

Kelompok 7 :

Ian Felix Jonathan : 05311940000008

Shavica Ihya Qurata Ayun L : 05311940000013

Daniel Evan Yudhiputra : 05311940000016

Bryan Yehuda Mannuel : 05311940000021

Revina Rahmanisa Harjanto : 05311940000046

Laporan ETS Teknologi Multimedia Shannon-Fanno

Sebelum menjalankan kode Shannon Fanno, kami membuat sebuah class yang didalamnya terdapat berbagai method yang kami butuhkan untuk menjalankan algoritma Shannon Fanno ini

```
class Char:
    def __init__(self, name, freq) -> None:
        self._name = name
        self._freq = freq
        self._code = ""

    def __lt__(self, other):
        return True if self._freq < other.get_freq() else False

    def __eq__(self, other):
        return True if self._name == other.get_name() and self._freq ==
other.get_freq() else False

    def __str__(self):
        return "{0}\t {1}\t {2}".format(self._name, str(self._freq),
self._code)

    def __iter__(self):
        return self

    def get_name(self):
        return self._name

    def get_freq(self):
        return self._freq

    def get_code(self):
        return self._code

    def append_code(self, code):
        self._code += str(code)
```

fungsi `get_name` digunakan untuk mengambil nama list nya, `get_freq` digunakan untuk mengambil frekuensi kemunculan char dari string yang dimasukkan. `Get_code` mengambil

kode(1,0) dari setiap urutan list yang sudah dibuat. Append_code digunakan untuk memasukkan kode(0,1) ke dalam list pada urutan tertentu dalam sebuah list.

Karena kode ini ingin dijalankan dalam proses client-server, maka kami membuat 2 buah kode dengan socket programming. Kode yang pertama merupakan kode untuk server-side dan yang kedua untuk client-side.

Pada client-side, user akan memasukkan string yang ingin dicompress. Lalu akan dilakukan algoritma Shannon fanno terhadap inputan user tersebut, dan hasil nya tersebut akan dimasukkan kedalam server.

```
HEADER = 64
PORT = 5050
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)
```

Kode diatas merupakan sebuah snippet untuk memulai socket antara client-server. Dapat dilihat bahwa proses ini dijalankan pada port 5050

Setelah membuat proses socket, selanjutnya kami membuat sebuah fungsi yang akan digunakan untuk mengirimkan inputan user ke server sebagai berikut

```
def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    client.send(send_length)
    client.send(message)
    print(client.recv(2048).decode(FORMAT))
```

Fungsi ini akan dipanggil ketika user ingin mengirimkan sesuatu kedalam server, fungsi ini menerima parameter msg yang merupakan sebuah string yang akan dikirimkan oleh user tersebut.

Setelah itu kami juga membuat sebuah fungsi untuk melakukan algoritma Shannon fanno sebagai berikut

```
def shannon_fano(lst):
    middle = find_middle(lst)
    if middle is None: return
```

```

for i in lst[: middle + 1]:
    i.append_code(0)
    lst1.append(0)
shannon_fano(lst[: middle + 1])
for i in lst[middle + 1:]:
    i.append_code(1)
    lst1.append(1)
shannon_fano(lst[middle + 1:])

```

Inti dari program ini adalah membagi list yang sudah dibuat berdasarkan frekuensi kemunculannya. Ketika sudah dibagi berdasarkan frekuensi kemunculan terdekat, maka akan dilakukan fungsi rekursif Shannon fanno ini sampai Semua karakter yang dimasukkan user diproses. Pada proses membagi 2 bagian tersebut, bagian sebelah kiri akan diberikan kode 0 dan bagian sebelah kanan akan diberikan kode 1. Dalam snippet code diatas, terdapat fungsi find_middle() untuk mencari tengah dari list yang kita gunakan. Berikut adalah implementasi dari find_middle tersebut :

```

def find_middle(lst):
    if len(lst) == 1: return None
    s = k = b = 0
    for p in lst: s += p.get_freq()
    s /= 2
    for p in range(len(lst)):
        k += lst[p].get_freq()
        if k == s: return p
        elif k > s:
            j = len(lst) - 1
            while b < s:
                b += lst[j].get_freq()
                j -= 1
            return p if abs(s - k) < abs(s - b) else j
    return

```

Lalu kami membuat proses handle input dari client sebagai berikut :

```

x = ' '
while x != 'quit':
    lst = list()
    m = input('Enter 1 to enter message ')
    if m == '1':
        m = input("message ---> ")
        s = frozenset(m)
        for c in s: lst.append(Char(c, m.count(c)/len(m)))
    elif m == '2':
        i = 1
        while True:
            m = input("{} --->".format(i))
            if m == "":

```

```

        print("end")
        break
    m = m.replace(' ', '')
    lst.append(Char(m[: m.find(':')], float(m[m.find(':') + 1:]))
    i += 1
else:
    lst.append(Char('a', 0.5))
    lst.append(Char('b', 0.25))
    lst.append(Char('c', 0.098))
    lst.append(Char('d', 0.052))
    lst.append(Char('e', 0.04))
    lst.append(Char('f', 0.03))
    lst.append(Char('g', 0.019))
    lst.append(Char('h', 0.011))

