บทที่ 4

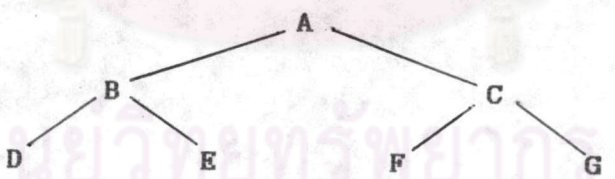
การออกแบบพจนานุกรมเสมือน

4.1 ลักษณะการจัดเก็บพจนานุกรม

ในการจัดเก็บพจนานุกรมจะต้องคำนึงถึงประสิทธิภาพในการดึงคำศัพท์ออกมาใช้คือ จะต้องพยายามหารูปแบบการจัดเก็บที่จะทำให้เกิดการขาดเพจ(page fault) น้อยๆ ในวิทยานิพนธ์ฉบับนี้ได้ทำการเปรียบเทียบ การจัดเก็บพจนานุกรม 2 แบบ คือ

4.1.1 การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางลึก(Depth First Order)

การเยี่ยมไปในแต่ละโหนดของทรายในทางลึก (Depth First Order) โดยจะเยี่ยมโหนดตามตัวเชื่อม(Link)ก่อนไปจนถึงโหนดสุดท้าย (Leave Node) แล้วจึงเยี่ยมโหนดถัดไปที่อยู่ในลำดับชั้นเดียวกัน (Next) เพื่อให้เข้าใจยิ่งขึ้นจะแสดงลำดับการเยี่ยมโหนดตามตัวอย่างต่อไปนี้

