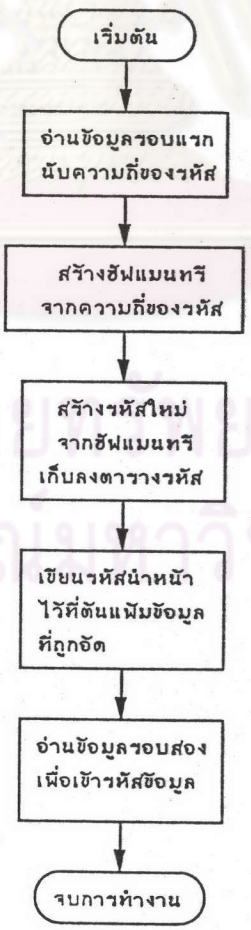
บทที่ 4
การปรับเปลี่ยนการอัดข้อความภาษาไทย

4.1 การปรับเปลี่ยนอัลกอริทึมฮัฟแมน

4.1.1 อัลกอริทึมการอัดข้อมูล

อัลกอริทึมการอัดข้อมูลที่ปรับเปลี่ยน ได้พัฒนาจากอัลกอริทึมการอัดข้อมูลเดิม ดังรูปที่ 4.1 ซึ่งมีขั้นตอนการทำงานดังนี้

1. การอ่านข้อมูลรอบแรกเพื่อนับความถี่ของรหัสต่าง ๆ ที่เกิดขึ้นในข้อมูล
2. นำความถี่ที่นับได้สร้างฮัฟแมนทรีตามหลักการของอัลกอริทึมฮัฟแมนโดยที่ทรีสร้างขึ้นจะมีลักษณะเป็นทรีแบบฟูลไบนารีทรี มีจำนวนลีฟเท่ากับจำนวนรหัสที่เกิดขึ้นในข้อมูล
3. สร้างรหัสใหม่จากทรีโดยการเดินจากรูทไปยังลีฟแต่ละลีฟ ค่าของรหัสใหม่ 1 บิตจะได้จากการกำหนดทิศทางการเดินของโหนดพ่อไปโหนดลูกทางซ้าย หรือทางขวามีค่า



รูปที่ 4.1 การอัดข้อมูลโดยอัลกอริทึมฮัฟแมน

เป็น 0 หรือ 1 โดยที่รหัสใหม่จะมีความยาวของรหัส (หน่วย : บิต) เท่ากับจำนวนทางเดินจากรูทไปยังลีฟของรหัสนั้น ๆ เก็บรหัสใหม่และความยาวของรหัสใหม่ลงตารางรหัส เราอาจจะไม่มีขั้นตอนการสร้างรหัสใหม่ก็ได้ แต่ในการเข้ารหัสข้อมูล การแทนรหัสต้องอ่านรหัสใหม่จากทรีโดยตรง แม้ว่าการสร้างรหัสใหม่เก็บไว้ในตารางรหัสจะต้องทำให้เสียเวลาเพิ่มขึ้น แต่ก็มีข้อดีทำให้การเข้ารหัสข้อมูลมีประสิทธิภาพมากกว่าการแทนรหัสใหม่โดยตรงจากทรี เพราะการแทนรหัสจากทรีจะสามารถแทนรหัสได้ทีละ 1 บิตเท่านั้น ทำให้ใช้เวลามากกว่าการแทนรหัสจากตารางรหัสซึ่งเราสามารถพัฒนาวิธีการแทนรหัสได้รวดเร็วขึ้น

4. การเขียนรหัสนำหน้าไว้ที่ต้นแฟ้มข้อมูลที่ถูกอัด มี 2 วิธี วิธีแรกเก็บรูปแบบของฮัฟแมนทรี ซึ่งจะต้องใช้เนื้อที่ทั้งหมด 2n-1 โหนดสำหรับรหัส n รหัสหรืออาจเก็บเป็นค่ารหัสและความถี่ของรหัสแต่ละรหัสที่ใช้ในการเข้ารหัสข้อมูล วิธีนี้จะใช้เนื้อที่ในการเก็บน้อยกว่าวิธีแรก แต่การขยายข้อมูลต้องเสียเวลาในการนำความถี่ของรหัสมาสร้างฮัฟแมนทรีอีก สำหรับโปรแกรมที่พัฒนาใช้วิธีการเก็บรหัสนำหน้าวิธีที่สองโดยมีรูปแบบของรหัสนำหน้าดังรูปที่ 4.2

