บทที่ 3

โครงสร้างการจัดเก็บและสืบค้นข้อมูล

การสืบค้นข้อมูลจะเร็วหรือช้าขึ้นกับโครงสร้างการจัดเก็บข้อมูลและอัลกอริทึมที่ใช้ในการสืบค้น ในบทนี้จะกล่าวถึงวิธีการจับเก็บและสืบค้นข้อมูลแบบต่างๆ และรวมทั้งอัลกอริทึมที่มีผู้ใช้ในการพัฒนาพจนานุกรมอิเล็กทรอนิกส์

การสืบค้นข้อมูลมีอยู่หลายวิธี ซึ่งแต่ละวิธีล้วนมีข้อดีข้อด้อยแตกต่างกัน ดังนี้

1. **การสืบค้นข้อมูลหรือข่าวสารแบบซีเควนเชียล** (Sequential Search) การสืบค้นข้อมูลวิธีนี้จะต้องทำการเปรียบเทียบระหว่างคีย์ กับค่าต่างๆในตารางตั้งแต่ค่าแรกจนถึงค่าที่ตรงกับคีย์ ทำให้ต้องมีการเปรียบเทียบกับค่าในตารางเกือบทุกตัว จึงจะรู้ว่าคีย์นั้นๆอยู่ในตารางหรือไม่ ฉะนั้นสามารถสรุปได้ว่า

กรณีดีที่สุด	ต้องเปรียบเทียบ 1 ครั้ง (คีย์ อยู่ที่ตัวแรก)
กรณีเลวร้ายที่สุด	ต้องเปรียบเทียบ n ครั้ง (คีย์ อยู่ที่ปลายแถว)
กรณีเฉลี่ยทั่วไป	ประมาณ n/2 ครั้ง

ข้อดีของการสืบค้นแบบซีเควนเชียล คือ ตารางที่ใช้เป็นตารางธรรมดา ข่าวสารในตารางไม่ต้องเรียงลำดับจากมากไปน้อยหรือน้อยไปมาก ดังนั้นการเพิ่มข่าวสารหรือนำข่าวออกจากตารางจะสะดวกมากถ้าตารางนั้นเป็นแบบลิงค์ลิสต์ (แทนที่จะเป็นอะเรย์ธรรมดา)

ถึงแม้ว่าการค้นหาแบบซีเควนเชียลตามที่กล่าวมาแล้วจะสะดวกเพราะค่าที่เก็บในตารางไม่ต้องเรียงตามค่าคีย์ แต่ว่าเวลาที่ใช้เพื่อหาคีย์ใดคีย์หนึ่งจะประมาณเท่ากับเวลาที่สูญเสียไปในการเปรียบเทียบค่า n ครั้ง ถ้าในตารางนั้นมีค่าคีย์อยู่ n ค่า นอกจากนี้ขนาดของตารางจะต้องกำหนดตั้งแต่เริ่มแรก ฉะนั้นการเพิ่มค่ามากๆอาจมีผลทำให้เกินตารางที่กำหนดไว้ในตอนแรกได้

2. **การค้นหาข่าวสารแบบขจัดครึ่ง** (Binary Search)

การค้นหาแบบไบนารี (binary search) หรือแบบขจัดครึ่ง ตารางที่ใช้กับอัลกอลิทึมแบบนี้ ค่าคีย์ต่างๆ ต้องเรียงลำดับจากมากไปน้อย (หรือน้อยไปมาก) การใช้เทคนิคนี้จะทำให้สามารถค้นหาค่าใดค่าหนึ่งในเวลาประมาณเท่ากับ $\log_2 n$ หลักการ "ขจัดครึ่ง" คือ ถ้าให้คีย์เรียงตามลำดับจากน้อยไปมากเป็น $(k_1, k_2, k_3, \ldots\ldots k_n)$ และถ้าเราต้องการหาว่า k_i อยู่ในชุดนี้หรือไม่ ก็ให้นำค่า k_i ไปเปรียบเทียบกับค่า k_{mid} ซึ่งก็คือค่าคีย์ที่อยู่ ณ จุดกึ่งกลางของชุดของคีย์ดังกล่าว ถ้า $k_i < k_{mid}$ แสดงว่า k_i ต้องอยู่ในครึ่งแรก แต่ถ้า $k_i > k_{mid}$ แสดงว่า k_i ต้องอยู่ในครึ่งหลัง ดังนั้นโดยอาศัยการเปรียบเทียบเพียงหนึ่งครั้ง เราก็

สามารถขจัดครึ่งหนึ่งของชุดคีย์ได้ โดยไม่ต้องนำมาพิจารณาต่อ เมื่อทราบว่า k_j อยู่ในส่วนครึ่งใด ก็ให้ทำแบบเดิมอีกสำหรับครึ่งนั้น เราก็สามารถขจัดครึ่งของชุดคีย์ในครึ่งนั้นโดยไม่ต้องนำมาพิจารณาอีก ในที่สุดก็สามารถหาได้ว่าค่าคีย์ k_j อยู่ในตารางนั้นหรือไม่ หลังจากทำการเปรียบเทียบไปแล้วประมาณ $\log_2 n$ ครั้ง ข้อเสียของการค้นหาแบบนี้คือ เวลาที่ใช้ในการค้นหาไม่แน่นอน จะใช้เวลาน้อยที่สุดเมื่อการเปรียบเทียบครั้งแรกพบคีย์ที่ต้องการ และหากไม่มีคีย์ที่ต้องการจะต้องทำการเปรียบเทียบไปจนข้อมูลหมด นอกจากนี้โครงสร้างข้อมูลสำหรับไบนารี ทรี จะต้องเรียงลำดับข้อมูล การเพิ่มหรือลบข้อมูลจึงค่อนข้างยุ่งยาก

3. การค้นหาข่าวสารแบบบี-ทรี (B-Tree)

