บทที่ 2

แนวคิดและทฤษฎี

2.1 แนวคิดและทฤษฎี

ระบบการค้นคืนข้อความเป็นเครื่องมือที่มีประสิทธิภาพต่อการค้นหาข้อมูล เพื่อตอบสนองในด้านความรวดเร็วและความถูกต้องต่อความต้องการของผู้ใช้ ดังนั้นระบบการค้นคืนส่วนมากแล้วจำเป็นต้องจัดเตรียมดรรชนีไว้ล่วงหน้าก่อน เพื่อความสะดวกในการค้นคืนภายหลัง โดยมีองค์ประกอบหลักอยู่ 2 ส่วน คือ

1. ส่วนของการจัดเก็บดรรชนีข้อมูลเพื่อการค้นคืนภายหลัง
2. ส่วนของการค้นหาข้อมูลตามความต้องการแบบต่างๆ ของผู้ใช้

2.2 โครงสร้างข้อมูลของระบบการค้นคืนข้อความ

โครงสร้างข้อมูล หมายถึงการจัดระเบียบแบบแผนให้กับข้อมูลต่างๆ [9] ดังนั้นในการประมวลผลข้อมูลด้วยคอมพิวเตอร์นั้น หากมีการประมวลผลกับข้อมูลที่ได้ออกแบบโครงสร้างที่เหมาะสมกับงานแล้วจะเอื้ออำนวยต่อการทำงาน ดังนี้

1. สามารถนำข้อมูลเหล่านั้นออกมาใช้งานได้ทันทีตามความต้องการ
2. ช่วยประหยัดเนื้อที่ของหน่วยความจำ เพื่อให้หน่วยความจำของคอมพิวเตอร์ถูกใช้งานอย่างมีประสิทธิภาพ
3. ช่วยประหยัดเนื้อที่ของหน่วยเก็บข้อมูลสำรอง
4. สามารถค้นคืนข้อมูลได้อย่างรวดเร็ว
5. สามารถปรับปรุงเปลี่ยนแปลงและพัฒนาโปรแกรมได้ง่าย

โครงสร้างข้อมูลที่รู้จักกันทั่วไปมีอยู่หลายประเภท โดยที่โครงสร้างข้อมูลแต่ละประเภทจะเหมาะสมสำหรับข้อมูลและการประมวลผลแบบหนึ่ง ดังนั้นในการที่จะเก็บข้อมูลจำเป็นต้อง

ศึกษาทั้งลักษณะข้อมูล วิธีการนำไปใช้งาน ซึ่งหลักเกณฑ์ที่นำมาใช้สำหรับการพิจารณาโครงสร้างข้อมูลประกอบด้วย

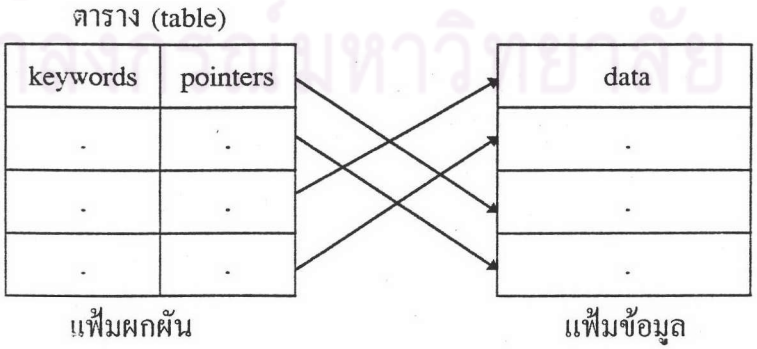
1. จำนวนข้อมูลที่ถูกจัดเก็บ
2. ขั้นตอนวิธีการค้นหาข้อมูลที่จัดเก็บไว้
3. ขั้นตอนวิธีการเพิ่มเติมแก้ไขข้อมูล
4. วัตถุประสงค์ในการนำไปใช้งาน