5. การเข้ารหัสข้อมูล เป็นการแทนรหัสเดิมในข้อมูลด้วยรหัสใหม่ที่สร้างขึ้น

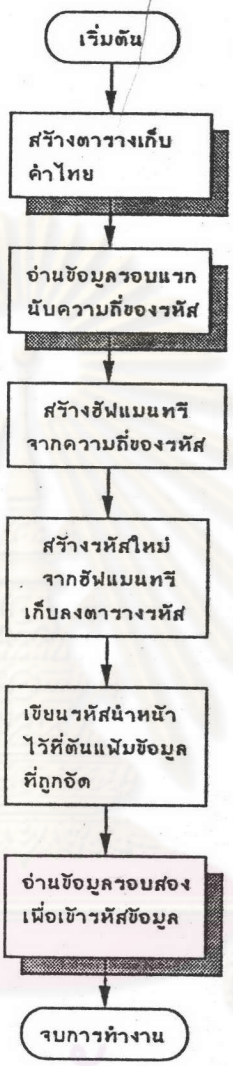
n (จำนวนรหัส)	รหัสที่ 1	ความถี่ รหัสที่ 1	รหัสที่ 2	ความถี่ รหัสที่ 2	.....	รหัสที่ n	ความถี่ รหัสที่ n

รูปที่ 4.2 รูปแบบของรหัสนำหน้า

สำหรับการปรับเปลี่ยนอัลกอริทึมฮัฟแมนในการอัดข้อมูล แสดงไว้ดังรูปที่ 4.3 มีขั้นตอนที่เพิ่มขึ้นมาคือ การสร้างตารางคำไทยจากแฟ้มข้อมูลคำไทย ตารางคำไทยนี้จะถูกนำไปใช้ในขั้นตอนการนับความถี่ของรหัสและการเข้ารหัสข้อมูล ซึ่งมีผลทำให้มีการเปลี่ยนแปลงอัลกอริทึมของวิธีการใน 2 ขั้นตอนนี้ ส่วนขั้นตอนอื่น ๆ ยังคงใช้หลักการตามอัลกอริทึมเดิมทุกประการ

การนับความถี่จะแตกต่างไปจากหลักการเดิมคือ มีตารางคำไทยไว้ตรวจสอบว่าข้อความส่วนใดที่ตรงกับคำในตารางคำไทย ข้อความส่วนนั้นจะไม่ถูกนับเป็นความถี่ของรหัสใดรหัสหนึ่ง แต่จะนับเป็นความถี่ของรหัสพิเศษแทน จำนวนเท่ากับความยาวของคำนั้น ในที่นี้รหัสพิเศษ หมายถึง รหัสตัวใด ๆ ในตารางแอสกีที่ถูกเลือกมาเพื่อใช้สำหรับเป็นรหัสของคำ และรหัสนี้ต้องเป็นรหัสที่ไม่มีการใช้งานในข้อมูลประเภทข้อความ หรืออาจกำหนดเป็นรหัสใหม่ตัวที่ 257 ก็ได้ เมื่อนับความถี่จากข้อมูลภาษาไทยโดยวิธีการเช่นนี้ จะได้ความถี่ของรหัสพิเศษมีค่าสูงสุด เพราะ ความถี่ของรหัสพิเศษเป็นค่าที่ได้จากจำนวนอักษรทั้งหมดของคำทุกคำในข้อมูลที่ตรงกับคำในตารางคำไทย

ต่อจากนั้นให้นำความถี่ของรหัสต่าง ๆ ที่นับได้จากข้อมูลสร้างฮัฟแมนทรี สร้างรหัสใหม่และเขียนรหัสนำหน้าตามหลักการของอัลกอริทึมเดิม



รูปที่ 4.3 การอัดข้อมูลโดยอัลกอริทึมฮัฟแมนที่ปรับเปลี่ยน

ตัวอย่าง สมมติแฟ้มข้อมูลหนึ่งมีข้อความดังนี้ "การค้นคิด"

