

Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Algoritma Brute Force



Disusun Oleh:

Aurelius Justin Philo Fanjaya / 13522020

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Daftar Pustaka

Daftar Pustaka	2
1. Algoritma Brute Force	3
2. Source program dalam bahasa Python	4
3. Tangkapan Layar input dan output	12
4. Link Repository	21
5. Lampiran	21

1. Algoritma Brute Force

Algoritma Brute Force yang saya gunakan untuk permasalahan di Tugas Kecil ini saya buat menggunakan Loop dan Fungsi Rekursif dengan penjelasan sebagai berikut.

1. Melakukan Looping pada setiap token di baris pertama.

Fungsi recursive yang saya buat memerlukan koordinat awal sebagai salah satu parameter fungsinya. Oleh sebab itu, setiap token pada baris pertama saya jadikan parameter dari setiap pemanggilan awal fungsi rekursif karena token pertama pada buffer harus dari baris pertama matrix.

2. Memanggil Fungsi Rekursif (**brute_force()**).

Fungsi rekursif **brute_force()** pada program menerima 5 buah parameter, yaitu **matrix**, **buffer**, **optimal_buffer**, **sequences**, **ukuran_buffer**. **Matrix** adalah matriks token yang akan dilakukan *brute force*. **Buffer** adalah rangkaian token yang akan dicek total poinnya. **Optimal_buffer** berisi buffer yang telah dicek dan memiliki poin paling tinggi (pada awal masukkan / *default*-nya berisi array token dan koordinat yang kosong serta memiliki 0 poin total). **Sequences** adalah beberapa rangkaian token yang akan dicocokkan dengan buffer beserta dengan *reward*-nya masing-masing. **Ukuran_buffer** adalah ukuran maksimal dari buffer.

Pada awal pemanggilan fungsi **brute_force()**, akan disimpan koordinat token terakhir dari parameter **buffer** sebagai variabel *coordinate (current coordinate)*. Kemudian, fungsi akan mengecek apakah panjang dari buffer genap atau ganjil. Jika genap, maka akan dilakukan pengecekan dengan looping secara horizontal (loop berdasarkan *matrix width*). Jika ganjil, maka akan dilakukan pengecekan dengan looping secara vertikal (loop berdasarkan *matrix height*).

Selanjutnya, tiap token pada baris/kolom yang sedang diloop akan dicek apakah koordinatnya sudah terdapat pada buffer untuk menghindari 2 token dengan koordinat yang sama berada pada buffer. Tiap token tersebut akan di-*append* ke dalam buffer dan dihitung poin totalnya berdasarkan parameter **sequences** menggunakan fungsi **countPoint()**. Jika hasilnya lebih optimal daripada **optimal_buffer** (poin lebih besar atau poin sama dengan jumlah token lebih sedikit), maka **optimal_buffer** akan diganti dengan kondisi buffer sekarang. Setelah itu, jika panjang buffer belum melebihi ukuran maksimum buffer, maka akan dipanggil fungsi **brute_force()** secara rekursif dan melanjutkan buffer dengan kondisi sekarang. Selanjutnya, buffer akan di-*pop* dan loop dapat berlanjut ke elemen selanjutnya. Setelah setiap token pada baris/kolom dicek, maka fungsi akan me-*return* **optimal_buffer**.

3. Output Algoritma Brute Force

Setelah setiap elemen pada baris pertama memanggil fungsi **brute_force()**, maka variabel **optimal_buffer** telah berisi kombinasi token paling optimal dengan constraint yang diberikan. Dari variabel **optimal_buffer**, dapat didapatkan urutan token beserta koordinatnya, serta poin total dari buffer tersebut yang kemudian akan di-*output*.