```

Kami memberikan 3 buah opsi bagi user, opsi pertama yaitu ketika user hanya memasukkan string dan program akan melakukan kompresi terhadap string dari user. Opsi kedua, ketika user ingin membuat library sendiri, sedangkan opsi ketiga ketika user ingin memasukkan secara manual teks dan probabilitas dari kemunculan kodenya.

```

lst.sort(reverse=True)
shannon_fano(lst)
h = l = 0
for c in lst:
    send(str(c))
    h += c.get_freq() * log2(c.get_freq())
    l += c.get_freq() * len(c.get_code())
    print(c.get_freq())
h = abs(h)
print("h = {}".format(h))
send(str(len(m)*8/len(lst1)/h))

```

setelah memproses inputan dari client, program akan melakukan sort terhadap list yang sudah dibuat user tersebut. Setelah dilakukan sort, maka dilakukan proses algoritma shanno fanno dengan mengirimkan list dan memanggil fungsi shanno_fanno. Setelah dilakukan kompresi, maka dilakukan perulangan sepanjang list. Untuk setiap index iterasi, maka kirimkan ke server hasil kompresi nya. Pada setiap perulangan juga dilakukan perhitungan entropy dengan rumus probabilitas dikali dengan $\log_2(\text{probabilitas})$

Setelah perulangan selesai, maka dilakukan perhitungan besar ratio kompresi dengan rumus

$$\frac{\text{Panjang character sebelum kompresi} \times 8}{\text{Panjang character sesudah } X \text{ entropi}}$$

Setelah itu hasil dari perhitungan tersebut dikirimkan ke server.

Pada server, kami juga membuat fungsi socket sebagai berikut

```

HEADER = 64
PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

```

Dapat dilihat bahwa server berjalan pada port yang sama dengan client yaitu 5050.

Untuk menjalankan server, kami membuat fungsi start sebagai berikut

```

def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")

print("[STARTING] server is starting...")
start()

```

fungsi ini digunakan untuk menghandle Semua koneksi dari user yang akan terhubung dengan server

Untuk melakukan handle terhadap text yang dikirimkan oleh user, maka kami membuat sebuah fungsi handle sebagai berikut :

```

def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")

    connected = True
    while connected:
        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                connected = False
            print(f"[{addr}] {msg}")
            conn.send("\n".encode(FORMAT))

    conn.close()

```

Fungsi ini akan melakukan print pada terminal terhadap hasil kompresi dari string yang sudah dikirim oleh user serta memunculkan rasio kompresi nya.

Berikut adalah kode lengkap :

1. Client.py

```

import socket
from functools import reduce
from functools import total_ordering

from math import log2
lst1 = list()

@total_ordering
class Char:
    def __init__(self, name, freq) -> None:
        self._name = name
        self._freq = freq
        self._code = ""

    def __lt__(self, other):
        return True if self._freq < other.get_freq() else False

    def __eq__(self, other):
        return True if self._name == other.get_name() and self._freq ==
other.get_freq() else False

    def __str__(self):
        return "{0}\t {1}\t {2}".format(self._name, str(self._freq),
self._code)

    def __iter__(self):
        return self

    def get_name(self):
        return self._name

    def get_freq(self):
        return self._freq

    def get_code(self):
        return self._code

    def append_code(self, code):
        self._code += str(code)

def find_middle(lst):
    if len(lst) == 1: return None
    s = k = b = 0
    for p in lst: s += p.get_freq()
    s /= 2
    for p in range(len(lst)):
        k += lst[p].get_freq()
        if k == s: return p
        elif k > s:
            j = len(lst) - 1
            while b < s:

```

```

        b += lst[j].get_freq()
        j -= 1
    return p if abs(s - k) < abs(s - b) else j
return

def shannon_fano(lst):
    middle = find_middle(lst)
    if middle is None: return
    for i in lst[: middle + 1]:
        i.append_code(0)
        lst1.append(0)
    shannon_fano(lst[: middle + 1])
    for i in lst[middle + 1:]:
        i.append_code(1)
        lst1.append(1)
    shannon_fano(lst[middle + 1:])

HEADER = 64
PORT = 5050
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
SERVER = socket.gethostbyname(socket.gethostbyname())
ADDR = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    client.send(send_length)
    client.send(message)
    print(client.recv(2048).decode(FORMAT))

x = ' '
while x != 'quit':
    lst = list()
    m = input('Enter 1 to enter message ')
    if m == '1':
        m = input("message ---> ")
        s = frozenset(m)
        for c in s: lst.append(Char(c, m.count(c)/len(m)))
    elif m == '2':
        i = 1
        while True:
            m = input("{} --->".format(i))
            if m == "":
                print("end")
                break
            m = m.replace(' ', '')