ระบบการค้นคืนข้อความที่ใช้งานโดยทั่วไปในปัจจุบัน ได้ออกแบบโครงสร้างข้อมูลหลายๆ แบบ สำหรับการวิจัยครั้งนี้ผู้วิจัยได้ศึกษาโครงสร้างข้อมูลบางโครงสร้างที่ใช้งานบ่อยๆ โดยที่โครงสร้างข้อมูลที่กล่าวถึงมีลักษณะเด่นที่แตกต่างกันไป โครงสร้างข้อมูลที่ศึกษามีดังนี้

1. โครงสร้างข้อมูลแบบแฟ้มผกผัน (inverted file)
2. โครงสร้างข้อมูลแฟ้มซิกเนเจอร์ (signature file)
3. โครงสร้างข้อมูลแบบทรัย (trie)
4. โครงสร้างข้อมูลแบบดับเบิลอะเรย์ (double array)
5. โครงสร้างข้อมูลแบบต้นไม้แพ็ต (PAT tree)

2.3 โครงสร้างข้อมูลแบบแฟ้มผกผัน (inverted file)

แฟ้มผกผัน คือแฟ้มดรรชนีของแฟ้มข้อมูล (data file) [4] แฟ้มผกผันเป็นโครงสร้างข้อมูลที่อาศัยตาราง (table) เป็นกลไกในการเข้าถึงข้อมูล โดยตารางดังกล่าวประกอบด้วยคำหลักกับตัวชี้ (pointer) ที่ชี้ไปยังข้อมูลในแฟ้มข้อมูลตามคำหลักนั้นๆ ดังรูปที่ 2.1



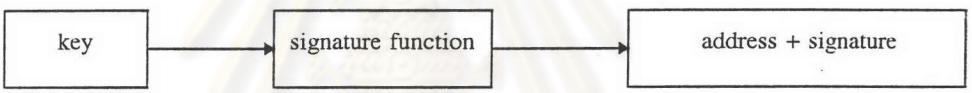
รูปที่ 2.1 ความสัมพันธ์ระหว่างแฟ้มผกผันกับแฟ้มข้อมูล

ข้อดีของแฟ้มผกผัน คือการเข้าถึงข้อมูลทำได้รวดเร็ว โครงสร้างข้อมูลไม่ซับซ้อนตรงไปตรงมาทำให้ง่ายต่อการเขียนโปรแกรม

ส่วนข้อเสีย คือใช้เนื้อที่จัดเก็บค่อนข้างมากและมีการจัดเก็บซ้ำๆ คือนำส่วนของข้อมูลที่เป็นดรรชนีไปเก็บไว้ในแฟ้มผกผัน นอกจากนี้การสร้างดรรชนีจำเป็นต้องเลือกส่วนหนึ่งของข้อมูลมาสร้างคำหลักเพื่อใช้ในการค้นคืนภายหลังอีกด้วย

2.4 โครงสร้างข้อมูลแบบแฟ้มซิกเนเจอร์ (signature file)

แฟ้มซิกเนเจอร์ เป็นโครงสร้างข้อมูลที่อาศัยฟังก์ชันซิกเนเจอร์ (signature function) เป็นกลไกในการเข้าถึงข้อมูล โดยใช้ค่าคีย์เป็นข้อมูลในการคำนวณ [7] ฟังก์ชันซิกเนเจอร์เป็นฟังก์ชันแฮช (Hashing Function) ประเภทหนึ่งที่คำนวณแล้วให้ค่าเลขที่อยู่ (address) กับซิกเนเจอร์ (signature) คือบิตที่เรียงต่อกันในรูปเลขฐานสองของข้อมูล ดังรูปที่ 2.2



รูปที่ 2.2 การคำนวณหาเลขที่อยู่ของข้อมูล

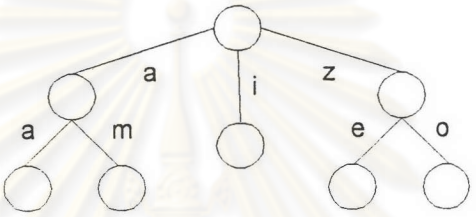
จะเห็นว่าการเข้าถึงข้อมูลทำได้โดยการส่งค่าคีย์ให้กับฟังก์ชันซิกเนเจอร์ เพื่อคำนวณหาค่าเลขที่อยู่กับซิกเนเจอร์จึงทำได้รวดเร็วกว่าการค้นหาคำหลัก

