วิธีการเข้ารหัสข้อมูลอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงโดยใช้ฟังก์ชันก่อกำเนิด

ว่าที่ร้อยตรี นัฐพล ปั้นทอง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

ISBN 974-14-2506-6

ON-LINE PREFIX-FREE INFINITE ENCODING USING GENERATING FUNCTION

Acting Sub Lieutenant Natthapon Punthong

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2006

ISBN 974-14-2506-6

หัวข้อวิทยานิพนธ์	วิธีการเข้ารหัสข้อมูลอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงโดยใช้ฟังก์ชันก่อกำเนิด
โดย	ว่าที่ร้อยตรี นัฐพล ปั้นทอง
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

..... อาจารย์ที่ปรึกษา
(อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์)

..... กรรมการ
(อาจารย์ ดร.เศรษฐา ปานงาม)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อานนท์ รุ่งสว่าง)

ว่าที่ร้อยตรี นัฐพล ปั้นทอง : วิธีการเข้ารหัสข้อมูลอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงโดยใช้ฟังก์ชันก่อกำเนิด (ON-LINE PREFIX-FREE INFINITE ENCODING USING GENERATING FUNCTION) อาจารย์ที่ปรึกษา: อ. ดร.อรรถสิทธิ์ สุรฤกษ์, 39 หน้า ISBN 974-14-2506-6

งานวิจัยนี้เป็นการเสนอวิธีการเข้ารหัสข้อมูลอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงแบบใหม่โดยใช้ฟังก์ชันก่อกำเนิดร่วมกับหลักการแทนที่แบบวนซ้ำ ซึ่งในงานวิจัยนี้ได้เสนอให้ใช้โครงสร้างต้นไม้ทวิภาคแบบอนันต์ในการแสดงรหัสข้อมูลและต้นไม้รหัสจะถูกสร้างโดยฟังก์ชันก่อกำเนิด ผลทางทฤษฎีแสดงให้เห็นจริงว่าโครงสร้างของต้นไม้รหัสแบบอนันต์ที่ใช้แสดงรหัสข้อมูลที่เสนอนั้นมีคุณสมบัติของความยาวรหัสที่โครงสร้างต้นไม้ทวิภาคที่เหมาะสมที่สุดพึงมี และรหัสข้อมูลที่ได้นั้นยังคงรักษาคุณสมบัติของการไม่ซ้ำตัวเติมหน้า นอกจากนั้น ผลที่ได้จากการทดลองยังแสดงให้เห็นว่าความยาวรหัสข้อมูลที่ใช้โดยเฉลี่ยที่ได้จากอัลกอริทึมที่เสนอมีค่าใกล้เคียงกับความยาวรหัสข้อมูลที่ใช้โดยเฉลี่ยที่ได้จากอัลกอริทึมแบบคลาสสิค และในงานวิจัยนี้ยังได้ทำการศึกษาอัลกอริทึมที่เหมาะสมสำหรับการสร้างโครงสร้างต้นไม้ทวิภาคแบบอนันต์ที่มีความสัมพันธ์กับการกระจายของข้อมูลที่จะใช้ในการเข้ารหัสอีกด้วย

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2549

#4770323021 : MAJOR COMPUTER SCIENCE
KEY WORD: ON-LINE INFINITE ENCODING/ PREFIX-FREE ENCODING/ INFINITE HUFFMAN CODES/ RECURSIVE REPLACEMENT METHOD

NATTHAPON PUNTHONG: ON-LINE PREFIX-FREE INFINITE ENCODING USING GENERATING FUNCTION. THESIS ADVISOR: ATHASIT SURARERKS, Ph.D., 39 pp. ISBN 974-14-2506-6.

This research proposes a novel on-line prefix-free infinite encoding algorithm using generating function combining with a recursive replacement method. The infinite binary tree can be generated by functions. The theoretical result demonstrates that the tree also satisfies the code length property of an optimal binary tree. The obtained codes preserve the prefix-free property. Moreover, the experimental results show that the average code length obtained from our algorithm is close to the result of classical algorithms. The research also focused on the algorithmic approach to construct an infinite binary tree that relates to the distribution of the input data.

Department Computer Engineering Student's signature Natthapon Punthong
Field of study Computer Science Advisor's signature
Academic year 2006

กิตติกรรมประกาศ

ข้าพเจ้าใคร่ขอกราบขอบพระคุณอาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ที่กรุณาแนะนำให้แนวคิด ความรู้ คำปรึกษา ความช่วยเหลือต่างๆ ตลอดจนดูแลการทำวิทยานิพนธ์ของข้าพเจ้าจนสำเร็จลุล่วงลงได้ด้วยดี

ขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา อาจารย์ ดร. เศรษฐา ปานงาม และผู้ช่วยศาตราจารย์ ดร. อานนท์ รุ่งสว่าง ซึ่งเป็นคณะกรรมการสอบวิทยานิพนธ์ ที่ได้สละเวลาและให้คำแนะนำต่างๆ ที่เป็นประโยชน์อย่างยิ่งต่อการจัดทำวิทยานิพนธ์ฉบับนี้

ขอขอบคุณ อาจารย์ทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาให้ข้าพเจ้า รวมถึงชี้แนะสิ่งดีๆ ตลอดเวลาที่ข้าพเจ้าได้ศึกษาเล่าเรียนระดับมหาบัณฑิต

สุดท้ายนี้ ข้าพเจ้าใคร่ขอกราบขอบพระคุณบิดา มารดาของข้าพเจ้า ที่คอยให้กำลังใจและสนับสนุนด้านการเงินแก่ข้าพเจ้า และขอขอบคุณเพื่อนๆ ทุกคน ที่ให้ความช่วยเหลือในทุกๆ ด้านจนสามารถทำวิทยานิพนธ์ฉบับนี้ได้สำเร็จลุล่วง

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเข้ารหัสเป็นปัญหาที่สำคัญปัญหาหนึ่งที่ได้มีการนำไปใช้กับงานวิจัยในหลายๆ ด้าน ตัวอย่างเช่น วิธีการในการบีบอัดคงรายละเอียดภาพหรือการเข้ารหัสแบบรัน-เลง (run-length encoding) [1] ได้ถูกเสนอไว้โดยใช้แนวคิดที่รหัสข้อมูลที่ออกไปแต่ละตัวมีความยาวแปรผันได้ (variable length) อัลกอริทึมในการเข้ารหัสที่เป็นที่รู้จักกันดีได้เสนอไว้โดยฮัฟฟ์แมน (Huffman) [2] ในปี ค.ศ. 1952 อัลกอริทึมนี้ได้ใช้วิธีการของโครงสร้างต้นไม้รหัสทวิภาค (binary tree structure) สำหรับการเข้ารหัสข้อมูล (ตัวอักษร) ในระหว่างกระบวนการทำงาน ได้มีการใช้เทคนิคการสร้างต้นไม้รหัสทวิภาคจากล่างขึ้นบน (bottom-up technique) นอกจากนี้อัลกอริทึมของฮัฟฟ์แมน ยังได้มีการนำไปพัฒนาต่อในอีกหลายๆ ด้าน [3]

เป็นที่รู้กันดีว่าอัลกอริทึมของฮัฟฟ์แมนนั้นได้เป็นที่รู้จักและมีการใช้งานกันในวงกว้าง แต่กระบวนการทำงานนั้นจำเป็นที่จะต้องทราบข้อมูลทั้งหมดและความถี่หรือความน่าจะเป็นของข้อมูลแต่ละตัวก่อนการเข้ารหัส นั่นคือข้อมูลทั้งหมดต้องผ่านการอ่านมาก่อนหนึ่งรอบก่อนที่อัลกอริทึมจะเริ่ม

การเข้ารหัสแบบเชื่อมตรง (on-line encoding) หมายถึงการที่มีข้อมูลเข้าผ่านเข้ามาในส่วนของการเข้ารหัสจะถูกอ่านเพียงครั้งเดียวทีละหนึ่งข้อมูลเข้า (ตัวอักษร) และทำการผลิตข้อมูลออกหรือรหัส (code) ออกมาทันที อาจเรียกได้ว่าเป็นปัญหาในการเข้ารหัสแบบการอ่านครั้งเดียว (one pass) ซึ่งวิธีการนี้จะเป็นประโยชน์อย่างมากสำหรับการเข้ารหัสที่ข้อมูลเข้ามีจำนวนเป็นอนันต์ การเข้ารหัสแบบเชื่อมตรงได้ถูกพิสูจน์ใน [4] ว่า ข้อมูลเข้าที่มีจำนวนเป็นอนันต์ จะมีข้อมูลออกที่เหมาะสมที่ไม่ซ้ำตัวเติมหน้า (prefix-free) แบบเดียวกับข้อมูลออกที่ได้จากอัลกอริทึมของฮัฟฟ์แมน สิ่งที่เห็นได้ชัดเจนจากอัลกอริทึมของฮัฟฟ์แมนคือ อัลกอริทึมนี้ไม่สามารถนำไปใช้กับปัญหาในการเข้ารหัสข้อมูลแบบเชื่อมตรงได้

ในการแก้ปัญหาการเข้ารหัสข้อมูล โครงสร้างต้นไม้รหัสได้ถูกทำการศึกษาโดยนักวิจัยหลายๆ คน โดยส่วนใหญ่จะสนใจโครงสร้างต้นไม้รหัสที่เหมาะสมแบบไม่ซ้ำตัวเติมหน้า (optimal prefix-free tree structure) [5, 6, 7, 8] ในกรณีที่ข้อมูลเข้ามีเป็นจำนวนอนันต์ ผู้วิจัยจึงสนใจโครงสร้างต้นไม้รหัสอนันต์ที่เหมาะสมแบบไม่ซ้ำตัวเติมหน้า (optimal infinite prefix-free tree structure) เสนอโดยกอลิน (Golin) และมา (Ma) ในปี ค.ศ. 2004 [9] นอกจากนี้ เลดลอว์ (Laidlaw) [10] ได้เสนออัลกอริทึมสำหรับการสร้างรหัส สำหรับข้อมูลเข้าที่มีจำนวนเป็นอนันต์โดยใช้วิธีหลักการแทนที่แบบวนซ้ำ (recursive replacement method)

ในปี ค.ศ. 1977 ลิมเพล (Lempel) และซิฟ (Ziv) [11] ได้เสนอวิธีการที่เรียกว่า ลิมเพล-ซิฟ อัลกอริทึม (Lempel-Ziv algorithm) โดยมีหลักการในการเข้ารหัสที่ใช้พจนานุกรมเป็นพื้นฐาน (dictionary-based) ผลที่ได้ทำให้ขนาดของข้อมูลออกหรือรหัสมีขนาดเล็กลง และได้มีการปรับปรุงและศึกษาวิจัยอัลกอริทึมนี้เรื่อยมา [11, 12, 13, 14]

ในงานวิจัยนี้ผู้วิจัยสนใจปัญหาของการเข้ารหัสแบบเชื่อมตรงที่ไม่ซ้ำตัวเติมหน้าด้วยเทคนิคแบบบนลงล่าง (top-down technique) ซึ่งเป็นการผันกลับของอัลกอริทึมของฮัฟฟ์แมน จะถูกนำมาสร้างข้อมูลออกที่เป็นแบบไม่ซ้ำตัวเติมหน้า โดยใช้โครงสร้างต้นไม้รหัสที่ไม่มีบัพใบ (terminal node หรือ leaf node) เป็นบัพพ่อแม่ (parent node) ของบัพอื่น ที่โครงสร้างต้นไม้รหัสดังกล่าวสร้างด้วยฟังก์ชันก่อกำเนิดร่วมกับวิธีหลักการแทนที่แบบวนซ้ำ

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อออกแบบอัลกอริทึมในการเข้ารหัสข้อมูลแบบเชื่อมตรงที่ข้อมูลเข้าสามารถมีได้เป็นจำนวนอนันต์โดยใช้ฟังก์ชันก่อกำเนิดซึ่งเหมาะสมกับการกระจายของข้อมูลที่อยู่ในรูปโค้งปกติ และสามารถทำงานได้อย่างรวดเร็ว

1.3 ขอบเขตของงานวิจัย

1) ในงานวิจัยนี้ผู้วิจัยใช้ข้อมูลเข้าในการทดสอบที่เป็นพยัญชนะในภาษาอังกฤษ (A-Z)
2) วิธีนี้ถูกวัดเปรียบเทียบกับเอนโทรปีของข้อมูล วิธีการเข้ารหัสของฮัฟฟ์แมน วิธีการเข้ารหัสของลิมเพล-ซิฟ (LZW และ LZC)
3) ค่าที่ใช้ในการวัดเปรียบเทียบประสิทธิภาพคือ ค่าความยาวของข้อมูลออกโดยเฉลี่ยจำนวนเท่าของเอนโทรปี

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1) ศึกษาวิธีการเข้ารหัสที่ใช้กันอยู่ในปัจจุบัน
2) ออกแบบอัลกอริทึมการเข้ารหัสข้อมูลวิธีใหม่โดยเน้นการทำงานแบบเชื่อมตรงได้และสามารถทำงานได้เร็ว
3) วิเคราะห์และพัฒนาฟังก์ชันก่อกำเนิดที่ใช้ในการสร้างโครงสร้างต้นไม้รหัสทวิภาคที่ใช้กับอัลกอริทึมการเข้ารหัสข้อมูลแบบเชื่อมตรงในทางทฤษฎี
4) ทดสอบวิธีการที่นำเสนอ
5) เปรียบเทียบผลที่ได้จากการทดลอง
6) วิเคราะห์ผลการทดลอง

7) สรุปผลการวิจัย พร้อมข้อเสนอแนะ และจัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย

1) ได้วิธีการเข้ารหัสข้อมูลเข้าแบบเชื่อมตรงที่สามารถทำงานได้กับข้อมูลที่มีจำนวนเป็นอนันต์ ใช้หน่วยความจำน้อยและรวดเร็ว
2) สามารถนำไปประยุกต์ใช้กับการลดขนาดของข้อมูลออกและการส่งข้อมูล
3) สามารถประยุกต์หลักวิธีการเข้ารหัสข้อมูลเข้าแบบเชื่อมตรงวิธีใหม่นี้ได้กับข้อมูลในลักษณะอื่นๆได้

1.6 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการ

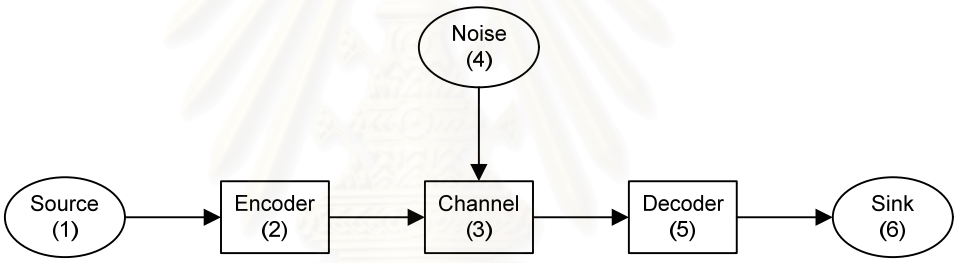
1) “Online Encoding Algorithm for Infinite Set” โดย นัฐพล ปั้นทอง และอรรถสิทธิ์ สุรฤกษ์ ในงานประชุมทางวิชาการ 9th National Computer Science and Engineering Conference (NCSEC2006) ณ มหาวิทยาลัยหอการค้าไทย กรุงเทพฯ ประเทศไทย ระหว่างวันที่ 27-28 ตุลาคม พ.ศ. 2548
2) “Online Prefix-Free Encoding Algorithm” โดย นัฐพล ปั้นทอง และอรรถสิทธิ์ สุรฤกษ์ ในงานประชุมวิชาการ 3rd IASTED International Conference on Signal Processing, Pattern Recognition, and Applications (SPPRA2006) ณ เมืองอินซ์บรัค ประเทศออสเตรีย ระหว่างวันที่ 15-17 กุมภาพันธ์ พ.ศ. 2549
3) “A Novel Approach for On-line Encoding Algorithm Using A Generating Function” โดย นัฐพล ปั้นทอง และอรรถสิทธิ์ สุรฤกษ์ ในงานประชุมวิชาการ IEEE International symposium on Intelligent Signal Processing and Communication Systems (IEEE-ISPACS 2006) ณ เมืองทอตโตริ ประเทศญี่ปุ่น ระหว่างวันที่ 12-15 ธันวาคม พ.ศ. 2549