การค้นหาข่าวสารแบบบี-ทรี ถูกตีพิมพ์ครั้งแรกโดยไบเยอร์และแมคไครส์ ในปี 1972 โครงสร้างข้อมูลต้นไม้แบบบี-ทรี เป็นโครงสร้างข้อมูลที่ใช้กันแพร่หลายมากในระบบฐานข้อมูลขนาดใหญ่ โครงสร้างข้อมูลแบบบี-ทรี มีคุณสมบัติ ดังนี้ (สุชาย ธนวเสถียร และวิชัย จิวังกูร, 2521)

3.1 ส่วนโหนดราก ถ้ามีโหนดลูกหลานแล้วจะต้องมีอย่างน้อยสองโหนดตามลงมา

3.2 โหนดอื่นๆที่ไม่ใช่โหนดรากหรือโหนดใบ จะมีโหนดลูกไม่น้อยกว่า $m/2$ สำหรับบี-ทรีที่มีลำดับ m เป็นเอ็มเวย์โหนด

3.3 โหนดใบต้องอยู่ในระดับเดียวกัน

เวลาในการค้นหาข่าวสารแบบนี้ คือ $\log n + 1$ เมื่อ n หมายถึง จำนวนคีย์ในบี-ทรี ข้อดีของบี-ทรี คือ เป็นโครงสร้างข้อมูลที่ใช้สำหรับจัดเก็บข้อมูลบนสื่อบันทึกข้อมูลประเภทดิสก์ จะใช้เวลาในการสืบค้นไม่มากนัก การเก็บข้อมูลไม่จำเป็นต้องรู้ถึงวิธีการจัดเก็บข้อมูลของสื่อบันทึกข้อมูล ส่วนข้อเสียของวิธีนี้ คือ การติดตั้งยุ่งยากและเวลาที่ใช้ในการค้นหาแปรตามขนาดของข้อมูล นอกจากนี้การเพิ่มหรือลบคีย์จะทำให้เกิดการเคลื่อนย้ายข้อมูลบ่อย และอีกประการหนึ่งก็คือ การกำหนดขนาดของโหนดจะต้องนำจำนวนระเบียนที่จัดเก็บในโหนดมาพิจารณาร่วมด้วย

4. การค้นหาข่าวสารแบบแฮชชิง (Hashing)

แฮชชิง เป็นเทคนิคการสร้างตารางเก็บคีย์แบบหนึ่งที่ช่วยให้การค้นหาคีย์ในตารางนั้นง่ายและรวดเร็ว แฮชชิงอาศัยเทคนิคการแปลงค่าคีย์เป็นแอดเดรสหรือที่อยู่ของคีย์นั้น การแปลงค่านี้ต้องอาศัยเทคนิคหรือฟังก์ชัน $H(k)$ เป็นตัวช่วยหาแอดเดรสของค่าคีย์ k (ค่า $H(k)$ ที่ดีต้องสามารถกระจายค่าคีย์ไปยังแอดเดรสต่างๆ ไม่ซ้ำกันเลยหรือซ้ำกันน้อยที่สุด ข้อเสียของวิธีนี้คือ การหาฟังก์ชันที่ใช้หาแอดเดรสเพื่อให้ค่าคีย์กระจายตัวอย่างดีนั้นหายาก แต่ข้อดีของวิธีนี้คือ การค้นหาคีย์ในตารางทำได้อย่างรวดเร็ว

5. การจับคู่เหมือนข้อความอย่างรวดเร็ว (First String Pattern Matching)

การจับคู่เหมือนข้อความ (String Pattern Matching) เป็นที่สนใจของโปรแกรมเมอร์มานานแล้ว และได้มีการพัฒนาโครงสร้างข้อมูลและอัลกอริทึมเพื่อปรับปรุงให้สามารถสืบค้นได้อย่างรวดเร็วและใช้เนื้อที่จัดเก็บข้อมูลน้อย วิธีการหนึ่งก็ คือ การจับคู่เหมือนข้อความอย่างรวดเร็ว (Fast Pattern Matching in String)

การสืบค้นคำสำคัญภายในข้อความที่กำหนด โดยทั่วไปมักจะเปรียบเทียบตั้งแต่จุดเริ่มต้นของข้อความและหยุดเมื่อพบตัวอักษรที่ไม่ตรง และเริ่มเปรียบเทียบใหม่อีกจนกระทั่งสิ้นสุดข้อความหรือพบตัวอักษรที่ตรงกันทั้งหมด วิธีนี้ไม่ค่อยมีประสิทธิภาพมากนัก ตัวอย่างเช่น การหาคำสำคัญ aaaaaaab ในข้อความ aaaaaaaaaaaaab ซึ่งมีรูปแบบเป็น a^nb และ $a^{2n}b$ ตามลำดับ พบว่าต้องมีการเปรียบเทียบจำนวน $(n+1)^2$ ตัวอักษร นอกจากนี้วิธีการจัดเก็บข้อความที่อ่านผ่านไปแล้วก็ค่อนข้างยุ่งยากมาก

ในปี 1987 คนุท และมอริส (Knuth and Morris) ได้ค้นพบอัลกอริทึมการจับคู่เหมือนข้อความ โดยถ้าคำสำคัญมีความยาว m และข้อความมีความยาว n เวลาที่ใช้ในการสืบค้นคำสำคัญในข้อความเท่ากับ O(m+n) โดยไม่ต้องมีการบันทึกข้อความที่อ่านผ่านมาแล้ว วิธีนี้ในการอ่านข้อความจากแฟ้มข้อมูลจะใช้เพียง O(m) ตำแหน่งในหน่วยความจำหลัก และใช้เวลาเพียง $O(\log_m)$ เพื่อจะอ่านตัวอักษรตัวต่อไปเข้ามา

สมมุติว่าเรานำคำสำคัญวางบนข้อความที่ต้องการสืบค้น และเริ่มจากปลายด้านซ้ายสุดและเลื่อนไปเรื่อยๆ เช่น ต้องการหาคำว่า " abcabcacab " ในข้อความ " babcbabcabcaabcabcabcacabc "

```
a b c a b c a c a b
b a b c b a b c a b c a a b c a b c a b c a c a b c
^
```