ข้อดีของแฟ้มซิกเนเจอร์ คือการหาตำแหน่งของข้อมูลอาศัยการคำนวณทางพีชคณิตจึงทำได้รวดเร็วกว่าโครงสร้างข้อมูลแบบแฟ้มผกผัน

ส่วนข้อเสีย คือในกรณีที่คำนวณแล้วได้เลขที่อยู่เดียวกันแสดงว่าเกิดการชนกัน (collision) ขึ้นแต่ได้ซิกเนเจอร์ต่างกัน การค้นหายังเป็นแบบลำดับ (sequential) และในกรณีที่เกิดปัญหาล้น (overflow) บ่อยๆ จะทำให้ประสิทธิภาพของแฟ้มซิกเนเจอร์ไม่ค่อยดี อีกทั้งการกำหนดฟังก์ชันในการคำนวณหาตำแหน่งของข้อมูลไม่ให้เกิดการซ้ำกันเลยกระทำได้ยาก นอกจากนี้แล้วการสร้างดรรชนีจำเป็นต้องเลือกส่วนหนึ่งของข้อมูลมาสร้างคำหลักเพื่อใช้ในการค้นคืนภายหลัง

2.5 โครงสร้างข้อมูลแบบทรัย (trie)

ทรัยเป็นโครงสร้างข้อมูลแบบต้นไม้ประเภทหนึ่ง ที่ประกอบด้วยโหนดต่างๆ กับเส้นทางเดินไปยังโหนดเหล่านั้น [7] โดยอาศัยส่วนของข้อมูล คือตัวอักษรของข้อมูลในต้นไม้กับข้อมูลที่ต้องการจะค้นหา นำมาเปรียบเทียบเพื่อกำหนดการเดินและการค้นหาข้อมูลในต้นไม้ โครงสร้างข้อมูลทรัยเป็นโครงสร้างข้อมูลที่มีประสิทธิภาพกับข้อมูลที่มีลักษณะเป็นตัวอักษรมาก เนื่องจากมีคุณสมบัติที่ช่วยลดการจัดเก็บข้อความร่วมที่ซ้ำๆ กันได้ดี ดังรูปที่ 2.3



รูปที่ 2.3 ตัวอย่างโครงสร้างข้อมูลทรัย

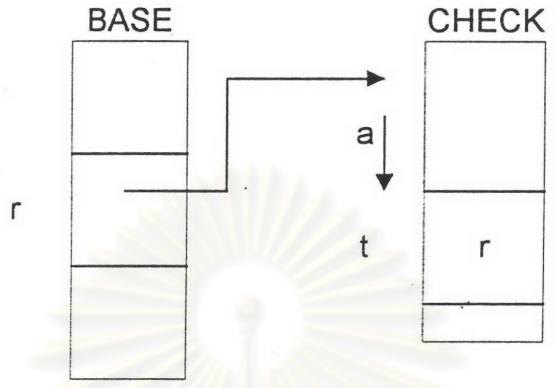
ข้อดีของทรัย คือเป็นโครงสร้างข้อมูลที่เหมาะสมกับข้อมูลที่เป็นคำๆ ที่ประกอบด้วยตัวอักษรเรียงต่อกัน และสามารถลดการจัดเก็บข้อมูลร่วม ทั้งยังเหมาะสมกับการค้นหาข้อความแบบเติมหน้า (prefix searching) ได้ดี ตลอดจนการเพิ่มข้อมูลและการลบข้อมูลกระทำได้ง่ายและรวดเร็วอีกด้วย

ส่วนข้อเสียของทรัย ก็คือสิ้นเปลืองเนื้อที่ในการจัดเก็บมาก และประสิทธิภาพในการจัดเก็บและการค้นหาขึ้นอยู่กับความยาวของข้อมูล โดยเวลาที่ใช้ในการค้นหาเฉลี่ยของโครงสร้างข้อมูลแบบทรัยของข้อมูล m คำที่ประกอบด้วยโหนด n โหนด เท่ากับ $\log_m n$ [7]

2.6 โครงสร้างข้อมูลแบบดับเบิลอะเรย์ (double array)