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ ผู้วิจัยขอกล่าวถึงความรู้พื้นฐาน ทฤษฎีและงานวิจัยที่เกี่ยวข้องกับงานวิจัยนี้ เริ่มด้วยแนวความคิดของแบบจำลองของระบบสัญญาณ ข้อมูลสารสนเทศ เอนโทรปีของข้อมูล อัลกอริทึมที่ทำการเข้ารหัสข้อมูลที่มีความยาวแปรผันได้ ตัวอย่างเช่น อัลกอริทึมของฮัฟฟ์แมน และอัลกอริทึมของลิมเพล-ซิฟ ซึ่งเป็นอัลกอริทึมที่ผู้วิจัยสนใจจะถูกอธิบายรวมอยู่ในบทนี้ด้วย นอกจากนั้น แนวคิดของหลักการการสร้างต้นไม้รหัสทวิภาคแบบอนันต์โดยใช้ฟังก์ชันก่อกำเนิดร่วมกับหลักการแทนที่แบบวนซ้ำจะถูกกล่าวถึงเป็นลำดับสุดท้ายของบทนี้

2.1 แบบจำลองของระบบสัญญาณ



รูปที่ 2.1 มาตราฐานระบบสัญญาณ

(1) แหล่งส่งข้อมูล
(2) ตัวเข้ารหัสของแหล่งส่งข้อมูล
(3) ช่องสัญญาณ
(4) สัญญาณรบกวนเมื่อสัญญาณอยู่ในช่องสัญญาณ
(5) ตัวถอดรหัสของแหล่งรับข้อมูลและข้อมูลที่ถอดออกมาได้ต้องเหมือนกับข้อมูลที่ทางแหล่งส่งข้อมูลต้องการส่งมา
(6) แหล่งรับข้อมูล

กล่องการเข้ารหัสหรือตัวเข้ารหัส (encoder) และกล่องการถอดรหัสหรือตัวถอดรหัส (decoder) ในกล่องการเข้ารหัส จะทำการเข้ารหัสข้อมูลที่มาจากแหล่งส่งข้อมูล โดยข้อดีของโครงสร้างนี้คือสามารถทำการเข้ารหัสข้อมูลได้แบบต่อเนื่อง (data stream) ข้อมูลที่ถูกเข้ารหัสแล้วจะถูกส่งไปที่ช่องสัญญาณ (channel) เพื่อส่งต่อไปยังแหล่งรับข้อมูลโดยจะต้องผ่านการถอดรหัสที่ตัวถอดรหัสข้อมูล ตัวถอดรหัสข้อมูลจะมีกระบวนการทำงานผันกลับกับตัวเข้ารหัส

ข้อมูลและข้อมูลที่ได้ทำการถอดรหัสออกมาแล้วนั้นจะต้องได้เป็นข้อมูลที่เหมือนกันกับตัวดั้งเดิมที่ถูกส่งมาจากแหล่งส่งข้อมูล

2.2 ข้อมูลสารสนเทศ

กำหนดให้แหล่งข้อมูลมีตัวอักษรจำนวน q ตัว แทนด้วยสัญสักษณ์

$$s_1, s_2, s_3, s_4 ..., s_q$$

และความน่าจะเป็น

$$p(s_1) = p_1, p(s_2) = p_2, p(s_3) = p_3, p(s_4) = p_4, ..., p(s_q) = p_q$$

ตามลำดับ ถ้าค่าความถี่หรือความน่าจะเป็นของข้อมูลของตัวอักษรแต่ละตัวมีค่าแตกต่างกันมากๆ แล้ว ตัวอักษรที่มีค่าความถี่หรือความน่าจะเป็นที่มีค่าต่ำ จะถือว่าเป็นข้อมูลที่มีค่าสารสนเทศมาก แต่ในทางกลับกันสำหรับข้อมูลตัวอักษรที่มีสาระน้อยๆ ค่าความถี่หรือความน่าจะเป็นของตัวอักษรนั้นๆ ก็จะมีค่ามาก จากงานวิจัยของชานอน (Shanon) ในปี ค.ศ. 1948 [15] ได้เสนอว่า สาระหรือความสำคัญของข้อมูลนั้นจะเป็นส่วนกลับกันกับค่าความถี่หรือความน่าจะเป็นที่ตัวอักษรนั้นปรากฏซึ่งอธิบายด้วยฟังก์ชัน

$$I(p) = -\log p = \log \frac{1}{p} \tag{2.1}$$

2.3 เอนโทรปี

วิทยาการทางด้านการบีบอัดข้อมูลเกิดขึ้นในปี ค.ศ. 1948 โดยงานวิจัยของชานอน [15] ได้อธิบายถึงการแจงสาระที่มีอยู่ในข้อมูลที่ปรากฏ โดยดูจากค่าความน่าจะเป็น p ที่ปรากฏขึ้น ซึ่งค่าความมีสาระของข้อมูลแต่ละตัวนั้นมีค่าเท่ากับ $-\log_2 p$ บิต กำหนดให้ค่าความมีสาระของข้อมูลตัวที่ s_i ใดๆ นั้น สามารถหาได้จากฟังก์ชัน $I(s_i)$ และนอกจากนั้นค่าความมีสาระของข้อมูลตัวที่ s_i โดยเฉลี่ยหาได้จาก

$$p_i I(s_i) = p_i \log_2 \frac{1}{p_i} \tag{2.2}$$

เมื่อ p_i คือค่าความน่าจะเป็นของข้อมูลตัวอักษรตัวที่ s_i

จากสมการ (2.2) ค่าความมีสาระของข้อมูลโดยเฉลี่ยของข้อมูลตัวอักษร s_i ทุกตัว (เอนโทรปี) คือ

$$\sum_{i=1}^{q} p_i \log_2 \frac{1}{p_i} \tag{2.3}$$

2.4 การเข้ารหัสข้อมูลที่มีความยาวแปรผันได้

การเข้ารหัสของข้อมูลที่ความยาวแปรผันได้หมายถึง การเข้ารหัสที่ความยาวของรหัสไม่จำเป็นต้องเท่ากันทุกรหัส ข้อดีของการเข้ารหัสข้อมูลที่ความยาวแปรผันได้นั้นคือ สามารถลดขนาดของการเข้ารหัสได้ ซึ่งจะใช้จำนวนสัญญาณ (จำนวนบิต) โดยเฉลี่ยต่ำกว่า เนื่องจากรหัสของข้อมูลแต่ละตัวไม่จำเป็นต้องมีความยาวเท่ากัน ทั้งนี้ขึ้นอยู่กับค่าความถี่หรือความน่าจะเป็นของข้อมูลที่จะทำการเข้ารหัส โดยทั่วไปถ้าข้อมูลทุกตัวมีค่าความถี่หรือความน่าจะเป็นแตกต่างกัน ด้วยความแตกต่างของค่าความถี่หรือความน่าจะเป็นนี้จะเป็นปัจจัยสำคัญในการเข้ารหัสข้อมูลที่มีความยาวแปรผันได้ โดยให้ข้อมูลที่มีความถี่หรือความน่าจะเป็นในการใช้มากมีรหัสที่สั้นกว่าและข้อมูลที่มีการใช้น้อยให้มีรหัสที่ยาวกว่า แต่ว่าปัญหาสำคัญของการเข้ารหัสข้อมูลที่แปรผันได้คือ เนื่องจากการที่มีความยาวของรหัสแต่ละตัวไม่เท่ากัน ที่แหล่งรับข้อมูลจะทราบได้อย่างไรว่าแหล่งส่งข้อมูลนั้นส่งรหัสของข้อมูลอะไรมาให้ (รหัสที่ส่งมามีความกำกวมเกิดขึ้น) ดังนั้นรหัสต่างๆ ก็จะต้องมีคุณสมบัติที่เรียกว่า การถอดรหัสที่เป็นไปได้อย่างเดียว

2.4.1 การถอดรหัสที่เป็นไปได้อย่างเดียว

ในที่นี้จะขอกล่าวว่า ข้อมูลเข้าที่ต้องการเข้ารหัสหรือข้อมูลเข้านี้จะหมายถึงตัวอักษรในภาษาอังกฤษ ส่วนข้อมูลออกคือ รหัสของตัวอักษรภาษาอังกฤษที่ใช้ในการส่ง (อยู่ในรูปของระบบของเลขฐานสองคือ 0 กับ 1) กำหนดให้ข้อมูลตัวอักษรก่อนทำการส่งมี q ตัวคือ s_1, s_2, s_3, ..., s_q

คุณสมบัติแรกที่ข้อมูลออกจะต้องมีคือ เมื่อไปถึงแหล่งรับข้อมูลแล้วจะต้องถอดรหัสออกมาได้เป็นอย่างเดียวเท่านั้น (uniquely decodable) ตัวอย่างเช่น กำหนดให้ข้อมูลตัวอักษร S มีทั้งหมด 4 ตัวที่ทำการเข้ารหัสแล้วอยู่ในรูปของเลขฐานสองคือ

$$s_1 = 0$$
$$s_2 = 01$$
$$s_3 = 11$$
$$s_4 = 00$$

เมื่อแหล่งรับข้อมูลได้รับข้อมูล 0011 สามารถถอดรหัสออกมาได้เป็นสองแบบ

$$0011 = \begin{cases} s_4, s_3 \\ s_1, s_1, s_3 \end{cases}$$

ดังนั้นข้อมูลออกนั้นไม่เป็นข้อมูลที่สามารถถอดรหัสได้เพียงอย่างเดียว การที่เราจะทำให้ข้อมูลออกสามารถถอดรหัสได้เพียงอย่างเดียวนั้น สามารถทำได้โดยการกำหนดรหัสของข้อมูลออกให้เป็นดังหนึ่งในตัวอย่างนี้

$$s_1 = 10$$
$$s_2 = 01$$
$$s_3 = 11$$
$$s_4 = 00$$

วิธีการเข้ารหัสที่การถอดรหัสสามารถตีความได้อย่างเดียวมีหลายวิธี วิธีหนึ่งที่เป็นที่นิยมคือ รหัสทุกรหัสต้องมีคุณสมบัติไม่ซ้ำตัวเติมหน้าซึ่งจะกล่าวโดยละเอียดในหัวข้อต่อไป

2.4.2 รหัสฮัฟฟ์แมน

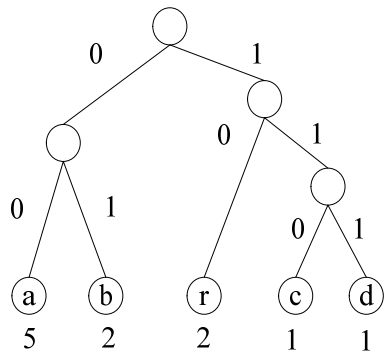
เมื่อข้อมูลเข้ามีความถี่หรือความน่าจะเป็นที่แตกต่างกัน การที่จะใช้ค่าความถี่หรือความน่าจะเป็นช่วยในการเข้ารหัสนั้น สิ่งที่ต้องการก็คือข้อมูลตัวอักษรที่มีความถี่หรือความน่าจะเป็นในการใช้มากก็ควรจะมีข้อมูลออกหรือรหัสที่สั้น ถ้าความน่าจะเป็นของข้อมูลตัวอักษรที่ i-th คือ p_i และความยาวของรหัส (ความยาวของจำนวนเลขโดดบนระบบเลขฐานสอง) คือ l_i แล้วค่าความยาวของรหัสเฉลี่ยคือ

$$L_{av} = \sum_{i=1}^{q} p_i l_i \qquad (2.4)$$

โดยที่รหัสทุกตัวที่ถูกสร้างขึ้นมาจะต้องรักษาคุณสมบัติของการไม่ซ้ำตัวเติมหน้า คือสำหรับรหัสสองรหัสใดๆ ที่แตกต่างกัน รหัสหนึ่งในนั้นจะต้องไม่เป็นตัวเติมหน้าของอีกตัวที่เหลือ

ในปี ค.ศ. 1952 เดวิด ฮัฟฟ์แมน [2] เสนอวิธีการเข้ารหัสที่เรียกว่า รหัสฮัฟฟ์แมน ฮัฟฟ์แมนได้แก้ปัญหาการสร้างความยาว (จำนวนบิต) ของรหัส ที่เหมาะสมได้ที่สามารถนำไปประยุกต์ใช้ในการบีบอัดข้อมูล

อัลกอริทึมของฮัฟฟ์แมนใช้เทคนิคแบบล่างขึ้นบนในการสร้างต้นไม้รหัสทวิภาคสำหรับการเข้ารหัสข้อมูลเพื่อให้ได้ค่าความยาวของรหัสเฉลี่ยต่ำสุด การทำงานของอัลกอริทึมเริ่มต้นที่ต้องทราบข้อมูลที่ต้องการเข้ารหัสทั้งหมดและค่าความถี่หรือความน่าจะเป็นของข้อมูลแต่ละตัว ทำการสร้างต้นไม้รหัสทวิภาคในการเข้ารหัสจากข้อมูลที่มีความถี่หรือความน่าจะเป็นน้อยสุดสองตัว ในการจัดรูปแบบให้เป็นต้นไม้รหัสทวิภาคย่อยๆ ที่รากของต้นไม้ย่อยนั้นๆ จะพิจารณาเป็นข้อมูลตัวใหม่ที่มีค่าความถี่หรือความน่าจะเป็นคือผลรวมของบัพลูก (child node) ของมัน กระบวนการจะทำซ้ำไปเรื่อยๆ จนกระทั่งข้อมูลทุกตัวถูกรวมเข้าไปอยู่ในต้นไม้รหัสทวิภาคต้นเดียวดังรูปที่ 2.2 เมื่อกำหนดให้ข้อมูลลำดับตัวอักษร abracadabra และตัวอย่างของรหัสของ c คือ 110 เป็นต้น



รหัสฮัฟฟ์แมน
a : 00
b : 01
r : 10
c : 110
d : 111

รูปที่ 2.2 ตัวอย่างอัลกอริทึมของฮัฟฟ์แมนที่ทำงานกับลำดับข้อมูล abracadabra

ตารางที่ 2.1 แสดงการสร้างต้นไม้ที่เหมาะสมโดยใช้อัลกอริทึมของฮัฟฟ์แมนที่ทำงานกับข้อมูลลำดับตัวอักษร abracadabra ส่วนในตารางที่ 2.2 แสดงการกำหนดรหัสของข้อมูลให้กับต้นไม้ที่เหมาะสม

ตาราง 2.1 การสร้างโครงสร้างต้นไม้ที่เหมาะสมด้วยวิธีการของฮัฟฟ์แมน

s_i	p_i				
a	5/11	5/11	5/11	5/11	11/11
b	2/11	2/11	2/11	6/11	
c	1/11	2/11	4/11		
d	1/11				
r	2/11	2/11			

ตารางที่ 2.2 การกำหนดรหัสของข้อมูลให้กับโครงสร้างต้นไม้ที่เหมาะสม

s_i	c_i	p_i							
a	0 ←	5/11	0	5/11	0	5/11	0	5/11	0 11/11
b	10 ←	2/11	10	2/11	10	2/11	10	6/11	1
c	1100 ←	1/11	1100	2/11	110	4/11	11		
d	1101 ←	1/11	1101						
r	111 ←	2/11	111	2/11	111				

2.5 การเข้ารหัสข้อมูลที่ใช้หลักการของพจนานุกรม

ลักษณะของการบีบอัดข้อมูลโดยใช้หลักการของพจนานุกรม (dictionary) นั้น สายอักษร (string) ที่เก็บอยู่ในพจนานุกรม สามารถมีความยาวได้หลากหลายขึ้นกับหน่วยความจำที่ใช้ในการเก็บสายอักษร สายอักษรที่ถูกจัดเก็บอยู่ในพจนานุกรมจะแสดงออกมาเป็นข้อมูลออกของการบีบอัดสายอักษรนั้นๆ ในรูปของดัชนีที่ชี้สายอักษรนั้นอยู่