จุดเริ่มต้นในการเปรียบเทียบคือ จุดที่ลูกศรชี้ในข้อความคือ อักษร b ส่วนอักษรเริ่มต้นของคำสำคัญคือ อักษร a ซึ่งพบว่าไม่ตรงกัน ฉะนั้นจะต้องเลื่อนคำสำคัญไปทางขวาหนึ่งตัวอักษร

```
  a b c a b c a c a b
b a b c b a b c a b c a a b c a b c a b c a c a b c
  ^
```

ในขั้นตอนนี้พบว่ามีอักษรที่ตรงกัน 3 ตัว คือ abc เริ่มไม่ตรงกันที่อักษรตัวที่ 4 จะได้ว่าตัวอักษรที่อ่านเข้ามาคือ abcx โดยที่ x ไม่เท่ากับ a ฉะนั้นจึงควรเลื่อนคำสำคัญไปอีก 4 ตัวอักษร

```
        a b c a b c a c a b
b a b c b a b c a b c a a b c a b c a b c a c a b c
        ^
```

เริ่มต้นเปรียบเทียบใหม่ พบว่าตัวอักษรตัวที่ 8 เริ่มไม่ตรงกัน ตัวอักษรที่อ่านเข้ามาได้ คือ abcabcax โดยที่ x ไม่เท่ากับ c ตอนนี้ควรจะเลื่อนคำสำคัญไปอีก 3 ตำแหน่ง

```
          a b c a b c a c a b
b a b c b a b c a b c a a b c a b c a b c a c a b c
            ^
```

เมื่อเปรียบเทียบแล้วพบว่าไม่ตรงทั้งหมดอีกเช่นกัน จึงควรเลื่อนคำสำคัญไปอีก 4 ตำแหน่ง

```
              a b c a b c a c a b
b a b c b a b c a b c a a b c a b c a b c a c a b c
              ^
```

ครั้งนี้พบว่าตำแหน่งที่ 8 เริ่มไม่ตรง จะต้องเลื่อนไปอีก 3 ตำแหน่ง

```
                  a b c a b c a c a b
b a b c b a b c a b c a a b c a b c a b c a c a b c
                  ^
```

ครั้งนี้พบว่าคำสำคัญตรงกับข้อความทั้ง 8 ตัวอักษร

วิธีการจับคู่คำเหมือนจะได้ผลดีถ้ามีตารางสำหรับจัดเก็บข้อมูลและจำนวนตำแหน่งที่ใช้ในการเลื่อนคำสำคัญ ถ้าเปรียบเทียบพบตัวอักษรที่ไม่ตรง ณ ตำแหน่ง j ของข้อความ pattern [j] จะเริ่มต้นเปรียบเทียบใหม่ที่ตำแหน่ง next [j] ฉะนั้นจะต้องเลื่อนคำสำคัญไปเท่ากับ pattern[j] - next[j]

ในแต่ละขั้นตอนในการเปรียบเทียบ สมมุติคำสำคัญถูกเลื่อนทั้งหมด n ครั้ง ฉะนั้นจะต้องมีทั้งหมด 2n ขั้นตอน การจับคู่เหมือนมีรูปแบบการทำงาน ดังนี้

1. นำคำสำคัญวางที่ตำแหน่งซ้ายสุดของข้อความ
2. เปรียบเทียบอักษรในคำสำคัญและอักษรในข้อความ

   ถ้าตรงกัน กลับไปท

   ถ้าไม่ตรงกัน ทำข้อ 3
3. เลื่อนคำสำคัญไปยังตำแหน่งถัดไป

   คนุทและมอริส ได้ใช้อะเรย์ 4 อะเรย์ คือ

Text[1:m]	ใช้สำหรับเก็บข้อความที่อ่านเข้ามา
Pattern[1:m]	ใช้สำหรับเก็บคำสำคัญ
Next[1:m]	ใช้สำหรับเก็บจำนวนตำแหน่งที่ใช้ในการเลื่อนคำสำคัญ
F[1:m]	ใช้สำหรับชี้ตำแหน่ง

ประสิทธิภาพของวิธีการนี้ขึ้นอยู่กับเนื้อที่จัดเก็บในหน่วยความจำหลักและใช้เวลาในการสืบค้นเท่ากับ O(m+n) ถ้าคำสำคัญยาว m และข้อความยาว n

6. การค้นหาข่าวสารแบบดิจิตอลเสิชทรี(Digital Search Tree)

จากเทคนิคการสืบค้นข้อมูลแบบอื่นๆที่ได้กล่าวข้างต้น การสืบค้นจะใช้คีย์(คำศัพท์ทั้งคำ) ในการเปรียบเทียบ แต่ในการสืบค้นแบบ ดี-เอส ทรี เป็นการใช้อักษรแต่ละตัวในคีย์มาเปรียบเทียบทีละตัว ฉะนั้นเวลาที่ใช้ในการสืบค้นจะขึ้นกับตัวอักษรที่ประกอบขึ้นมาเป็นคีย์ทำให้สามารถสืบค้นข้อมูลได้อย่างรวดเร็ว ในการแสดง ดี-เอส ทรี ในเชิงโปรแกรมจะใช้โครงสร้างข้อมูลแบบดับเบิลอะเรย์ทรัย (Double Array Trie) โครงสร้างข้อมูลแบบทรัยประกอบด้วยโหนดต่างๆ ซึ่งสร้างจากตัวอักษรกับตัวชี้ที่ชี้ไปยังโหนดที่เป็นโหนดลูก โครงสร้างข้อมูลแบบทรัยนี้เหมาะสำหรับจัดเก็บพจนานุกรม เนื่องจากเป็นโครงสร้างข้อมูลที่มีประสิทธิภาพในการสืบค้นข้อมูลแบบพรีฟิกซ์เสิชชิง (Prefix Searching) คือ เป็นการสืบค้นข้อมูลโดยระบุเฉพาะส่วนต้นของข้อมูลและปิดท้ายด้วย "*" เพื่อแสดงว่าส่วนที่เหลือของคำศัพท์เป็นอะไรก็ได้ ซึ่งจะได้ผลลัพท์เป็นคำที่ขึ้นต้นด้วยตัวอักษรที่ระบุทั้งหมด