กำหนดให้รหัสพิเศษคือ รหัส 255 ในตารางแอสกี และกำหนดให้ตารางคำไทยมีคำ "การ" และ "คิด" ในตำแหน่งที่ 17 และ 18 ตามลำดับ ดังรูปที่ 4.4 (ข)

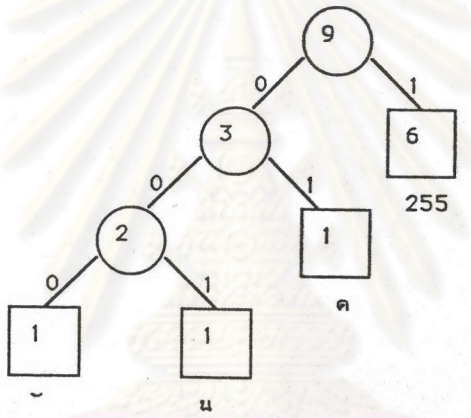
เมื่อนับความถี่ตามหลักการที่ปรับเปลี่ยนจะได้ความถี่แสดงดังรูปที่ 4.4 (ก) นำความถี่จากตารางสร้างฮัฟแมนทรีแสดงดังรูปที่ 4.4 (ค) และอ่านรหัสใหม่จากทรี พร้อมทั้งนับความยาวรหัสเก็บลงตารางรหัส

รหัส	ความถี่	รหัสใหม่	ความยาวรหัสใหม่
ค	1	01	2
น	1	001	3
ั	1	000	3
.	.	.	
.	.	.	
.	.	.	
.	.	.	
255	6	1	1

(ก) ตารางรหัส

รหัสคำ	คำไทย
.	.
.	.
.	.
17	การ
18	คิด
.	.
.	.
.	.
.	.
.	.

(ข) ตารางคำไทย



รูปที่ 4.4 (ค) ฮัฟแมนทรี

การเข้ารหัสจะเป็นการแทนรหัสเดิมของข้อมูลด้วยรหัสใหม่ ตามหลักการเดิมของอัลกอริทึมฮัฟแมนเป็นการแทนรหัส 1 ครั้งต่อ 1 อักษร แต่ตามหลักการใหม่นี้การแทนรหัสจะต้องตรวจสอบว่ากลุ่มอักษรที่เข้ารหัสนั้นตรงกับคำในตารางคำไทยหรือไม่ ถ้าไม่ตรงก็จะเป็นการแทนรหัสตามปกติเหมือนหลักการเดิม แต่ถ้ากลุ่มอักษรนั้นตรงกับคำในตารางไทย การแทนรหัสจะต้องทำ 2 ครั้ง ครั้งแรกเป็นการแทนรหัสพิเศษ และครั้งที่ 2 เป็นการแแทนรหัสของคำคำนั้น จากตัวอย่างข้อมูล "การค้นคิด" ที่มีรหัสใหม่ดังตารางในรูปที่ 4.4 (ก) และตารางคำไทยดังรูปที่ 4.4 (ข) จะได้ข้อมูลที่เข้ารหัสแล้ว ดังรูปที่ 4.5

1	00010001	01	000	001	1	00010010
255	(การ)	ค	ั	น	255	(คิด)

รูปที่ 4.5 ข้อมูลที่เข้ารหัส

การแทนรหัสคำ จะใช้ขนาดรหัสของคำกี่บิตขึ้นอยู่กับจำนวนคำไทยที่นำมาใช้ในการอัดข้อมูล ถ้าต้องการแแทนรหัสคำมากขึ้นก็สามารถทำได้โดยการเพิ่มขนาดของรหัส แต่ประสิทธิภาพการอัดข้อมูลอาจลดลงก็ได้ เพราะถ้าเราพิจารณาค่าเปอร์เซ็นต์สะสมจากตารางที่ 2 ในภาคผนวก พบว่าคำ 100 อันดับแรกมีเปอร์เซ็นต์สะสมสูงถึง 48.70 เปอร์เซ็นต์ หมายความว่า ข้อความภาษาไทยจะประกอบด้วยคำเหล่านี้ 48.70 เปอร์เซ็นต์ ดังนั้นในการวิจัยครั้งนี้ใช้ขนาดรหัสของคำเพียง 2 ขนาดคือ 8 บิตและ 9 บิต โดยที่