2.5.1 อัลกอริทึมของลิมเพล-ซิฟ

อัลกอริทึมหนึ่งที่เป็นที่นิยมใช้ในการบีบอัดข้อมูลคือ วิธีการเข้ารหัสของลิมเพลและซิฟ (ลิมเพล-ซิฟอัลกอริทึม) แบบแรกมีชื่อเรียกว่า แอลแซด77 (LZ77) เสนอไว้ในปี ค.ศ. 1977 [11] โดยอัลกอริทึมนี้คือ ระหว่างที่กระบวนการบีบอัดข้อมูลกำลังทำงานอยู่ก็จะมีการสร้างพจนานุกรม เก็บรหัส การสร้างรหัสจะสร้างให้กับสายอักษรที่มีสายอักษรที่ปรากฏก่อนหน้าเป็นส่วนเติมหน้าที่ยาวที่สุด รายละเอียดของอัลกอริทึมแอลแซด77 อธิบายได้ดังนี้

Algorithm: LZ77

```
while (lookAheadBuffer not empty) do
    get a reference (position, length) to longest match
    if (length > 0)
        output (position, length, next symbol)
        shift the window length +1 positions along
    else
        output (0, 0, first symbol in the lookahead buffer)
        shift the window 1 character along
    endif
enddo
```

จากตารางที่ 2.3 ถ้ากำหนดให้มีข้อมูลที่ต้องการส่งเป็น abracadabra ก่อนเริ่มต้นการทำงานต้องกำหนดขนาดของบัฟเฟอร์ที่ใช้สำหรับการเข้ารหัส ในที่นี้ให้ขนาดของบัฟเฟอร์เท่ากับสิบสอง โดยแบ่งเป็นสองส่วน ส่วนแรกเป็นหน้าต่างสำหรับอ่านข้อมูลเข้าที่มีขนาดเท่ากับสี่ (ตำแหน่งที่ขีดเส้นใต้ในตารางที่ 2.3) และส่วนที่เก็บข้อมูลที่ทำการเข้ารหัสแล้วให้มีขนาดเท่ากับแปด เมื่อเริ่มทำการเข้ารหัสจะต้องมีการกำหนดค่าเริ่มต้นของบัฟเฟอร์ดังนี้ ส่วนแรกหน้าต่างสำหรับอ่านข้อมูลเข้าที่มีขนาดสี่อ่านข้อมูลมาสี่ตัวอักษร (abra) ส่วนที่สองเป็นส่วนที่เก็บข้อมูลที่เข้ารหัสแล้วจะต้องถูกเติมให้เต็มด้วยข้อมูลตัวแรกสุดในหน้าต่างคือ a จะได้บัฟเฟอร์ที่กำหนดค่าเริ่มต้นแล้วเป็น aaaaaaaaabra หลังจากทำการกำหนดค่าเริ่มต้นแล้วจึง

เริ่มทำการเข้ารหัสโดยเริ่มอ่านข้อมูลในหน้าต่างจากตัวแรกสุดก่อนคือ a ทีละตัว แล้วนำ a ไปตรวจสอบกับส่วนเก็บข้อมูลที่ทำการเข้ารหัสแล้ว จากตัวอย่างเราพบว่ามี a อยู่จึงทำการอ่านข้อมูลลำดับถัดไปในหน้าต่างคือ b แล้วนำ ab ไปตรวจสอบกับส่วนเก็บข้อมูลที่ทำการเข้ารหัสแล้ว (ส่วนที่สอง) ต่อไปอีก ปรากฏว่าไม่พบ ab จึงสร้างรหัสออกมาเป็น <8, 1, b> 8 คือเลขที่แสดงถึงตำแหน่งในบัฟเฟอร์ที่มีตัวซ้ำ, 1 คือความยาวของข้อมูลที่ซ้ำ (a) และ b คือตัวอักษรที่ตามหลังจากตัวที่มีการซ้ำ หลังจากนั้นทำการเลื่อนหน้าต่างออกไปจนถึงตัวที่อ่านมาเป็นตัวสุดท้ายได้เป็น aaaaaaabraca และทำการวนซ้ำเริ่มกระบวนเข้ารหัสใหม่

ตารางที่ 2.3 ตัวอย่างอัลกอริทึมแอลแซด77 ที่ทำงานกับลำดับข้อมูล abracadabra

Remaining sequence	Buffer	Longest prefix	Code
abracadabra	aaaaaaaaabra	a	<8, 1, b>
racadabra	aaaaaaabraca		<1, 0, r>
acadabra	aaaaaabracad	a	<6, 1, c>
adabra	aaaabracadab	a	<4, 1, d>
abra	aabracadabra	abra	<2, 4, □>

อัลกอริทึมแบบที่สองถูกพัฒนาโดยลิมเพลและซิฟคือ แอลแซด78 (LZ78) [16] รหัสที่ได้จากอัลกอริทึมนี้ประกอบด้วยสองส่วน ส่วนที่หนึ่งคือดัชนีที่อ้างอิงถึงข้อมูลที่เหมือนกันที่ยาวที่สุดในพจนานุกรม และส่วนที่สองคือตัวแรกสุดที่ไม่เหมือนตัวใดๆ ในพจนานุกรม ข้อด้อยของแอลแซด78 อยู่ที่ข้อมูลในพจนานุกรม ขยายใหญ่โดยที่ไม่มีที่สิ้นสุด เหตุนี้เองทำให้มีการพัฒนาอัลกอริทึมนี้ในหลากหลายวิธี จากหลายๆ งานวิจัยที่เกี่ยวข้องกับแอลแซด78 ก็มี แอลแซดดับเบิ้ลยู (LZW) ที่พัฒนาโดยเวล์ช (Welch) [17] เป็นงานวิจัยที่น่าสนใจและแอลแซดดับเบิ้ลยูนี้ยังถูกนำไปใช้ในโปรแกรมบีบอัดของระบบยูนิกซ์

จากตารางที่ 2.4 ทำการเข้ารหัสข้อมูล abracadabra เริ่มต้นการเข้ารหัสด้วยพจนานุกรมที่ว่าง พจนานุกรมจะถูกสร้างขึ้นในระหว่างการเข้ารหัสข้อมูลที่พบข้อมูลปรากฏเป็นครั้งแรก การเข้ารหัสแอลแซด78 อัลกอริทึมจะทำการหาสายอักษรที่ซ้ำกันกับที่มีอยู่ในพจนานุกรมที่มีความยาวที่สุด โดยรหัสที่ได้ในตำแหน่งแรกจะแสดงถึงดัชนีที่พบสายอักษรที่ซ้ำกันกับที่มีอยู่ในพจนานุกรมที่มีความยาวที่สุดและตำแหน่งที่สองคือตัวอักษรลำดับถัดมา ทำการวนซ้ำกระบวนการเข้ารหัสใหม่โดยเริ่มอ่านข้อมูลในลำดับต่อไป

ตารางที่ 2.4 ตัวอย่างอัลกอริทึมแอลแซด78 ที่ทำงานกับลำดับข้อมูล abracadabra

Compressor		Dictionary	
Remaining sequence	Code	Index	Sequence
abracadabra	<0, a>	1	a
bracadabra	<0, b>	2	b
racadabra	<0, r>	3	r
acadabra	<1, c>	4	ac
adabra	<1, d>	5	ad
abra	<1, b>	6	ab
ra	<3, a>	7	ra

ในระหว่างการเข้ารหัสพจนานุกรมสามารถขยายขนาดได้อย่างไม่จำกัด นั่นคือดัชนีจะมีค่ามากขึ้นและต้องการจำนวนบิตมากขึ้นในการเข้ารหัส ปัญหาดังกล่าวจึงได้มีการทำการปรับปรุงอัลกอริทึมของลิมเพล-ซิฟในเวอร์ชันต่างๆ ตามมา

2.5.2 อัลกอริทึมที่ปรับปรุงมาจากลิมเพล-ซิฟ

แอลแซดดับเบิ้ลยู เป็นอัลกอริทึมที่ได้รับการปรับปรุงมาจากอัลกอริทึมแอลแซด78 เสนอโดยเทอรี่ เวล์ช ในปี ค.ศ. 1984 [17] โดยอัลกอริทึมที่ได้รับการปรับปรุงใหม่นั้นยังคงเป็นที่ใช้งานในด้านการบีบอัดข้อมูลกันอย่างกว้างขวางเพราะว่าความง่ายและความรวดเร็วในการทำงานของอัลกอริทึม ซึ่งกระบวนการทำงานของแอลแซดดับเบิ้ลยูนั้นไม่ได้ทำการส่งรหัสของข้อมูลตัวอักษรที่ไม่เหมือนออกไปอย่างชัดเจนเหมือนกับแอลแซด78 รหัสที่ได้จากแอลแซดดับเบิ้ลยูจะเป็น ค่าดัชนีที่อ้างอิงถึงข้อมูลในพจนานุกรม โดยมีความยาวคงตัว อีกอัลกอริทึมหนึ่งที่ดัดแปรจากแอลแซดดับเบิ้ลยูคือแอลแซดซี (LZC) [18] กระบวนการทำงานของอัลกอริทึมแอลแซดซีนั้นจะมีการทำงานเหมือนกับแอลแซดดับเบิ้ลยูแต่ต่างกันตรงที่รหัสที่ได้นั้นจะเป็นดัชนีที่อ้างอิงถึงข้อมูลในพจนานุกรมที่มีความยาวแปรผันได้

2.6 การสร้างโครงสร้างต้นไม้ทวิภาคแบบอนันต์โดยใช้ฟังก์ชันก่อกำเนิด

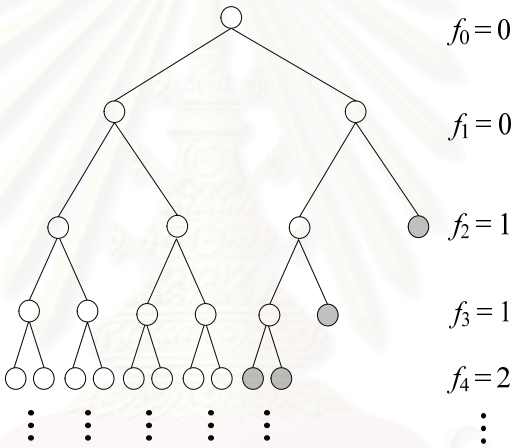
ในส่วนนี้ได้ทำการศึกษาในปัญหาที่เกี่ยวกับการการสร้างต้นไม้ทวิภาคแบบอนันต์ โดยผู้วิจัยสนใจในการสร้างต้นไม้ทวิภาคแบบอนันต์ที่บัพใบ (leaf node หรือ terminal node) สามารถเกิดขึ้นได้ที่ตำแหน่งใดก็ได้ในโครงสร้างต้นไม้ ซึ่งงานวิจัยของเลดลอว์ [10] ได้เสนอวิธีการเข้ารหัสข้อมูลตัวอักษรทีละตัวที่มีจำนวนเป็นเซตอนันต์โดยอาศัยหลักการในการสร้างโครงสร้างต้นไม้รหัสแบบไม่จำกัด (infinite tree) ที่จำนวนบัพใบในแต่ละระดับจะถูกกำหนดด้วย

ฟังก์ชันก่อกำเนิด (generating function) วิธีการดังกล่าวเรียกว่าวิธีการแทนที่แบบวนซ้ำ (recursive replacement method)

ฟังก์ชันก่อกำเนิด $G(x)$ โดย $G(0)$ ต้องมีค่าเป็น 0 และในแต่ละระดับในของการสร้างต้นไม้ทวิภาค บัพถูกแบ่งออกได้เป็น 2 ส่วน ส่วนแรกเรียกว่าส่วนที่เป็นบัพใบ (leaf part หรือ terminal part) ส่วนที่สองเรียกว่าส่วนที่ไม่เป็นส่วนสุดท้าย (non-terminal part หรือ generating part) ตัวอย่างเช่น ลำดับของฟิโบนักชีถูกสร้างได้โดยฟังก์ชัน

$$f_L = f_{L-1} + f_{L-2} \tag{2.5}$$

เมื่อ $f_0 = 0, f_1 = 0, f_2 = 1$ และจำนวนบัพใบที่ระดับที่ L คือ f_L ดังแสดงในรูปที่ 2.3 (นั่นคือ ลำดับของฟิโบนักชี {0, 0, 1, 1, 2, 3, 5, 8, 13, …})



รูปที่ 2.3 ต้นไม้รหัสทวิภาคที่สร้างโดยการแทนที่แบบวนซ้ำของลำดับฟิโบนักชี

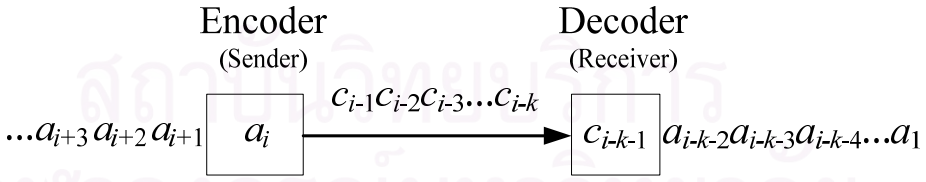
บทที่ 3

การออกแบบอัลกอริทึมการเข้ารหัสข้อมูลอนันต์ที่ไม่ซ้ำตัวเติมหน้า แบบเชื่อมตรงโดยใช้ฟังก์ชันก่อกำเนิด

ผู้วิจัยสนใจปัญหาในการเข้ารหัสแบบเชื่อมตรงที่สามารถรองรับการทำงานกับข้อมูลที่มีจำนวนเป็นอนันต์ได้ โดยจะใช้โครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ที่สร้างจากฟังก์ชันก่อกำเนิดด้วยหลักการแทนที่แบบวนซ้ำ ซึ่งในส่วนนี้ผู้วิจัยจะเริ่มต้นด้วยการอธิบายการเข้ารหัสแบบเชื่อมตรง โครงสร้างของต้นไม้รหัสทวิภาคแบบอนันต์ในการเข้ารหัสแบบเชื่อมตรงและวิธีการสร้างต้นไม้รหัสทวิภาคดังกล่าว โดยสามารถแบ่งออกได้เป็นขั้นตอนดังนี้

3.1 การเข้ารหัสข้อมูลแบบเชื่อมตรง

กำหนดให้ข้อมูลเข้ามีลักษณะเป็นลำดับของตัวอักษร อัลกอริทึมการเข้ารหัสแบบเชื่อมตรงจะทำงานเป็นลำดับโดยเริ่มต้นจากตัวอักษรตัวแรกทำการอ่านข้อมูลเข้าตัวอักษรทีละตัวอักษร (character-by-character) นั่นคือการทำงานไม่ได้พิจารณาข้อมูลเข้าทั้งหมดในทีเดียว การเข้ารหัสของตัวอักษรในลำดับถัดมาจะพิจารณาจากผลลัพธ์และข้อมูลเข้าทั้งหมดในลำดับก่อนหน้านี้ และการถอดรหัสก็จะทำงานในลักษณะเดียวกัน การเข้ารหัสแบบเชื่อมตรงสามารถอธิบายดังแสดงในรูปที่ 3.1 เมื่อกำหนดให้ข้อมูลเข้าเป็น $a_1a_2a_3...$ ตามลำดับ ในขณะที่ a_i กำลังถูกกำหนดรหัส a_1 ถึง a_{i-1} ได้มีการเข้ารหัสเป็น c_1 ถึง c_{i-1} เรียบร้อยแล้วและถูกส่งไปให้ผู้รับตามลำดับ ผู้รับเมื่อได้รับรหัสแล้วจะทำการถอดรหัสโดยในรูปรหัส c_{i-k-1} กำลังถูกถอดรหัสนั่นคือ c_1 ถึง c_{i-k-2} ได้ถูกถอดรหัสเรียบร้อยแล้ว ผลรับที่ได้จะเป็น a_1 ถึง a_{i-k-2} ตามลำดับ



รูปที่ 3.1 การทำงานของการเข้ารหัสและถอดรหัสแบบเชื่อมตรง

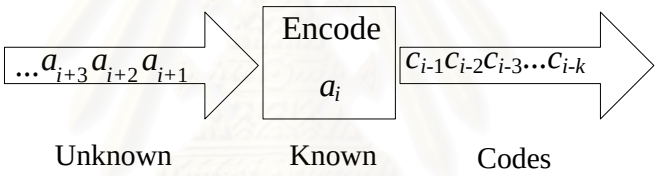
3.2 จำนวนรูปแบบตัวอักษรเป็นอนันต์

ในกระบวนการทำงานของอัลกอริทึมแบบเชื่อมตรง ข้อมูลเข้าจะถูกพิจารณาเป็นลำดับ ดังนั้นในระหว่างการเข้ารหัส ข้อมูลเข้าที่ยังไม่ผ่านการเข้ารหัสจะเป็นข้อมูลที่เรายังไม่ทราบ ทั้งนี้ถ้าในกรณีที่จำนวนรูปแบบตัวอักษรเป็นอนันต์ ในระหว่างการทำงานก็อาจมีตัวอักษรใหม่ปรากฏขึ้นได้ตลอดเวลาทำให้ไม่สามารถระบุจำนวนตัวอักษรที่แตกต่างกันทั้งหมดได้อย่าง