โครงสร้างข้อมูลแบบดับเบิลอะเรย์ เป็นโครงสร้างข้อมูลที่พัฒนาขึ้นมาเพื่อใช้กับโครงสร้างข้อมูลแบบดีเอส-ทรี [1] โดยนำหลักการของโครงสร้างข้อมูลแบบทริปเปิ้ลอะเรย์ที่ใช้งานกับทรานซิชั่นเทเบิ้ลสำหรับแยคซ์ (Yacc) ภายใต้ข้อกำหนดที่ว่าจะนำมาประยุกต์ใช้กับทรานซิชั่นเทเบิ้ลสำหรับสตริงแพทเทินแมชชิงแมชชีน (string pattern matching machine) โดยโครงสร้างข้อมูลแบบดับเบิลอะเรย์ประกอบด้วยอะเรย์ 1 มิติ 2 อะเรย์ คือ อะเรย์เบส (BASE) กับอะเรย์เช็ค

(CHECK) และมีหมายเลขประจำโหนดเป็นดรรชนีเพื่อแสดงความสัมพันธ์ระหว่างเบสและเช็ค ซึ่งรูปแบบของโครงสร้างข้อมูลแบบดับเบิลอะเรย์แสดงตามรูปที่ 2.4



รูปที่ 2.4 ลักษณะโครงสร้างข้อมูลแบบดับเบิลอะเรย์

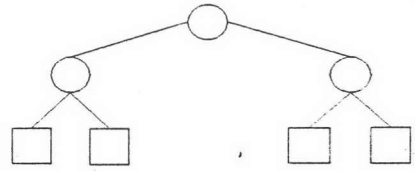
การคำนวณหาค่าหมายเลขของโหนดถัดไปและเส้นทางระหว่างโหนดของโครงสร้างข้อมูลแบบดับเบิลอะเรย์นั้นอาศัยสมการ BASE[r] + a = t และ CHECK[t] = r

ข้อดีของดับเบิลอะเรย์ คือมีความสามารถคล้ายๆ กับโครงสร้างข้อมูลแบบทรัย และมีข้อดีเพิ่มที่สามารถลดความสิ้นเปลืองที่ใช้ในการจัดเก็บส่วนของข้อมูลที่ไม่เป็นข้อมูลร่วม

ส่วนข้อเสียของดับเบิลอะเรย์ คือเหมือนๆ กับโครงสร้างข้อมูลแบบทรัย ยกเว้นในส่วนของความสิ้นเปลืองของเนื้อที่ในการจัดเก็บ ที่ใช้เนื้อที่ลดน้อยลงกว่าโครงสร้างข้อมูลแบบทรัย แต่ก็ยังคงสิ้นเปลืองเนื้อที่ ในส่วนของการจัดเก็บข้อมูลที่เป็นข้อมูลร่วมเหมือนกับ โครงสร้างข้อมูลแบบทรัยเหมือนเดิม

2.7 โครงสร้างข้อมูลแบบต้นไม้แพ็ต (PAT tree)

ต้นไม้แพ็ตย่อมาจาก Practical Algorithm To Retrieve Information Coded In Alphanumeric [10][14][17] เป็นต้นไม้ค้นหาดิจิตอลแบบทวิภาค (binary digital search tree) หมายถึงโครงสร้างข้อมูลแบบต้นไม้ทวิภาคที่มีการสืบค้นแบบดิจิตอล ดังนั้นโครงสร้างข้อมูลต้นไม้แพ็ตจึงเป็นโครงสร้างข้อมูลที่ประกอบด้วยโหนดต่างๆกับเส้นทางเดินระหว่างโหนด โดยมีโหนดเริ่มต้นเรียกว่าโหนดราก (root node) และมีโหนดภายใน (internal node) เป็นรูปวงกลมกับโหนดภายนอก (external node) เป็นรูปสี่เหลี่ยม ดังรูปที่ 2.5