จากวิธีการสืบค้นข้อมูลที่กล่าวมาข้างต้นนั้นเมื่อพิจารณาข้อดีข้อเสียของแต่ละวิธีแล้วพบว่าควรจะนำวิธีการสืบค้นข้อมูลแบบดิจิตอลเสิชทรีมาใช้ และใช้โครงสร้างการจัดเก็บข้อมูลแบบทรัย เนื่องจากเวลาที่ใช้ในการสืบค้นขึ้นกับความยาวของคีย์ที่ใช้ ซึ่งทำให้การสืบค้นเป็นไปอย่างรวดเร็ว และยังเหมาะกับการทำพรีฟิกซ์เสิชชิง ซึ่งเหมาะสมที่จะนำมาใช้กับพจนานุกรม

โครงสร้างของพจนานุกรมอิเล็กทรอนิกส์ที่มีผู้พัฒนา

ได้มีผู้คิดค้น พัฒนาโครงสร้างและวิธีการสืบค้นข้อมูลสำหรับพจนานุกรมอิเล็กทรอนิกส์ ไว้หลายท่านด้วยกัน ดังจะกล่าวต่อไปนี้

1. การพัฒนาพจนานุกรมอิเล็กทรอนิกส์โดยใช้เทคนิคการลดขนาดข้อมูล

เป็นผลงานวิจัยของ รศ. ยืน ภู่วรวรรณ และ ผศ.ดร.ชัยยงค์ วงศ์ชัยสุวัฒน์ โดยตีพิมพ์ในเอกสารประกอบการสัมมนาทางวิชาการ Proceeding of The Regionnal Workshop on Computer Processing of Asian Language (CPAL) เพื่อออกแบบและพัฒนาพจนานุกรมอิเล็กทรอนิกส์ ซึ่งพจนานุกรมดังกล่าวประกอบด้วยคำศัพท์ภาษาไทย ประเภทคำศัพท์ (category) และความหมายของคำศัพท์ โดยคำนึงถึงการเข้าถึงที่รวดเร็ว

(ยืนและชัยยงค์, 2535) เนื่องจากคำไทยมีรูปแบบลักษณะของการผสมคำค่อนข้างมาก เช่น ช่อง เป็นคำโดด และมีความหมาย แต่เมื่อผสมเป็น ช่องแคบ ช่องเขา และช่องไฟ จะได้ความหมายใหม่

หรือ คง เมื่อผสมคำอื่นเป็น คงคลัง คงเหลือ และคงตัว การผสมกับคำอื่นเป็นคำใหม่นี้จะทำให้เปลี่ยนรูปทางไวยากรณ์ไปด้วย เช่น คง เป็นกริยา ส่วนคงตัว เป็นวิเศษณ์ เป็นต้น โครงสร้างพจนานุกรมจะแยกเป็นคำหลัก และกลุ่มคำดังกล่าวข้างต้น โครงสร้างของพจนานุกรมจึงแยกออกเป็นหลายระดับ คือโครงสร้างระดับแรก จะเก็บข้อมูลในระดับคำ เพื่อประโยชน์ในการนำไปใช้ในกรณีที่ไม่ต้องคำนึงถึงรูปแบบไวยากรณ์หรือความหมายของคำ เช่น การตัดแบ่งคำในประโยค การตรวจสอบตัวสะกดในโปรแกรมประมวลผลคำต่างๆ เป็นต้น โครงสร้างที่จัดเก็บนี้เป็นโครงสร้างที่ลดขนาดข้อมูลโดยการจัดเก็บข้อมูลในลักษณะความยาวไม่คงที่ ขึ้นอยู่กับความยาวของข้อมูล และสามารถค้นหาได้อย่างรวดเร็วเพราะเป็นการค้นหาในหน่วยความจำ ด้วยหลักการของไบนารีเสิช ทรี

สำหรับคำหลักที่ไวยากรณ์และความหมายต่างกัน จะเก็บในสดมภ์ต่างกันและใช้ตัวชี้ ชี้ต่อไปยังคำหลักของแต่ละคำ การจัดเก็บข้อมูลของแต่ละคำหลักจะแยกเก็บเป็น 3 ระดับ ที่เชื่อมต่อกันด้วยรหัส โดยมีโครงสร้าง แบบทรีเชื่อมต่อกันทั้ง 3 ระดับ โดยการจัดเก็บข้อมูลในระดับที่สามเป็นสดมภ์ย่อยๆ ซึ่งสามารถเพิ่มสดมภ์ได้อีก แต่ในขั้นตอนนี้จัดเก็บเพียงแค่สดมภ์พื้นฐาน

การจัดรหัสในที่นี้ รหัสของแต่ละคำหลักใช้เลข 5 หลัก โดยแยกเข้าหาระดับที่ 2 แบบทรี (tree) เป็นแบบหนึ่งต่อหลาย (one to many) และเชื่อมโยงด้วยรหัสแบบไปกลับสองทิศทางได้ ทำนองเดียวกันระดับที่ 2 เชื่อมต่อกับระดับที่ 3 ด้วยโครงสร้างแบบสัมพันธ์ชนิดหนึ่งต่อหลายอีกเช่นกัน

<u>การลดขนาดข้อมูลเพื่อเพิ่มประสิทธิภาพการสืบค้นข้อมูล ในพจนานุกรมอิเล็กทรอนิกส์</u>

ข้อมูลที่จัดเก็บในพจนานุกรมจะมีการจัดเรียงตามลำดับของคำตามรหัสตัวอักษรที่วางไว้ เพื่อให้การค้นหาข้อมูลทำได้ง่ายขึ้น โครงสร้างแฟ้มข้อมูลจึงเน้นรูปแบบของข้อมูลที่อยู่ในหน่วยความจำ โครงสร้างที่ใช้จัดเก็บพจนานุกรม มีดังนี้

