

ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÀI TẬP LỚN

Lý thuyết thông tin (Lập trình)

TRẦN QUỐC DUY

duy.tq207643@sis.hust.edu.vn

NGUYỄN QUANG TIỆP

tiiep.nq207634@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn:

TS. Trịnh Văn Chiến

Chữ ký GVHD

Khoa:

Kỹ thuật máy tính

Trường:

Công nghệ thông tin và Truyền thông

HÀ NỘI, 07/2023

TÓM TẮT NỘI DUNG BÀI TẬP LỚN

Giới thiệu vấn đề

Tìm hiểu nguyên lý của các kiến thức lý thuyết dựa vào việc sử dụng ngôn ngữ lập trình.

Hướng tiếp cận

Sử dụng ngôn ngữ lập trình Python và Java, sử dụng thuật toán phù hợp để giải quyết bài toán.

Tổng quan giải pháp

Với ngôn ngữ Python, tìm hiểu và sử dụng các thư viện hỗ trợ tốt nhất. Tính toán dựa trên cơ sở lý thuyết và đưa ra màn hình kết quả.

Với ngôn ngữ Java, tạo các hàm chức năng giải quyết các vấn đề nhỏ của bài toán, sử dụng cấu trúc cây để lưu trữ các đối tượng sử dụng.

Kết quả đạt được

Sau khi hoàn thành, hệ thống có thể đáp ứng được các yêu cầu của bài toán và đưa ra kết quả một cách chính xác.

BÀI TẬP LỚN BAO GỒM

Bài 1:

- a) Nhập vào ma trận xác suất kết hợp $P(x,y)$ có kích cỡ $M \times N$ với M và N nhập từ bàn phím. Cảnh báo nếu nhập xác suất âm, yêu cầu nhập lại.
- b) Tính và hiển thị $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, $H(X,Y)$, $H(Y) - H(Y|X)$, $I(X,Y)$.
- c) Tính $D(P(x)||P(y))$ và $D(P(y)||P(x))$.

Bài 2:

- a) Nhập vào một chuỗi không dấu có chiều dài bất kỳ, không phân biệt chữ hoa, chữ thường.
- b) Mã hóa Huffman cho chuỗi trên.
- c) Mã hóa Shannon Fano cho chuỗi trên, tính hiệu suất mã hóa, và tính dư thừa.

BẢNG PHÂN CHIA CÔNG VIỆC

| MSSV | Họ và tên | Công việc | Ngôn ngữ |
|----------|-------------------|-----------|----------|
| 20207643 | Trần Quốc Duy | Bài tập 1 | Python |
| 20207634 | Nguyễn Quang Tiệp | Bài tập 2 | Java |

MỤC LỤC

| | |
|---|----------|
| CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI..... | 1 |
| 1.1 Đặt vấn đề..... | 1 |
| 1.2 Mục tiêu và phạm vi đề tài..... | 1 |
| 1.3 Định hướng giải pháp..... | 1 |
| 1.4 Bố cục đồ án | 2 |
| CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH BÀI TOÁN | 3 |
| 2.1 Khảo sát yêu cầu bài toán | 3 |
| 2.2 Các kiến thức cần thiết | 4 |
| 2.2.1 Entropy | 4 |
| 2.2.2 Joint Entropy | 4 |
| 2.2.3 Conditional Entropy | 4 |
| 2.2.4 Quy tắc chuỗi | 5 |
| 2.2.5 Relative Entropy | 5 |
| 2.2.6 Mutual Information | 6 |
| 2.2.7 Huffman Code | 6 |
| 2.2.8 Shannon Fano Code | 6 |
| CHƯƠNG 3. CÀI ĐẶT | 8 |
| 3.1 Công nghệ sử dụng | 8 |
| 3.1.1 Ngôn ngữ Python | 8 |
| 3.1.2 Ngôn ngữ Java | 8 |
| 3.2 Cài đặt và kết quả | 8 |
| 3.2.1 Bài tập 1..... | 8 |

| | |
|---------------------------------|-----------|
| 3.2.2 Bài tập 2..... | 13 |
| CHƯƠNG 4. KẾT LUẬN | 25 |

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Để hiểu kỹ hơn về các công thức liên quan đến lượng tin riêng (Entropy), đối với sinh viên chuyên ngành Công nghệ thông tin, đưa các công thức đó vào lập trình, tạo thành một bài toán mà hệ thống có thể tự tính toán và trả kết quả, đó là một cách khá dễ dàng để sinh viên đó có thể hiểu rõ cách hoạt động của công thức và tạo hứng thú trong việc học tập và nghiên cứu.