รูปที่ 2.5 องค์ประกอบของต้นไม้แพ็ต

ต้นไม้แพ็ตเป็นโครงสร้างข้อมูลที่มีประสิทธิภาพอย่างมากกับการค้นข้อมูลที่มีการสร้างดรรชนีเตรียมไว้ก่อน เพื่อการค้นข้อมูลในภายหลังที่สะดวกรวดเร็วขึ้น [9] โดย Gonnet เป็นคนแรกที่กล่าวถึงต้นไม้แพ็ตไว้ในบทความ "Unstructured Data Bases" by Gonnet (1983) และในปี 1985 ศูนย์คอมพิวเตอร์ของ Oxford English Dictionary ได้พัฒนาระบบแพ็ต (PAT™ System) ขึ้น จนในเวลาต่อมารู้จักกันอย่างแพร่หลายว่าเป็นระบบที่สามารถค้นหาข้อมูลได้รวดเร็วมาก

2.7.1 สายอักขระแบบกึ่งอนันต์ (Semi-infinite string) หรือซิสตริง (Sistring)

เมื่อพิจารณาถึงข้อมูลในระบบค้นคืนข้อความพบว่าข้อมูลมีลักษณะเป็นลำดับสายอักขระ (array of character) ที่เรียงต่อกันไปเรื่อยๆจนจบข้อมูล ซึ่งซิสตริงก็คือลำดับสายอักขระย่อย (subsequence) ในข้อมูลนั้น โดยมีตำแหน่งเริ่มต้นที่แน่นอนตำแหน่งหนึ่งแล้วเลื่อนไปทางขวาจนจบข้อมูลนั้น ดังตัวอย่าง

Text	คือ Once upon a time, in a far away land...
Sistring 1	คือ Once upon a time, in a far away land...
Sistring 2	คือ nce upon a time, in a far away land...
Sistring 6	คือ upon a time, in a far away land...
Sistring 8	คือ on a time, in a far away land...
Sistring 11	คือ a time, in a far away land...
Sistring 22	คือ a far away land...

โดยที่ Sistring 1 หมายความว่าเป็นซิสตริงตำแหน่งเริ่มต้นที่ 1 ไปจนจบข้อมูลตามลำดับ

การเปรียบเทียบซิสตริงสองตัวนั้นจะเปรียบเทียบตามลำดับรหัสค่าของซิสตริงนั้น เช่นลำดับแอสกี (ASCII ordering) เป็นต้น ซึ่งจะได้ผลลัพธ์ตามตัวอย่างข้างบน ดังนี้

Sistring 22 < Sistring 11 < Sistring 2 < Sistring 8 < Sistring 1 < Sistring 6

โดยมี Sistring 22 คือ "a far away land..." เป็นค่าน้อยสุดส่วน Sistring 6 คือ "upon a time, in a far away land..." เป็นค่ามากที่สุด

2.7.2 ต้นไม้แพ็ต (PAT tree)

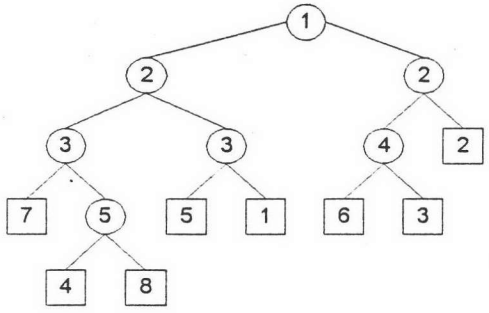
ต้นไม้แพ็ตเป็นต้นไม้ค้นหาดิจิตอลแบบทวิภาค (binary digital search tree) [3][7][8][10][12][14][17] ซึ่งประกอบไปด้วยซิสตริงที่เป็นไปได้ของข้อมูลนั้น (ไม่จำเป็นต้องเป็นทุกๆ ซิสตริง) โดยมีแต่ละบิตของคีย์เป็นตัวระบุให้เลือกเดินไปทางไหน ถ้าบิตนั้นมีค่าเป็น 0 จะเดินไปทางซ้าย ถ้าเป็น 1 จะเดินไปทางขวา นอกจากนี้แล้วต้นไม้แพ็ตยังประกอบไปด้วยโหนดจำนวน 2 แบบ คือ

1. โหนดภายในจะไม่เก็บคีย์ แต่เก็บค่าของตัวนับข้าม (skip counter) กับตัวชี้ (pointer) ที่ชี้โหนดข้างซ้ายและโหนดข้างขวา โดยโหนดภายในจะไม่ชี้โหนดว่าง (null node) เสมอ