ถูกต้อง ดังนั้นวิธีการเข้ารหัสจะต้องรองรับการกำหนดรหัสได้ไม่จำกัด นอกจากนี้จำนวนรูปแบบตัวอักษรทั้งหมดเป็นปัจจัยสำคัญต่อการเข้ารหัสอย่างมีประสิทธิภาพ ทั้งนี้ในงานวิจัยนี้ผู้วิจัยเลือกใช้โครงสร้างต้นไม้รหัสทวิภาคเป็นโครงสร้างของการเข้ารหัส ต้นไม้ดังกล่าวจึงต้องมีความสามารถในการเข้ารหัสได้แบบไม่จำกัด

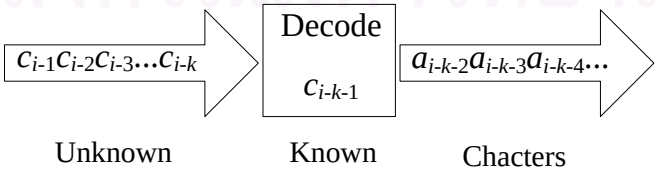
ข้อมูลตัวอักษรเข้าทั้งหมดในงานวิจัยนี้จึงถูกแยกพิจารณาออกเป็นสองกลุ่มคือ กลุ่มที่ปรากฏแล้วและกลุ่มที่ยังไม่เคยเกิดขึ้นมาก่อนสามารถกำหนดได้ดังนิยามต่อไปนี้

นิยาม 3.1 *กำหนดให้* Σ *เป็นชุดตัวอักษรเข้าและกำหนดให้ลำดับของข้อมูลเข้าเป็น* $a_1a_2a_3...a_n...$ *โดยที่* a_i *เป็นอักษรใน* Σ *กระบวนการเข้ารหัสข้อมูลจะทำงานกับข้อมูลทีละตัว* $a_1, a_2, a_3, ..., a_n, ...$ *ตามลำดับ จะกล่าวว่าข้อมูลลำดับที่* a_i *เป็นข้อมูลที่ทราบค่า* (known) *ก็ต่อเมื่อ ข้อมูลลำดับที่* a_i *เป็นข้อมูลที่ทำการเข้ารหัสแล้ว ส่วนลำดับข้อมูลเข้าที่เหลือ* $a_{i+1}a_{i+2}a_{i+3}...a_n...$ *จะถูกพิจารณาเป็นข้อมูลที่ไม่ทราบค่า* (unknown)



รูปที่ 3.2 แผนภาพการเข้ารหัสข้อมูล

รูปที่ 3.2 แสดงการเข้ารหัสข้อมูลเพื่อให้ได้รหัสออกมา ในส่วนของข้อมูลเข้าที่เหลือที่จะถูกพิจารณาเป็นข้อมูลที่ไม่ทราบค่าไป ในทำนองเดียวกันการถอดรหัสก็จะพิจารณารหัสทีละรหัสและทำการถอดรหัสเพื่อให้ได้ข้อมูลอักษรที่เหมือนกันกับต้นฉบับที่ได้ทำการเข้ารหัสมา และในส่วนของรหัสที่ยังไม่ได้ถูกถอดรหัสก็จะถูกพิจารณาเป็นรหัสที่ไม่ทราบค่าเช่นกัน ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 แผนภาพการถอดรหัสข้อมูล

3.3 คุณสมบัติของต้นไม้ที่เหมาะสมที่สุด

เนื่องจากต้นไม้ทวิภาคถูกนำมาใช้เป็นโครงสร้างในการเข้ารหัสและเราทราบมาแล้วจากการเข้ารหัสของฮัฟฟ์แมนว่า รหัสของข้อมูลจะถูกกำหนดอยู่ที่บัพใบของโครงสร้างต้นไม้ทวิภาค ซึ่งตำแหน่งของใบในโครงสร้างต้นไม้จะมีผลต่อความยาวของรหัส การเข้ารหัสที่มีประสิทธิภาพปัจจัยหนึ่งที่เราพิจารณาก็คือทำอย่างไรให้รหัสที่ได้มีความยาวรวมน้อยที่สุดดังนั้นในการเข้ารหัสที่ดีจะต้องมีการพิจารณาถึงโครงสร้างต้นไม้ที่เหมาะสมที่สุดเพื่อให้ได้ความยาวรวมของรหัสต่ำสุด

อนึ่งการพิจารณาความยาวรวมของรหัสต่ำสุดนั้น ในการทำงานแบบเชื่อมตรงเราไม่สามารถรับประกันได้ว่าความยาวรวมต่ำสุดมีค่าเป็นเท่าใดทั้งนี้เนื่องจากว่า ในระหว่างการทำงานอาจเกิดตัวอักษรใหม่ขึ้นได้ตลอดเวลาและเราไม่สามารถบอกได้ว่าตัวอักษรใดมีปริมาณมากน้อยแค่ไหนซึ่งจากอัลกอริทึมของฮัฟฟ์แมนแสดงให้เห็นว่า ความถี่ของการปรากฏของตัวอักษรก็มีผลต่อการเข้ารหัสและความยาวของรหัสด้วย ดังนั้นในงานวิจัยนี้จึงพิจารณาเลือกใช้โครงสร้างต้นไม้ที่มีคุณสมบัติใกล้เคียงกับโครงสร้างต้นไม้ที่เหมาะสมที่สุดที่ทำให้ค่าความยาวเฉลี่ยของรหัสที่ใช้สำหรับข้อมูลแต่ละตัวให้มีค่าต่ำที่สุดซึ่งกำหนดได้ดังสมการ (3.1)

$$\min L_{av} = \sum_{all\ i} p_i l_i \qquad (3.1)$$

เมื่อกำหนดให้ p_i คือค่าความน่าจะเป็นของการปรากฏของตัวอักษรที่ i และ l_i คือความยาวของรหัสของตัวอักษรที่ i

การสร้างโครงสร้างต้นไม้ให้ใกล้เคียงกับโครงสร้างต้นไม้ที่เหมาะสมที่สุดนั้น เราพบว่าต้นไม้ที่สร้างควรมีคุณสมบัติที่ใกล้เคียงกับต้นไม้ที่เหมาะสมที่สุด โดยเราพบว่าต้นไม้ที่เหมาะสมที่สุดของฮัฟฟ์แมนนั้นมีคุณสมบัติของความยาวรหัส กำหนดได้ดังนิยาม 3.2

นิยาม 3.2 *กำหนดให้ Σ เป็นชุดตัวอักษรและให้ x_i และ x_j เป็นตัวอักษรใดๆ ใน Σ กำหนดให้ c_i และ c_j เป็นรหัสของ x_i และ x_j ตามลำดับโดยที่ p_i และ p_j คือค่าความน่าจะเป็นของการปรากฏของ x_i และ x_j ตามลำดับ จะกล่าวว่าต้นไม้ที่ใช้ในการเข้ารหัสมีคุณสมบัติของความยาวรหัส* (code length property) *ก็ต่อเมื่อ ถ้า p_i มากกว่าหรือเท่ากับ p_j แล้วความยาวของรหัส c_i จะต้องสั้นกว่าหรือเท่ากับความยาวของรหัส c_j*

ในบทตั้งที่จะกล่าวต่อไปนี้แสดงให้เห็นว่าต้นไม้ที่เหมาะสมที่สุดนั้นจะมีคุณสมบัติของความยาวรหัสด้วย

บทตั้ง 3.1 *กำหนดให้ S เป็นต้นไม้ที่เหมาะสมที่สุดสำหรับการเข้ารหัสข้อมูล ให้ c_i และ c_j เป็นรหัสที่สร้างได้โดย S และ p_i กับ p_j คือค่าความน่าจะเป็นของรหัสตามลำดับ ถ้า p_i มากกว่าหรือเท่ากับ p_j แล้วความยาวของรหัส c_i จะต้องสั้นกว่าหรือเท่ากับความยาวของรหัส c_j*

พิสูจน์บทตั้ง 3.1

วิธีการพิสูจน์บทตั้ง 3.1 จะต้องแสดงว่า ถ้า p_i มากกว่าหรือเท่ากับ p_j แล้วความยาวของรหัส c_i จะต้องสั้นกว่าหรือเท่ากับความยาวของรหัส c_j

สมมติให้ S เป็นต้นไม้ที่เหมาะสมที่สุดสำหรับการเข้ารหัสข้อมูล และ c_i กับ c_j เป็นรหัสที่สร้างได้จาก S ที่สอดคล้องกับข้อมูลอักษร a_i และ a_j กำหนดให้ p_i มากกว่าหรือเท่ากับ p_j ต้องแสดงให้เห็นจริงว่า

ความยาวของรหัส c_i จะต้องสั้นกว่าหรือเท่ากับความยาวของรหัส c_j

โดยวิธีการพิสูจน์แบบขัดแย้ง กำหนดให้ ความยาวของรหัส c_i ยาวกว่าความยาวของรหัส c_j เมื่อ S เป็นต้นไม้ที่เหมาะสมที่สุดสำหรับการเข้ารหัสข้อมูล ดังนั้นค่าความยาวเฉลี่ยของรหัสต่ำสุดคือ

$$\sum_{k=1}^{n} length(c_k)p_k \tag{3.2}$$

ให้ $n = length(c_i)$ และ $m = length(c_j)$ แล้วความยาวเฉลี่ยของรหัสที่ต่ำสุดสามารถเขียนใหม่ได้เป็น

$$\sum_{k=1,k\neq i,k\neq j}^{n} length(c_k)\,p_k + np_i + mp_j \tag{3.3}$$

เพราะว่า p_i มีค่ามากกว่า p_j ทำให้ข้อมูลอักษร a_j และ a_i จึงมีรหัสเป็น c_i และ c_j ตามลำดับ ดังนั้นความยาวเฉลี่ยของรหัสจึงมีค่าเท่ากับ

$$\sum_{k=1,k\neq i,k\neq j}^{n} length(c_k)\,p_k + mp_i + np_j \tag{3.4}$$

เห็นได้ชัดว่า ค่าความยาวเฉลี่ยของรหัสที่ได้จากสมการที่ (3.4) มีค่าน้อยกว่า ค่าที่ได้จากสมการที่ (3.3) เกิดข้อขัดแย้ง ดังนั้นทำให้บทตั้ง 3.1 เป็นจริง ■

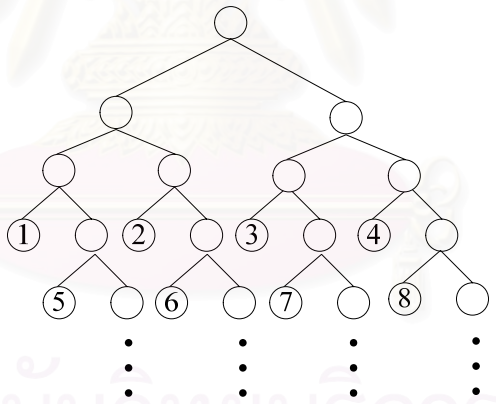
3.4 การสร้างต้นไม้รหัสแบบอนันต์

เมื่อข้อมูลที่ใช้ในการเข้ารหัสถูกพิจารณาอยู่ในลักษณะที่ไม่ทราบจำนวนและความถี่หรือความน่าจะเป็นอย่างชัดเจน โครงสร้างต้นไม้รหัสทวิภาคที่ใช้ในการเข้ารหัสจึงต้องเป็นโครงสร้างที่สามารถที่จะทำการเพิ่มข้อมูลตัวที่ปรากฏขึ้นมาใหม่ได้เสมอ ในการสร้างโครงสร้างต้นไม้รหัสทวิภาคดังกล่าวนี้ได้ใช้วิธีการสร้างด้วยหลักการแทนที่แบบวนซ้ำที่ขึ้นกับฟังก์ชันก่อกำเนิด (generating function) ที่กำหนด นอกจากนั้นในงานวิจัยนี้ ผู้วิจัยได้ใช้หลักการดังกล่าวในการสร้างโครงสร้างต้นไม้รหัสทวิภาคสำหรับการเข้ารหัสข้อมูล

ตัวอย่างของหลักการสร้างโครงสร้างต้นไม้รหัสทวิภาคที่ใช้ในการเข้ารหัสที่ถูกสร้างด้วยหลักการแทนที่แบบวนซ้ำที่ขึ้นกับฟังก์ชันก่อกำเนิด เช่น กำหนดให้

$$f(L) = c \tag{3.5}$$

เมื่อ c อยู่ในรูปของ 2^m เมื่อ m คือจำนวนเต็มบวกและ L คือระดับในต้นไม้ จะเห็นได้ว่าฟังก์ชันก่อกำเนิดที่กำหนดมาให้นี้เป็นฟังก์ชันค่าคงที่ ที่มีคุณสมบัติว่าทุกระดับในโครงสร้างต้นไม้มีจำนวนของใบ (รหัส) เป็นจำนวน c เสมอ เช่น ถ้ากำหนดให้ c เท่ากับ 4 โครงสร้างต้นไม้รหัสทวิภาคที่ถูกสร้างด้วยฟังก์ชันก่อกำเนิดนี้จะแสดงดังรูปที่ 3.4



รูปที่ 3.4 โครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ที่ถูกสร้างโดย $f(L) = 4$

จากการสร้างต้นไม้ด้วยวิธีดังกล่าวต้นไม้สามารถเกิดลูกใหม่ได้ตลอดเวลาที่มีรหัสใหม่เกิดขึ้น ดังนั้นบัพในโครงสร้างต้นไม้จึงสามารถแบ่งออกได้เป็นสองประเภทคือ บัพที่แสดงรหัส (ใบ) แสดงด้วยวงกลมระบุหมายเลขในรูปที่ 3.4 และบัพที่สามารถจะแตกกิ่ง (branching หรือ non-terminal nodes) ต่อไปได้ สามารถแสดงได้ด้วยวงกลมไม่ระบุหมายเลข

ในการกำหนดรหัสจากโครงสร้างต้นไม้จะพิจารณาจากบัพใบ ซึ่งในแต่ละบัพใบจะประกอบด้วยข้อมูล 3 ค่า ตามสมการ (3.6)

$$node_i = < a_i, \, frequency(a_i), c_i > \tag{3.6}$$

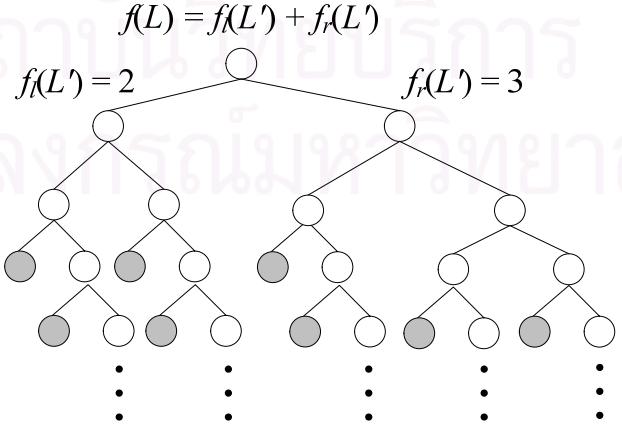
เมื่อ $node_i$ เป็นสมาชิกในโครงสร้างต้นไม้ ($Tree$) a_i และ $frequency(a_i)$ คือตัวอักษรข้อมูลเข้าและค่าความถี่ของข้อมูลเข้าตัวนั้น c_i คือข้อมูลออกหรือรหัสแสดงด้วยเส้นทางจากบัพเริ่มต้นหรือบัพราก (root node) ไปยังบัพสุดท้าย ($node_i$)