Bài tập lớn này sẽ giải quyết hai bài toán sau:

Bài 1:

- a) Nhập vào ma trận xác suất kết hợp $P(x,y)$ có kích cỡ $M \times N$ với M và N nhập từ bàn phím. Cảnh báo nếu nhập xác suất âm, yêu cầu nhập lại.
- b) Tính và hiển thị $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, $H(X,Y)$, $H(Y) - H(Y|X)$, $I(X,Y)$.
- c) Tính $D(P(x)||P(y))$ và $D(P(y)||P(x))$.

Bài 2:

- a) Nhập vào một chuỗi không dấu có chiều dài bất kỳ, không phân biệt chữ hoa, chữ thường.
- b) Mã hóa Huffman cho chuỗi trên.
- c) Mã hóa Shannon Fano cho chuỗi trên, tính hiệu suất mã hóa, và tính dư thừa.

1.2 Mục tiêu và phạm vi đề tài

Mục tiêu chính của báo cáo này là tìm hiểu, phân tích và triển khai các kiến thức lý thuyết thông qua việc lập trình. Hệ thống được lập trình đòi hỏi phải cho ra kết quả đúng với lý thuyết.

Bài tập lớn sẽ tập trung vào việc nghiên cứu và phân tích những lý thuyết được sử dụng trong việc tính Entropy và các loại mã hóa.

1.3 Định hướng giải pháp

Đối với bài tập 1, hướng giải quyết bài toán này sẽ xoay quanh việc sử dụng các công thức tính liên quan đến lượng tin riêng (Entropy) đã học.

Đối với bài tập 2, cần sử dụng các thuật toán liên quan để giúp cho việc mã hóa được dễ dàng hơn.

1.4 Bố cục đề án

Phần còn lại của báo cáo đề án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về yêu cầu của bài toán, các kiến thức cần có để giải quyết bài toán bằng phương pháp toán học.

Trong chương 3, chúng tôi trình bày về việc sử dụng các ngôn ngữ lập trình đã nêu trên để giải quyết các bài toán đã có, đưa ra hình ảnh chạy các kết quả tương ứng.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH BÀI TOÁN

2.1 Khảo sát yêu cầu bài toán

Ở bài 1, yêu cầu của bài toán đưa ra như sau:

- a) *Nhập vào ma trận xác suất kết hợp $P(x,y)$ có kích cỡ $M \times N$ với M và N nhập từ bàn phím. Cảnh báo nếu nhập xác suất âm, yêu cầu nhập lại.*
- b) *Tính và hiển thị $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, $H(X,Y)$, $H(Y) - H(Y|X)$, $I(X,Y)$.*
- c) *Tính $D(P(x)||P(y))$ và $D(P(y)||P(x))$.*

Với câu a), hệ thống sẽ cho phép nhập số hàng và số cột được nhập từ bàn phím. Sau đó, người dùng sẽ lần lượt nhập các xác suất kết hợp $P(x,y)$ cho từng cặp (x,y) tương ứng. Nếu xác suất nhận được là số âm, hệ thống sẽ hiện ra thông báo cảnh báo và yêu cầu người dùng nhập lại.

Với câu b), hệ thống sau khi đã có ma trận xác suất kết hợp $P(x,y)$ sẽ tự động tính $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, $H(X,Y)$, $H(Y) - H(Y|X)$, $I(X,Y)$ và in kết quả ra màn hình.

Với câu c), hệ thống sẽ tiếp tục tính $D(P(x)||P(y))$ và $D(P(y)||P(x))$ và hiển thị kết quả lên màn hình.

Đến bài 2, yêu cầu như sau:

- a) *Nhập vào một chuỗi không dấu có chiều dài bất kỳ, không phân biệt chữ hoa, chữ thường.*
- b) *Mã hóa Huffman cho chuỗi trên.*
- c) *Mã hóa Shannon Fano cho chuỗi trên, tính hiệu suất mã hóa, và tính dư thừa.*

Dữ liệu được nhập vào ở câu a) sẽ được dùng để mã hóa trong 2 câu b) và c), chuỗi sau khi nhập vào cần có các xử lý chữ hoa và chữ thường để sao cho thống nhất một loại chữ.

Câu b) yêu cầu mã hóa chuỗi đã nhập vào thành mã Huffman.

Câu c) yêu cầu mã hóa chuỗi đó thành mã Shannon Fano, sau đó tính hiệu suất mã và tính dư thừa.