2. โหนดภายนอกเก็บค่าของตำแหน่งเริ่มต้นของซิสตริง

ดังตัวอย่างต่อไปนี้

01100100010111...	Text
123456789.....	Position



รูปที่ 2.6 ต้นไม้แพ็ตเมื่อเพิ่มซิสตริงไปแล้ว 8 ซิสตริง

รูปที่ 2.6 แสดงต้นไม้แพ็ตที่เพิ่มซิสตริงไปแล้ว 8 ซิสตริง โดยโหนดภายนอกเป็นโหนดรูปสี่เหลี่ยม และเก็บค่าตำแหน่งเริ่มต้นของซิสตริงนั้น ส่วนโหนดภายในจะเป็น โหนดรูปวงกลมและเก็บค่าตัวนับข้ามของซิสตริงนั้น

สมมติว่าต้องการค้นข้อมูล “00101” เริ่มต้นตรวจบิตที่ 1 (มีค่าเป็น 0, ไปทางซ้าย) แล้วตรวจบิตที่ 2 (มีค่าเป็น 0, ไปทางซ้าย) แล้วตรวจบิตที่ 3 (มีค่าเป็น 1, ไปทางขวา) แล้วตรวจบิตที่ 5 (มีค่าเป็น 1, ไปทางขวา) จะถึงโหนดภายนอกที่ 8 จากนั้นทำการเปรียบเทียบกับซิสตริงของโหนดภายนอกที่ 8 ซึ่งมีค่าเท่ากัน คือ “00101” จะสังเกตุเห็นว่าเราได้ข้ามบิต (skip bit) ที่ 4 ไป เนื่องจากบิตที่ 4 ของซิสตริงของโหนดภายนอกที่ 4 และ 8 มีค่าเท่ากันคือ 0 จึงไม่จำเป็นต้องตรวจสอบบิตที่ 4

2.7.3 ตัวชี้ดรรชนี (indexing point)

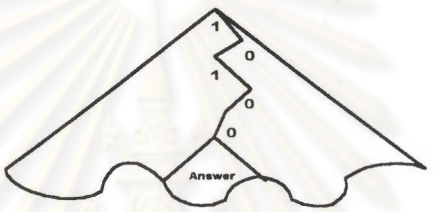
จากต้นไม้แพ็ตในหัวข้อที่แล้วจะเห็นว่าเราใช้ทุกๆ ตำแหน่งในข้อมูลเป็นตัวชี้ดรรชนีทั้งหมด ซึ่งจะมีจำนวนเท่ากับความยาวของข้อมูลนั้น แต่ในความเป็นจริงแล้วเราอาจสนใจเฉพาะข้อมูลในลักษณะของคำหรือวลีก็ได้ซึ่งจะช่วยให้ตัวชี้ดรรชนีมีขนาดเล็กลงไปได้ ทั้งนี้ขึ้นอยู่กับการพิจารณาของนักออกแบบและวิเคราะห์ระบบด้วย

2.7.4 การค้นข้อมูลในต้นไม้แพ็ต

การค้นข้อมูลในต้นไม้แพ็ตทำได้ดังนี้

2.7.4.1 การค้นหาแบบเติมหน้า (prefix)

เนื่องจากทุกๆ ต้นไม้ย่อยในต้นไม้แพ็ตประกอบด้วยโหนดภายนอกที่เป็นซิสตริงที่มีโหนดภายในเป็นข้อมูลร่วม ดังนั้นการค้นหาแบบเติมหน้าจะสิ้นสุดที่โหนดภายในหรือบางครั้งถึงโหนดภายนอกก็ได้ ในกรณีที่การค้นหาสิ้นสุดที่โหนดภายในนั้นจะได้ผลลัพธ์ คือจำนวน ซิสตริงทั้งหมดในต้นไม้ย่อยนั้น ส่วนในกรณีการค้นหาสิ้นสุดที่โหนดภายนอกก็จะทำการเปรียบเทียบกับค่าซิสตริงที่โหนดภายนอกนั้น ถ้าเท่ากันผลลัพธ์คือซิสตริงนั้น แต่ถ้าไม่เท่ากันผลลัพธ์ก็คือการค้นหานั้นไม่พบข้อมูล ดังรูปที่ 2.7