ด้วยวิธีการสร้างโครงสร้างต้นไม้รหัสทวิภาคข้างต้นทำให้ผู้วิจัยสรุปได้ว่า ทุกข้อมูลออกหรือรหัสที่ได้จากโครงสร้างต้นไม้รหัสทวิภาคที่สร้างโดยฟังก์ชันก่อกำเนิดที่แสดงอยู่ที่บัพใบเท่านั้น คงลักษณะของการที่ไม่ซ้ำตัวเติมหน้า ซึ่งมีลักษณะแบบเดียวกับข้อมูลออกที่ได้จากอัลกอริทึมของฮัฟฟ์แมน นอกจากนี้โครงสร้างต้นไม้รหัสทวิภาคดังกล่าวยังสามารถทำการเข้ารหัสข้อมูลที่ปรากฏใหม่ได้เสมอด้วยฟังก์ชันก่อกำเนิดที่กำหนด

นอกจากนี้โครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ยังสามารถสร้างได้หลายเทคนิคด้วยกัน เช่น ในกรณีที่เราต้องการสร้างฟังก์ชันค่าคงที่ที่ค่าของ c ไม่อยู่ในรูปของ 2^m เมื่อ m เป็นจำนวนเต็มบวกก็สามารถทำได้เช่นเดียวกัน เช่น กำหนดให้ c มีค่าเป็น 5 ฟังก์ชันนี้สามารถสร้างได้จากการรวมกันของฟังก์ชันก่อกำเนิดสองฟังก์ชัน ซึ่งต้นไม้รหัสทางซ้ายสร้างด้วยฟังก์ชันหนึ่ง ส่วนทางขวาสร้างด้วยอีกฟังก์ชันหนึ่ง ฟังก์ชันที่เกิดจากการรวมกันดังกล่าวสามารถแสดงได้ดังสมการ (3.7)

$$f(L) = f_l(L') + f_r(L'), \; L = L' + 1 \tag{3.7}$$

เมื่อ $f_l(L')$ คือฟังก์ชันก่อกำเนิดสำหรับต้นไม้รหัสทวิภาคแบบอนันต์ทางซ้าย โดย $f_l(L')$ เท่ากับ 2 L' มีค่ามากกว่าหรือเท่ากับ 2 $f_l(0)$ $f_l(1)$ เท่ากับ 0 และ $f_r(L')$ คือฟังก์ชันก่อกำเนิดสำหรับต้นไม้รหัสทวิภาคแบบอนันต์ทางขวา โดย $f_r(L')$ เท่ากับ 3 L' มากกว่าหรือเท่ากับ 3 และค่าเริ่มต้นคือ $f_r(0)$ $f_r(1)$ เท่ากับ 0 และ $f_r(2)$ เท่ากับ 1 ส่วน L คือระดับในโครงสร้างต้นไม้รหัสที่ได้จากการรวมกันของทั้งสองฟังก์ชัน ซึ่งโครงสร้างต้นไม้ที่สร้างได้จากฟังก์ชันดังกล่าวแสดงโดยรูปที่ 3.5



รูปที่ 3.5 โครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ที่สร้างโดย
การรวมกันของฟังก์ชันก่อกำเนิดสองฟังก์ชัน

ในระหว่างกระบวนการเข้ารหัสข้อมูลที่ใช้โครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์โดยสร้างด้วยหลักการแทนที่แบบวนซ้ำที่ขึ้นกับฟังก์ชันก่อกำเนิด ซึ่งเป็นการเข้ารหัสที่ได้รหัสที่มีความยาวแปรผันได้นั้น สิ่งที่ควรคำนึงเป็นอย่างยิ่งก็คือ ความถี่หรือความน่าจะเป็นของข้อมูลแต่ละตัวและจำนวนข้อมูลทั้งหมด ในการเข้ารหัสแบบเชื่อมตรงก็เช่นเดียวกันแต่เนื่องจากข้อมูลทั้งหมดไม่ทราบล่วงหน้าระหว่างการเข้ารหัส ดังนั้นจำนวนรหัสและความถี่หรือความน่าจะเป็นจึงเปลี่ยนแปลงได้ตลอดเวลาของการเข้ารหัส จากหัวข้อ 3.3 เราต้องการให้โครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์มีคุณสมบัติของความยาวรหัสเป็นเหมือนคุณสมบัติของต้นไม้ที่เหมาะสมที่สุดคือ ความยาวรหัสที่ว่าข้อมูลที่มีความถี่หรือความน่าจะเป็นในการใช้มากควรมีรหัสที่สั้นกว่าและข้อมูลที่มีการใช้น้อยควรมีรหัสที่ยาวกว่านั้นจำเป็นที่จะต้องมีการปรับโครงสร้างของต้นไม้รหัสทวิภาคแบบอนันต์ตลอดเวลาของการเข้ารหัสแบบเชื่อมตรง

เพื่อรักษาคุณสมบัติของความยาวรหัส กระบวนการปรับโครงสร้างของต้นไม้รหัสทวิภาคแบบอนันต์ที่ใช้ในการเข้ารหัสข้อมูลนั้นจะมีการปรับรหัสของข้อมูลที่มีความถี่เพิ่มขึ้นจากข้อมูลเข้าที่เกิดขึ้นใหม่ให้อยู่ในตำแหน่งที่เหมาะสมและสอดคล้องกับคุณสมบัติความยาวรหัส โดยจะอธิบายในหัวข้อถัดไป

3.5 อัลกอริทึมการสลับรหัส

ในระหว่างกระบวนการทำการเข้ารหัสแบบเชื่อมตรง ความถี่หรือความน่าจะเป็นของข้อมูลอักษรสามารถเปลี่ยนแปลงได้ขึ้นกับข้อมูลที่เข้ามา เพื่อที่จะรักษาให้ต้นไม้ที่ใช้ในการเข้ารหัสมีคุณสมบัติของความยาวรหัสไว้ ดังนั้นต้นไม้รหัสอาจจะต้องทำการปรับเมื่อมีข้อมูลใหม่เข้ามา หลักการในการปรับต้นไม้รหัสสามารถทำได้โดยยึดหลักที่ว่า เมื่อใดก็ตามที่โครงสร้างต้นไม้ไม่มีคุณสมบัติของความยาวรหัส เราจะทำการปรับโครงสร้างต้นไม้เพื่อให้ได้คุณสมบัติดังกล่าว โดยสามารถแยกพิจารณาได้เป็นสองกรณี

กรณีที่หนึ่ง หากข้อมูลเข้าใหม่ที่เกิดขึ้นเป็นการปรากฏครั้งแรกแสดงว่าความถี่ของการปรากฏของข้อมูลย่อมต้องน้อยที่สุดเมื่อเทียบกับข้อมูลที่ได้อ่านมาทั้งหมดแล้ว (known)

กรณีที่สอง ข้อมูลเข้าใหม่ที่เกิดขึ้นไม่ได้เป็นการปรากฏครั้งแรก นั่นคือความถี่ของการปรากฏของข้อมูลนี้จะต้องเพิ่มขึ้น ซึ่งอาจมีผลทำให้โครงสร้างต้นไม้ขาดคุณสมบัติความยาวรหัสได้

จากสองกรณีดังกล่าว จะเห็นว่ามีโอกาสเป็นไปได้ที่โครงสร้างต้นไม้ที่มีคุณสมบัติของความยาวรหัสอาจขาดคุณสมบัติได้เมื่อมีข้อมูลใหม่ปรากฏ การขาดคุณสมบัติของความยาวรหัสหมายถึง มีรหัสของข้อมูลสองรหัสคือ c_i และ c_j ซึ่งความยาวของ c_i มากกว่าความยาวของ c_j โดยที่ความถี่หรือความน่าจะเป็นของ c_i มีค่าสูงกว่าของ c_j แต่เนื่องจากว่า ถ้าเรามีการพิจารณาปรับโครงสร้างต้นไม้ทุกรอบของการเกิดข้อมูลใหม่ ความถี่ที่เพิ่มขึ้นจะมีค่าเท่ากับหนึ่งเสมอ ดังนั้น การปรับตำแหน่งของรหัสในโครงสร้างต้นไม้จะพิจารณาเฉพาะรหัสที่มีค่าความถี่เท่ากับ

ค่าความถี่ของรหัสของข้อมูลใหม่ก่อนที่ข้อมูลใหม่จะปรากฏเท่านั้น โดยความยาวของรหัสของข้อมูลใหม่จะต้องสั้นกว่าหรือเท่ากับความยาวรหัสของข้อมูลที่พิจารณาทุกตัว ซึ่งนำไปสู่การพิจารณาเลือกตำแหน่งที่เหมาะสมของรหัสของข้อมูลใหม่โดยใช้วิธีการสลับตำแหน่ง อธิบายได้โดย *อัลกอริทึมการสลับรหัส* (*code interchanging algorithm*) ต่อไปนี้

Algorithm: Code Interchanging

input: Unsatisfied tree

a (input character)

c_a (code of *a*)

p_a (its frequency)

output: Satisfied tree

begin

for each node *b*: *length*(c_b) < *length*(c_a) **do**

if $p_b < p_a$

buffer $\leftarrow c_a$

$c_a \leftarrow c_b$

$c_b \leftarrow$ *buffer*

exit

endif

enddo

end

บทตั้ง 3.2 แสดงให้เห็นว่าต้นไม้รหัสทวิภาคแบบอนันต์ที่ได้ทำการปรับด้วยอัลกอริทึมการเปลี่ยนรหัสนั้นมีคุณสมบัติของความยาวรหัส

บทตั้ง 3.2 *กำหนดให้ S เป็นต้นไม้รหัสทวิภาคแบบอนันต์ที่ไม่มีคุณสมบัติของความยาวรหัส เมื่อได้ทำการปรับด้วยอัลกอริทึมการเปลี่ยนรหัสแล้วจะมีคุณสมบัติของความยาวรหัสเสมอ*

พิสูจน์บทตั้ง 3.2

จะแสดงว่าบทตั้ง 3.2 เป็นจริง นั่นคือต้องแสดงว่า ค่าความยาวรหัสเฉลี่ยที่คำนวณได้หลังจากการที่ต้นไม้รหัสทำการปรับด้วยอัลกอริทึมการสลับรหัสแล้วมีค่าต่ำกว่าค่าความยาวเฉลี่ยที่คำนวณได้ก่อนที่ต้นไม้รหัสจะถูกปรับ

กำหนดให้ $a_1a_2a_3...a_k$ คือลำดับของข้อมูลที่ต้องการเข้ารหัสและให้ $c_1, c_2, c_3, ..., c_k$ เป็นรหัสของข้อมูล $a_1, a_2, a_3, ..., a_k$ ตามลำดับ กำหนดให้ต้นไม้รหัสทวิภาคแบบอนันต์มีคุณสมบัติของความยาวรหัสและค่าความยาวรหัสเฉลี่ยคือ

$$\sum_{i=1}^{k} length(c_i)p_i \tag{3.8}$$

สมมติให้ข้อมูลอักษร a_k เป็นข้อมูลกำลังถูกเข้ารหัส ดังนั้นค่าความน่าจะเป็นของข้อมูล a_k (p_k) จึงมีค่าเพิ่มขึ้น ในกรณีที่ทำให้มีข้อมูลอักษร a_j ที่ค่าความน่าจะเป็น $p_j < p_k$ แต่ $length(c_j) < length(c_k)$ ดังนั้นจึงเป็นเหตุให้ต้นไม้รหัสสูญเสียคุณสมบัติของความยาวรหัสและค่าความยาวรหัสเฉลี่ยจึงกลายเป็น

$$L_k = \sum_{i=1}^{k-1} length(c_i)\, p_i + length(c_k)\, p_k \tag{3.9}$$

เมื่อนำอัลกอริทึมการสลับรหัสมาใช้กับต้นไม้รหัส ทำให้ต้นไม้รหัสถูกปรับ และค่าความยาวรหัสเฉลี่ยที่ได้คือ

$$L'_k = \sum_{i=1}^{k-1} length(c_i)\, p_i + length(c_j) p_k \tag{3.10}$$

เนื่องจาก $length(c_j) < length(c_k)$ ดังนั้น $L'_k < L_k$ ■

จากบทพิสูจน์ของบทตั้งที่ 3.2 เราจะเห็นว่าในกรณีที่มีเพียงรหัสเดียวเท่านั้นที่ทำให้ต้นไม้ขาดคุณสมบัติของความยาวรหัส การสลับตำแหน่งของรหัสในโครงสร้างต้นไม้จะเกิดขึ้นเพียงครั้งเดียว

3.6 อัลกอริทึมการเข้ารหัสและถอดรหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรง

อัลกอริทึมในการเข้ารหัสและถอดรหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงได้ถูกเสนอในส่วนนี้ หลักการคือการสร้างโครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ที่มีคุณสมบัติของความยาวรหัส เมื่อมีข้อมูลอักษรผ่านเข้ามาในกระบวนการเข้ารหัสสามารถแยกได้เป็นสองกรณีที่มีผลต่อการสร้างรหัสดังนี้

กรณีที่หนึ่ง ถ้าข้อมูลใหม่ที่ปรากฏขึ้นนั้นเป็นการปรากฏครั้งแรก จะต้องมีการกำหนดรหัสให้กับข้อมูลใหม่นั้นซึ่งแน่นอนว่าค่าความถี่ของข้อมูลใหม่นั้นจะเท่ากับหนึ่งเสมอ ซึ่งจะเป็นความถี่ที่น้อยที่สุดเมื่อเทียบกับข้อมูลที่เคยถูกเข้ารหัสมาแล้วทั้งหมด ดังนั้นรหัสของข้อมูลใหม่นี้จะต้องถูกสร้างและจัดเก็บในโครงสร้างต้นไม้ ณ ตำแหน่งของใบที่อยู่ในระดับสูงที่สุด (ค่า L สูงสุด) ในโครงสร้างต้นไม้

กรณีที่สอง ถ้าข้อมูลใหม่เป็นข้อมูลที่เคยปรากฏมาก่อนหน้านี้ ดังนั้นรหัสของข้อมูลนี้จะต้องมีการถูกบันทึกไว้แล้วในโครงสร้างต้นไม้ ในกรณีเช่นนี้ค่าความถี่ของข้อมูลนั้นจะมีค่าเพิ่มขึ้นหนึ่ง เราต้องพิจารณาว่าการเพิ่มขึ้นของความถี่นี้ทำให้โครงสร้างต้นไม้ขาดคุณสมบัติความยาวรหัสหรือไม่ ถ้าต้นไม้ขาดคุณสมบัติของความยาวรหัสเราจะต้องมีการปรับโครงสร้างต้นไม้โดยใช้อัลกอริทึมการสลับรหัส

อัลกอริทึมในการเข้ารหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรง (on-line prefix-free infinite encoding algorithm) สามารถแสดงดังต่อไปนี้

Algorithm: On-Line Prefix-Free Infinite Encoding

input: $a_1a_2a_3a_4\ldots$ (a sequence of characters)

output: $c_1c_2c_3c_4\ldots$ (a sequence of binary code)

begin

Tree $\leftarrow \varnothing$ (empty set)

$i \leftarrow 1$ (current input character)

$j \leftarrow 0$ (number of terminal nodes in *Tree*)

while (not-end-of-data) **do**

if ($a_i = x \mid \langle x, f_x, c_x \rangle$ = *node* in *Tree*)

$f_x \leftarrow f_x + 1$

$c_i \leftarrow c_x$

call Interchanging algorithm

else $j \leftarrow j + 1$

new $node_j$

$Tree \leftarrow Tree \cup node_j$

*** $node_j \leftarrow \langle a_i, 1,$ path from root to $node_j \rangle$

endif

$i \leftarrow i + 1$

enddo

end

หมายเหตุ *** เมื่อข้อมูล a_j เป็นข้อมูลใหม่ที่ปรากฏเป็นครั้งแรกในการเข้ารหัสแล้ว ข้อมูล a_j จะต้องถูกส่งไปพร้อมกับรหัสด้วย

ด้วยเทคนิคเดียวกัน อัลกอริทึมการถอดรหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรง (on-line prefix-free infinite decoding algorithm) สามารถแสดงดังต่อไปนี้

Algorithm: On-Line Prefix-Free Infinite Decoding

input: $c_1c_2c_3c_4\ldots$ (a sequence of binary code)

output: $a_1a_2a_3a_4\ldots$ (a sequence of characters)

begin

Tree $\leftarrow \varnothing$ (empty set)

$i \leftarrow 1$ (current input code)

$j \leftarrow 0$ (number of terminal nodes in *Tree*)

while (not-end-of-data) **do**

if ($c_i = y \mid \langle x, f_x, y\rangle$ = *node* in *Tree*)