2.2 Các kiến thức cần thiết

2.2.1 Entropy

- Entropy (Self-information - lượng tin riêng):

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$

- Lượng thông tin trong một biến ngẫu nhiên.
- Độ không chắc chắn trung bình của một biến ngẫu nhiên.
- Cho biết không gian của biến ngẫu nhiên (bao gồm giá trị có thể xuất hiện và xác suất xuất hiện).
- Đặc tính của Entropy:
 - $H(X) \geq 0$. Nếu $H(X) = 0$ có nghĩa là không có thông tin mới.
 - Từ tính đơn điệu (monotonicity): Entropy (lượng tin riêng) tăng khi mà chiều dài bản tin tăng.

2.2.2 Joint Entropy

- Mở rộng khái niệm cho nhiều biến ngẫu nhiên. Ví dụ một cặp biến ngẫu nhiên (X, Y) .
- Joint Entropy được định nghĩa như sau:

$$H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(x, y)$$

- Đây là khái niệm mở rộng của Entropy cho một biến. Nếu thiết lập một véc-tơ: $Z = [X, Y]^T$ thì quay về khái niệm Entropy cho một biến.
- Lượng thông tin trung bình để biểu diễn 02 biến ngẫu nhiên.

2.2.3 Conditional Entropy

- Xác suất có điều kiện:

$$p(Y|X) = \frac{p(X, Y)}{p(X)}$$

- Một cặp biến ngẫu nhiên (X, Y) , conditional entropy (entropy có điều kiện) được

định nghĩa:

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X=x) = \sum_{x \in X} p(x) \left[- \sum_{y \in Y} p(y|x) \log_2 p(y|x) \right]$$

$$H(Y|X) = - \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(y|x)$$

- Lượng thông tin thêm cần để truyền Y nếu đã biết X.

2.2.4 Quy tắc chuỗi

- Quy tắc chuỗi (chain rule):

- Đối với cặp biến ngẫu nhiên (X,Y): $H(X,Y) = H(X) + H(Y|X)$
- Đối với nhiều biến ngẫu nhiên (X_1, \dots, X_n)

$$H(X_1, \dots, X_n) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \dots + H(X_n|X_1, \dots, X_{n-1})$$

- Tính chất:

$$H(X|Y) \neq H(Y|X)$$

$$H(X) - H(X|Y) = H(Y) - H(Y|X)$$

2.2.5 Relative Entropy

- Relative Entropy (Entropy tương đối):

$$D(p||q) = \sum_{x \in \tilde{X}} p(x) \log_2 \frac{p(x)}{q(x)}$$

- Tên gọi khác: Khoảng cách Kullback-Leibler.
- Đo độ không hiệu quả nếu giả sử phân bố xác suất là q trong khi phân bố xác suất đúng là p .
- Nếu chúng ta dùng phân bố q để mã hóa dữ liệu, cần số bits để biểu diễn biến ngẫu nhiên

$$H(p) + D(p||q)$$

2.2.6 Mutual Information

- Mutual Information (thông tin tương hỗ):

$$I(X, Y) = \sum_{x \in \tilde{X}} \sum_{y \in \tilde{Y}} p(x, y) \log_2 \frac{p(x, y)}{p(x)q(y)} = D(p(x, y) || p(x)q(y))$$

- Giảm tính không chắc chắn của một biến ngẫu nhiên khi biết thông tin của một biến ngẫu nhiên khác.
- Mọi quan hệ giữa thông tin tương hỗ và entropy

$$I(X, Y) = H(Y) - H(Y|X)$$

2.2.7 Huffman Code

Cho một nguồn tin và xác suất xuất hiện của mỗi ký tự, làm cách nào để thiết kế được một mã tối ưu (prefix code và ngắn nhất về mặt trung bình)

Huffman Code (Mã Huffman):

- Bước 1: Hợp nhất D ký hiệu với xác suất nhỏ nhất để tạo ra một biểu tượng mới có xác suất là tổng của D ký hiệu trên.
- Bước 2: Gán D ký hiệu trên với các số 0, 1, ..., D-1 và quay lại bước 1.
- Lặp lại bước 1, 2 đến khi tổng xác suất là 1.