- รหัส 8 บิต จะใช้แทนค่าของคำได้ทั้งหมด 255 คำ (ไม่ใช้ค่า 0) มีจำนวนเปอร์เซ็นต์สะสมเท่ากับ 65.61 เปอร์เซ็นต์

- รหัส 9 บิต จะใช้แทนค่าของคำได้ทั้งหมด 511 คำ (ไม่ใช้ค่า 0) มีจำนวนเปอร์เซ็นต์สะสมเท่ากับ 78.55 เปอร์เซ็นต์ (มากกว่ารหัส 8 บิต 12.94 เปอร์เซ็นต์)

เนื่องจากคำไทยที่ใช้ในการตรวจสอบในการวิจัยนี้ จะใช้ไทร์ (Trie) เป็นโครงสร้างข้อมูลที่ใช้เก็บคำไทย โดยที่รหัสของคำจะถูกเก็บไว้ที่ลีฟเสมอและโหนดภายในของไทร์จะมีรหัสของคำเป็น 0 ดังนั้นเราจึงไม่ใช้ค่า 0 เป็นรหัสของคำ (ดูรายละเอียดโครงสร้างข้อมูลที่ข้อ 4.1.4)

4.1.2 อัลกอริทึมการขยายข้อมูล

รูปที่ 4.6 การขยายข้อมูลโดยอัลกอริทึมฮัฟแมนที่ปรับเปลี่ยน

อัลกอริทึมของการขยายข้อมูลที่ปรับเปลี่ยนแสดงดังรูปที่ 4.6 ขั้นตอนที่เพิ่มขึ้นมาคือ การสร้างตารางคำไทยที่เป็นชุดเดียวกันกับชุดที่ใช้ในขั้นตอนการอัดข้อมูล ต่อจากนั้นให้อ่านรหัสนำหน้าจากแฟ้มข้อมูลที่ถูกอัดเพื่อนำมาสร้างฮัฟแมนทรีที่มีลักษณะเหมือนกับทรีที่สร้างในการอัดข้อมูลทุกประการ ขั้นตอนสุดท้ายคือ การถอดรหัสข้อมูลจะเป็นการแทนรหัสให้กลับคืนสู่รหัสเดิมตามที่ปรากฏในข้อมูลต้นกำเนิด

การถอดรหัสแต่ละรหัส เป็นการเดินตามทรีจากรูทไปยังลีฟทิศทางการเดินจะกำหนดได้จากการอ่านข้อมูลทีละ 1 บิต ถ้าบิตมีค่าเป็น 0 จะเดินไปหาโหนดลูกทางขวาและถ้าบิตเป็น 1 เดินไปหาโหนดลูกทางซ้าย พร้อมกับให้มีการตรวจสอบด้วยว่าโหนดนั้นเป็นลีฟหรือไม่ ถ้าไม่ใช่ลีฟให้อ่านข้อมูลบิตถัดไป แต่ถ้าโหนดนั้นเป็นลีฟสามารถแทนรหัสได้ทันที ถ้ารหัสที่แทนที่ได้เป็นรหัสพิเศษแสดงว่ารหัสที่ตามมาเป็นรหัสคำ ดังนั้นให้แทนรหัสคำตามขนาดของรหัสคำที่ใช้ในขั้นตอนการเข้ารหัส หลังจากแทนรหัสได้ในแต่ละรหัสให้กลับมาเริ่มต้นที่รูท ทำซ้ำขั้นตอนเช่นนี้ไปเรื่อย ๆ จนจบข้อความที่ถอดรหัส

ตัวอย่าง การถอดรหัสโดยฮัฟแมนทรีจากรูปที่ 4.4 (ค) และตารางคำไทยดังรูปที่ 4.4 (ข)

- อ่านข้อมูลบิตแรกจากข้อมูลที่เข้ารหัสจากรูปที่ 4.5 คือ 1 แสดงว่าต้องเดินไปทางซ้าย ปรากฏว่าโหนดทางซ้ายนี้เป็นลีฟ และเป็นลีฟของรหัสพิเศษด้วย ดังนั้นแสดงว่าข้อมูลที่เข้ารหัสอีก 8 บิตถัดไปเป็นรหัสของคำ คือคำที่ 17 ของตารางคำไทย คือ "การ" เมื่อเดินไปถึงลีฟในการถอดรหัสแต่ละครั้งเราจะต้องกลับมาเริ่มต้นใหม่ที่รูท เพื่อถอดรหัสตัวถัดไป
- จากนั้นอ่านข้อมูลบิตถัดไป คือ 0 แสดงว่าต้องเดินไปทางขวา และไม่ใช่ลีฟ อ่านข้อมูลบิตถัดไป คือ 1 แสดงว่าต้องเดินไปทางซ้าย ปรากฏว่าโหนดนี้เป็นลีฟของรหัส "ค"
- ทำการถอดรหัสโดยวิธีการดังกล่าวจนจบก็จะได้ข้อมูลเดิมกลับคืนมา