$f_x \leftarrow f_x + 1$

$a_i \leftarrow x$

call Interchanging algorithm

else $j \leftarrow j + 1$

new $node_j$

$Tree \leftarrow Tree \cup node_j$

$node_j \leftarrow \langle a_i, 1,$ path from root to $node_j\rangle$

endif

$i \leftarrow i + 1$

enddo

end

จากอัลกอริทึมการเข้ารหัสและถอดรหัสทั้งสองนี้เราสามารถพิสูจน์ได้ว่า การเข้ารหัสแบบเชื่อมตรงด้วยอัลกอริทึมทั้งสองรหัสที่ได้จะมีคุณสมบัติไม่ซ้ำตัวเติมหน้าและมีคุณสมบัติของความยาวรหัสดังพิสูจน์ได้ด้วยทฤษฎีบท 3.1

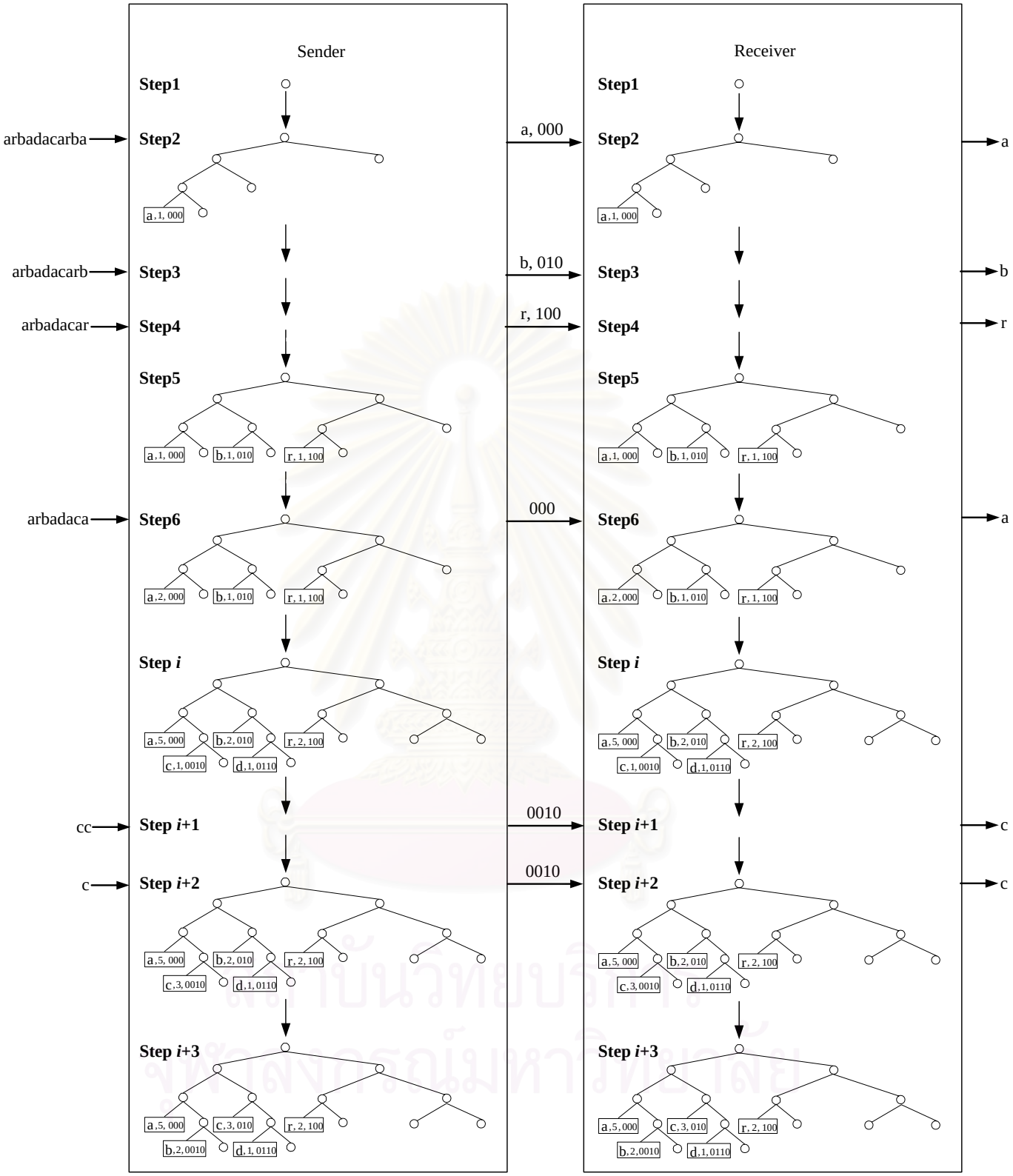
ทฤษฎีบท 3.1 *ต้นไม้รหัสทวิภาคแบบอนันต์ที่สร้างโดยอัลกอริทึมการเข้ารหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงที่ได้เสนอนั้น จะมีคุณสมบัติของความยาวรหัสด้วย*

พิสูจน์ทฤษฎีบท 3.1

เนื่องจากการใช้ต้นไม้รหัสทวิภาคเป็นโครงสร้างในการกำหนดรหัสดังนั้นรหัสที่ได้จึงมีคุณสมบัติไม่ซ้ำตัวเติมหน้า และจากผลลัพธ์ที่ได้จากการพิสูจน์บทตั้ง 3.2 แสดงให้เห็นว่าต้นไม้รหัสทวิภาคแบบอนันต์ที่สร้างโดยอัลกอริทึมการเข้ารหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรง จะมีคุณสมบัติของความยาวรหัสด้วย นั่นคือทฤษฎีบท 3.1 เป็นจริง ■

เพื่อให้เกิดความเข้าใจในหลักการทำงานของอัลกอริทึมการเข้ารหัส การถอดรหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงและอัลกอริทึมการสลับรหัสมากขึ้น ผู้วิจัยจะขออธิบายอัลกอริทึมดังกล่าวโดยการยกตัวอย่างที่แสดงดังรูปที่ 3.6

กำหนดให้ฟังก์ชัน $f(L) = f_l(L') + f_r(L')$ $L = L' + 1$ คือฟังก์ชันก่อกำเนิดที่ใช้ในตัวอย่างนี้ เมื่อ $f_l(L')$ คือฟังก์ชันก่อกำเนิดสำหรับต้นไม้รหัสทวิภาคแบบอนันต์ทางซ้าย โดย $f_l(L')$ เท่ากับ 2 L' มีค่ามากกว่าหรือเท่ากับ 2 $f_l(0)$ $f_l(1)$ เท่ากับ 0 และ $f_r(L')$ คือฟังก์ชันก่อกำเนิดสำหรับต้นไม้รหัสทวิภาคแบบอนันต์ทางขวา โดย $f_r(L')$ เท่ากับ 3 L' มากกว่าหรือเท่ากับ 3 และค่าเริ่มต้นคือ $f_r(0)$ $f_r(1)$ เท่ากับ 0 และ $f_r(2)$ เท่ากับ 1 ส่วน L คือระดับในโครงสร้างต้นไม้รหัสที่ได้จากการรวมกันของทั้งสองฟังก์ชัน สมมติให้ข้อมูลที่ใช้ในการเข้ารหัสเป็นลำดับของ abracadabra ในขั้นตอนแรกจากรูปที่ 3.6 (Step1) ก่อนการเข้ารหัสนั้นต้นไม้รหัสต้องว่างอยู่ เมื่อเริ่มทำการเข้ารหัสลำดับของข้อมูลเข้าจะถูกเข้ารหัสทีละตัว (character-by-character) ในที่นี้ a b และ r เป็นข้อมูลตัวใหม่ที่ไม่เคยปรากฏมาก่อนในต้นไม้รหัสแสดงใน (Step2 Step3 และ Step4) ตามลำดับ รหัสของข้อมูลใหม่ที่ปรากฏขึ้นจะถูกกำหนดให้โดยต้นไม้รหัสที่สร้างจากฟังก์ชันก่อกำเนิดและทำการเพิ่มค่าความถี่ด้วยการบวกหนึ่ง และรหัสที่ได้แสดงโดยเส้นทางจากบัพรากไปยังบัพใบของแต่ละข้อมูลดังนี้ a b และ r คือ <a, 000>, <b, 010>, และ <r, 100> ตามลำดับ ซึ่งรหัสเหล่านั้นจะถูกส่งไปยังส่วนของกระบวนการถอดรหัส โดยในส่วนของการถอดรหัสนี้ต้นไม้รหัสสามารถสร้างได้จากรหัสข้อมูลที่รับมา <a, 000>, <b, 010>, และ <r, 100> ซึ่งทำให้ได้ต้นไม้รหัสที่ใช้ในการเข้ารหัสและถอดรหัสแสดงดังรูปที่ 3.6 (Step5) ในส่วนที่เหลือนั้นจะทำกระบวนการวนซ้ำแบบเดิมเมื่อมีรหัสใหม่เกิดขึ้น ณ เวลาใดๆ (Step *i*) สมมติว่ามีข้อมูลเพิ่มเติมที่ต้องการเข้ารหัสคือ cc ตามลำดับ (Step *i*+1 และ Step *i*+2) ทำให้ต้นไม้ที่ใช้เข้ารหัสสูญเสียคุณสมบัติของความยาวรหัสไป ดังนั้นต้นไม้ที่ใช้เข้ารหัสต้องทำการปรับโดยใช้อัลกอริทึมการสลับรหัส ซึ่งจะทำการสลับรหัสระหว่างข้อมูล c กับ b และในทำนองเดียวกันเมื่อต้นไม้ที่ใช้ถอดรหัสทำการถอดรหัส 0010 แล้วทำให้สูญเสียคุณสมบัติเช่นกัน ดังนั้นจึงต้องทำการปรับต้นไม้ที่ใช้ถอดรหัสด้วยเทคนิคเดียวกัน ซึ่งเห็นได้อย่างชัดเจน (Step *i*+3) ว่าทั้งต้นไม้เข้ารหัสและถอดรหัสมีโครงสร้างเหมือนกันและหลังจากทำการปรับต้นไม้แล้วยังคงมีคุณสมบัติของความยาวรหัสด้วย



รูปที่ 3.6 ตัวอย่างการทำงานของอัลกอริทึมการเข้ารหัส การถอดรหัสอนันต์ ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงและอัลกอริทึมการสลับรหัส กับลำดับข้อมูล abracadabracc

บทที่ 4

อัลกอริทึมการเข้ารหัสข้อมูลอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรง สำหรับการกระจายของข้อมูลแบบโค้งปกติ

แม้ว่าตัวแปรสำคัญของอัลกอริทึมที่ใช้ในการเข้ารหัสข้อมูลอนันต์แบบเชื่อมตรงที่ผู้วิจัยทำการวิจัยอยู่นั้นคือโครงสร้างของต้นไม้รหัสทวิภาคแบบอนันต์ที่ใช้ในการเข้ารหัส แต่ก็เป็นเรื่องยากที่จะทำการหาฟังก์ชันก่อกำเนิดที่ใช้ในการสร้างโครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ดังกล่าวสำหรับข้อมูลที่ไม่ทราบทั้งจำนวนและความถี่หรือความน่าจะเป็น การแก้ปัญหาโดยการปรับโครงสร้างต้นไม้เมื่อข้อมูลเข้ามีการเปลี่ยนแปลงก็สามารถลดขนาดของความยาวเฉลี่ยของรหัสได้ในระดับหนึ่ง อีกปัจจัยหนึ่งซึ่งมีความสำคัญในการกำหนดความยาวของรหัสก็คือโครงสร้างต้นไม้ที่เลือกใช้ ทั้งนี้เนื่องจากว่าความยาวรหัสขึ้นกับระดับของรหัสที่ปรากฏในโครงสร้างต้นไม้ โครงสร้างต้นไม้ที่มีประสิทธิภาพสามารถที่จะลดความยาวของรหัสลงได้ ซึ่งในบทนี้ผู้วิจัยจะศึกษาถึงความสัมพันธ์ของโครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์ว่ามีส่วนสัมพันธ์กับการกระจายของข้อมูลเข้าหรือไม่ ซึ่งผู้วิจัยจะพิจารณาโครงสร้างต้นไม้ที่เป็นต้นไม้เหมาะสมที่สุดตามแนวความคิดของฮัฟฟ์แมนที่สร้างมาจากข้อมูลเข้าที่ทราบลักษณะการกระจายของข้อมูลมาเป็นตัวกำหนดฟังก์ชันก่อกำเนิด โดยในงานวิจัยนี้ผู้วิจัยเลือกใช้ข้อมูลที่มีการกระจายของข้อมูลแบบโค้งปกติ (normal source) เทคนิคที่เลือกใช้ในงานวิจัยนี้ผู้วิจัยได้เสนอฟังก์ชันค่าถ่วงน้ำหนักขึ้นเพื่อใช้แทนการกำหนดฟังก์ชันก่อกำเนิดของโครงสร้างต้นไม้

4.1 อัลกอริทึมการเข้ารหัสและถอดรหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงสำหรับการกระจายของข้อมูลแบบโค้งปกติ

อัลกอริทึมการเข้ารหัสและถอดรหัสสำหรับการกระจายของข้อมูลแบบโค้งปกติจะถูกเสนอในส่วนนี้ หลักการคือการสร้างต้นไม้รหัสทวิภาคแบบอนันต์สำหรับการเข้ารหัสและถอดรหัสที่ยังคงรักษาคุณสมบัติของความยาวรหัส เมื่อมีข้อมูลอักษรเข้าที่ปรากฏขึ้นเป็นครั้งแรกจะทำการเพิ่มข้อมูลนั้นเข้าไปยังต้นไม้รหัสตามฟังก์ชันค่าถ่วงน้ำหนักที่กำหนด (ฟังก์ชันค่าถ่วงน้ำหนักจะอธิบายโดยละเอียดในหัวข้อ 4.2) ถ้าเป็นข้อมูลที่เคยปรากฏแล้วจะทำการปรับต้นไม้รหัสเมื่อต้นไม้รหัสสูญเสียคุณสมบัติของความยาวรหัสด้วยอัลกอริทึมการสลับรหัส อัลกอริทึมการเข้ารหัสข้อมูลอนันต์แบบเชื่อมตรง (on-line prefix-free infinite code encoding algorithm) แสดงดังต่อไปนี้

Algorithm: On-Line Prefix-Free Infinite Code Encoding

input: $a_1a_2a_3a_4\ldots$ (a sequence of characters)

output: $c_1c_2c_3c_4\ldots$ (a sequence of binary code)

begin

Tree $\leftarrow \varnothing$ (empty set)

$N \leftarrow 0$ (number of different input characters)

$i \leftarrow 1$ (current input character)

while (not-end-of-data) **do**

if ($a_i = x \mid \langle x, f_x, c_x \rangle$ = *node* in *Tree*)

$f_x \leftarrow f_x + 1$

$c_i \leftarrow c_x$

call Interchanging algorithm

else

$N \leftarrow N + 1$

$L \leftarrow$ weight cost function(N)

new $node_{\text{left}}$, $node_{\text{right}}$

$node_{\text{left}} \leftarrow \langle node_{\text{minFreq}(L-1)} \rangle$

$node_{\text{minFreq}(L-1)} \leftarrow$ <null>

left.$node_{\text{minFreq}(L-1)} \leftarrow node_{\text{left}}$

right.$node_{\text{minFreq}(L-1)} \leftarrow node_{\text{right}}$

$node_{\text{right}} \leftarrow \langle a_i$, 1, path from root to $node_{\text{right}} \rangle$

endif

$i \leftarrow i + 1$

enddo

end

ด้วยหลักการเดียวกับอัลกอริทึมการเข้ารหัสอนันต์แบบเชื่อมตรง อัลกอริทึมการถอดรหัสอนันต์แบบเชื่อมตรง (on-line prefix-free infinite code decoding) แสดงโดย