2.2.8 Shannon Fano Code

- Shannon Fano Code (Mã Shannon-Fano):

- Bước 1: Các ký tự được sắp xếp theo thứ tự giảm dần của xác suất.
- Bước 2: Chia các ký tự thành hai tập X và Y với xác suất tương đương.
- Bước 3: Tập X gán nhãn 1, tập Y gán nhãn 0.
- Bước 4: Lặp lại Bước 2 và Bước 3 đến khi không chia được nữa.
- Bước 5: Ghi lại các từ mã.
- Hiệu suất mã hóa Shannon-Fano:

$$\eta = \frac{H}{\hat{H}}$$

- H là lượng tin riêng:

$$H = - \sum_{i=1}^n p_i \log_2 p_i$$

- \hat{H} được tính như sau:

$$\hat{H} = \sum_{i=1}^n p_i l_i$$

- Tính dư thừa:

$$R = 1 - \eta$$

CHƯƠNG 3. CÀI ĐẶT

3.1 Công nghệ sử dụng

3.1.1 Ngôn ngữ Python

Python là một ngôn ngữ lập trình cao cấp, có cú pháp đơn giản và dễ hiểu. Một số điểm mạnh của Python bao gồm:

Đa năng: Python được sử dụng trong nhiều lĩnh vực như phân tích dữ liệu, trí tuệ nhân tạo, web development, automation, và nhiều hơn nữa.

Cộng đồng lớn: Python có một cộng đồng lớn và phát triển mạnh mẽ. Điều này nhờ vào việc có nhiều thư viện, framework và công cụ hỗ trợ phong phú.

Dễ học và đọc mã: Cú pháp đơn giản và trực quan của Python giúp cho người mới học dễ dàng tiếp cận và hiểu code dễ dàng hơn.

3.1.2 Ngôn ngữ Java

Java là một ngôn ngữ lập trình mạnh mẽ và được sử dụng rộng rãi trong nhiều ứng dụng công nghệ. Một số lợi ích của Java bao gồm:

Độ tin cậy cao: Java được thiết kế để tạo ra các ứng dụng đáng tin cậy. Với quản lý bộ nhớ tự động và kiểm tra lỗi, mã Java ít gặp lỗi hơn và có khả năng chống lỗi tốt.

Cross-platform: Java được biên dịch thành bytecode có thể chạy trên nhiều hệ điều hành khác nhau. Điều này giúp việc phát triển ứng dụng diễn ra linh hoạt và dễ dàng.

Quản lý tài nguyên: Java được biên dịch thành bytecode có thể chạy trên nhiều hệ điều hành khác nhau. Điều này giúp việc phát triển ứng dụng diễn ra linh hoạt và dễ dàng.

3.2 Cài đặt và kết quả

3.2.1 Bài tập 1

Sử dụng ngôn ngữ Python

Câu a: Nhập vào ma trận xác suất kết hợp $P(x,y)$ có cỡ $M \times N$ với M và N nhập từ bàn phím. Cảnh báo nếu nhập xác suất âm, yêu cầu nhập lại.

Hàm xử lý:

```
def xacSuatDauVao() :
```

```

2     M = int(input("Nhap so hang M: "))
3     N = int(input("Nhap so cot N: "))
4
5     xacSuat = []
6
7     for i in range(M):
8         row = []
9         for j in range(N):
10            while True:
11                xacSuat_str = input(f"Nhap xac suat P({
12                    i}, {j}): ")
13
14                # Kiem tra neu gia tri nhap vao la so
15                thap phan
16                try:
17                    probability = float(xacSuat_str)
18                    if probability >= 0:
19                        break
20                    else:
21                        print("Xac suat khong duoc am.
22                            Vui long nhap lai.")
23                except ValueError:
24                    # Neu gia tri nhap vao khong phai
25                    la so thap phan, thu kiem tra xem
26                    co phai la dang phan so khong
27                    try:
28                        numerator, denominator = map(
29                            int, xacSuat_str.split('/'))
30                        if numerator >= 0 and
31                            denominator > 0:
32                            probability = numerator /
33                                denominator
34                            break
35                        else:
36                            print("Xac suat khong hop
37                                le, vui long nhap lai.")
38                    except ValueError:
39                        print("Gia tri khong hop le,

```

```

31         vui long nhap lai.")
32         row.append(probability)
33
34     xacSuat.append(row)
35
36     return xacSuat

```

Hàm hiển thị ma trận $P(x,y)$:

```

1 def print_maTranXacSuat(matrix):
2     print("\nMa tran ket hop P(x,y)")
3     for row in matrix:
4         print(','.join(str(prob) for prob in row))

```

Màn hình hiển thị:

```

Nhap so hang M: 3
Nhap so cot N: 3
Nhap xac suat P(0, 0): 1/16
Nhap xac suat P(0, 1): 3/8
Nhap xac suat P(0, 2): 1/16
Nhap xac suat P(1, 0): 1/16
Nhap xac suat P(1, 1): 3/16
Nhap xac suat P(1, 2): 0
Nhap xac suat P(2, 0): 0
Nhap xac suat P(2, 1): 3/16
Nhap xac suat P(2, 2): 1/16

Ma tran ket hop P(x,y)
0.0625, 0.375, 0.0625
0.0625, 0.1875, 0.0
0.0, 0.1875, 0.0625