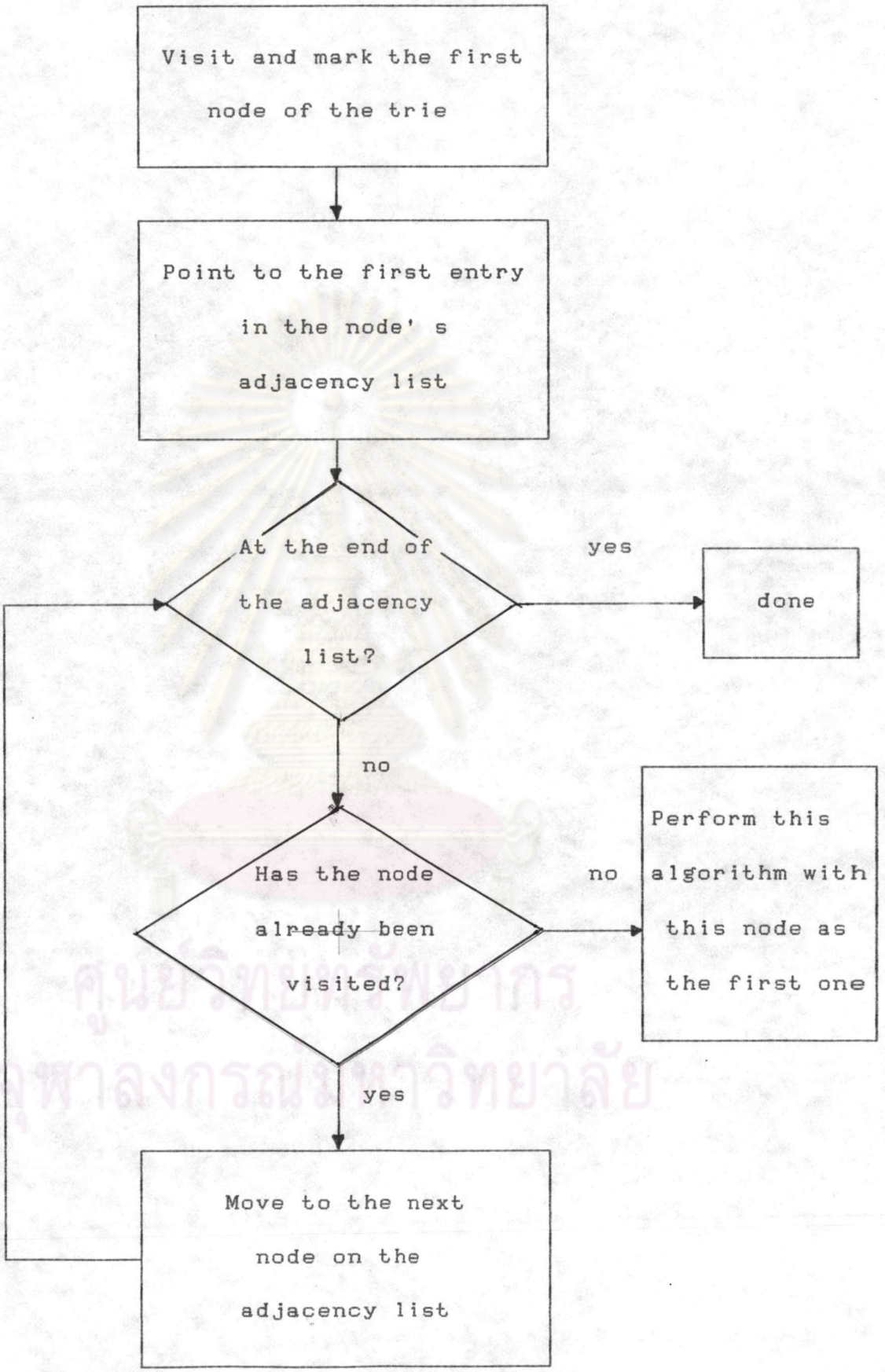


จากโครงสร้างทรายข้างต้น การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางลึก จะมีลำดับดังนี้ ABDECFG นั่นคือ เส้นทางการเยี่ยมโหนด จะเยี่ยมไปตามตัวเชื่อม (Link)จนกว่าจะถึงโหนดสุดท้าย จากนั้นเลือกโหนดถัดไปในลำดับชั้นเดียวกัน (Next) ทำเช่นนี้ไปจนเยี่ยมโหนดครบทุกโหนดในทราย

การเยี่ยมโหนดโดยวิธีนี้สามารถอธิบายโดยใช้อัลกอลิธึม ภาษาปาสคาลได้ดังนี้

```
procedure depth(var thisnode:nodeid);
var savenode : nodeid;
    adjptr : edgeptr;
begin {visit thisnode here}
    trie[thisnode].mark := 1;
    adjptr := trie[thisnode].adjlist;
    while adjptr <> nil
    do begin savenode := adjptr↑.node;
        if(trie[savenode].mark = 0)
        then depth(savenode);
        adjptr := adjptr↑.next
    end;
end;
```

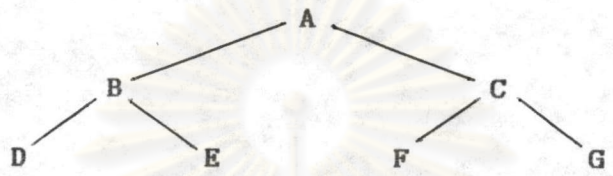
จากอัลกอลิธึมนี้สามารถแสดงเป็น ผังงาน (Flowchart) ได้ดังรูปที่ 4.1



รูปที่ 4.1 แสดงขั้นตอนการเยี่ยมโหนดในทางลึก

4.1.2 การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางกว้าง (Breath First Order)

การเยี่ยมไปในแต่ละโหนดของทรายในทางกว้าง (Breath First Order) โดยจะเยี่ยมโหนดตามลำดับชั้นเดียวกัน (Next) ก่อน จากนั้นจึงจะเยี่ยมไปตามตัวเชื่อม (Link) โดยจะแสดงลำดับการเยี่ยมโหนดดังตัวอย่างต่อไปนี้