Algorithm: On-Line Prefix-Free Infinite Code Decoding

input: $c_1c_2c_3c_4$… (a sequence of binary code)

output: $a_1a_2a_3a_4$… (a sequence of characters)

begin

Tree ⟵ ∅ (empty set)

i ⟵ 1 (current input character)

while (not-end-of-data) **do**

if (c_i = y | <x, f_x, y> = *node* in *Tree*)

f_x ⟵ f_x +1

a_i ⟵ x

call Interchanging algorithm

else

new $node_{left}$, $node_{new}$

$node_{new}$ ⟵ <a_i, 1, c_i>

$node_{left}$ ⟵ <left.parent.$node_{new}$>

parent.$node_{new}$ ⟵ <null>

endif

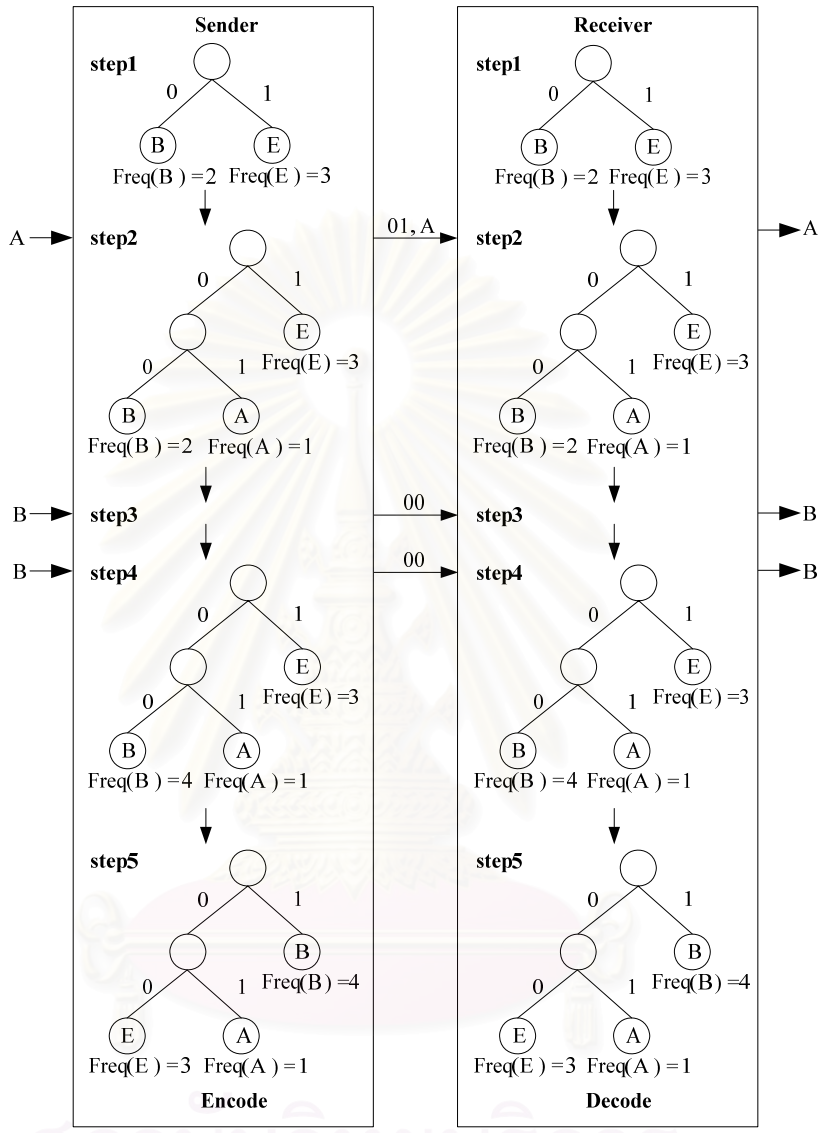
i ⟵ i + 1

enddo

end

ตัวอย่างการทำงานของอัลกอริทึมข้างต้นสามารถอธิบายได้โดยรูปที่ 4.1 สมมติให้โครงสร้างของต้นไม้เข้ารหัสและถอดรหัสมีข้อมูลที่ทำการเข้าและถอดรหัสแล้วคือ B และ E เมื่อค่าความถี่ของข้อมูล B และ E คือ 2 และ 3 ตามลำดับ (step 1) ในรูปที่ 4.1 กำหนดให้ ลำดับของข้อมูลเข้าเป็น ABB พิจารณาข้อมูลอักษร A เป็นข้อมูลใหม่ปรากฏขึ้นในกระบวนการเข้ารหัส (step 2) ดังนั้นข้อมูลอักษร A จะต้องถูกเพิ่มเข้าไปใหม่ในต้นไม้รหัส ซึ่งจะอยู่ที่ระดับใดนั้นจะถูกกำหนดโดยฟังก์ชันค่าถ่วงน้ำหนัก โดยจะขอกล่าวรายละเอียดในหัวข้อถัดไป หลังจากทำการเพิ่มข้อมูล A เข้าไปแล้วทำให้รหัสของ B เปลี่ยนจาก 0 เป็น 00 และรหัสของ A คือ 01 ทำการส่งรหัส <A, 01> ไปยังส่วนของการถอดรหัส ซึ่งจะทำการถอดรหัสจากข้อมูลที่รับมาได้ หลังจากนั้นพิจารณาข้อมูล BB เป็นข้อมูลเข้า จะได้ว่าเมื่อทำการเข้ารหัสแล้วทำให้ต้นไม้ที่ใช้เข้ารหัสสูญเสียคุณสมบัติของความยาวรหัส (step 4) ดังนั้นต้นไม้สำหรับการเข้ารหัสจึงต้องทำการปรับโดยอัลกอริทึมการสลับรหัส นั่นคือการสลับรหัสระหว่างข้อมูล B กับ C ก็จะ

ได้ผลลัพธ์ดังรูปที่ 4.1 (step 5) ทำนองเดียวกันการถอดรหัสก็ใช้เทคนิคเดียวกันนี้ในการสร้างและปรับต้นไม้รหัส ผลลัพธ์ที่ได้จะได้ต้นไม้การเข้ารหัสที่เหมือนกันทั้งสองฝั่งตลอดเวลา



รูปที่ 4.1 อัลกอริทึมการเข้ารหัส การถอดรหัสและการสลับรหัส

4.2 ฟังก์ชันค่าถ่วงน้ำหนัก

โครงสร้างของต้นไม้รหัสเป็นปัจจัยสำคัญสำหรับการเพิ่มประสิทธิภาพของอัลกอริทึมการเข้ารหัส ดังนั้นประเด็นสำคัญคือการสร้างโครงสร้างต้นไม้สำหรับการเข้ารหัสที่มีความสอดคล้องกับการกระจายของข้อมูลที่จะทำการเข้ารหัสได้ ซึ่งในหัวข้อนี้ ผู้วิจัยได้ทำการศึกษาโครงสร้างต้นไม้รหัสที่เหมาะสมสำหรับการกระจายข้อมูล โดยผู้วิจัยได้เลือกกรณีของข้อมูลที่มีการกระจายของข้อมูลแบบโค้งปกติ

เพื่อที่จะทำให้ได้ค่าความยาวเฉลี่ยในการเข้ารหัสของแต่ละข้อมูลต่ำสุด หลักการสำคัญคือการเลือกระดับที่เหมาะสมสำหรับการเพิ่มข้อมูลตัวใหม่ที่ปรากฏเข้าไปในต้นไม้ ซึ่งผู้วิจัยได้ทำการศึกษากับข้อมูลที่มีการกระจายแบบโค้งปกติ และได้ทำการวิเคราะห์กับโครงสร้างต้นไม้ที่เหมาะสมที่สุดด้วยอัลกอริทึมของฮัฟฟ์แมนแล้วได้ฟังก์ชันค่าถ่วงน้ำหนัก (weight cost function) คือ

$$z_i = 1/\left(a_z + b_z\left(i^{c_z}\right)\right) \tag{4.1}$$

เมื่อ z_i คือเปอร์เซนต์ของจำนวนโหนดที่มีในระดับที่ i ในโครงสร้างต้นไม้ ส่วน a_z b_z และ c_z เป็นฟังก์ชันที่ขึ้นอยู่กับค่าของ N ซึ่งเป็นจำนวนข้อมูลอักษรที่แตกต่างกันที่อยู่ในต้นไม้รหัส แสดงโดยฟังก์ชันดังต่อไปนี้

$$a_z = 1.5693629 + (-0.062556165)N \tag{4.2}$$

$$b_z = (-1.4371658) + 0.057994939N \tag{4.3}$$

$$c_z = 0.065868934 + (0.000027899102)N \tag{4.4}$$

ในการเพิ่มรหัสใหม่ให้กับต้นไม้ การเลือกระดับที่เหมาะสมให้กับรหัสเป็นสิ่งจำเป็น วิธีในการพิจารณาค่าระดับสามารถทำได้โดย การพิจารณาว่าความแตกต่างระหว่างต้นไม้รหัสที่เพิ่มรหัสใหม่เข้าไปแล้ว ณ ตำแหน่งใดจะให้ค่าความแตกต่างน้อยที่สุดเมื่อเทียบกับฟังก์ชันก่อกำเนิดที่เหมาะสมของต้นไม้นั้น แต่ทั้งนี้เนื่องจากว่าเราไม่ทราบฟังก์ชันก่อกำเนิด เราทราบเพียงว่าในแต่ละระดับควรมีจำนวนรหัสเป็นเท่าใด (ซึ่งได้มาจากฟังก์ชันค่าถ่วงน้ำหนัก) ดังนั้นเราจึงพิจารณาค่าที่แตกต่างกันน้อยที่สุดระหว่างเปอร์เซนต์ของรหัสที่คำนวณได้ในแต่ละระดับเทียบกับเปอร์เซนต์ของจำนวนรหัสจริงเมื่อเพิ่มรหัสใหม่เข้าไปแล้วในทุกระดับที่เป็นไปได้ การคำนวณค่าระดับดังกล่าวสามารถทำได้จากสมการ (4.5)

$$\left|z_i - x_i\right| = \min_{all\ level\ j}\{\left|z_j - x_j\right|, z_j \in \Re^+\} \tag{4.5}$$

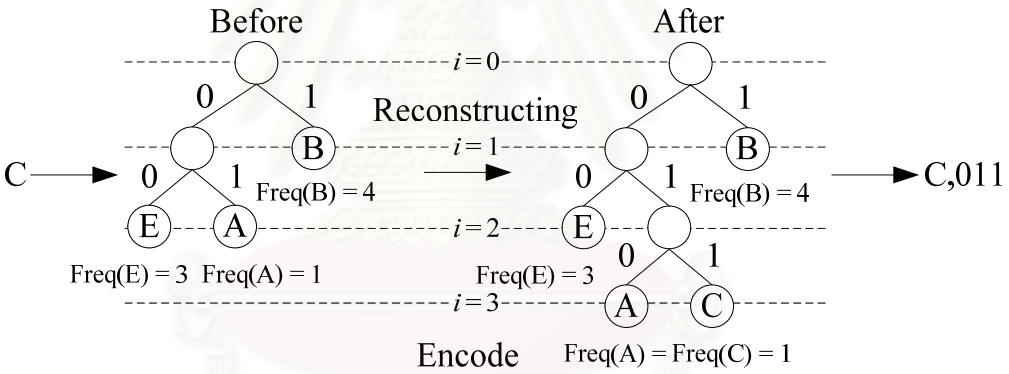
ตัวอย่างการสร้างโครงสร้างต้นไม้รหัสใหม่โดยใช้ฟังก์ชันค่าถ่วงน้ำหนักแสดงในรูปที่ 4.2 สมมติให้โครงสร้างต้นไม้รหัสมีข้อมูลอักษร A B และ E อยู่ ซึ่งมีค่าถี่ของการปรากฏของข้อมูลแต่ละตัวคือ 1 4 และ 3 ตามลำดับ แสดงโดยต้นไม้รหัสทางซ้ายในรูปที่ 4.2 กำหนดให้ข้อมูลอักษร C เป็นข้อมูลตัวถัดไปที่กำลังจะทำการเข้ารหัส ปรากฏว่า C เป็นข้อมูลใหม่ที่ปรากฏเป็นครั้งแรก ดังนั้นข้อมูล C จะต้องถูกใส่เข้าไปใหม่ในต้นไม้รหัสที่ทำการคำนวณจากฟังก์ชันค่าถ่วงน้ำหนักที่ระดับ i เท่ากับ 2 และ 3 (เนื่องจากว่ารหัสใหม่สามารถเกิดขึ้นได้เพียงสองระดับเท่านั้น)

จากตัวอย่างจำนวนรหัสทั้งหมด N มีค่าเท่ากับ 4 ทำให้ได้ค่า z_2 เป็น 17.37495 และ z_3 เป็น 42.83078 เราพิจารณาสองกรณี

กรณีที่หนึ่ง รหัสใหม่ถูกจัดเก็บในระดับที่สอง ผลที่ตามมาคือรหัสของ B จะถูกเปลี่ยนมาอยู่ในระดับที่สองด้วยเช่นรหัสของ B เป็น 10 และจะทำให้รหัสของ C เป็น 11 ทำให้จำนวนรหัสในระดับที่หนึ่งเป็นศูนย์และระดับที่สองเป็นสี่ เพราะฉะนั้นจำนวนรหัสในระดับที่สองนี้จะเป็นร้อยเปอร์เซนต์เมื่อเทียบกับจำนวนรหัสทั้งหมด (x_2 เป็น 100)

กรณีที่สอง รหัสใหม่ถูกจัดเก็บในระดับที่สาม ผลที่ได้รับคือรหัสของ A จะถูกเปลี่ยนมาอยู่ในระดับที่สามเช่นเดียวกับรหัสของ C และจะทำให้จำนวนรหัสในระดับที่หนึ่งและสองเป็นหนึ่ง จำนวนรหัสในระดับที่สามเป็นสอง เพราะฉะนั้นจำนวนรหัสในระดับที่สามจะคิดเป็นห้าสิบเปอร์เซนต์เมื่อเทียบกับจำนวนรหัสทั้งหมด (x_3 เป็น 50)

เมื่อพิจารณาตามสมการ (4.5) เป็นเกณฑ์ในการตัดสินใจ ดังนั้นระดับที่เหมาะสมสำหรับรหัสของ C คือระดับที่สาม (L เป็น 3 เนื่องจาก $\min_{all\ level\ j}\{|z_j - x_j|\}$ มีค่าเท่ากับ 7.169216) และหลังจากทำการปรับต้นไม้ใหม่แล้วได้ต้นไม้รหัสดังแสดงทางขวาในรูปที่ 4.2



รูปที่ 4.2 การสร้างโครงสร้างต้นไม้รหัสใหม่โดยใช้ฟังก์ชันค่าถ่วงน้ำหนัก

บทที่ 5

การเปรียบเทียบผลการทดลอง

เนื่องจากงานวิจัยนี้ ได้นำเสนออัลกอริทึมการเข้ารหัสอนันต์แบบเชื่อมตรงโดยใช้ฟังก์ชันก่อกำเนิด ดังนั้น จะทำการวัดประสิทธิภาพของอัลกอริทึม โดยทำการเปรียบเทียบค่าความยาวรหัสเฉลี่ยที่ได้จากการเข้ารหัสด้วยอัลกอริทึมที่เสนอ กับอัลกอริทึมฮัฟฟ์แมน อัลกอริทึมลิมเพล-ซิฟ (ในงานวิจัยนี้ได้เลือกแอลแซดซีและแอลแซดดับเบิ้ลยู) และเอนโทรปีของข้อมูล ในส่วนของข้อมูลที่ใช้ในการทดสอบการเข้ารหัสในงานวิจัยนี้ใช้ข้อความภาษาอังกฤษเป็นข้อมูลทดสอบ

ในการทดลองนี้ ผู้วิจัยได้แบ่งข้อมูลที่ใช้ในการทดลองออกเป็นสองแบบ แบบแรกเป็นข้อมูลที่มีการกระจายของอักษรภาษาอังกฤษแบบโค้งปกติ ส่วนแบบที่สองเป็นข้อความในภาษาอังกฤษทั่วไป โดยแต่ละแบบมีจำนวนสิบแฟ้มข้อมูลซึ่งมีขนาดของข้อมูลแตกต่างกัน