```

Hình 3.1: Xác suất đầu vào

Câu b: Tính và hiển thị $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, $H(X,Y)$, $H(Y) - H(Y|X)$, $I(X,Y)$

Mã nguồn:

```

1 def tinhDoLechKullbackLeibler(xacSuat_x, xacSuat_y):
2     return entropy(xacSuat_x, xacSuat_y, base=2)

```

```
3
4 def main():
5     # Nhập ma tran xac suat ket hop P(x, y)
6     maTranXacSuat = xacSuatDauVao()
7     print_maTranXacSuat(maTranXacSuat)
8
9     joint_probabilities = np.array(maTranXacSuat)
10
11     # Tinh xac suat bien
12     marginal_probabilities_y = joint_probabilities.sum(
13         axis=1)
14     marginal_probabilities_x = joint_probabilities.sum(
15         axis=0)
16
17     # Tinh entropy
18     entropy_x = entropy(marginal_probabilities_x, base
19         =2)
20     entropy_y = entropy(marginal_probabilities_y, base
21         =2)
22
23     # Tinh joint entropy:
24     joint_entropy = entropy(joint_probabilities.flatten
25         (), base=2)
26
27     # Tinh conditional entropy
28     conditional_entropy_x_given_y = joint_entropy -
29         entropy_y
30     conditional_entropy_y_given_x = joint_entropy -
31         entropy_x
32
33     # Tinh mutual information = H(Y)-H(Y|X)
34     mutual_information_1 = entropy(
35         marginal_probabilities_y, base=2)+entropy(
36         marginal_probabilities_x,base=2) - joint_entropy
```

Đoạn mã hiển thị kết quả:

```
1 # Hien thi cac ket qua
2 print("\nHien thi ket qua:")
```



```

3     print(f"H(X) : {entropy_x}")
4     print(f"H(Y) : {entropy_y}")
5     print(f"H(X | Y) : {conditional_entropy_x_given_y}")
6     print(f"H(Y | X) : {conditional_entropy_y_given_x}")
7     print(f"H(X, Y) : {joint_entropy}")
8     print(f"H(Y) - H(Y | X) : {entropy_y -
          conditional_entropy_y_given_x}")
9     print(f"I(X; Y) : {mutual_information_1}")

```

Hiển thị kết quả:

```

Hien thi ket qua:
H(X): 1.061278124459133
H(Y): 1.5
H(X | Y): 0.9362781244591329
H(Y | X): 1.375
H(X, Y): 2.436278124459133
H(Y) - H(Y | X): 0.125
I(X; Y): 0.125

```

Hình 3.2: Kết quả 1b

Câu c: Tính $D(P(x)||P(y))$ và $D(P(y)||P(x))$

Mã nguồn:

```

1 # Tính D(P(x) || P(y)) và D(P(y) || P(x))
2 kl_divergence_xy = tinhDoLechKullbackLeibler(
    marginal_probabilities_x, marginal_probabilities_y)
3 kl_divergence_yx = tinhDoLechKullbackLeibler(
    marginal_probabilities_y, marginal_probabilities_x)

```

Với hàm được sử dụng:

```

1 def tinhDoLechKullbackLeibler(xacSuat_x, xacSuat_y):
2     return entropy(xacSuat_x, xacSuat_y, base=2)

```

Trong đó, hàm entropy được import từ thư viện `scipy.stats`

Đoạn mã hiển thị kết quả:

```

1 print(f"D(P(x) || P(y)) : {kl_divergence_xy}")

```

```
2 print(f"D(P(y)||P(x)): {kl_divergence_yx}")
```

Kết quả:

```
D(P(x)||P(y)): 0.8137218755408672
D(P(y)||P(x)): 0.8537593748197109
PS C:\Users\Duy Tran>
```

Hình 3.3: Kết quả 1c

3.2.2 Bài tập 2

Sử dụng ngôn ngữ Java

Câu a: Nhập vào một chuỗi ký tự không dấu có chiều dài bất kỳ, không phân biệt chữ hoa, chữ thường.

Mã nguồn:

```
1 // a) Nhập vào một chuỗi ký tự không dấu có chiều dài
    bất kỳ, không phân biệt chữ hoa, chữ thường
2 input();
3 showInput();
```

Trong đó, hàm `input()` là hàm nhập chuỗi ký tự và hàm `showInput()` để in chuỗi đó sau khi chuẩn hóa.