2. Source program dalam bahasa Python

a. main.py

```
import time
from random import randint
from configReader import readConfig
from pointCounter import countPoint
from bruteForce import brute_force

def main():
    # INPUT
    print('Selamat datang!')
    print('Pilih metode Input!')
    print('1. File ".txt"')
    print('2. Otomatis\n')
    pilihan = input('Pilihan : ')
    print()
    while (pilihan != '1' and pilihan != '2'):
        print("Masukkan tidak tepat! Coba lagi.")
        pilihan = input("pilihan : ")

    if (pilihan == '1'): # Baca file txt
        print('Pastikan file berada di folder "test"!')
        path = '../test/' + input('Nama file ".txt" : ')
        config = readConfig(path)
        ukuran_buffer = config['buffer_size']
        matrix_width = config['matrix_width']
        matrix_height = config['matrix_height']
        jumlah_sekuens = config['number_of_sequences']
        sequences = config['sequences']
        matrix = config['matrix']

    else: # Otomatis
        jumlah_token_unik = input('Jumlah Token Unik : ')
        while not jumlah_token_unik.isdigit():
            print("Input tidak valid!")
            jumlah_token_unik = input('Jumlah Token Unik: ')
        jumlah_token_unik = int(jumlah_token_unik)

        token_unik = input('Token : ').split()
        # Cek Token Validity
        token_valid = True
```

```

for token in token_unik:
    if len(token) != 2 or not token.isalnum():
        token_valid = False
        break
token_set = set(token_unik)
while not token_valid or len(token_unik) != jumlah_token_unik or
len(token_unik) != len(token_set):
    print('Token tidak valid, tidak unik, atau tidak sesuai
jumlah!')
    token_unik = input('Token : ').split()
    token_valid = True
    for token in token_unik:
        if len(token) != 2 or not token.isalnum():
            token_valid = False
            break
    token_set = set(token_unik)

ukuran_buffer = input('Ukuran Buffer : ')
while not ukuran_buffer.isdigit():
    print("Input tidak valid!")
    ukuran_buffer = input('Ukuran Buffer : ')
ukuran_buffer = int(ukuran_buffer)

matrix_size = input('Matrix Size : ').split()
while len(matrix_size) != 2 or not (matrix_size[0].isdigit() and
matrix_size[1].isdigit()):
    print('Input tidak valid!')
    matrix_size = input('Matrix Size : ').split()
matrix_width = int(matrix_size[0])
matrix_height = int(matrix_size[1])

jumlah_sekuens = input('Jumlah Sekuens : ')
while not jumlah_sekuens.isdigit():
    print("Input tidak valid!")
    jumlah_sekuens = input('Jumlah Sekuens : ')
jumlah_sekuens = int(jumlah_sekuens)

ukuran_maksimal_sekuens = input('Ukuran Maksimal Sekuens : ')
while not ukuran_maksimal_sekuens.isdigit():
    print("Input tidak valid!")
    ukuran_maksimal_sekuens = input('Ukuran Maksimal Sekuens : ')
ukuran_maksimal_sekuens = int(ukuran_maksimal_sekuens)

```

```

print()

# Create Random Matrix
matrix = []
for i in range(matrix_height):
    line = []
    for j in range(matrix_width):
        line.append(token_unik[randint(0, jumlah_token_unik-1)])
    matrix.append(line)

# Create Random Sequence
sequences = []
for i in range(jumlah_sekuens):
    valid = False
    while not valid:
        sekuens = []
        for j in range(randint(1, ukuran_maksimal_sekuens)):
            sekuens.append(token_unik[randint(0,
jumlah_token_unik-1)])
        valid = True # Cek apakah sequence yang digenerate unik
        for sequence in sequences:
            if (sequence[0] == sekuens):
                valid = False
                break
        sequences.append((sekuens, randint(-50, 50)))

print()

# Print Matrix
print("MATRIX")
for baris in matrix:
    for token in baris:
        print(token, end=' ')
    print()
print()

# Print Sequence
print("SEQUENCES")
for sequence in sequences:
    print("Sequence : ", end='')
    for token in sequence[0]:

```

```

        print(token, end=' ')

    print()
    print("Reward : ", sequence[1])
    print()
print()

# PROCESS
start_time = time.time()
buffer = ([],[]) # (list of tokens, list of coordinates)
optimal_buffer = ([], [], 0)

for i in range(matrix_width):
    buffer[0].append(matrix[0][i])
    buffer[1].append((0, i))
    point = countPoint(buffer[0], sequences)
    if point > optimal_buffer[2]:
        optimal_buffer = (buffer[0].copy(), buffer[1].copy(), point)
    optimal_buffer = brute_force(matrix, buffer, optimal_buffer,
sequences, ukuran_buffer)
    buffer[0].pop()
    buffer[1].pop()

# output
output = ""
output += str(optimal_buffer[2]) + "\n"
for token in optimal_buffer[0]:
    output += token + " "
output += "\n"
for coordinate in optimal_buffer[1]:
    output += str(coordinate[1]+1) + ',' + str(coordinate[0]+1) + "\n"
output += "\n"

run_time = round((time.time() - start_time) * 1000)
output += str(run_time) + " ms"

print(output)

pilihan = input("Apakah ingin menyimpan solusi? (y/n) ")
print()
while pilihan != 'y' and pilihan != 'n':
    print('Pilihan tidak valid! Ulangi masukan!')
```

```

        pilihan = input("Apakah ingin menyimpan solusi? (y/n) ")
        print()

    if pilihan == 'y':
        nama_output = input("Nama file output : ")
        f = open("../test/" + nama_output, "w")
        f.write(output)
        f.close()
        print('Output berhasil disimpan di
"../test/{}".format(nama_output))

if __name__ == "__main__":
    main()