```

```

        lst.append(Char(m[: m.find(':')], float(m[m.find(':') +
1:]))))
        i += 1
    else:
        lst.append(Char('a', 0.5))
        lst.append(Char('b', 0.25))
        lst.append(Char('c', 0.098))
        lst.append(Char('d', 0.052))
        lst.append(Char('e', 0.04))
        lst.append(Char('f', 0.03))
        lst.append(Char('g', 0.019))
        lst.append(Char('h', 0.011))

    lst.sort(reverse=True)
    shannon_fano(lst)
    h = l = 0
    for c in lst:
        send(str(c))
        # print(c)
        h += c.get_freq() * log2(c.get_freq())
        l += c.get_freq() * len(c.get_code())
        print(c.get_freq())
    h = abs(h)
    # print("H_max = {}".format(log2(len(lst))))
    print("h = {}".format(h))
    # print("l_cp = {}".format(l))
    # print("K_c.c. = {}".format(log2(len(lst))/l))
    # print("K_o.ə. = {}".format(h/l))
    # print(f"message: {lst}")
    # print(len(m))
    # print(len(m)*8/len(lst1)/h)
    # pembagian = len(lst1)/len(m)/l
    send(str(len(m)*8/len(lst1)/h))

send(DISCONNECT_MESSAGE)

```

2. Server.py

```

import socket
import threading
from functools import reduce

HEADER = 64
PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

```



```

def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")

    connected = True
    while connected:
        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                connected = False
            print(f"[{addr}] {msg}")
            conn.send("\n".encode(FORMAT))

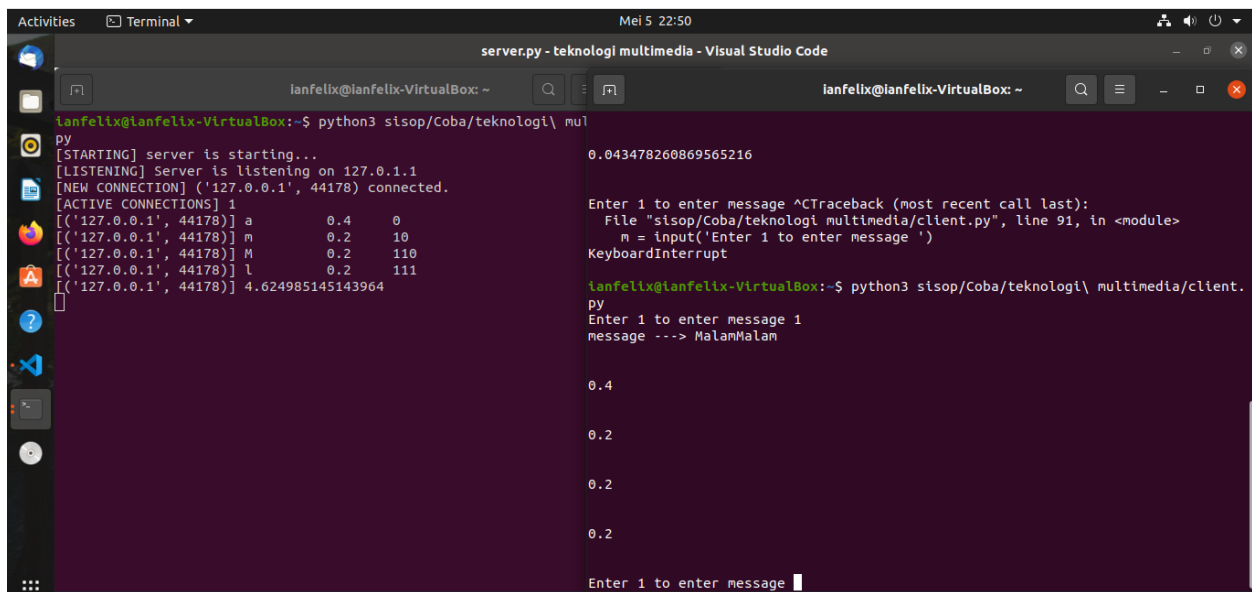
    conn.close()

def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn,
addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")

print("[STARTING] server is starting...")
start()

```

Hasil Run Program



```

Mei 5 22:50
server.py - teknologi multimedia - Visual Studio Code
ianfelix@ianfelix-VirtualBox: ~
ianfelix@ianfelix-VirtualBox:~$ python3 sisop/Coba/teknologi\ mul
py
[STARTING] server is starting...
[LISTENING] Server is listening on 127.0.1.1
[NEW CONNECTION] ('127.0.0.1', 44178) connected.
[ACTIVE CONNECTIONS] 1
[('127.0.0.1', 44178)] a 0.4 0
[('127.0.0.1', 44178)] m 0.2 10
[('127.0.0.1', 44178)] M 0.2 110
[('127.0.0.1', 44178)] l 0.2 111
[('127.0.0.1', 44178)] 4.624985145143964

Enter 1 to enter message ^CTraceback (most recent call last):
  File "sisop/Coba/teknologi multimedia/client.py", line 91, in <module>
    m = input('Enter 1 to enter message ')
KeyboardInterrupt

ianfelix@ianfelix-VirtualBox:~$ python3 sisop/Coba/teknologi\ multimedia/client.
py
Enter 1 to enter message 1
message ---> MalamMalam

0.4

0.2

0.2

0.2

Enter 1 to enter message

```