ก. โครงสร้างแบบ Table-Table-Linear Search

โครงสร้างแบบนี้จะลดขนาดข้อมูลด้วยการเก็บข้อมูลแบบเรียงต่อกัน ดังนี้

ก_____ 2กา 3กาก 3กาง 6กางเกง 3กาจ 5กาฬ_____

ด้วยวิธีการเรียงพจนานุกรมใช้วิธี Table-Table จะใช้ตารางนำหน้า 2 ตัวอักษร ดังนั้น โครงสร้างในพจนานุกรมจะลด 2 ตัวอักษรแรกมาอยู่ที่ตาราง ดังนี้

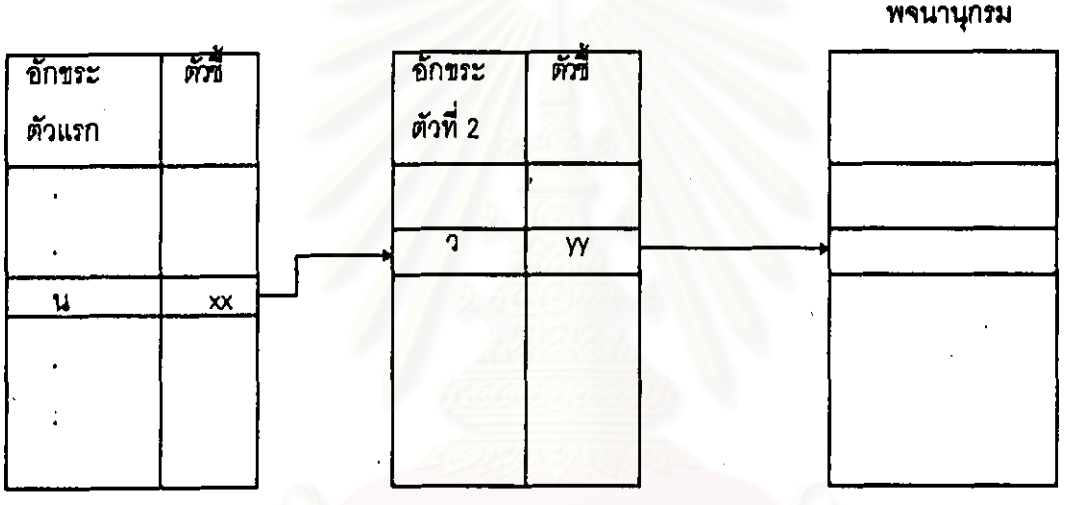
ก_____01ก.1ง 4งเกง 1จ 3ฬาด_____

ด้วยวิธีการลดขนาดของตัวอักษร 2 ตัวอักษรแรก มาที่ตาราง ทำให้ขนาดของพจนานุกรมจาก 84,642 ไบต์เหลือเพียง 22,339 ไบต์ โดยการจัดพจนานุกรมนี้จะใช้ตัวเลข 1 ไบต์ ซึ่งเป็นเลขไบนารีเป็นตัวคั่น ถ้าเป็นเลข 0 จะหมายถึง คำ 2 ตัวแรก

การค้นหาข้อมูลโดยวิธี Table-Table-Linear Search

ตารางระดับที่ 1 จะชี้ไปยังตารางในระดับที่ 2 เพื่อหาตำแหน่งของข้อมูลในหน่วยความจำ ซึ่งแต่ละตัวอักษรจะเป็นตัวชี้ขนาด 2 ไบต์ ดังนั้น ตารางในระดับที่ 1 จะมีขนาดข้อมูลเท่ากับ 49*3 คือ 147 ไบต์

ตารางระดับที่ 2 เป็นตารางที่เสมือนจะหาตำแหน่งคำในพจนานุกรม 2 ตัวแรก โดยตารางในระดับที่ 2 เป็นเสมือนตัวอักษรตัวที่ 2 ดังนั้น จึงมีจำนวนตารางทั้งหมด 49 ตารางในแต่ละตารางมีอักษรที่ใช้ 68 ตัว จำนวนข้อมูลที่ใช้ทั้งหมดจึงเท่ากับ 68*3*49 = 9,996 ไบต์ ตารางระดับที่ 2 นี้จะชี้ตำแหน่งของพจนานุกรมในตำแหน่งตัวอักษร 2 ตัวแรก สรุปวิธีการจัดเก็บพจนานุกรมวิธีนี้จะใช้จำนวนหน่วยความจำทั้งสิ้น 32,482 ไบต์



รูปที่ 3-1 โครงสร้างแบบเทเบิล-เทเบิล ไลน์เนียเสิช

ข. โครงสร้างข้อมูลแบบเทเบิล-อินเด็กซ์เสิช

โครงสร้างข้อมูลแบบนี้ ยึดหลักการเพื่อให้ง่ายต่อการลดขนาด โดยนำข้อมูลพจนานุกรมเดิมมาเรียงต่อกันเสมือนเป็นสตริงตัวหนึ่ง โดยมีตัวเลขคั่นระหว่างคำ ดังนี้

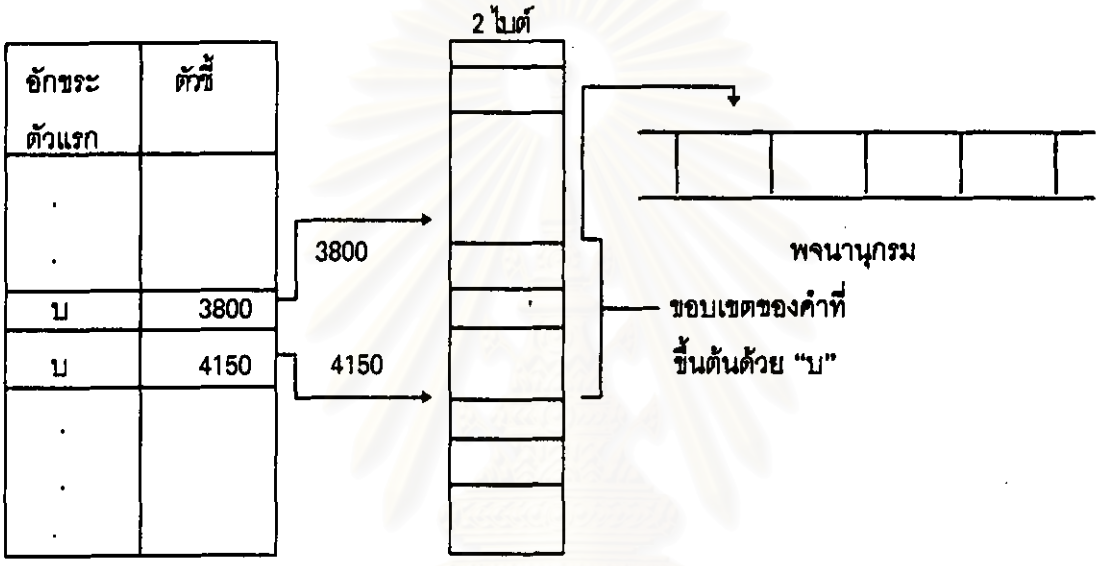
_____2ก 3กาก 3กาง 6กางเกง 3กาจ 5กาชาด_____