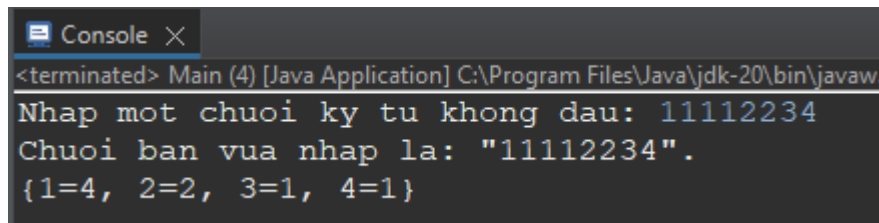
```
1 static void input() {
2     System.out.print("Nhập một chuỗi ký tự không dấu: "
3         );
4     s = sc.nextLine().toLowerCase();
5     n = s.length();
6     for (int i = 0; i < n; i++) {
7         if (map.containsKey(s.charAt(i)))
8             map.put(s.charAt(i), map.get(s.charAt(i)) +
9                 1);
10        else
11            {
12                map.put(s.charAt(i), 1);
13            }
14    }
```

```

14     }
15 }
16
17 static void showInput() {
18     System.out.println("Chuoi ban vua nhap la: \"" + s
19         + "\".");
20     System.out.println(map);
21 }

```

Màn hình hiển thị:



Hình 3.4: Nhập chuỗi ký tự

Câu b: Mã hóa Huffman cho chuỗi trên.

Mã Huffman được thiết kế qua việc xây dựng cấu trúc cây lưu trữ, được thể hiện qua lớp HuffmanNode và mã code sẽ được triển khai trong lớp HuffmanCode.

Mã nguồn:

```

1 // b) Ma hoa Huffman
2 HuffmanCode huf = new HuffmanCode(map.size(), map);
3 huf.displayResult();

```

Trong đó, lớp HuffmanCode:

```

1 public class HuffmanCode {
2
3
4     private HuffmanNode root;
5     private PriorityQueue<HuffmanNode> queue;
6     private HashMap<Character, Integer> map;
7     private int n;
8
9     public HuffmanCode(int n, HashMap<Character,
10         Integer> map) {

```

```
10         this.n = n;
11         this.map = map;
12         createQueue();
13         createTree();
14     }
15
16
17     private void createQueue() {
18         queue = new PriorityQueue<>(n, new
19             Comparator<HuffmanNode>() {
20
21             @Override
22             public int compare(HuffmanNode
23                 n1, HuffmanNode n2) {
24                 // TODO Auto-generated
25                 // method stub
26                 return n1.getValue() -
27                     n2.getValue();
28             }
29         });
30     }
31
32     private void createTree() {
33         Set<Character> keys = map.keySet();
34         Iterator<Character> i = keys.iterator();
35         ;
36         while (i.hasNext()) {
37             char c = i.next();
38             HuffmanNode n = new HuffmanNode
39                 ();
40             n.setName(c);
41             n.setValue(map.get(c));
42             n.setLeftChild(null);
43             n.setLeftChild(null);
44
45             queue.add(n);
46         }
```

```
42
43     root = null;
44
45     while (queue.size() > 1) {
46
47         HuffmanNode x = queue.peek();
48         queue.poll();
49
50         HuffmanNode y = queue.peek();
51         queue.poll();
52
53         HuffmanNode f = new HuffmanNode
54             ();
55
56         f.setValue(x.getValue() + y.
57             getValue());
58         f.setName('-');
59
60         f.setLeftChild(x);
61         f.setRightChild(y);
62
63         root = f;
64
65         queue.add(f);
66     }
67
68     private void printCode(HuffmanNode node, String
69         str) {
70         if (node.getLeftChild() == null && node
71             .getRightChild() == null && node.
72             getName() != '-') {
73             System.out.println(node.getName
74                 () + ": " + str);
75             return;
76         }
77
78         printCode(node.getLeftChild(), str + "0
```

```

        ");
74         printCode(node.getRightChild(), str + "
           1");
75     }
76
77     public void displayResult() {
78         System.out.println("\nMa hoa Huffman");
79         printCode(root, "");
80     }
81 }