รูปที่ 2.7 การค้นหาแบบเติมหน้า

สำหรับต้นไม้แพ็ตที่ใช้ข้อมูลสุ่มๆ ปรากฏว่ามีความสูงประมาณ O(log n) [2][13] [15][16] โดยที่ n คือ จำนวนโหนดทั้งหมดในต้นไม้ จะทำให้ประสิทธิภาพในการค้นแบบเติมหน้าจะประมาณ O(log n) โดยไม่ขึ้นกับขนาดของคำตอบ (จะสั้นหรือยาวก็มีประสิทธิภาพเท่ากัน)

2.7.4.2 การค้นหาแบบข้างเคียง (Proximity)

การค้นหาแบบข้างเคียง คือการค้นหาข้อมูลทั้งหมดที่สตริง (String) S_1 ห่างจากสตริง S_2 ไปตามค่าที่กำหนด ส่วนใหญ่เป็นค่าคงที่ที่ผู้ใช้เป็นผู้กำหนดให้ โดยวิธีการง่ายที่สุด คือการค้นหาสตริง S_1 และสตริง S_2ก่อน แล้วเรียงตำแหน่งคำตอบจากน้อยไปมากของแต่ละคำตอบแล้วค้นหาข้อมูลตามตำแหน่งต่างๆ ที่เรียงลำดับไว้แล้ว จากนั้นตรวจสอบความห่างของตำแหน่งว่าตรงตามเงื่อนไขหรือไม่

สำหรับเซตของคำตอบ m_1 และ m_2 ($m_1 < m_2$) ประสิทธิภาพรวมในการค้นแบบข้างเคียงจะประมาณ $(m_1 + m_2) \log m_1$ แต่ถ้า m_1 หรือ m_2เป็น O(n) [15][16]โดยที่ n คือจำนวนโหนดทั้งหมดในต้นไม้แล้วประสิทธิภาพรวมก็เป็น O(n) ด้วย

2.7.4.3 การค้นหาแบบช่วง (Range)

การค้นหาแบบช่วง คือการค้นหาสตริงทั้งหมดที่มีค่าเริ่มต้นและสิ้นสุดตามสตริงที่กำหนดมาทั้งสองค่า เช่นการค้นหาแบบช่วงของสตริง "abc" ... "acc" ซึ่งอาจจะได้คำตอบเป็น "abracadabra," "acacia," "aboriginal" ก็ได้แต่ต้องไม่ใช่ "abacus" หรือ"acrimonious" โดยการเริ่มต้นค้นสตริงทั้งสองพร้อมกับเก็บสะสมต้นไม้ย่อยที่เริ่มต้นและสิ้นสุดของสตริงทั้งสอง ซึ่งประสิทธิภาพในการค้นของแต่ละสตริงคือ O(height) [15][16] โดยที่ height คือความสูงของต้นไม้ หรือประสิทธิภาพรวมก็คือประมาณ O(log n) [15][16] โดยที่ n คือ จำนวนโหนดทั้งหมดในต้นไม้

2.7.4.4 การค้นหาแบบนัยสำคัญสูงสุด (Most Significant) หรือความถี่สูงสุด (Most Frequent)

การค้นหาแบบนัยสำคัญสูงสุดหรือความถี่สูงสุด คือการค้นหาสตริงที่ปรากฏอยู่ในข้อมูลบ่อยมากที่สุด เช่นการค้นข้อมูล 3 ตัวเรียงต่อกันที่ปรากฏอยู่ในเอกสารบ่อยมากที่สุดเป็นต้น สำหรับต้นไม้แพ็ตทำได้โดยการหาต้นไม้ย่อยที่ใหญ่ที่สุดที่อยู่ห่างจากราก (root) ไป 3 เพราะต้นไม้ย่อยนี้จะมีข้อมูลร่วมของซิสตริงที่เป็นตัวเดียวกันมากที่สุดที่ปรากฏในเอกสาร โดยมีประสิทธิภาพประมาณ O(n/average size of answer) [15][16] โดยที่ n คือ จำนวนโหนดทั้งหมดในต้นไม้