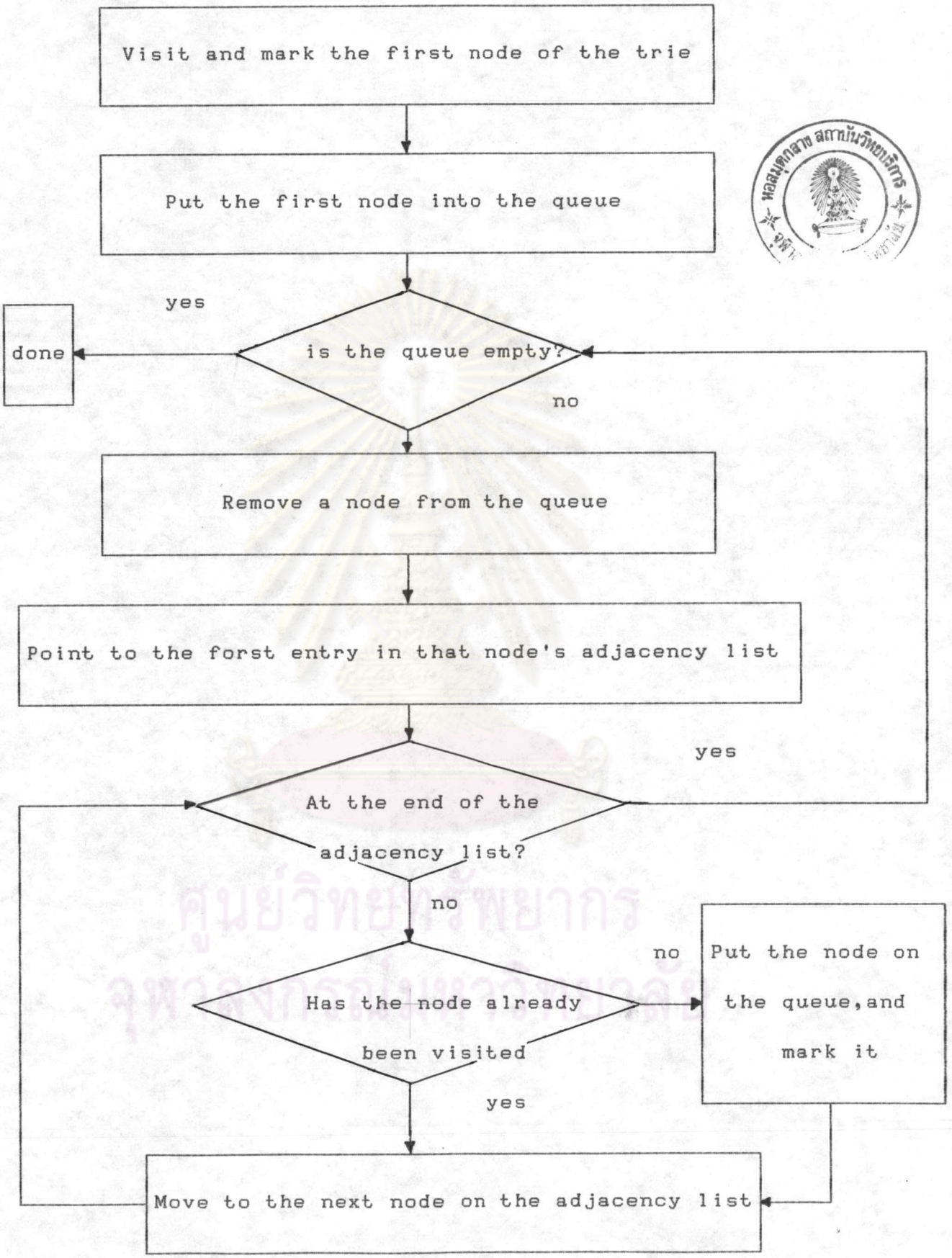


จากโครงสร้างทรายข้างต้น การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางกว้าง จะมีลำดับดังนี้ ABCDEFG การเยี่ยมโหนดโดยวิธีนี้สามารถอธิบายโดยใช้อัลกอลิธึม ภาษาปาสคาลได้ดังนี้

```
type nodeid = 0 .. ordertrie;
     edgeptr = ↑edgeinfo;
     nodetype = record mark : 0..1;
                       adjlist : edgeptr
                end;
     edgeinfo = record node : nodeid;
                       weight : interger;
                       next : edgeptr
                end;
     trietype = array[1..ordertrie] of nodetype;
var  trie : trietype;
     firstnode : nodeid;
```

```
procedure breadth(firstnode:nodeid);
var  queuestruct;
     savenode : nodeid;
     adjptr : edgeptr;
begin {visit firstnode here}
     trie[firstnode].mark := 1;
     insert(firstnode);
     while t.f <> 0
     do begin remove(savenode);
               adjptr := trie[savenode].adjlist;
               {visit nodes adjacent to savenode}
               while adjptr <> nil;
               do begin savenode := adjptr^.node;
                        if(trie[savenode].mark = 0)
                        then begin insert(savenode);
                                   {visit savenode here}
                                   trie[savenode].mark := 1;
                             end;
                        adjptr := adjptr^.next
                  end;
        end;
end;
```