การค้นหาจะใช้ตารางคล้ายกับวิธีที่ 1 โดยใช้ตัวอักษรแรกปรากฏอยู่ในตารางขอบเขตของกลุ่มตัวอักษรเก็บไว้ในตารางที่ 2 ซึ่งเป็นตัวชี้หาตำแหน่งคำในพจนานุกรมที่เก็บเรียงเป็นสตริง ในการค้นหา เช่น ต้องการคำว่า บ้าง ซึ่งขึ้นต้นด้วยตัว "บ" การมองตารางระดับที่ 1 จะบอกได้ว่ากลุ่มตัวอักษรขึ้นต้น

ด้วยตัว บ อยู่ที่ตำแหน่ง 3,800 - 4,150 ดังนั้น เราจะทำ binary search ในตัวชี้ในตารางระดับ 2 ตั้งแต่ตำแหน่ง 3,800 - 4,150 เพื่อหาข้อมูลในพจนานุกรมที่เก็บแบบลดขนาด

ด้วยวิธีนี้จะมีขนาดข้อมูลในตารางระดับ 1 มี 147 ไบต์ ตารางระดับ 2 มี 10,800 ไบต์ และตัวอย่างพจนานุกรม 29,911 ไบต์ รวมเป็น 40,858 ไบต์

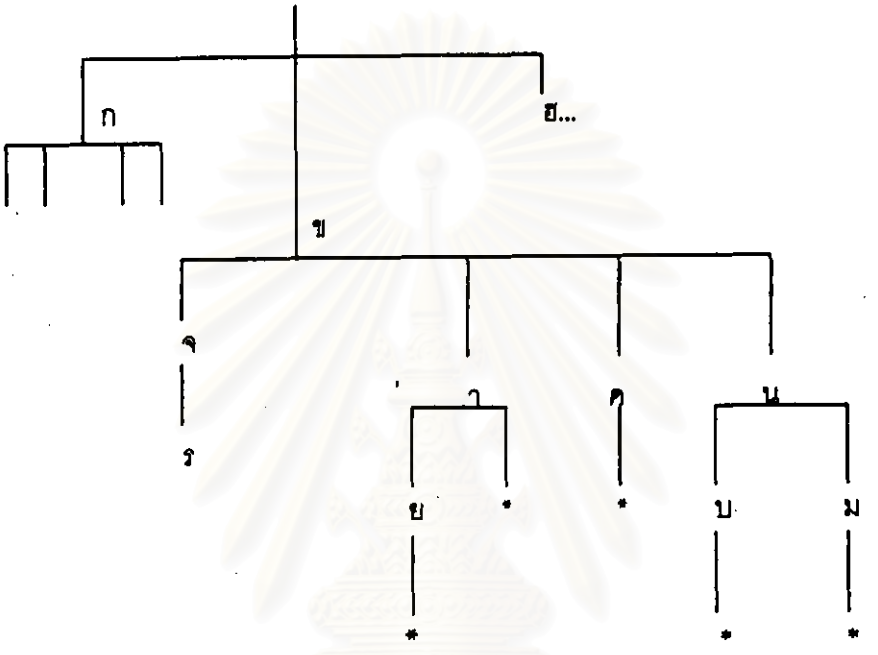
อนึ่ง การจัดเก็บข้อมูลแบบนี้สามารถลดขนาดหน่วยความจำได้อีก โดยไม่ต้องเก็บตัวเลขจำนวนไบต์และตัวอักษรตัวแรก แต่การค้นหาจะเสียเวลาเพิ่มขึ้น เพราะต้องมองหาตารางในลำดับต่อไปเพื่อทำการคำนวณจำนวนไบต์



รูปที่ 3-2 โครงสร้างข้อมูลแบบเทเบิล-อินเด็กซ์ เสิช

ค. โครงสร้างข้อมูลแบบต้นไม้

โครงสร้างข้อมูลแบบนี้ประกอบด้วยโหนด (node) ต่างๆ และโหนดเหล่านี้เป็นที่เก็บตัวอักขระและมีตัวชี้ ชี้ไปยังโหนดอื่น การจัดเก็บข้อมูลจะแยกเป็นกลุ่มๆโดยใช้อักขระตัวแรกของคำศัพท์ในการจัดกลุ่ม



รูปที่ 3-3 โครงสร้างข้อมูลพจนานุกรมแบบทรี

2. โครงสร้างพจนานุกรมซึ่งพัฒนาโดย ดร.รัตติกร วรากูลศิริพันธุ์ และสิงห์ ตรงงาม

(รัตติกร และสิงห์, 1995) จากงานวิจัยเรื่อง "ต้นแบบการเก็บบันทึกพจนานุกรมภาษาอังกฤษ-ไทย ด้วยระบบคอมพิวเตอร์ (A Prototype of Computerized English-Thai Dictionary) " จากงานวิจัยนี้ได้เสนอแนวทางแก้ไข การค้นหาคำศัพท์ด้วยการเก็บบันทึกพจนานุกรมในรูปฐานข้อมูลไว้ในหน่วยความจำของระบบคอมพิวเตอร์ โดยใช้หลักการประมวลผลภาษาธรรมชาติในการจัดเก็บบันทึกคำศัพท์เป็นหมวดหมู่ ทำให้เกิดความรวดเร็วในการค้นหาคำศัพท์และความหมาย งานวิจัยนี้ได้ออกแบบระบบพจนานุกรมอิเล็กทรอนิกส์สำหรับภาษาอังกฤษ-ไทย ประกอบด้วยการเก็บบันทึก การแก้ไข การลบและการค้นหาคำศัพท์