```

Và lớp HuffmanNode:

```

1  public class HuffmanNode {
2      private int value;
3      private char name;
4      private HuffmanNode leftChild;
5      private HuffmanNode rightChild;
6
7      public HuffmanNode() {
8      }
9
10     public int getValue() {
11         return value;
12     }
13
14     public void setValue(int value) {
15         this.value = value;
16     }
17
18     public char getName() {
19         return name;
20     }
21
22     public void setName(char name) {
23         this.name = name;
24     }
25
26     public HuffmanNode getLeftChild() {

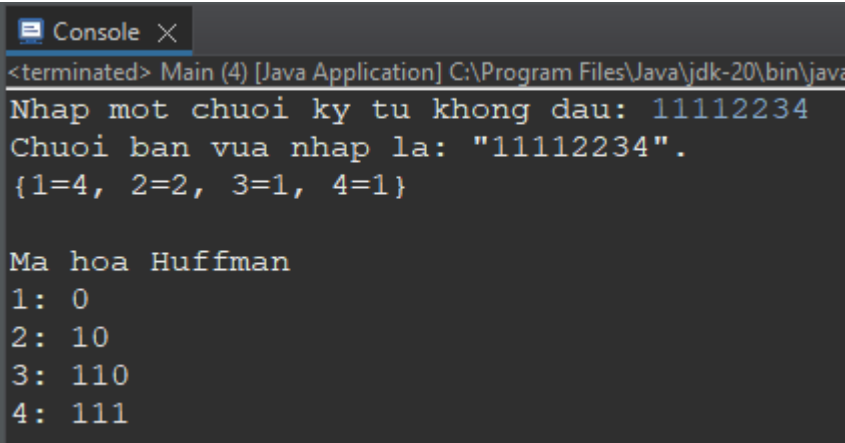
```

```

27         return leftChild;
28     }
29
30     public void setLeftChild(HuffmanNode leftChild)
31     {
32         this.leftChild = leftChild;
33     }
34
35     public HuffmanNode getRightChild() {
36         return rightChild;
37     }
38
39     public void setRightChild(HuffmanNode
40         rightChild) {
41         this.rightChild = rightChild;
42     }
43
44     public boolean isLeaf() {
45         return this.getLeftChild() == null &&
46             this.getRightChild() == null;
47     }
48 }

```

Kết quả sau khi mã hóa:



The screenshot shows a Java console window with the following output:

```

<terminated> Main (4) [Java Application] C:\Program Files\Java\jdk-20\bin\java
Nhập một chuỗi ký tự không dấu: 11112234
Chuỗi bạn vừa nhập là: "11112234".
{1=4, 2=2, 3=1, 4=1}

Mã hóa Huffman
1: 0
2: 10
3: 110
4: 111

```

Hình 3.5: Mã hóa Huffman

Câu c: Mã hóa Shannon-Fano cho chuỗi trên, tính hiệu suất mã hóa, và tính dư thừa.

Mã hóa Shannon-Fano được triển khai qua việc chia nhỏ các tập X rồi mã hóa dần dần các tập đó.

Mã nguồn:

```
1 // c) Ma hoa Shannon-Fano, tinh hieu suat ma hoa va
   tinh du thua
2 ShannonFanoCode fano = new ShannonFanoCode(n, map);
3 fano.printCode();
4 fano.showPerform();
5 fano.showRedundancy();
```

Trong đó, lớp ShannonFanoCode:

```
1 public class ShannonFanoCode {
2
3     private int n;
4     private HashMap<Character, Integer> map;
5     private ShannonFanoNode[] nodeArr;
6     private double perform;
7
8     public ShannonFanoCode(int n, HashMap<Character
9         , Integer> map) {
10         this.n = n;
11         this.map = map;
12         nodeArr = new ShannonFanoNode[map.size
13             ()];
14         createArr();
15         sortArr();
16         generateCode(0, nodeArr.length-1);
17     }
18
19     private void createArr() {
20         Set<Character> set = map.keySet();
21         Iterator<Character> i = set.iterator();
22         int index = 0;
23         while (i.hasNext()) {
24             ShannonFanoNode node = new
25                 ShannonFanoNode();
26             char name = i.next();
```



```
24         node.setName(name);
25         node.setValue(map.get(name));
26         nodeArr[index] = node;
27         index++;
28     }
29 }
30
31 private void sortArr() {
32     for (int i = 0; i < nodeArr.length - 1;
33         i++) {
34         for (int j = i + 1; j < nodeArr
35             .length; j++) {
36             if (nodeArr[i].getValue
37                 () < nodeArr[j].
38                 getValue())
39             {
40                 ShannonFanoNode
41                     tmp = new
42                     ShannonFanoNode
43                     ();
44
45                 tmp.setName(
46                     nodeArr[i].
47                     getName());
48                 tmp.setValue(
49                     nodeArr[i].
50                     getValue());
51
52                 nodeArr[i].
53                     setName(
54                     nodeArr[j].
55                     getName());
56                 nodeArr[i].
57                     setValue(
58                     nodeArr[j].
59                     getValue());
60
61                 nodeArr[j].
```

```

45         setName(tmp.
            getName());
        nodeArr[j].
            setValue(tmp.
                getValue());
46     }
47 }
48 }
49 }
50
51 private void generateCode(int l, int r) {
52     if (l == r) return;
53     int sumLeft = 0, sum = 0;
54     int mid = l;
55     for (int i = l; i <= r; i++) {
56         sum += nodeArr[i].getValue();
57     }
58     int beforeDif = 0;
59     int afterDif = sum;
60     while (mid < r) {
61         sumLeft += nodeArr[mid].
            getValue();
62         sum -= nodeArr[mid].getValue();
63         beforeDif = Math.abs(afterDif);
64         afterDif = Math.abs(sum -
            sumLeft);
65         if (afterDif >= beforeDif) {
66             break;
67         }
68         mid++;
69     }
70     for (int i = l; i < mid; i++) {
71         nodeArr[i].setCode(nodeArr[i].
            getCode() + "0");
72     }
73     for (int i = mid; i <= r; i++) {
74         nodeArr[i].setCode(nodeArr[i].
            getCode() + "1");