4.1.3 การจัดเตรียมแฟ้มข้อมูลคำไทย

นำคำจากตารางที่ 2 จากภาคผนวกจัดเตรียมลงแฟ้มข้อมูล

4.1.4 โครงสร้างข้อมูลหลักสำหรับอัลกอริทึม

โครงสร้างข้อมูลที่ใช้มี 3 โครงสร้างคือ

4.1.4.1 โครงสร้างทรีที่ใช้สำหรับการเข้ารหัส มีรายละเอียดดังนี้คือ

- รหัสของอักษร
- ตัวแปรที่ใช้เก็บความถี่ของรหัส
- รหัสใหม่ได้จากทรีที่สร้าง
- ความยาวของรหัสใหม่

- พอยเตอร์ชี้โหนดซ้าย
- พอยเตอร์ชี้โหนดขวา

4.1.4.2 โครงสร้างไทร์ เป็นโครงสร้างข้อมูลที่ใช้เก็บคำไทยจากแฟ้มข้อมูลคำไทย

```
struct branchnode {
        int num;
        struct branchnode *ptr[77];
}
struct infnode {
        int num;
        unsigned char *key;
}
```

โครงสร้าง branchnode เป็นโหนดของไทร์ ซึ่งมีตัวแปรชี้ ptr สำหรับชี้ไปยังโหนดย่อยของแต่ละโหนดจำนวนเท่ากับจำนวนอักษรไทยรหัสส.ม.อ. 76 โหนด และเพิ่มอีกหนึ่งโหนดเป็นโหนด ptr[0] เพื่อใช้ในการชี้ไปยังโครงสร้าง infnode สำหรับตัวแปร num มีค่าเท่ากับ 0 เสมอเพื่อใช้แสดงว่าเป็นโหนดของ branchnode

โครงสร้าง infnode ทำหน้าที่เป็นลีฟของไทร์ ประกอบด้วยตัวแปร num จะเป็นค่าของรหัสของคำไทย และตัวแปร key ทำหน้าที่ชี้ไปยังคำไทยแต่ละคำของตัวแปร num (Ammeraal 1987 : 149-157)

4.1.4.3 โครงสร้างทรีที่ใช้ในการถอดรหัส มีรายละเอียดดังนี้

- รหัสของอักษร
- ตัวแปรที่ใช้เก็บความถี่ของรหัส
- พอยเตอร์ชี้โหนดซ้าย
- พอยเตอร์ชี้โหนดขวา

4.2 การปรับเปลี่ยนอัลกอริทึมแอลแซดดับบลิว

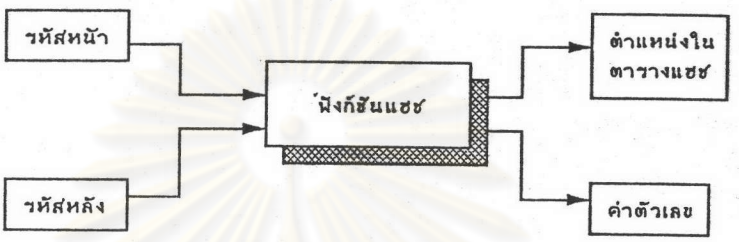
4.2.1 โครงสร้างข้อมูลหลักสำหรับอัลกอริทึม

โครงสร้างข้อมูลที่ใช้ในการอัดข้อมูล มี 2 โครงสร้าง คือ

4.2.1.1 ตารางรหัส จะกำหนดให้มีรหัสเริ่มต้นอยู่ 256 รหัสตามค่าในตารางแอสกี เมื่อทำการอัดข้อมูลจะเรียนรู้ข้อมูล โดยการเปลี่ยนกลุ่มข้อมูลให้อยู่ในลักษณะ <รหัสหน้า><รหัสท้าย> ในที่นี้เราจะใช้ขนาดตารางที่เก็บรหัสได้ 4096 รหัส ซึ่งกำหนดจากความ