โครงสร้างของพจนานุกรม ประกอบด้วยโครงสร้าง 2 ส่วน ใหญ่ๆ ดังนี้

2.1 โครงสร้างของโปรแกรม

2.2 โครงสร้างของฐานข้อมูล

2.1 โครงสร้างของโปรแกรม คือชุดซอฟต์แวร์ ที่ใช้ในการจัดเก็บคำศัพท์และการค้นหาคำศัพท์ ฉบับอังกฤษ-ไทย บนเครื่องไมโครคอมพิวเตอร์ ประกอบด้วย 2 กลุ่มโปรแกรม ดังนี้

2.1.1 กลุ่มโปรแกรมต้นฉบับ เพื่อจัดเก็บคำศัพท์เป็นฐานข้อมูลและการบำรุงรักษากลุ่มโปรแกรมชุดนี้ ประกอบด้วย 5 โปรแกรมย่อย

2.1.1.1 โปรแกรมสร้าง Search tree และ Text block

2.1.1.2 โปรแกรมแก้ไข Search tree และ Text block

2.1.1.3 โปรแกรมลบ Search tree และ Text block

2.1.1.4 โปรแกรมค้นหาและแสดงคำศัพท์

2.1.1.5 โปรแกรมพิมพ์คำศัพท์

2.1.2 กลุ่มโปรแกรมเพื่อบริการผู้ใช้งาน มีลักษณะที่สำคัญ ดังนี้

2.1.2.1 เป็นโปรแกรมที่ฝังตัวในหน่วยความจำ และเรียกขึ้นมาใช้งานเมื่อไรก็ได้

2.1.2.2 ในการค้นหา สามารถหาเป็นกลุ่มคำได้ โดยใช้เครื่องหมาย "*" ช่วย เช่น ถ้าต้องการค้นหากลุ่มคำที่ขึ้นต้นด้วย an เราสามารถหาจาก an*

2.1.2.3 สามารถพิมพ์คำศัพท์ออกทางเครื่องพิมพ์

2.2 โครงสร้างของฐานข้อมูล คือแฟ้มข้อมูลที่ใช้ในการจัดเก็บคำศัพท์และการค้นหาคำศัพท์ ประกอบด้วยแฟ้มข้อมูล ดังนี้

2.2.1 แฟ้มข้อมูลพจนานุกรม ใช้เก็บคำศัพท์ทั้งหมดในพจนานุกรม คำศัพท์ทุกตัว จะมีตัวชี้โยงไปหาความหมายใน Text File การจัดเรียงลำดับของคำศัพท์ได้ใช้โครงสร้างของต้นไม้ (Tree Structure) โดยมี ตัวชี้โยงไปยังลำดับต่างๆของต้นไม้

2.2.2 แฟ้มข้อมูล Text File เป็นข้อมูลที่เก็บความหมายของคำศัพท์ที่จัดเก็บในระบบคอมพิวเตอร์ ข้อมูลนี้ได้จากโปรแกรมที่ใช้สร้างเลิชทรี และเท็กซ์บอกซ์ ซึ่งจะนำความหมายของคำศัพท์ผ่านขบวนการแปรรูป ตามรูปแบบที่กำหนด ทำให้พร้อมที่จะแสดงผลที่จอภาพหรือเครื่องพิมพ์ได้

2.2.1 โครงสร้างแฟ้มข้อมูลพจนานุกรม ใช้เก็บคำศัพท์เป็นแบบตารางดรรชนี(Table index Search) รวมกับไบนารีทรี ซึ่งโครงสร้างแบบนี้ง่ายต่อการลดขนาด และมีความเร็วในการค้นหาข้อมูล อยู่ในขั้นที่เร็วพอสมควร มีส่วนประกอบของโครงสร้าง ดังนี้

2.2.1.1 ส่วนตัวอักษรที่เป็นตัวชี้ (Letter Pointer) มีโครงสร้างการเก็บข้อมูลแบบตารางดรรชนี คือ อักษรแต่ละชุด จะมีค่าดรรชนีชี้ไปยังระเบียนเริ่มต้นของตัวอักษรชุดนั้น เพื่อเป็นการประหยัดเวลาในการค้นหาข้อมูล เพราะแทนที่จะเริ่มต้นที่คำศัพท์ตัวแรก ก็จะไปเริ่มต้นที่กลุ่มข้อมูลของคำศัพท์ที่ต้องการได้ทันที