จากอัลกอลิธึมนี้สามารถแสดงเป็น ผังงาน (Flowchart) ได้ดังรูปที่ 4.2



รูปที่ 4.2 แสดงขั้นตอนการเยี่ยมโหนดในทางกว้าง

4.2 โครงสร้างพจนานุกรมเสมือน

จากโครงสร้างของทรายที่กล่าวข้างต้นใช้เนื้อที่ 6ไบต์ ในการจัดเก็บเป็นไฟล์พจนานุกรมจริงๆสามารถตัดฟิลด์(field) next ออกไปได้

โครงสร้างการจัดเก็บพจนานุกรมประกอบด้วยค่าต่างๆตามตารางที่ 4.1

บิทเริ่มต้น	จำนวนบิท	ค่าที่เก็บ
0	7	ตัวอักษร
7	1	ตัวบ่งชี้(pointer)ว่าเป็น ตัวอักษรตัวสุดท้ายของคำศัพท์
8	1	ตัวบ่งชี้(pointer)ว่าเป็นโหนดสุดท้ายในเพจหรือไม่
15	15	ตัวเชื่อมโยงไปยังเพจถัดไป ซึ่งประกอบตำแหน่งของเพจ และ ตำแหน่งของเยื้อง (Offset)

ตารางที่ 4.1 โครงสร้างการจัดเก็บพจนานุกรม

จากโครงสร้างดังกล่าวนี้จะได้ยกตัวอย่าง การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางลึก และ การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางกว้าง ซึ่งประกอบด้วย คำศัพท์ 5 คำดังนี้

กด

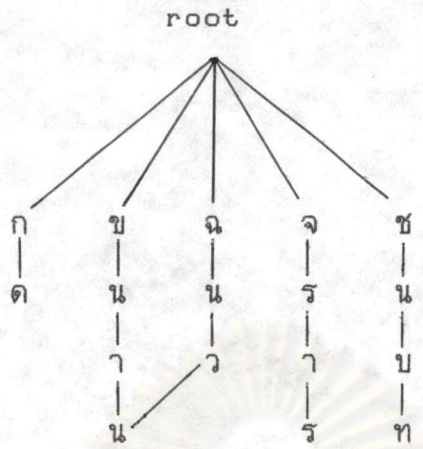
ขนาน

ฉนวน

จราจร

ชนบท

สามารถแสดงในรูปกราฟโครงสร้างแบบ ทรายได้ดังนี้



ในการจัดเก็บพจนานุกรมนั้นจะเก็บตัวอักษรที่อยู่ในระดับ (level) แรกไว้ก่อน เพื่อใช้เป็นจุดเริ่มต้นในการค้นหาพจนานุกรม

จากคำทั้ง 5 นี้ ถ้าสมมุติให้ 1 เพจประกอบด้วย 5 ตัวอักษร การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางกว้าง จะมีลำดับการจัดเก็บดังนี้

ก(1) ข(2) ฉ(3) จ(4) ช(5)	อยู่ในเพจที่ 1
ด(1) น(2) น(3) ร(4) น(5)	อยู่ในเพจที่ 2
า(2) ว(3) า(4) บ(5) น(2,3)	อยู่ในเพจที่ 3
จ(4) ท(5) ร(4)	อยู่ในเพจที่ 4