```

b. bruteForce.py

```

from pointCounter import countPoint

def brute_force(matrix, buffer, optimal_buffer, sequences,
ukuran_buffer):
    # Algoritma bruteForce
    coordinate = buffer[1][-1]
    if (len(buffer[0]) % 2 == 0) :
        for i in range(len(matrix[0])):
            if (not (coordinate[0], i) in buffer[1]):
                buffer[0].append(matrix[coordinate[0]][i])
                buffer[1].append((coordinate[0], i))
                point = countPoint(buffer[0], sequences)
                if(point > optimal_buffer[2] or (point ==
optimal_buffer[2] and len(buffer[0]) < len(optimal_buffer[0]))):
                    optimal_buffer = (buffer[0].copy(), buffer[1].copy(),
point)

                if (len(buffer[0]) < ukuran_buffer):
                    optimal_buffer = brute_force(matrix, buffer,
optimal_buffer, sequences, ukuran_buffer)

                buffer[0].pop()
                buffer[1].pop()
    else:
        for i in range(len(matrix)):
            if (not (i, coordinate[1]) in buffer[1]):
                buffer[0].append(matrix[i][coordinate[1]])
                buffer[1].append((i, coordinate[1]))

```



```

        point = countPoint(buffer[0], sequences)
        if(point > optimal_buffer[2] or (point ==
optimal_buffer[2] and len(buffer[0]) < len(optimal_buffer[0]))):
            optimal_buffer = (buffer[0].copy(), buffer[1].copy(),
point)

            if (len(buffer[0]) < ukuran_buffer):
                optimal_buffer = brute_force(matrix, buffer,
optimal_buffer, sequences, ukuran_buffer)

            buffer[0].pop()
            buffer[1].pop()

    return optimal_buffer

```

c. pointCounter.py

```

def countPoint(buffer, sequences):
    # Menghitung jumlah point dari suatu array of token (buffer)
    berdasarkan sequences input.

    point = 0
    for sequence in sequences:
        for i in range(len(buffer)):
            j = i
            match = True
            for token in sequence[0]:
                if (j >= len(buffer)):
                    match = False
                    break

                if (token != buffer[j]):
                    match = False
                    break

                j += 1
            if (match):
                point += sequence[1]
                break

    return point

```

d. configReader.py

```

import os

def readConfig(path):

```

```

# Membaca file .txt menjadi config program.
if(os.path.isfile(path)):
    try:
        with open(path) as file:
            lines = []
            for line in file:
                lines.append(line.rstrip())
    except IOError as e:
        print("File tidak berhasil dibuka!")
        exit()
else:
    print(path, "Tidak ditemukan! Cek nama file")
    exit()

# input
try:
    if (len(lines[1].split()) != 2):
        print("File tidak sesuai format!")
        exit()

    matrix_width = int(lines[1].split()[0])
    matrix_height = int(lines[1].split()[1])
    matrix = []
    for i in range(2, 2+matrix_height):
        row = lines[i].split()
        if (len(row) != matrix_width):
            print("Jumlah kolom pada matrix tidak sesuai dengan
matrix_width!")
            quit()
        for token in row:
            if len(token) != 2 or not token.isalnum():
                print("Token terdiri atas 2 Karakter Alfanumerik!")
                quit()
        matrix.append(row)

    # input sequence
    number_of_sequences = int(lines[2+matrix_height])
    sequences = []
    for i in range(3+matrix_height, 3 + matrix_height +
2*number_of_sequences, 2):
        sequences.append(((lines[i].split()), int(lines[i+1])))

