TC1.txt

Input :

Tucil1_13522049 > test > TC1.txt

```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

Output :

Saved file :

Tucil1_13522049 > test > Solution_TC1.txt

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

581 ms
```

Terminal :

```
PS C:\ITB\Akademik\Tingkat 2\sem 2\Stima\Tucil1_13522049\src> python -u "c:\ITB\Akademik\Tingkat 2
\sem 2\Stima\Tucil1_13522049\src\BruteForce.py"
--------------------------------------
|| BREACH PROTOCOL OPTIMAL SOLUTION ||
--------------------------------------

Do you want to load existing txt file? (y/n) y
Enter file name : TC1

Loading...

--------------
|| Solution ||
--------------
50
7A BD 7A BD 1C BD 55
1,1
1,4
3,4
3,5
6,5
6,3
1,3


581 ms
Do you want to save the solution? (y/n) y
Enter file name : Solution_TC1
PS C:\ITB\Akademik\Tingkat 2\sem 2\Stima\Tucil1_13522049\src>
```

2) Test Case 2

Program men-*generate* input sesuai dengan spesifikasi user dan menyimpan file pada Solusi_TC2.txt.

Input :

```
--------------------------------------
|| BREACH PROTOCOL OPTIMAL SOLUTION ||
--------------------------------------

Do you want to load existing txt file? (y/n) n

Requesting Information...
Number of unique token : 5
Enter 5 unique token separated by blank: 7A BD 1C E9 55

Buffer size : 7

|| Matrix size ||
Column size : 6
Row size : 6

Number of Sequence : 3
Maximum sequence size : 3

Generating...

|| Generated Matrix ||
1C BD 55 7A 55 55
55 E9 55 7A E9 7A
7A BD 7A 1C 1C 55
1C 1C 55 1C 55 1C
E9 7A 55 7A 7A 55
55 7A 7A E9 BD 7A
```

```
|| Generated Sequences ||
1C BD
30 points

BD 7A
15 points

BD 1C 1C
10 points
```

Output :

Saved file :

Tucil1_13522049 > test > Solution_TC2.txt

```
55
BD 1C 1C 1C BD 7A
2,1
2,4
4,4
4,3
2,3
2,5

570 ms
```

Terminal :

```
Loading...

--------------
|| Solution ||
--------------
55
BD 1C 1C 1C BD 7A
2,1
2,4
4,4
4,3
2,3
2,5


570 ms
Do you want to save the solution? (y/n) y
Enter file name : Solution_TC2
PS C:\ITB\Akademik\Tingkat 2\sem 2\Stima\Tucil1_13522049\src>
```

3) Test Case 3

Program menerima input dari file TC3.txt dan menyimpan solusi pada file Solusi_TC3.txt

Input :

Tucil1_13522049 > test > TC3.txt

```
3
2 5
11 22
33 44
55 66
77 88
99 00
2
99 00
15
00 88
20
```

Output :

Saved file :

Tucil1_13522049 > test > Solution_TC3.txt

```
15
11 99 00
1,1
1,5
2,5

0 ms
```

Terminal :

```
--------------------------------------
|| BREACH PROTOCOL OPTIMAL SOLUTION ||
--------------------------------------

Do you want to load existing txt file? (y/n) y
Enter file name : TC3

Loading...

--------------
|| Solution ||
--------------
15
11 99 00
1,1
1,5
2,5


0 ms
Do you want to save the solution? (y/n) y
Enter file name : Solution_TC3
PS C:\ITB\Akademik\Tingkat 2\sem 2\Stima\Tucil1_13522049\src>
```

4) Test Case 4

Program men-generate input sesuai dengan spesifikasi user dan menyimpan file pada Solusi_TC4.txt.

Input :

```
--------------------------------------
|| BREACH PROTOCOL OPTIMAL SOLUTION ||
--------------------------------------

Do you want to load existing txt file? (y/n) n

Requesting Information...
Number of unique token : 3
Enter 3 unique token separated by blank: IF TB 23

Buffer size : 2

|| Matrix size ||
Column size : 3
Row size : 3

Number of Sequence : 2
Maximum sequence size : 2

Generating...

|| Generated Matrix ||
23 23 IF
23 23 23
IF IF TB

|| Generated Sequences ||
23 IF
20 points

23 23
20 points
```

Output :

Saved file :

Tucil1_13522049 > test > Solution_TC4.txt

```
1   20
2   23 23
3   1,1
4   1,2
5
6   0 ms
```

Terminal :

```
Loading...

--------------
|| Solution ||
--------------
20
23 23
1,1
1,2


0 ms
Do you want to save the solution? (y/n) y
Enter file name : Solution_TC4
PS C:\ITB\Akademik\Tingkat 2\sem 2\Stima\Tucil1_13522049\src>
```

5) Test Case 5

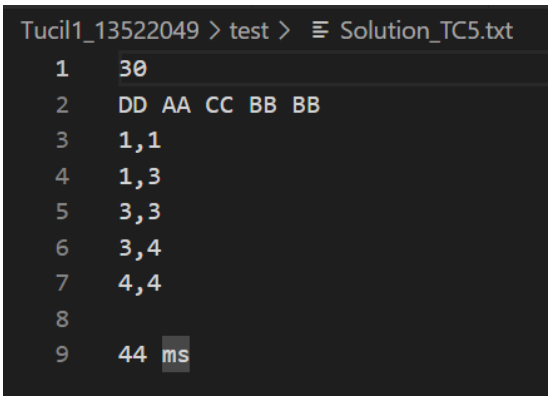
Program men-generate input sesuai dengan spesifikasi user dan menyimpan file pada Solusi_TC5.txt.