ตัวเลขที่อยู่ข้างหลังตัวอักษรเป็นตัวเลขที่ใช้บอกว่าตัวอักษรนั้น มาจากคำที่เท่าไร

การจัดเก็บพจนานุกรมตามลำดับการเยี่ยมโหนดในทางลึก จะมีลำดับการจัดเก็บดังนี้

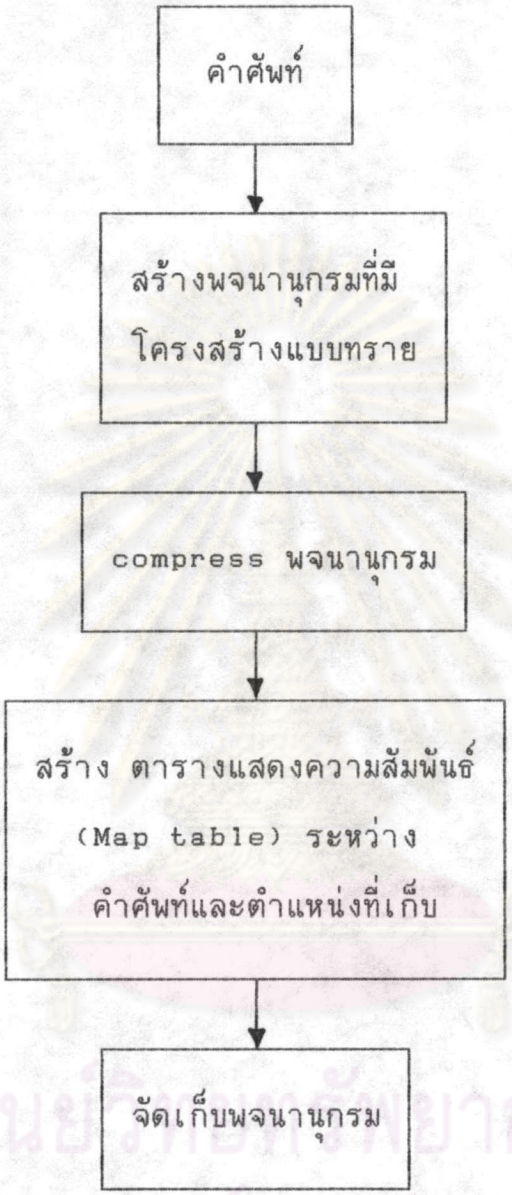
ก(1)	ข(2)	ฉ(3)	จ(4)	ช(5)	เป็นเพจที่ 1
ด(1)	น(2,3)	า(2)	น(2)	ว(3)	เป็นเพจที่ 2
น(3)	ร(4)	จ(4)	า(4)	ร(4)	เป็นเพจที่ 3
ท(5)	บ(5)	น(5)			เป็นเพจที่ 4

4.3 ฟังก์ชันพื้นฐาน

ในการออกแบบโครงสร้างพจนานุกรมเสมือน ใช้ภาษาซีเป็นเครื่องมือในการพัฒนา เนื่องจากเป็นภาษาที่มีประสิทธิภาพและมีความยืดหยุ่นสูง สามารถนำไปใช้ในเครื่องคอมพิวเตอร์ได้ หลายๆเครื่องโดยไม่ต้องมีการเปลี่ยนแปลงแก้ไข ซึ่งฟังก์ชันพื้นฐานสำหรับโครงสร้างพจนานุกรมเสมือน ประกอบด้วย

MakeVirtualAddr	- ใช้สร้างตำแหน่งเสมือนสำหรับพจนานุกรมเสมือน
SaveTrie	- ใช้ในการเก็บพจนานุกรมเสมือนลงไฟล์โดยใช้การจัดเก็บพจนานุกรมในทางลึก
GetEdgeMatch	- ใช้ในการเปรียบเทียบคำจากข้อความที่ต้องการตัดคำกับคำศัพท์จากพจนานุกรม
Check_mem	- ใช้ในการตรวจสอบดูเนื้อที่ในหน่วยความจำหลักว่าเต็มหรือยัง ถ้าเต็มแล้วจะเลือกคำศัพท์ที่ใช้น้อยที่สุดออกไปเป็นการนำหลักการของ LRU มาใช้