```

```
config = {  
    'buffer_size' : int(lines[0]),  
    'matrix_width' : matrix_width,  
    'matrix_height' : matrix_height,  
    'number_of_sequences' : number_of_sequences,  
    'sequences' : sequences,  
    'matrix' : matrix  
}  
except:  
    print("Isi file tidak sesuai format! Program dihentikan.")  
    exit()  
  
return config
```

3. Tangkapan Layar input dan output

No	INPUT	OUTPUT
1	<pre>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</pre> <p>tes1.txt</p>	<pre>50 7A BD 7A BD 1C BD 55 1,1 1,4 3,4 3,5 6,5 6,3 1,3 459 ms</pre> <p>solution1.txt</p>

2

```
10
5 5
AA AA BB BB BB
BB AA AA AA AA
AA BB BB AA AA
BB BB BB BB AA
AA AA BB AA AA
10
BB AA BB BB BB AA AA BB AA
50
BB AA BB BB BB BB AA
-45
AA BB BB AA AA
-20
BB AA
-20
BB BB AA BB
-10
AA AA BB
-10
BB BB
-10
AA AA BB AA BB BB BB
10
BB AA BB BB AA
-15
BB AA BB AA AA BB
-5
```

tes2.txt

```
10
BB AA BB BB BB AA AA BB AA
3,1
3,2
1,2
1,4
2,4
2,2
4,2
4,1
1,1

12425 ms
```

solution2.txt

3

```
10
7 6
BD 1C BD E9 55 BD E9
1C 7A 1C 1C 1C 55 BD
1C 1C 1C E9 55 1C 7A
7A 55 BD 7A 55 E9 55
55 7A 55 E9 55 7A BD
7A BD BD 55 1C 55 E9
4
1C BD E9
45
7A BD 1C
15
7A 1C E9 55
50
BD 7A E9 1C 1C
45
```

tes3.txt

```
110
BD 7A BD 1C BD E9 7A 1C E9 55
1,1
1,4
3,4
3,2
7,2
7,6
1,6
1,3
4,3
4,6
```

168217 ms

solution3.txt

4

```
Selamat datang!
Pilih metode Input!
1. File ".txt"
2. Otomatis

Pilihan : 2

Jumlah Token Unik : 5
Token : A1 B2 C3 D4 E5
Ukuran Buffer : 7
Matrix Size : 3 8
Jumlah Sekuens : 10
Ukuran Maksimal Sekuens : 7
```

MATRIX

```
A1 B2 B2
C3 A1 B2
E5 A1 D4
E5 B2 E5
E5 D4 E5
E5 D4 A1
C3 B2 E5
B2 B2 D4
```

SEQUENCES

```
Sequence : B2 B2 E5
Reward : -23

Sequence : D4 B2 C3 E5 E5 D4 A1
Reward : 8

Sequence : A1 B2 E5 A1 A1 E5 E5
Reward : 50

Sequence : D4 B2 C3 A1
Reward : -11

Sequence : B2 A1
Reward : 40

Sequence : C3
Reward : -32

Sequence : B2 E5 C3 D4
Reward : -11

Sequence : B2 A1 C3 A1 D4 C3 E5
Reward : -37

Sequence : B2 E5 E5
Reward : -30

Sequence : D4 E5 D4 A1
Reward : 5
```

45

B2 A1 D4 E5 D4 A1

2,1

2,3

3,3

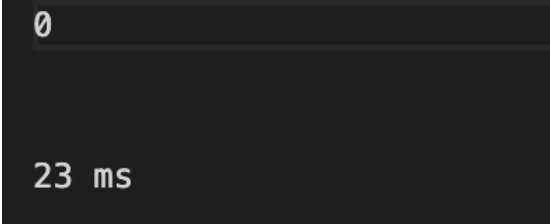
3,5

2,5

2,2

164 ms

random1.txt

5	<pre>Selamat datang! Pilih metode Input! 1. File ".txt" 2. Otomatis Pilihan : 2 Jumlah Token Unik : 5 Token : AA BB CC DD EE Ukuran Buffer : 5 Matrix Size : 5 5 Jumlah Sekuens : 5 Ukuran Maksimal Sekuens : 5 MATRIX AA CC BB CC CC AA AA EE CC DD CC CC DD CC EE DD DD EE DD BB EE EE EE CC CC SEQUENCES Sequence : EE AA DD Reward : -40 Sequence : CC AA DD BB AA Reward : 37 Sequence : EE CC BB Reward : -44 Sequence : CC Reward : -27 Sequence : EE AA BB DD BB Reward : 42</pre>	 <p>0</p> <p>23 ms</p> <p>random2.txt</p>
---	--	---