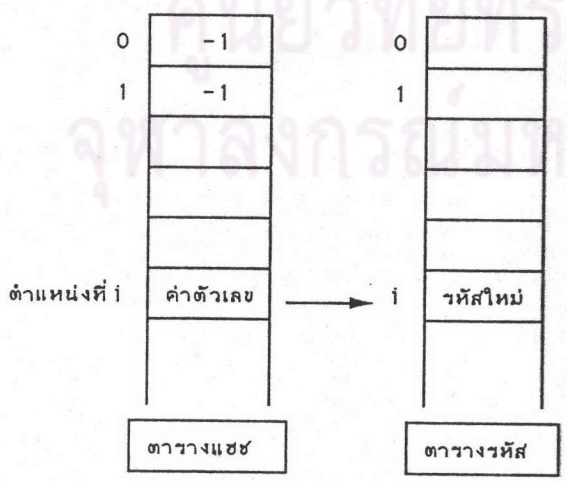
ยาวสูงสุดของรหัสหน้า 12 บิต

4.2.1.2 ตารางแฮช เป็นตารางที่ใช้เก็บค่าตัวเลขที่คำนวณมาโดยฟังก์ชันแฮชจากค่าของ <รหัสหน้า> และ <รหัสท้าย> ค่าที่ได้คือ ตำแหน่งในตารางแฮช และค่าตัวเลข ดังรูปที่ 4.7



รูปที่ 4.7 ฟังก์ชันแฮช

ตารางแฮช กำหนดให้มีค่าเริ่มต้น -1 ทุกตัว <รหัสหน้า><รหัสท้าย> จะเป็นรหัสใหม่ ถ้าตำแหน่งที่คำนวณได้ในตารางแฮชมีเป็นค่า -1 ดังนั้นให้เก็บค่าตัวเลขลงตารางแฮช และเก็บรหัสใหม่ลงตารางรหัส ดังรูปที่ 4.8



รูปที่ 4.8 ตารางที่ใช้ในการเข้ารหัส

ตำแหน่ง	รหัสหน้า	รหัสหลัง
0	-1	0
1	-1	1
.	.	.
.	.	.
.	.	.
255	-1	255
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
4094		
4095		

รูปที่ 4.9 ตารางที่ใช้ในการถอดรหัส

สำหรับโครงสร้างข้อมูลที่ใช้ในการขยายข้อมูล มีโครงสร้างเดียวคือ ตารางรหัส ซึ่งกำหนดรายละเอียดไว้ 2 ส่วน ส่วนแรก เป็นค่ารหัสหน้า และส่วนหลัง เป็นค่ารหัสหลัง ตารางรหัสสามารถเก็บรหัสได้ 4096 รหัส โดยที่รหัส 256 รหัสแรกของตาราง (0-255) ไม่มีค่าของรหัสนำหน้าจึงกำหนดให้มีค่าเป็น -1 ดังรูปที่ 4.9

4.2.2 การจัดเตรียมแฟ้มรหัสคำไทย

เนื่องจากอัลกอริทึมของวิธีการนี้ ใช้คุณสมบัติการนำหน้าในการสร้างรหัส ดังนั้นคำจากตารางคำไทยต้องสร้างให้อยู่ในรูปแบบใหม่ตามขั้นตอนดังนี้

1. กระจายกลุ่มอักษรของคำจากตัวแรกและตัวถัดไปจนถึงตัวสุดท้ายของคำ ตัวอย่างการกระจายคำไทย ดังรูปที่ 4.10

2. เปลี่ยนรูปของกลุ่มอักษรให้เป็นรูปแบบของ <รหัสหน้า><รหัสท้าย> นำไปผ่านฟังก์ชันแฮชได้ตำแหน่งและค่าในตารางแฮช