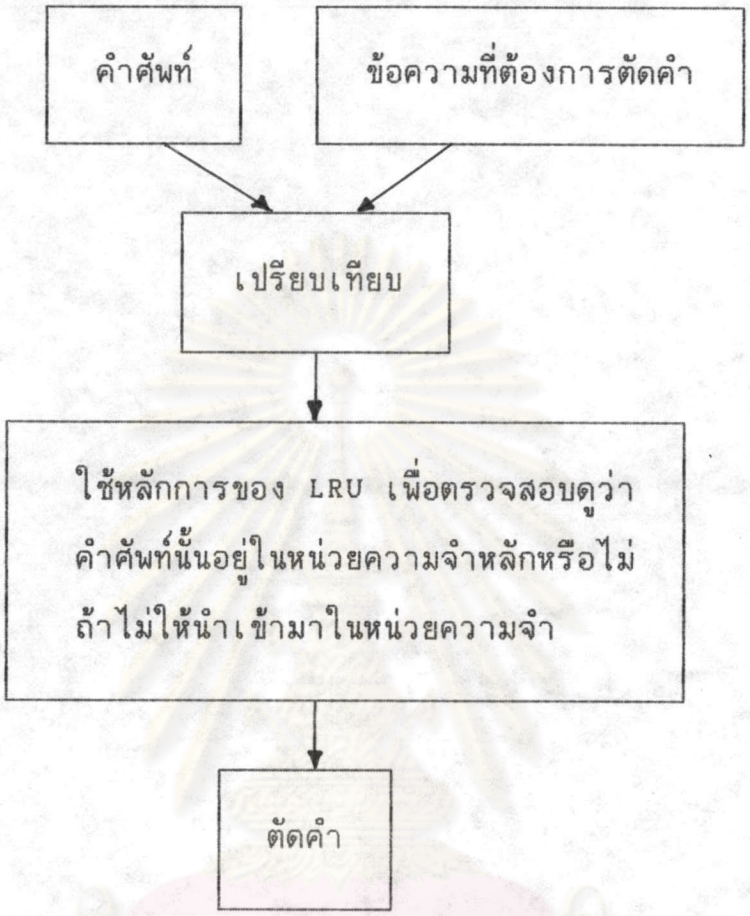
4.4 ขั้นตอนการสร้างพจนานุกรมเสมือน



รูปที่ 4.3 แสดงขั้นตอนในการสร้างพจนานุกรมเสมือน

ขั้นตอนในการสร้างพจนานุกรมเสมือน เริ่มจากการนำคำศัพท์จากไฟล์มาสร้างพจนานุกรมที่มีโครงสร้างแบบทราย ตามหัวข้อที่ 2.2 มีการอัดข้อมูลเพื่อขจัดความซ้ำซ้อนของตัวอักษรออกไป และเพื่อที่จะทำให้การสร้างพจนานุกรมเสมือนนี้มีประโยชน์มากยิ่งขึ้น จึงมีการสร้างตารางแสดง ความสัมพันธ์ระหว่างตัวอักษรในคำศัพท์ กับตำแหน่งของตัวอักษรนั้น จากนั้นจะนำพจนานุกรมเสมือนที่ได้เก็บลงไฟล์

4.5 ขั้นตอนการนำพจนานุกรมเสมือนมาใช้งาน



รูปที่ 4.4 แสดงขั้นตอนในการนำพจนานุกรมเสมือนมาใช้งาน

ในการนำพจนานุกรมเสมือนมาใช้งาน เริ่มจากมีข้อความใดๆที่ต้องการตัดคำ และ มีพจนานุกรมที่ใช้ในการค้นหาคำ ตอนเริ่มต้นของการทำงาน จะมีการนำพจนานุกรมบางส่วนมาไว้ในหน่วยความจำหลัก จากนั้นเปรียบเทียบคำศัพท์ในพจนานุกรม กับ ข้อความเพื่อหาจุดที่ตัดคำทุกครั้งที่มีการค้นหาคำในพจนานุกรม แล้วไม่เจอคำศัพท์ดังกล่าว จะทำการค้นหาคำศัพท์ที่ถูกเรียกใช้น้อยที่สุดนำออกจากหน่วยความจำหลัก แล้วจึงนำคำศัพท์ที่ต้องการใช้จากหน่วยความจำสำรองเข้ามาแทนที่ และเปรียบเทียบหาจุดตัดคำต่อไป