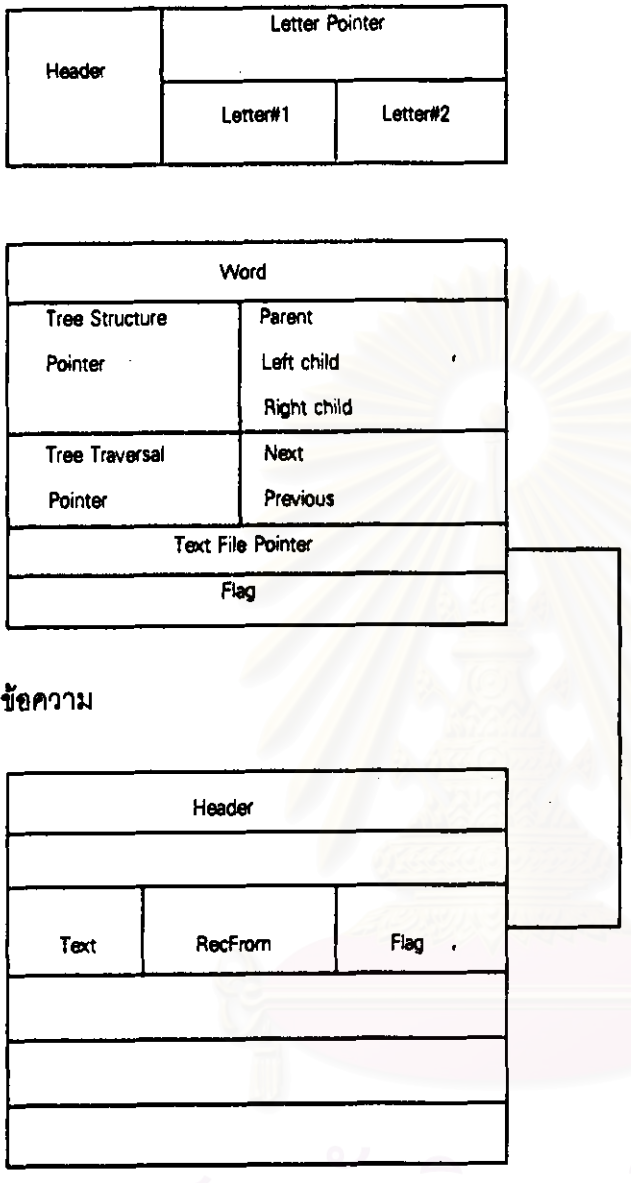
2.2.1.2 ส่วนของคำศัพท์ มีโครงสร้างการเก็บข้อมูลแบบไบนารี ทรี เพื่อสะดวกในการค้นหาคำศัพท์ที่ต้องการ สำหรับโครงสร้างส่วนนี้ จะมีการเก็บค่าต่อไปนี้

Word	ใช้เก็บคำศัพท์
Parent	ใช้เก็บหมายเลขระเบียน ของโหนดราก
Left Child	ใช้เก็บหมายเลขระเบียน ของคำที่อยู่ด้านซ้ายของโหนด
Right Child	ใช้เก็บหมายเลขระเบียน ของคำที่อยู่ด้านขวาของโหนด
Next	ใช้เก็บหมายเลขระเบียน ของคำที่ต่อจากคำนี้
Previous	ใช้เก็บหมายเลขระเบียน ของคำที่อยู่หน้าคำนี้
Text File Pointer	ใช้เก็บหมายเลขระเบียน ในแฟ้มข้อความ (Text File)
Flag	ใช้เก็บสถานะของระเบียนนี้

2.2.2 โครงสร้างแฟ้มข้อมูล Text File

Text	ใช้เก็บความหมายของคำศัพท์
RecFrom	ใช้เก็บหมายเลขระเบียน ของคำศัพท์ที่เป็นเจ้าของความหมาย
Flag	ใช้เก็บค่าสถานะภาพของระเบียนนี้

แฟ้มข้อมูลพจนานุกรม



แฟ้มข้อความ

รูปที่ 3-4 โครงสร้างแฟ้มข้อมูลของพจนานุกรมภาษาอังกฤษ-ไทย

2.3 การจัดเก็บและการสืบค้นคำศัพท์

ข้อมูลของพจนานุกรมจะถูกป้อนผ่านกลุ่มโปรแกรมต้นฉบับ ซึ่งเป็นโปรแกรมสำหรับจัดเก็บคำศัพท์และความหมายของคำศัพท์ พร้อมทั้งสร้างเสิชทรี และช่องสำหรับบันทึกข้อมูล โปรแกรมชุดนี้ประกอบด้วย 5 เมนูย่อยคือ บันทึกคำศัพท์ แก้ไขและลบคำศัพท์ ค้นหาคำศัพท์ พิมพ์คำศัพท์ และจบการทำงาน

2.3.1 วิธีการเก็บข้อมูล

คำศัพท์จะถูกป้อนจากเมนูบันทึกคำศัพท์ เมื่อได้คำศัพท์และความหมายที่ต้องการแล้ว ก็จะนำคำศัพท์ไปตรวจสอบก่อนว่า ซ้ำกับคำศัพท์เดิมหรือไม่ เนื่องจากโปรแกรมจะไม่ยอมให้ป้อนคำศัพท์ซ้ำ ในกรณีที่คำศัพท์ไม่ซ้ำ โปรแกรมจะนำคำศัพท์นี้ไปผ่านขบวนการสร้างตารางดรรชนี และไบนารีทรี ส่วนความหมายจะไปผ่านขบวนการสร้างเท็กซ์บอกซ์

2.3.2 วิธีการค้นหาคำศัพท์

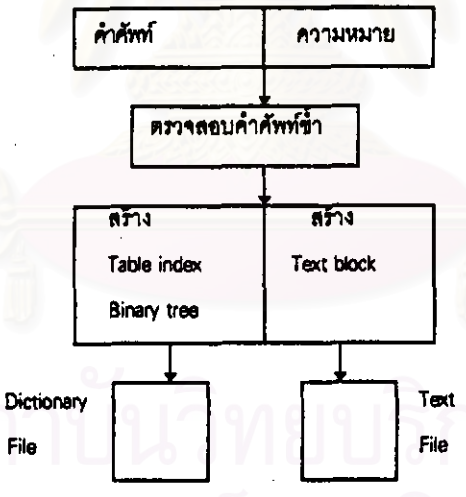
การค้นหาคำศัพท์ เลือกจากเมนูค้นหาคำศัพท์ ซึ่งมีขั้นตอน ดังนี้

2.3.2.1 พิมพ์คำศัพท์ที่ต้องการค้นหาความหมาย

2.3.2.2 นำอักษร 2 ตัวแรกไปเปิดตารางดรรชนี เพื่อหาตำแหน่งเริ่มต้นของ ดรรชนีแฟ้มข้อมูล (File Index)

2.3.2.3 นำคำศัพท์ที่ต้องการค้นหาความหมาย ไปเปรียบเทียบกับคำศัพท์ในดรรชนีแฟ้มข้อมูล เพื่อหาตำแหน่งของความหมายในแฟ้มข้อความ

2.3.2.4 นำความหมายแสดงผลบนจอภาพ



รูปที่ 3-5 แสดงวิธีการจัดเก็บข้อมูล