6

```
Selamat datang!  
Pilih metode Input!  
1. File ".txt"  
2. Otomatis  
  
Pilihan : 2  
  
Jumlah Token Unik : 5  
Token : ab cd ef gh ij  
Ukuran Buffer : 2  
Matrix Size : 1 5  
Jumlah Sekuens : 5  
Ukuran Maksimal Sekuens : 2
```

MATRIX

```
cd  
ab  
ab  
ab  
ef
```

SEQUENCES

```
Sequence : ij  
Reward : 27
```

```
Sequence : ab ef  
Reward : 4
```

```
Sequence : cd  
Reward : 3
```

```
Sequence : ef  
Reward : -12
```

```
Sequence : ab ij  
Reward : 8
```

3

cd

1,1

0 ms

random3.txt

7

```
Selamat datang!  
Pilih metode Input!  
1. File ".txt"  
2. Otomatis  
  
Pilihan : 2  
  
Jumlah Token Unik : 5  
Token : AA BB CC DD EE  
Ukuran Buffer : 2  
Matrix Size : 10 1  
Jumlah Sekuens : 10  
Ukuran Maksimal Sekuens : 2
```

```
MATRIX  
EE AA EE DD CC AA BB CC DD EE
```

```
SEQUENCES  
Sequence : CC  
Reward : -17
```

```
Sequence : AA  
Reward : 36
```

```
Sequence : AA CC  
Reward : -16
```

```
Sequence : AA BB  
Reward : -42
```

```
Sequence : EE CC  
Reward : 11
```

```
Sequence : EE  
Reward : 23
```

```
Sequence : BB  
Reward : -4
```

```
Sequence : CC CC  
Reward : 2
```

```
Sequence : EE DD  
Reward : 11
```

```
Sequence : CC DD  
Reward : -36
```

36

AA

2,1

0 ms

random4.txt

8

Selamat datang!
Pilih metode Input!
1. File ".txt"
2. Otomatis

Pilihan : 2

Jumlah Token Unik : 6
Token : AA BB CC DD EE FF
Ukuran Buffer : 10
Matrix Size : 6 6
Jumlah Sekuens : 10
Ukuran Maksimal Sekuens : 10

MATRIX
CC CC CC FF EE EE
EE DD AA EE DD EE
AA AA FF AA AA BB
BB AA DD CC EE AA
CC FF EE DD CC FF
EE EE CC BB EE EE

SEQUENCES
Sequence : CC DD EE
Reward : -1

Sequence : CC BB BB DD CC AA EE BB BB BB
Reward : 27

Sequence : BB
Reward : 29

Sequence : CC BB CC FF CC BB FF CC AA CC
Reward : -3

Sequence : EE
Reward : 50

Sequence : EE AA DD CC EE AA AA CC
Reward : -45

Sequence : FF BB AA FF
Reward : -33

Sequence : CC FF CC AA
Reward : -4

Sequence : AA BB FF FF FF FF BB
Reward : -3

Sequence : EE EE FF BB AA DD EE CC CC
Reward : -11

79
EE BB
6,1
6,3

126528 ms

9

```
Selamat datang!  
Pilih metode Input!  
1. File ".txt"  
2. Otomatis  
  
Pilihan : 2  
  
Jumlah Token Unik : 1  
Token : 11  
Ukuran Buffer : 5  
Matrix Size : 5 5  
Jumlah Sekuens : 5  
Ukuran Maksimal Sekuens : 5
```

MATRIX

```
11 11 11 11 11  
11 11 11 11 11  
11 11 11 11 11  
11 11 11 11 11  
11 11 11 11 11
```

SEQUENCES

```
Sequence : 11 11 11  
Reward : -5
```

```
Sequence : 11 11  
Reward : 40
```

```
Sequence : 11 11 11 11 11  
Reward : -50
```

```
Sequence : 11 11 11 11  
Reward : 26
```

```
Sequence : 11  
Reward : -4
```

57

11 11 11 11

1,1

1,2

2,2

2,1

20 ms

random6.txt

4. Link Repository

https://github.com/AureliusJustin/Tucil1_13522020

5. 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		✓