3. เก็บรูปแบบของ <รหัสหน้า><รหัสท้าย> และค่าที่ได้จากฟังก์ชันแฮชเก็บลงแฟ้มข้อมูลรหัส แฟ้มข้อมูลนี้จะนำไปใช้ทั้งการอัดข้อมูลและขยายข้อมูล โดยที่การอัดข้อมูลจะใช้ค่าของฟังก์ชันแฮช และการขยายข้อมูลจะใช้ค่าของ <รหัสหน้า><รหัสท้าย>

คำไทย	กลุ่มอักษร
กรรม	กร
	กรร
	กรรม
กระ	กระ (กร-มีแล้ว)
กล่าว	กล
	กล่
	กล่า
	กล่าว

รูปที่ 4.10 การกระจายคำไทย

รหัสแอสกี	
**	**รหัสที่ได้จากแฟ้มข้อมูลรหัสของคำไทย
รหัสที่ได้จากข้อมูล	

รูปที่ 4.11 ตารางรหัสของอัลกอริทึมแอลแซดดับบลิวที่ปรับเปลี่ยน

4.2.3 อัลกอริทึมการอัดข้อมูลและอัลกอริทึมการขยายข้อมูล

หลักการอัดข้อมูลของอัลกอริทึมแอลแซดดับบลิว รหัสใหม่จะได้จากการเรียนรู้

จากข้อมูลที่กำลังทำการอัดอยู่ แต่สำหรับหลักการใหม่นั้นจะกำหนดให้อัลกอริทึมมีการเรียนรู้รหัสคำไทยจากแฟ้มข้อมูลรหัสที่ได้จัดเตรียมไว้ก่อนตามขั้นตอนที่ 4.2.2 แล้วจึงเรียนรู้รหัสใหม่อื่น ๆ จากข้อมูลตามหลักการของอัลกอริทึมเดิม ดังรูปที่ 4.11

ตัวอย่าง สมมติแฟ้มข้อมูลหนึ่งมีข้อความดังนี้ "การงานการบ้านการงาม" นำมาเข้ารหัสโดยอัลกอริทึมแอลแซดดับบลิว จะได้ตารางรหัสและข้อความที่ถูกกระจายเพื่อเข้ารหัสด้วยรหัสเริ่มต้นขนาด 9 บิต ดังรูปที่ 4.12 (ก) และ 4.12 (ข) ตามลำดับ

0 255	รหัสแอสกี (รหัสภาษาไทย ส.ม.อ.)
256	กา -> 161า
257	าร -> 210ร
258	รง -> 195ง
259	งา -> 167า
260	าน -> 210น
261	นก -> 185ก
262	การ -> 256ร
263	รบ -> 195บ
264	บ้ -> 186◌้
265	้า -> 233า
266	านก -> 260ก
267	การง -> 262ง
268	งาม -> 259ม

รูปที่ 4.12 (ก) ตารางรหัสที่สร้างจากอัลกอริทึมแอลแซดดับบลิว

ก	า	ร	ง	า	น	กา	ร	บ	◌้	าน	การ	งา	ม

รูปที่ 4.12 (ข) ข้อมูลที่เข้ารหัสโดยอัลกอริทึมแอลแซดดับบลิว

เมื่อนำมาเข้ารหัสโดยอัลกอริทึมแอลแซดดับบลิวที่ปรับเปลี่ยน จะได้ตารางรหัสและข้อความที่ถูกกระจายเพื่อเข้ารหัสด้วยรหัสเริ่มต้นขนาด 10 บิต ดังรูปที่ 4.13 (ก) และ 4.13 (ข) ตามลำดับ

0 255	รหัสแอสกี (รหัสภาษาไทย ส.ม.อ.)
256 257 ... 274 275 ... ... 323 324 ... 780	.... กา -> 161า การ -> 274ร .... งา -> 167า งาน -> 323น ....
781 782 783 784 785 786	การง -> 275ง งานก -> 324ก การบ -> 275บ บ้านก -> 456ก การงา -> 781า าม -> 210ม