Input :

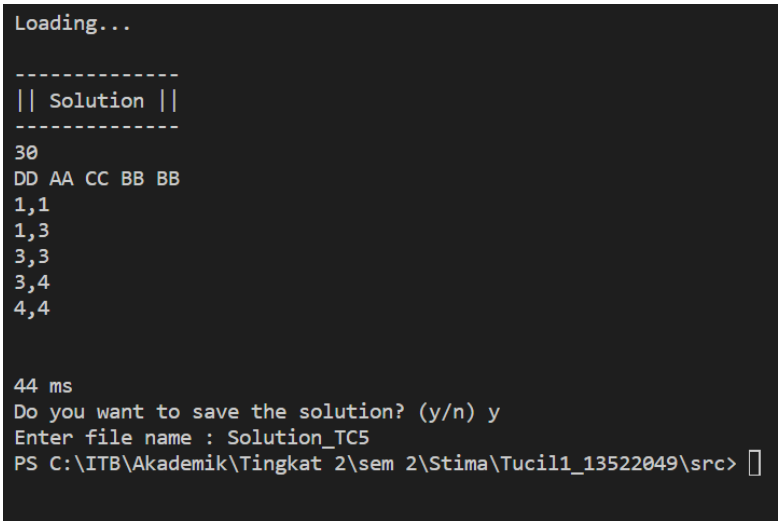
```
Requesting Information...
Number of unique token : 4
Enter 4 unique token separated by blank: AA BB CC DD EE

Buffer size : 6

|| Matrix size ||
Column size : 5
Row size : 5

Number of Sequence : 4
Maximum sequence size : 4

Generating...

|| Generated Matrix ||
DD DD BB DD AA
DD CC CC DD CC
AA AA CC CC BB
DD DD BB BB BB
AA AA DD CC DD

|| Generated Sequences ||
AA CC BB BB
30 points

BB AA DD
25 points

AA CC BB CC
10 points

CC DD CC
20 points
```

Output :

Saved file :



Terminal :



6) Test Case 6

Program men-generate input sesuai dengan spesifikasi user dan menyimpan file pada Solusi_TC6.txt.

Input :

```
--------------------------------------
|| BREACH PROTOCOL OPTIMAL SOLUTION ||
--------------------------------------

Do you want to load existing txt file? (y/n) n

Requesting Information...
Number of unique token : 6
Enter 6 unique token separated by blank: AA BB CC DD EE FF

Buffer size : 8

|| Matrix size ||
Column size : 7
Row size : 7

Number of Sequence : 4
Maximum sequence size : 5

Generating...

|| Generated Matrix ||
AA FF DD CC DD AA DD
CC EE DD AA BB DD DD
BB CC AA DD FF AA EE
BB FF AA CC FF EE AA
AA FF AA FF FF DD FF
CC CC AA AA AA DD AA
DD FF FF DD DD AA FF
```

```
|| Generated Sequences ||
CC EE CC FF EE
30 points

AA CC AA
25 points

FF AA CC BB
10 points

FF EE DD
15 points
```

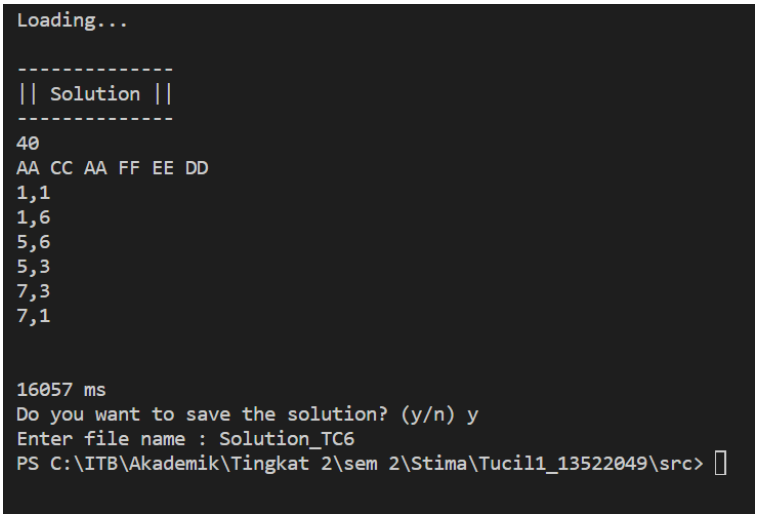
Output :

Saved file :

Tucil1_13522049 > test > Solution_TC6.txt

```
1   40
2   AA CC AA FF EE DD
3   1,1
4   1,6
5   5,6
6   5,3
7   7,3
8   7,1
9
10  16057 ms
```

Terminal :



Pada test case 6 terlihat bahwa hasil *buffer* paling optimal hanya berisi 6 token walaupun *buffer size* yang diinput sebesar 8.

Github

Link Repository : https://github.com/VansonK/Tucil1_13522049

Mohon menghubungi LINE vansonk ataupun melalui teams jika terdapat kendala dalam mengakses Github.

CheckList

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	V	
2. Program berhasil dijalankan	V	
3. Program dapat membaca masukan berkas .txt	V	
4. Program dapat menghasilkan masukan secara acak	V	
5. Solusi yang diberikan program optimal	V	
6. Program dapat menyimpan solusi dalam berkas .txt	V	
7. Program memiliki GUI		V