```

```
75         }
76         generateCode(l, mid - 1);
77         generateCode(mid, r);
78     }
79
80     public void printCode() {
81         System.out.println("\nMa hoa Shannon
82                             Fano");
83         for (int i = 0; i < nodeArr.length; i
84             ++){
85             System.out.println(nodeArr[i].
86                 getName() + ": " + nodeArr[i]
87                 .getCode());
88         }
89     }
90
91     public void showPerform() {
92         double a = 0.0, b = 0.0;
93         for (int i = 0; i < nodeArr.length; i
94             ++){
95             double p = (double)nodeArr[i].
96                 getValue() / n;
97             a -= p * (Math.log(p) / Math.
98                 log(2));
99             b += p * nodeArr[i].getCode().
100                 length();
101         }
102         perform = a/b;
103         System.out.println("\nHieu suat: " +
104             perform);
105     }
106
107     public void showRedundancy() {
108         System.out.println("\nTinh du thua R =
109                             " + (1 - perform));
110     }
111 }
```

Và lớp ShannonFanoNode:

```
1 public class ShannonFanoNode {
2
3     private char name;
4     private int value;
5     private String code;
6
7     public ShannonFanoNode() {
8         this.code = "";
9     }
10
11    public char getName() {
12        return name;
13    }
14
15    public void setName(char name) {
16        this.name = name;
17    }
18
19    public int getValue() {
20        return value;
21    }
22
23    public void setValue(int value) {
24        this.value = value;
25    }
26
27    public String getCode() {
28        return code;
29    }
30
31    public void setCode(String code) {
32        this.code = code;
33    }
34
35 }
```

Kết quả sau khi mã hóa Shannon-Fano:

```
Ma hoa Shannon Fano
1: 0
2: 10
3: 110
4: 111

Hieu suat: 1.0

Tinh du thua R = 0.0
```

Hình 3.6: Mã hóa Shannon Fano

CHƯƠNG 4. KẾT LUẬN

Trong bài báo cáo này, chúng tôi đã thực hiện việc ứng dụng việc lập trình để phân tích và tìm hiểu các công thức toán học về mặt lý thuyết. Qua việc nghiên cứu và thực hành, chúng tôi nhận thấy đây là một việc hữu hiệu để phát triển các kỹ năng như lập trình, nghiên cứu, phân tích.

Với bài đầu tiên, chúng tôi đã tìm hiểu được rằng ngôn ngữ Python có thư viện `scipy.stats` hỗ trợ việc tính toán công thức Entropy một cách dễ dàng. Do đó, bài toán được giải quyết dễ dàng và đạt được kết quả chính xác.

Ở bài thứ hai, chúng tôi đã sử dụng ngôn ngữ Java, ngôn ngữ hướng đối tượng mạnh mẽ giúp cho việc quản lý dễ dàng. Chúng tôi đã hiểu được cách triển khai từ một kiến thức lý thuyết vào việc lập trình. Việc mã hóa hai loại mã Huffman và Shannon Fano làm tăng khả năng tư duy và giúp cho việc khi nhớ kiến thức lâu dài.

Cuối cùng, tuy các đoạn mã nguồn còn nhiều hạn chế và chỉ đáp ứng được nhu cầu cơ bản trong bài tập lớn, nhưng trong tương lai, chúng tôi sẽ phát triển đoạn mã trở nên chặt chẽ và có thể kiểm thử tất cả các khả năng còn lại.