รูปที่ 4.13 (ก) ตารางรหัสที่สร้างจากอัลกอริทึมแอลแซดดับบลิวที่ปรับเปลี่ยน

ก า ร	ง า น	ก า ร	บ้ า น	ก า ร ง	า	ม

รูปที่ 4.13 (ข) ข้อมูลที่เข้ารหัสโดยอัลกอริทึมแอลแซดดับบลิวที่ปรับเปลี่ยน

4.3 การวิเคราะห์เชิงประสิทธิภาพของอัลกอริทึมฮัฟแมนที่ปรับเปลี่ยน

4.3.1 ประสิทธิภาพการอัด

ประสิทธิภาพการอัดจะมากหรือน้อย ขึ้นอยู่กับชีแมนติคคำไทยในข้อความ และความยาวของคำที่พบด้วย ถ้าพบชีแมนติคคำไทยมากจะได้ประสิทธิภาพการอัดมากแต่ถ้าพบชีแมนติคคำไทยน้อยจะได้ประสิทธิภาพการอัดน้อย อย่างไรก็ตามในกรณีที่ไม่พบชีแมนติคคำไทยในข้อความ ประสิทธิภาพการอัดจะยังคงเท่ากับวิธีการเดิม

4.3.2 การแทนรหัส

ตามหลักการเดิม

จำนวนครั้งของการแทนรหัส = O(n)

เมื่อ n เป็นจำนวนตัวอักษรทั้งหมดของข้อความ

สำหรับหลักการที่ปรับเปลี่ยน

จำนวนครั้งของการแทนรหัส = O(b) + O (2w)

เมื่อ b เป็นจำนวนตัวอักษรทั้งหมดที่ไม่ตรงกับชีแมนติคคำไทย

w เป็นจำนวนคำทั้งหมดที่ตรงกับชีแมนติคคำไทย

4.3.3 เวลา

เวลาที่ใช้ในการนับความถี่และการเข้ารหัสจะใช้เวลามากขึ้นกว่าวิธีการเดิม เพราะต้องใช้เวลาส่วนหนึ่ง ในการตรวจสอบข้อมูลกันชีแมนติคคำไทยด้วย ส่วนเวลาที่ใช้ในการถอดรหัสจะลดลง เพราะจำนวนครั้งของการแทนรหัสน้อยลง

4.4 การวิเคราะห์เชิงประสิทธิภาพของอัลกอริทึมแอลแซดดับบลิวที่ปรับเปลี่ยน

4.4.1 ประสิทธิภาพการอัด

ตามหลักการเดิมของอัลกอริทึมแอลแซดดับบลิว ซึ่งเป็นวิธีการที่สามารถเรียนรู้ลักษณะของข้อมูลได้เองโดยที่ไม่ต้องศึกษาชีแมนติคคำไทย แต่จะให้ประสิทธิภาพการอัดจะต่ำในช่วงแรกและประสิทธิภาพการอัดจะไม่ดีถ้าข้อความนั้นมีขนาดไม่ใหญ่มากพอ ดังนั้นการสร้างรหัสคำไทยลงตารางรหัสก่อนเข้ารหัสข้อมูล จะมีผลทำให้ประสิทธิภาพการอัดของวิธีการนี้ไม่ขึ้นกับขนาดของแฟ้มข้อมูล นอกจากนี้ยังสามารถเพิ่มประสิทธิภาพการอัดข้อมูลขึ้นจากเดิม

4.4.2 การแทนรหัส

การแทนรหัสของอัลกอริทึมแอลแซดดับบลิว เป็นการแทนรหัสแบบแปรได้-แปรได้ จำนวนครั้งของการแทนรหัสไม่คงที่ขึ้นอยู่กับลักษณะข้อมูล แต่อย่างไรก็ตามหลักการที่ปรับเปลี่ยน จะมีจำนวนครั้งในการแทนรหัสน้อยกว่าหลักการเดิมเสมอ ในกรณีที่ข้อมูลที่เข้ารหัสมี

ลักษณะไม่ตรงกับรหัสของชีแมนติคคำไทย จำนวนครั้งของการแทนรหัสจะเท่ากับหลักการเดิม

4.4.3 เวลา

ตามหลักการเดิม จะใช้เวลาในการเข้ารหัสตามจำนวนครั้งของการแทนรหัสเพียงอย่างเดียว สำหรับหลักการที่ปรับเปลี่ยนจะต้องใช้เวลาส่วนหนึ่งในการสร้างชีแมนติคคำไทยลงตารางรหัสแต่เวลาที่ใช้ในการเข้ารหัสจะลดลงจากเดิม เพราะจำนวนครั้งการแทนรหัสจะน้อยกว่าจำนวนครั้งการแทนรหัสของหลักการเดิม