รูปที่ 5.1 แสดงเปอร์เซนต์ของการปรากฏของข้อมูลอักษรทั้งสองแบบ โดยกราฟแท่งที่แสดงด้วยสีขาวเป็นกราฟที่แสดงเปอร์เซนต์ของการปรากฏของข้อมูลอักษรที่มีการกระจายแบบโค้งปกติ ส่วนสีดำแสดงถึงเปอร์เซนต์ของการปรากฏของข้อมูลอักษรในข้อความภาษาอังกฤษทั่วไป ซึ่งมีความคล้ายคลึงกับงานที่ได้ทำการศึกษาโดยเบเกอร์และไปเปอร์ [19] และในส่วนของค่าความยาวรหัสเฉลี่ยที่ได้จากอัลกอริทึมที่เสนอ อัลกอริทึมฮัฟฟ์แมน อัลกอริทึมแอลแซดซี อัลกอริทึมแอลแซดดับเบิ้ลยูและเอนโทรปีของข้อมูลแสดงในตารางที่ 5.1

ในการทดลองฟังก์ชันก่อกำเนิดสามแบบที่เลือกใช้คือ ฟังก์ชันค่าคงที่ ฟังก์ชันประกอบ และฟังก์ชันค่าถ่วงน้ำหนักโดยมีรายละเอียดของฟังก์ชันดังต่อไปนี้

1. ฟังก์ชันค่าคงที่

$$f(L) = \lfloor \log_2 n \rfloor \text{ สำหรับ } L \geq 3 \qquad (5.1)$$

เมื่อ $f(0)$ $f(1)$ $f(2)$ มีค่าเท่ากับศูนย์ และ n คือจำนวนข้อมูลอักษรที่แตกต่างกัน (n เป็น 26)

2. ฟังก์ชันประกอบ

$$f(L) = f_l(L') + f_r(L'), L = L' + 1 \qquad (5.2)$$

เมื่อ $f_l(L')$ เท่ากับ 2 L' มีค่ามากกว่าหรือเท่ากับ 2 $f_l(0)$ $f_l(1)$ เท่ากับ 0 และ $f_r(L')$ เท่ากับ 3 L' มากกว่าหรือเท่ากับ 3 และค่าเริ่มต้นคือ $f_r(0)$ $f_r(1)$ เท่ากับ 0 และ $f_r(2)$ เท่ากับ 1

3. ฟังก์ชันถ่วงค่าน้ำหนัก

$$z_i = 1/\left(a_z + b_z\left(i^{c_z}\right)\right) \tag{5.3}$$

เมื่อ z_i คือเปอร์เซนต์ของจำนวนโหนดที่มีในระดับที่ i ในโครงสร้างต้นไม้ ส่วน a_z b_z และ c_z เป็นฟังก์ชันที่ขึ้นอยู่กับค่าของ N ซึ่งเป็นจำนวนข้อมูลอักษรที่แตกต่างกันที่อยู่ในต้นไม้รหัส แสดงโดยฟังก์ชันดังต่อไปนี้

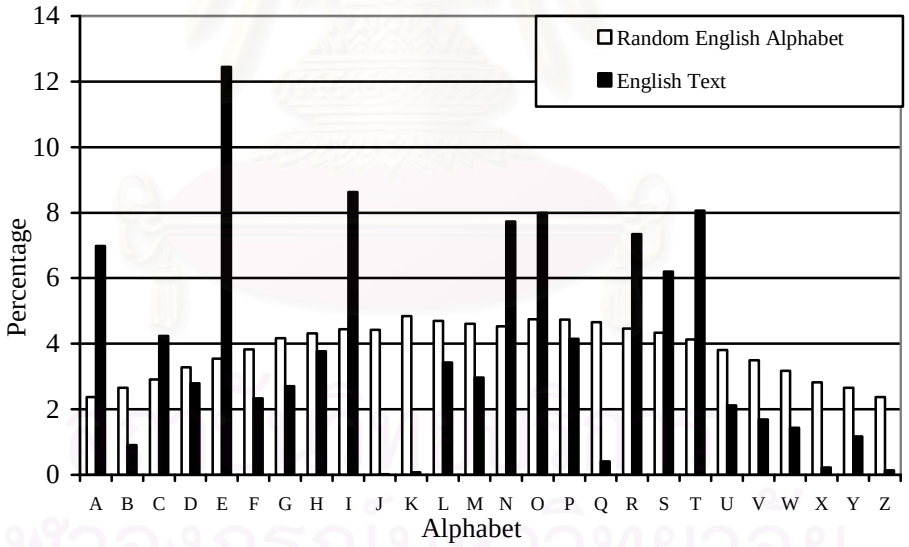
$$a_z = 1.5693629 + (-0.062556165)N \tag{5.4}$$

$$b_z = (-1.4371658) + 0.057994939N \tag{5.5}$$

$$c_z = 0.065868934 + (0.000027899102)N \tag{5.6}$$

ระดับที่เหมาะสมคือ L เท่ากับ i ถ้า

$$\left|z_i - x_i\right| = \min_{all\ level\ j}\{\left|z_j - x_j\right|, z_j \in \Re^+\} \tag{5.7}$$



รูปที่ 5.1 ค่าเปอร์เซนต์ของการปรากฏของข้อมูลอักษรภาษาอังกฤษ

ตารางที่ 5.1 การเปรียบเทียบผลการทดลอง

size	Random English Alphabet							English Text						
(bytes)	1st Fn	2nd Fn	3rd Fn	LZW	LZC	Huffman	Entropy	1st Fn	2nd Fn	3rd Fn	LZW	LZC	Huffman	Entropy
1000	6.47100	5.50000	4.73900	8.58000	6.66400	4.72900	4.63410	4.50200	4.30700	4.29000	6.48000	4.91400	4.16800	4.13471
2000	6.49400	5.51100	4.75100	7.67016	6.28936	4.73800	4.63759	4.48150	4.28450	4.35300	5.67000	4.48200	4.17100	4.13251
3000	6.49833	5.51233	4.75667	7.16561	6.07398	4.74233	4.63944	4.47167	4.28133	4.39567	5.23600	4.30500	4.18567	4.14054
4000	6.48750	5.50075	4.75575	6.85329	5.97701	4.74400	4.63972	4.46800	4.27375	4.41550	4.93077	4.14896	4.18700	4.14259
5000	6.48140	5.49380	4.75720	6.63360	5.93240	4.74560	4.64074	4.45040	4.26160	4.42760	4.72945	4.03859	4.17980	4.13774
6000	6.50650	5.51217	4.76267	6.47600	5.89167	4.75167	4.64208	4.44733	4.26050	4.43667	4.72989	4.03882	4.18150	4.13808
7000	6.44186	5.47043	4.75757	6.34881	5.84802	4.74743	4.63977	4.45040	4.26160	4.42760	4.52698	3.94304	4.17980	4.13774
8000	6.36950	5.42525	4.75538	6.40800	5.81463	4.74450	4.63666	4.45850	4.26825	4.44875	4.30350	3.86525	4.18888	4.14588
9000	6.25289	5.35322	4.74956	7.02878	5.81920	4.73622	4.62792	4.45167	4.26700	4.45000	4.20047	3.81087	4.18689	4.14635
10000	6.04850	5.22980	4.72230	7.52595	5.77793	4.70760	4.60920	4.44450	4.26080	4.45030	4.10286	3.75205	4.18350	4.14436
Average	6.40515	5.45088	4.75071	7.06902	6.00882	4.73864	4.63472	4.46260	4.27263	4.40951	4.89099	4.12986	4.18120	4.14005

ตารางที่ 5.1 แสดงให้เห็นว่าการเข้ารหัสด้วยอัลกอริทึมที่เสนอโดยใช้ฟังก์ชันที่มีความเหมาะสมกับการกระจายของข้อมูลนั้นทำให้สามารถทำการเข้ารหัสข้อมูลได้ดีกว่าอัลกอริทึมอื่นๆ นอกจากนั้นค่าความยาวรหัสเฉลี่ยของข้อมูลภาษาอังกฤษที่มีการกระจายแบบโค้งปกติและข้อความภาษาอังกฤษทั่วไปที่ได้จากการเข้ารหัสด้วยอัลกอริทึมที่เสนอมีค่าใกล้เคียงกับอัลกอริทึมของฮัฟฟ์แมน

ตารางที่ 5.2 การเปรียบเทียบกับเอนโทรปี

Type of Algorithm	Average/Entropy	
	Random English Alphabet	English Text
1^{st} $F^{\underline{n}}$	1.38199	1.07791
2^{nd} $F^{\underline{n}}$	1.17610	1.03202
3^{rd} $F^{\underline{n}}$	1.02503	1.06509
Huffman	1.02242	1.00994
LZW	1.52523	1.19118
LZC	1.29648	0.99754

จากตารางที่ 5.2 แสดงให้เห็นว่าอัลกอริทึมที่เสนอในกรณีที่ทราบการกระจายของข้อมูลสามารถให้ผลลัพธ์ค่าเฉลี่ยของความยาวของรหัสทั้งหมดใกล้เคียงค่าเอนโทรปีของข้อมูลมากที่สุดเมื่อเทียบกับอัลกอริทึมของฮัฟฟ์แมน แอลแซดดับเบิ้ลยู และแอลแซดซี สำหรับข้อความภาษาอังกฤษทั่วไปความยาวเฉลี่ยของรหัสที่ได้จากอัลกอริทึมแบบเชื่อมตรงมีค่าใกล้เคียงกับผลลัพธ์ที่ได้จากอัลกอริทึมของฮัฟฟ์แมน

บทที่ 6

สรุปผลการวิจัย

ในงานวิจัยนี้ ผู้วิจัยได้เสนออัลกอริทึมการเข้ารหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงโดยใช้ฟังก์ชันก่อกำเนิด ซึ่งฟังก์ชันดังกล่าวเป็นฟังก์ชันที่ใช้ในการสร้างโครงสร้างต้นไม้รหัสทวิภาคแบบอนันต์สำหรับการแสดงรหัส โดยผลทางทฤษฎีแสดงให้เห็นจริงว่าโครงสร้างต้นไม้ที่ได้จากอัลกอริทึมที่เสนอมีคุณสมบัติของความยาวรหัส ในส่วนของผลการทดลองกับข้อมูลภาษาอังกฤษซึ่งมีทั้งข้อมูลที่มีการกระจายของอักษรภาษาอังกฤษในรูปของโค้งปกติและข้อความภาษาอังกฤษทั่วไป แสดงให้เห็นว่าค่าความยาวรหัสเฉลี่ยที่ได้จากอัลกอริทึมที่เสนอมีค่าใกล้เคียงกับอัลกอริทึมคลาสสิค

นอกจากนั้นผู้วิจัยยังได้ทำการศึกษาฟังก์ชันที่เหมาะสมกับการกระจายของข้อมูลด้วย ซึ่งในงานวิจัยนี้ผู้วิจัยได้เลือกการกระจายของข้อมูลแบบโค้งปกติเป็นกรณีศึกษา จากผลที่ได้นั้นผู้วิจัยสรุปได้ว่าเมื่อสามารถทำการหาฟังก์ชันค่าถ่วงน้ำหนักที่ใช้สำหรับการสร้างโครงสร้างต้นไม้รหัส ได้เหมาะสมกับการกระจายของข้อมูลที่ต้องการเข้ารหัส แล้วนำไปประยุกต์ใช้กับอัลกอริทึมการเข้ารหัสอนันต์ที่ไม่ซ้ำตัวเติมหน้าแบบเชื่อมตรงและอัลกอริทึมการสลับรหัส จะทำให้การเข้ารหัสมีประสิทธิภาพเพิ่มขึ้น

จากงานวิจัยนี้ผู้วิจัยเห็นว่าการปรับรหัสให้อยู่ในระดับที่เหมาะสมโดยพิจารณาจากฟังก์ชันค่าถ่วงน้ำหนักอย่างเดียวนั้นอาจนำไปสู่ผลลัพธ์ที่มีประสิทธิภาพมากขึ้นก็จริง แต่ปัญหาหนึ่งก็คือเราต้องทราบค่าการกระจายของข้อมูลก่อนเพื่อคำนวณฟังก์ชันค่าถ่วงน้ำหนักซึ่งอาจจะเป็นเรื่องยาก ผู้วิจัยพบว่าในทางปฏิบัติแล้วเมื่อข้อมูลถูกอ่านเข้ามาตามลำดับเราน่าจะมีวิธีการในการตรวจสอบหรือคำนวณลักษณะการกระจายตัวของข้อมูลได้โดยไม่จำเป็นต้องกำหนดล่วงหน้า ซึ่งแนวคิดนี้น่าจะนำไปสู่การปรับปรุงประสิทธิภาพของอัลกอริทึมให้โครงสร้างต้นไม้สอดคล้องกับการกระจายตัวของข้อมูลจริงอันนำไปสู่การลดค่าเฉลี่ยของความยาวรหัสทั้งหมดได้

รายการอ้างอิง

[1] Golomb, S.W. Run Length Encodings. In **IEEE Transactions on Information Theory**, pp.399-401. July, 1966.

[2] Huffman, D.A. A method for the construction of minimum redundancy codes. In **Proc. IRE 40**, pp.1098-1101. September, 1952.

[3] Golin, M. and Rote, G. A dynamic programming algorithm for constructing optimal prefix-free codes for unequal letter costs. In **IEEE Trans. Inform. Theory**, pp.1770-1781. September, 1998.

[4] Linder, T. Tarokh, V. and Zeger, K. Existence of Optimal Prefix Codes for Infinite Source Alphabets. In **IEEE Transactions on Information Theory**, pp.2026-28. November, 1997.

[5] Golin, M. Kenyon, C. and Young, N. Huffman Coding with Unequal Letter Costs. In **Proceeding of the 34th ACM Symposium on Theory of Computing (STOC2002)**, pp.785-791. May, 2002.

[6] Golin, M. and Na, H. Optimal prefix-free codes that end in a specified pattern and similar problems: the uniform probability case (Extended Abstract)*. In **IEEE Trans. Inform. Theory**, pp.143-152. 2001.

[7] Golin, M. and Young, N. Prefix Codes: Equiprobable words, unequal letter costs. In **SIAM Journal on Computing**, pp.1281-1292. December, 1996.

[8] Chan, S.L. and Golin, M. A Dynamic Programming Algorithm for Construction Optimal ”1”-ended Binary Prefix-Free Codes. In **IEEE Transactions on Information Theory**, pp.1637-44. July, 2000.

[9] Ma, K.K. and Golin, M.J. Algorithms for Infinite Huffman-Codes*. In **Proceedings of the Association for Computer Machinery**, pp.758-767. 2004.

[10] Laidlaw, M.G.G. The Construction of Codes for Infinite Sets. In **Proceedings of SAICSIT**, pp.157-165. 2004.

[11] Ziv, J. and Lempel, A. A Universal algorithm for sequential data compression. In **IEEE Trans. Inform Theory**, pp. 337-343. May, 1977.

[12] Fiala, E.R. and Greene, D.H. Data compression with finite windows. In **Communications of ACM**, pp. 490-505. April, 1989.

[13] Jiang, J. and Jones, S. Word-based dynamic algorithms for data compression. In **IEEE Proceedings-1**. December, 1992.

[14] Hidetoshi, Y. Improved Variations Relating the Ziv-Lempel and Welch-Type Algorithms for Sequential Data Compression. In **IEEE Trans. on Inform Theory**. January, 1992.

[15] Shannon, C.E. A Mathematical Theory of Communication. In **Bell Systems Technical Journal**, vol 27, pp 379-423 and pp 623-656. July, 1948.

[16] Ziv, J. and Lempel, A. Compression of individual sequences via variable-rate coding. In **IEEE Transactions on Information Theory**, vol. 24, pp. 530–536. September, 1978.

[17] Welch, T.A. A technique for high-performance data compression. In **IEEE Computer**, pp. 8-19. June, 1984.

[18] Horspool, R.N. Improving LZW. In **Proceedings of Data Compression, DCC' 91, IEEE Computer Society Press**, pp. 332-341. April, 1991.

[19] Beker, H. and Piper, F. **Cipher Systems**. Wiely-Interscience. 1982.

ประวัติผู้เขียนวิทยานิพนธ์

ว่าที่ร้อยตรี นัฐพล ปั้นทอง เกิดเมื่อวันที่ 14 สิงหาคม พ.ศ. 2524 จบการศึกษาระดับมัธยมศึกษาตอนต้น และมัธยมศึกษาตอนปลายจากโรงเรียนชลบุรี "สุขบท" เข้าศึกษาต่อในระดับปริญญาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย จนสำเร็จการศึกษาในปี พ.ศ. 2547 และในปี พ.ศ. 2547 เข้าศึกษาต่อในระดับปริญญามหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย