**Received:** 3 ม.ค. 2568 **Revised:** 13 มี.ค. 2568 **Accepted:** 18 มี.ค. 2568

# การเปรียบเทียบประสิทธิภาพระหว่างโครงสร้างข้อมูลต้นไม้กับโครงสร้างข้อมูลลิงค์ลิสต์ในการจัดการข้อมูลบุคคลของฝ่ายทรัพยากรมนุษย์

# Performance Comparison between Tree Data Structure and Linked List Data Structure in Managing Employee Information for Human Resources

ผดุงเกียรติ สุตาโย[1],พาสน์ ปราโมกข์ชน[1], สมนึก สินธุปวน[1], ก่องกาญจน์ ดุลยไชย[1*]
[1]สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้

Phadungkiat Sutayo[1], Part Pramokchon[1], Somnuek Sinthupuan[1],
Kongkarn Dullayachai[1*]
[1]Computer Science, Faculty of Science, Maejo University
*Corresponding author: kongkarn@gmaejo.mju.ac.th

## Abstract

This article presents a performance comparison between tree data structures and linked list data structures for managing human resources data with hierarchical relationships, including employee information such as ID, full name, position, department, and event records. The study aims to compare the speed of insertion, deletion, and searching for each structure, tested with datasets of 100, 1,000, and 10,000 records, each tested three times. The results show that the tree structure outperforms the linked list in data insertion, particularly with the 10,000-record dataset, achieving an insertion rate of 3.785 records per second and an average time of 264.28 milliseconds. In contrast, the linked list achieves only 0.107 records per second and an average time of 9,345.85 milliseconds. For searching and deleting data, both structures perform similarly. In conclusion, the tree structure is better suited for systems handling hierarchical data, as it can insert data quickly and handle larger datasets more efficiently, while the linked list may be more suitable for tasks with smaller datasets or without complex hierarchical relationships

***Keywords:*** *Tree data structure; linked list; Hierarchy*

**บทคัดย่อ**

บทความนี้นำเสนอการเปรียบเทียบประสิทธิภาพระหว่างโครงสร้างข้อมูลต้นไม้กับโครงสร้างข้อมูลลิงค์ลิสต์สำหรับการจัดการข้อมูลฝ่ายทรัพยากรมนุษย์ที่มีความสัมพันธ์แบบลำดับ โดยมีข้อมูลพนักงาน เช่น รหัส ชื่อ-นามสกุล ตำแหน่ง แผนก และบันทึกเหตุการณ์ งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบความเร็วในการเพิ่ม ลบ และค้นหาข้อมูลในแต่ละโครงสร้าง โดยทดสอบกับชุดข้อมูลขนาด 100, 1,000 และ 10,000 รายการ แต่ละชุดข้อมูลถูกทดสอบ 3 ครั้ง ผลการทดสอบพบว่าโครงสร้างข้อมูลต้นไม้มีประสิทธิภาพดีกว่าในการเพิ่มข้อมูล โดยเฉพาะกับข้อมูลขนาด 10,000 รายการ ที่มีอัตราการเพิ่มข้อมูล 3.785 รายการต่อวินาที และใช้เวลาเฉลี่ย 264.28 มิลลิวินาที ในทางกลับกันโครงสร้างข้อมูลลิงค์ลิสต์มีอัตราการเพิ่มข้อมูลเพียง 0.107 รายการต่อวินาที และใช้เวลาเฉลี่ยสูงถึง 9,345.85 มิลลิวินาที สำหรับการค้นหาและลบข้อมูลทั้งสองโครงสร้างมีประสิทธิภาพใกล้เคียงกัน สรุปได้ว่า โครงสร้างข้อมูลต้นไม้เหมาะสำหรับระบบที่จัดการข้อมูลแบบลำดับชั้น เนื่องจากสามารถเพิ่มข้อมูลได้รวดเร็วและรองรับข้อมูลจำนวนมากได้ดีกว่า ในขณะที่โครงสร้างข้อมูลลิงค์ลิสต์อาจเหมาะกับงานที่มีข้อมูลน้อยหรือไม่ต้องการความสัมพันธ์แบบลำดับชั้นที่ซับซ้อน

***คำสำคัญ**: โครงสร้างข้อมูลต้นไม้; โครงสร้างข้อมูลลิงค์ลิสต์; ลำดับชั้น*

## 1. บทนำ

การเลือกโครงสร้างข้อมูลที่เหมาะสมเป็นปัจจัยสำคัญสำหรับการพัฒนาระบบที่ต้องจัดเก็บและประมวลผลข้อมูลในลักษณะลำดับชั้น โดยเฉพาะในระบบที่เกี่ยวข้องกับการจัดการข้อมูลพนักงานในองค์กรประกอบไปด้วย รหัสพนักงาน ชื่อ-นามสกุล ตำแหน่ง แผนก และบันทึกเหตุการณ์ต่าง ๆ เช่น การบันทึกเหตุการณ์ที่เกี่ยวกับการพัฒนาตนเอง การประเมินผล หรือเหตุการณ์อื่น ๆ ที่เกี่ยวข้อง ช่วยให้การจัดการข้อมูลลำดับชั้นมีประสิทธิภาพและวิเคราะห์ข้อมูลได้อย่างสะดวก โดยโครงสร้างข้อมูลที่นิยมใช้ในการจัดเก็บข้อมูลประเภทนี้คือ โครงสร้างข้อมูลต้นไม้เป็นโครงสร้างที่ใช้จัดระเบียบข้อมูลในลักษณะลำดับชั้น โดยมีจุดเริ่มต้นหรือรากที่เชื่อมโยงกับข้อมูลอื่น ๆ ในรูปแบบของโหนด ซึ่งแต่ละโหนดสามารถเชื่อมโยงไปยังโหนดอื่น ๆ ที่เรียกว่าลูก และโครงสร้างข้อมูลลิงค์ลิสต์เป็นโครงสร้างที่ใช้จัดเก็บข้อมูลในลำดับที่เชื่อมโยงกัน โดยแต่ละข้อมูลจะถูกเก็บไว้ในโหนด ซึ่งมีข้อมูลและการเชื่อมโยงไปยังโหนดถัดไป

ดังนั้น งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อเปรียบเทียบประสิทธิภาพของโครงสร้างข้อมูลต้นไม้และโครงสร้างข้อมูลลิงค์ลิสต์ในด้านการเพิ่ม ลบ และค้นหาข้อมูลพนักงาน โดยพิจารณาจากอัตราการจัดการข้อมูลและเวลาที่ใช้ในการประมวลผลต่อครั้ง เพื่อช่วยให้ผู้พัฒนาระบบสามารถเลือกใช้

โครงสร้างข้อมูลที่เหมาะสมที่สุดในการจัดการข้อมูลพนักงานภายในองค์กรอย่างมีประสิทธิภาพ พร้อมรองรับการขยายตัวและการจัดการข้อมูลในปริมาณมากในอนาคต

## 2. วัตถุประสงค์

1. เพื่อศึกษาประสิทธิภาพของโครงสร้างข้อมูลในการจัดการข้อมูลที่มีความสัมพันธ์แบบลำดับชั้น
2. เพื่อระบุโครงสร้างข้อมูลที่เหมาะสมสำหรับการพัฒนาระบบจัดการข้อมูลฝ่ายทรัพยากรมนุษย์อย่างมีประสิทธิภาพ

## 3. งานวิจัยที่เกี่ยวข้อง

Božinovski, A., Tanev, G., Stojčevska, B., Pačovski, V., & Ackovska, N. (2017) ได้ศึกษาการวิเคราะห์ความซับซ้อนของเวลาของขั้นตอนวิธีการสร้างไบนารีทรี (Time Complexity Analysis of the Binary Tree Roll Algorithm) การศึกษานี้ทำการวิเคราะห์ความซับซ้อนของเวลาในอัลกอริธึม Binary Tree Roll โดยเน้นที่เวอร์ชันทวนเข็มนาฬิกา CCW และเปรียบเทียบกับเวอร์ชันตามเข็มนาฬิกา CW ซึ่งมีการวิเคราะห์ทางทฤษฎีและการทดสอบเชิงประจักษ์ที่สอดคล้องกัน โดยการวิเคราะห์ทางทฤษฎีได้หาความสัมพันธ์เชิงซ้ำสำหรับกรณีต่างๆ และนำมาแก้ไขเพื่อหาค่าความซับซ้อน ในขณะที่การทดสอบเชิงประจักษ์ได้ทดสอบต้นไม้ที่มีจำนวนโหนดต่างๆ และนับจำนวนขั้นตอนที่ใช้ในการดำเนินการ ผลการทดลองยืนยันว่าอัลกอริธึม CCW จะมีความซับซ้อนเชิงเส้นในกรณีที่ดีที่สุด และเชิงกำลังสองในกรณีที่แย่ที่สุด สำหรับต้นไม้ที่มีจำนวนโหนดไม่เกิน 645 โหนด ผลลัพธ์จะเป็นเชิงเส้น แต่เมื่อจำนวนโหนดเพิ่มขึ้น ความซับซ้อนเชิงกำลังสองจะเริ่มเด่นชัดขึ้น

Fetaji, M., Ebibi, M., & Fetaji, B. (2021) ได้ศึกษาการวัดประสิทธิภาพของอัลกอริทึมระหว่าง Array กับ Dynamic Linked List (Measuring Algorithms Performance in Dynamic Linked List and Arrays) โดยเปรียบเทียบประสิทธิภาพของโครงสร้างข้อมูล ในการค้นหา การเพิ่มและการลบข้อมูล โดยผลการศึกษาแสดงว่า Array เหมาะสำหรับกรณีที่ข้อมูลเรียงลำดับแล้วและต้องการการเข้าถึงข้อมูลโดยตรง เช่น การค้นหาผ่าน Binary Search ที่ทำงานในเวลา O(log n) ในขณะที่ DLL เหมาะสำหรับการเพิ่มหรือลบข้อมูลที่ตำแหน่งต่างๆ ในรายการ เพราะสามารถจัดการหน่วยความจำได้ตามต้องการ การเลือกใช้โครงสร้างข้อมูลขึ้นอยู่กับลักษณะของการดำเนินการที่ต้องการ เช่น Array เหมาะกับการเข้าถึงข้อมูลบ่อยครั้งและขนาดข้อมูลที่รู้ล่วงหน้าส่วน Dynamic Linked List เหมาะกับข้อมูลที่มีขนาดไม่แน่นอนหรือเมื่อการเพิ่มและลบข้อมูลต้องเกิดบ่อยๆ

Samoladas, D., Karras, C., Karras, A., Sioutas, S., & Theodorakopoulos, L. (2023) ได้ศึกษาโครงสร้างข้อมูลแบบต้นไม้และเทคนิคการสร้างดัชนีที่มีประสิทธิภาพสำหรับการจัดการข้อมูลขนาดใหญ่ (Tree data structures and efficient indexing techniques for big data management: A comprehensive study) โดยมีการแยกแยะโครงสร้างข้อมูลต้นไม้ที่ใช้บ่อย ได้แก่ B+ Tree, QuadTree, KD Tree และ R Tree ซึ่งแต่ละโครงสร้างเหมาะสมกับประเภทของข้อมูลที่แตกต่างกัน นอกจากนี้ยังมีการอภิปรายถึงเทคนิคการทำดัชนีทั้งในเชิงศูนย์กลางและกระจาย ซึ่งช่วยเพิ่มประสิทธิภาพในการแทรกข้อมูลและการค้นหาภายใต้สภาวะต่างๆ ผลการศึกษาเหล่านี้สามารถเสริมสร้างความเข้าใจในกลไกการทำงานของโครงสร้างข้อมูลต้นไม้และการทำดัชนีในบริบทของข้อมูลขนาดใหญ่ และยังช่วยให้ผู้อ่านเข้าใจความสามารถของเทคนิคต่างๆ ในการจัดการข้อมูลอย่างมีประสิทธิภาพ

Afrati, F. N., Delorey, D., Pasumansky, M., & Ullman, J. D. (2014) ได้ศึกษาการจัดเก็บและสอบถามข้อมูลโครงสร้างแบบต้นไม้ใน Dremel (Storing and Querying Tree-Structured Records in Dremel) โดยได้นำเสนอเทคนิค semi-flattening ในการจัดเก็บและสืบค้นข้อมูลที่มีโครงสร้างแบบต้นไม้ในระบบ Dremel เพื่อตอบสนองต่อความท้าทายในการจัดการข้อมูลขนาดใหญ่ โดยเทคนิคนี้ช่วยให้การสืบค้นข้อมูลในโครงสร้างที่ซับซ้อนมีประสิทธิภาพมากขึ้น โดยการปรับโครงสร้างข้อมูลให้เรียบง่ายขึ้นบางส่วน แต่ยังคงรักษาความสัมพันธ์ของข้อมูลในรูปแบบต้นไม้ ซึ่งช่วยสนับสนุนแนวทางในการจัดการข้อมูลที่มีโครงสร้างซับซ้อน ผลลัพธ์จากงานวิจัยแสดงให้เห็นว่า Dremel สามารถจัดการข้อมูลที่มีหลายระดับความลึกได้อย่างรวดเร็วและมีประสิทธิภาพ ทำให้ Dremel เหมาะสมสำหรับการใช้งานในระบบฐานข้อมูลที่ต้องรองรับข้อมูลขนาดใหญ่และซับซ้อน

Joshi, N., Satpute, N., Walgude, K., Korpad, D., & Ransing, E. (2024) ได้ศึกษาโครงสร้างข้อมูลลิงค์ลิสต์และระบบการจัดการห้องสมุด (Linked List Data Structure and Library Management System) โดยมีเป้าหมายเพื่อเพิ่มประสิทธิภาพและความคล่องตัวในการทำงานต่างๆ เช่น การเพิ่มหรือลบหนังสือออกจากห้องสมุด การติดตามประวัติการยืม รวมถึงสามารถช่วยจัดการกับทรัพยากรห้องสมุดได้อย่างมีประสิทธิภาพ เนื่องจากสามารถแทรกและลบโหนดที่แทนข้อมูลหนังสือได้ง่ายและรวดเร็ว โดยเฉพาะการใช้ Singly Linked List ที่มีประโยชน์อย่างยิ่งสำหรับการติดตามหนังสือที่ยืมมา ซึ่งช่วยให้ระบบสามารถจัดระเบียบข้อมูลและจัดการทรัพยากรได้อย่างมีประสิทธิภาพ ผลลัพธ์จากการศึกษาแสดงให้เห็นว่าโครงสร้างข้อมูลลิงค์ลิสต์สามารถเอาชนะข้อจำกัดของอาร์เรย์ โดยการจัดสรรหน่วยความจำแบบไดนามิกและการดำเนินการที่ยืดหยุ่น ทำให้ระบบการจัดการห้องสมุดมีความยืดหยุ่นมากขึ้น และสามารถปรับปรุงเวลาในการค้นหาและการเรียกค้นข้อมูลได้ดีขึ้น โดยเฉพาะเมื่อการจัดการข้อมูลขนาดเล็กหรือไม่ซับซ้อน

**4. วิธีการดำเนินการ**

การเตรียมข้อมูล ข้อมูลที่ใช้ในการทดสอบในครั้งนี้จะประกอบด้วยชุดข้อมูลที่มีลักษณะแบบสุ่มตัวเลข โดยชุดข้อมูลที่เลือกจะมีขนาด 3 ขนาด ได้แก่ 100, 1,000 และ 10,000 รายการ แต่ละรายการประกอบไปด้วย รหัสพนักงาน ชื่อ-นามสกุล ตำแหน่ง แผนก และบันทึกเหตุการณ์ต่าง ๆ เช่น การพัฒนาตนเอง การประเมินผล และการอนุมัติการขอขึ้นเงินเดือน จะถูกนำมาทดสอบในทั้งสองโครงสร้างข้อมูลตัวอย่างเช่น ชุดข้อมูล 100 การทดสอบเป็นการนำข้อมูลเข้าโดยเริ่มจากไม่มีข้อมูลอยู่เลยจนถึง 100 ชุด เป็นต้น โดยโครงสร้างข้อมูลต้นไม้ใช้ประเภท ไบนารีทรี (Binary Tree) และโครงสร้างข้อมูลลิงค์สิลต์ใช้ประเภท Singly Linked List ในการทำการทดสอบ

การทดสอบประสิทธิภาพของทั้งสองโครงสร้างข้อมูลจะประกอบด้วยการจัดการข้อมูล 3 ประเภท ได้แก่

1. การเพิ่มข้อมูล (Insertion) ทดสอบการเพิ่มข้อมูลใหม่ในโครงสร้างข้อมูลต้นไม้และโครงสร้างข้อมูลลิงค์ลิสต์ โดยจะวัดเวลาในการเพิ่มข้อมูลในแต่ละโครงสร้าง
2. การค้นหาข้อมูล (Search) ทดสอบการค้นหาข้อมูลในทั้งสองโครงสร้าง โดยการค้นหาข้อมูลที่มีลำดับชั้นในต้นไม้และการค้นหาในโครงสร้างข้อมูลลิงค์ลิสต์
3. การลบข้อมูล (Deletion) ทดสอบการลบข้อมูลในทั้งสองโครงสร้าง โดยจะวัดเวลาในการลบข้อมูลและปรับปรุงโครงสร้างข้อมูลหลังจากการลบ

เครื่องมือและเทคโนโลยี การทดสอบจะใช้ ภาษา Ruby ในการเขียนโค้ดทดสอบ โดยใช้ Benchmark.ips ในการวัดประสิทธิภาพของแต่ละโครงสร้างข้อมูล ผลการทดสอบจะถูกบันทึกในตาราง

**5. ผลการศึกษา**

จากการวิจัยเกี่ยวกับการเปรียบเทียบประสิทธิภาพระหว่างโครงสร้างข้อมูลต้นไม้ (Tree Data Structure) กับโครงสร้างข้อมูลลิงค์ลิสต์ (Linked List Structure) ได้ทำการทดสอบโดยใช้ 3 ชุดข้อมูล ได้แก่ ชุดข้อมูลขนาด 100, 1,000, 10,000 รายการ ซึ่งการทดสอบแต่ละชุดข้อมูลแบ่งออกเป็น 3 รอบการทดสอบ โดยแต่ละรอบมีเกณฑ์การวัดผลดังนี้

การจัดการข้อมูล หมายถึง การเพิ่ม ลบ และค้นหาข้อมูล ภายในโครงสร้างข้อมูล

1. การวัดจำนวนอัตราการจัดการข้อมูล (i/s) ย่อมาจาก iterations per second คือ จำนวนครั้งสูงสุดที่สามารถจัดการข้อมูลข้อมูลได้ภายในระยะเวลา 1 วินาที ซึ่งใช้ในการประเมินความเร็วโดยรวมของโครงสร้างข้อมูล

2. การวัดเวลาเฉลี่ยต่อการจัดการข้อมูล (ms/i) ย่อมาจาก milliseconds per iteration คือเวลาที่ใช้เฉลี่ยต่อการจัดการข้อมูลต่อครั้งในหน่วยมิลลิวินาที ใช้ในการวิเคราะห์ในเชิงลึก
3. การวัดเวลาเฉลี่ยต่อการจัดการข้อมูล (ns/i) ย่อมาจาก nanoseconds per iteration คือเวลาที่ใช้เฉลี่ยต่อการจัดการข้อมูลต่อครั้งในหน่วยนาโนวินาที

ผลการทดสอบประสิทธิภาพบันทึกในรูปแบบตาราง โดยประกอบด้วย ชุดข้อมูล ค่าความแปรปรวน อัตราการการจัดการข้อมูลต่อวินาที (i/s) และค่าเฉลี่ยเวลาในการจัดการข้อมูล (ms/i) ต่อครั้ง โดยมีรายละเอียดดังนี้

**ตารางที่ 1** ค่าเฉลี่ยการเพิ่มข้อมูลโครงสร้างข้อมูลต้นไม้จากการทดสอบจำนวน 3 ครั้ง

| ชุดข้อมูล | อัตราการเพิ่มข้อมูล (i/s) | ค่าความแปรปรวน (%) | เวลาเฉลี่ยในการเพิ่มข้อมูล (ms/i) ต่อครั้ง |
|---|---|---|---|
| 100 | 53.899 | ±7.4% | 18.56 |
| 1,000 | 12.639 | ±7.9% | 78.79 |
| 10,000 | 3.785 | ± 0.0% | 264.28 |

จากผลการทดสอบประสิทธิภาพการเพิ่มข้อมูล พบว่า เมื่อชุดข้อมูลเพิ่มขึ้น อัตราการเพิ่มข้อมูลลดลง ขณะที่เวลาเฉลี่ยในการเพิ่มข้อมูลต่อครั้งเพิ่มขึ้น โดยชุดข้อมูล 100 รายการ อัตราการเพิ่มที่ 53.899 รายการต่อวินาที และใช้เวลาเฉลี่ย 18.56 มิลลิวินาที ชุดข้อมูล 1,000 รายการ อัตราการเพิ่มลดลงเหลือ 12.639 รายการต่อวินาที ใช้เวลาเฉลี่ย 78.79 มิลลิวินาที และชุดข้อมูล 10,000 รายการ อัตราการเพิ่มลดลงเหลือ 3.785 รายการต่อวินาที ใช้เวลาเฉลี่ย 264.28 มิลลิวินาที

**ตารางที่ 2** ค่าเฉลี่ยการค้นหาข้อมูลโครงสร้างข้อมูลต้นไม้จากการทดสอบจำนวน 3 ครั้ง

| ชุดข้อมูล | อัตราการค้นหาข้อมูล (i/s) | ค่าความแปรปรวน (%) | เวลาเฉลี่ยในการค้นหาข้อมูล (ns/i) ต่อครั้ง |
|---|---|---|---|
| 100 | 5.312M | ±1.5% | 188.23 |
| 1,000 | 5.293M | ±1.6% | 188.78 |
| 10,000 | 5.080M | ± 1.5% | 196.83 |

จากผลการทดสอบประสิทธิภาพการค้นหาข้อมูล พบว่า เมื่อชุดข้อมูลเพิ่มขึ้น อัตราการค้นหาข้อมูลลดลงเล็กน้อย ขณะที่เวลาเฉลี่ยในการค้นหาข้อมูลต่อครั้งเพิ่มขึ้นเล็กน้อย โดยสำหรับชุดข้อมูล 100 รายการ อัตราการค้นหาที่ 5.312M รายการต่อวินาที ใช้เวลาเฉลี่ย 188.23 นาโนวินาที ชุดข้อมูล 1,000 รายการ อัตราการค้นหาที่ 5.293M รายการต่อวินาที ใช้เวลาเฉลี่ย 188.78 นาโนวินาที และชุดข้อมูล 10,000 รายการ อัตราการค้นหาลดลงเหลือ 5.080M รายการต่อวินาที ใช้เวลาเฉลี่ย 196.83 นาโนวินาที

**ตารางที่ 3** ค่าเฉลี่ยการลบข้อมูลโครงสร้างข้อมูลต้นไม้จากการทดสอบจำนวน 3 ครั้ง

| ชุดข้อมูล | อัตราการลบข้อมูล (i/s) | ค่าความแปรปรวน (%) | เวลาเฉลี่ยในการลบข้อมูล (ns/i) ต่อครั้ง |
|---|---|---|---|
| 100 | 5.180M | ±1.3% | 193.08 |
| 1,000 | 5.172M | ±1.6% | 193.34 |
| 10,000 | 4.924M | ±1.6% | 203.08 |

จากผลการทดสอบประสิทธิภาพการลบข้อมูล พบว่า เมื่อชุดข้อมูลเพิ่มขึ้น อัตราการลบข้อมูลลดลงเล็กน้อย ขณะที่เวลาเฉลี่ยในการลบข้อมูลต่อครั้งเพิ่มขึ้นเล็กน้อย โดยชุดข้อมูล 100 รายการ อัตราการลบที่ 5.312M รายการต่อวินาที ใช้เวลาเฉลี่ย 193.08 นาโนวินาที ชุดข้อมูล 1,000 รายการ อัตราการลบที่ 5.172M รายการต่อวินาที ใช้เวลาเฉลี่ย 193.34 นาโนวินาที และชุดข้อมูล 10,000 รายการ อัตราการลบลดลงเหลือ 4.924M รายการต่อวินาที เวลาเฉลี่ย 203.08 นาโนวินาที

**ตารางที่ 4** ค่าเฉลี่ยการเพิ่มข้อมูลโครงสร้างข้อมูลลิงค์สิสต์จากการทดสอบจำนวน 3 ครั้ง

| ชุดข้อมูล | อัตราการเพิ่มข้อมูล (i/s) | ค่าความแปรปรวน (%) | เวลาเฉลี่ยในการเพิ่มข้อมูล (ms/i) ต่อครั้ง |
|---|---|---|---|
| 100 | 13.982 | ±7.1% | 71.52 |
| 1,000 | 1.377 | ±0.0% | 726.04 |
| 10,000 | 0.107 | ± 0.0% | 9,345.85 |

จากผลการทดสอบประสิทธิภาพการเพิ่มข้อมูล พบว่า เมื่อชุดข้อมูลเพิ่มขึ้น อัตราการเพิ่มข้อมูลลดลงต่อเนื่อง ขณะที่เวลาเฉลี่ยในการเพิ่มข้อมูลต่อครั้งเพิ่มขึ้นอย่างชัดเจน โดยชุดข้อมูล 100 รายการ มีอัตราการเพิ่มที่ 13.982 รายการต่อวินาที ใช้เวลาเฉลี่ย 71.52 มิลลิวินาที ชุดข้อมูล 1,000 รายการ อัตราการเพิ่มที่ 1.377 รายการต่อวินาที ใช้เวลาเฉลี่ย 726.04 มิลลิวินาที และชุดข้อมูล 10,000 รายการ อัตราการเพิ่มลดลงเหลือ 0.107 รายการต่อวินาที ใช้เวลาเฉลี่ย 9,345.85 มิลลิวินาที

**ตารางที่ 5** ค่าเฉลี่ยการค้นหาข้อมูลโครงสร้างข้อมูลลิงค์สิสต์จากการทดสอบจำนวน 3 ครั้ง

| ชุดข้อมูล | อัตราการค้นหาข้อมูล (i/s) | ค่าความแปรปรวน (%) | เวลาในการค้นหาข้อมูล (ns/i) ต่อครั้ง |
|---|---|---|---|
| 100 | 5.665M | ±2.0% | 176.50 |
| 1,000 | 5.669M | ±2.1% | 176.40 |
| 10,000 | 5.500M | ±1.5% | 181.79 |

จากผลการทดสอบประสิทธิภาพการค้นหาข้อมูล พบว่า เมื่อชุดข้อมูลเพิ่มขึ้น อัตราการค้นหาข้อมูลลดลงเล็กน้อย ขณะที่เวลาเฉลี่ยต่อการค้นหาข้อมูลแต่ละครั้งเพิ่มขึ้นเล็กน้อย โดยสำหรับชุดข้อมูล 100 รายการ อัตราการค้นหาที่ 5.665M รายการต่อวินาที ใช้เวลาเฉลี่ย 176.50 นาโนวินาที ชุดข้อมูล 1,000 รายการ อัตราการค้นหาที่ 5.669M รายการต่อวินาที ใช้เวลาเฉลี่ย 176.40 นาโนวินาที และชุดข้อมูล 10,000 รายการ อัตราการการค้นหาลดลงเล็กน้อยเหลือ 5.500M รายการต่อวินาที ใช้เวลาเฉลี่ย 181.79 นาโนวินาที

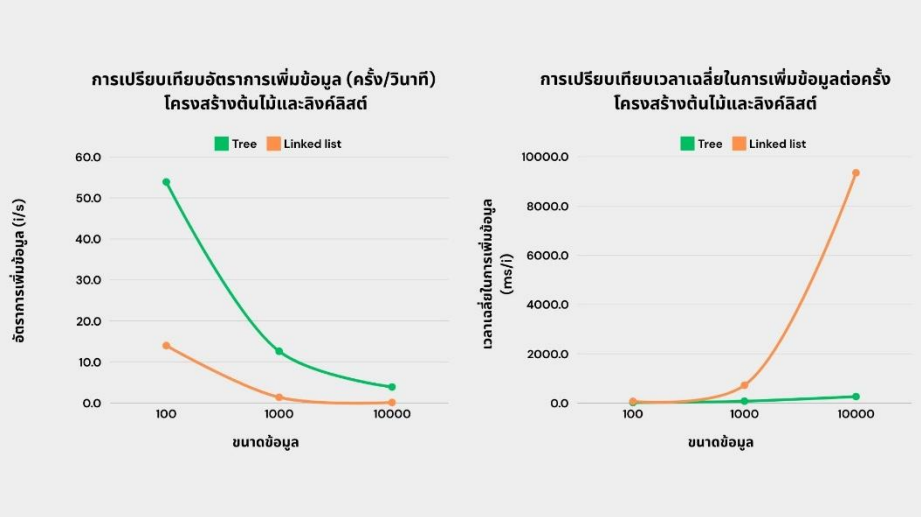
**ตารางที่ 6** ค่าเฉลี่ยผลการลบข้อมูลโครงสร้างข้อมูลลิงค์สิสต์จากการทดสอบจำนวน 3 ครั้ง

| ชุดข้อมูล | อัตราการลบข้อมูล (i/s) | ค่าความแปรปรวน (%) | เวลาในการลบข้อมูล (ns/i) ต่อครั้ง |
|---|---|---|---|
| 100 | 5.758M | ±1.5% | 173.59 |
| 1,000 | 5.855M | ±1.4% | 170.81 |
| 10,000 | 5.600M | ± 1.0% | 178.61 |

จากผลการทดสอบประสิทธิภาพการลบข้อมูล พบว่า เมื่อชุดข้อมูลเพิ่มขึ้น อัตราการลบข้อมูลลดลงเล็กน้อย ขณะที่เวลาในการลบข้อมูลต่อครั้งเพิ่มขึ้นเล็กน้อย โดยชุดข้อมูล 100 รายการ อัตราการลบที่ 5.758M รายการต่อวินาที ใช้เวลาเฉลี่ย 173.59 นาโนวินาที ชุดข้อมูล 1,000 รายการ อัตราการลบที่ 5.855M รายการต่อวินาที ใช้เวลาเฉลี่ย 170.81 นาโนวินาที และชุดข้อมูล 10,000 รายการ อัตราการลบลดลงเล็กน้อยเหลือ 5.600M รายการต่อวินาที ใช้เวลาเฉลี่ย 178.61 นาโนวินาที

## 6. อภิปรายผล

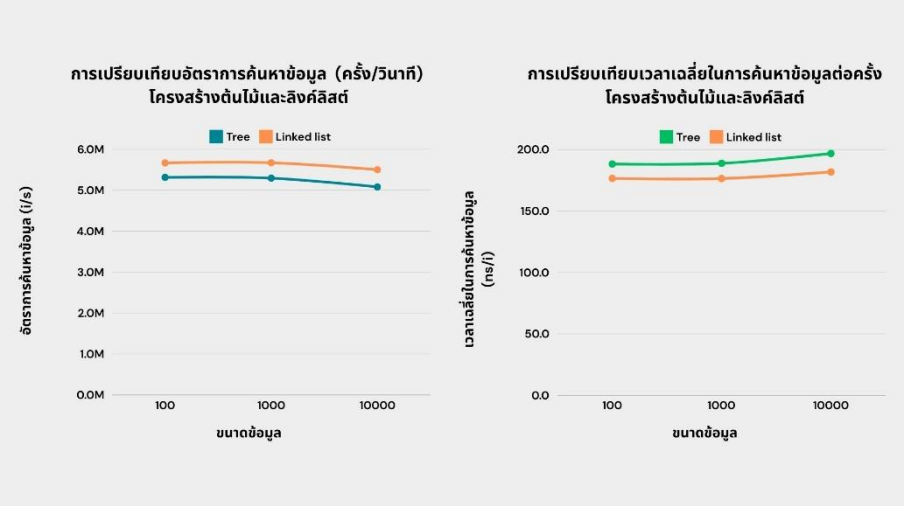
ในการศึกษาครั้งนี้ ได้ทำการเปรียบเทียบประสิทธิภาพระหว่างโครงสร้างข้อมูลต้นไม้กับโครงสร้างข้อมูลลิงค์ลิสต์ในการจัดการข้อมูลบุคคลของฝ่ายทรัพยากรมนุษย์ โดยทำการทดสอบประสิทธิภาพทั้ง 2 โครงสร้างและบันทึกผลการทดสอบ สรุปผลจากการทดลองและแสดงข้อมูลในกราฟดังนี้




**ภาพที่ 1** การเปรียบเทียบประสิทธิภาพการเพิ่มข้อมูล

จากภาพที่ 1 เป็นการแสดงผลการเปรียบเทียบประสิทธิภาพสำหรับการเพิ่มข้อมูลระหว่างโครงสร้างข้อมูลต้นไม้ กับโครงสร้างข้อมูลลิงค์ลิสต์ โดยกราฟทางซ้ายมือแสดงการเปรียบเทียบอัตราการเพิ่มข้อมูล (ครั้ง/วินาที) จากกราฟจะเห็นได้ว่า โครงสร้างข้อมูลต้นไม้มีอัตราการเพิ่มข้อมูลสูงกว่าโครงสร้างข้อมูลลิงค์ลิสต์ในทุกชุดข้อมูล โดยเฉพาะในชุดข้อมูลขนาด 10,000 รายการ ผลลัพธ์จากการเพิ่มข้อมูลในโครงสร้างข้อมูลต้นไม้มากกว่า โครงสร้างข้อมูลลิงค์ลิสต์ถึง 35 เท่า

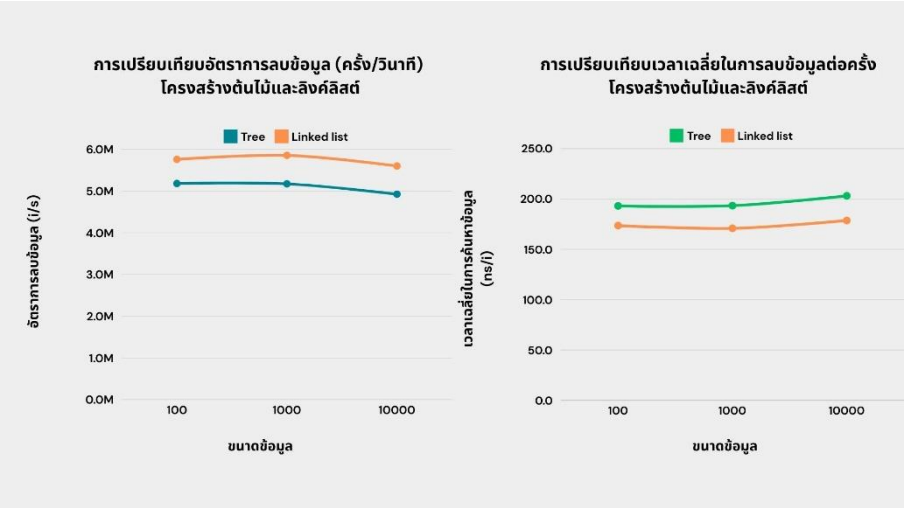
กราฟทางขวามือแสดงการเปรียบเทียบเวลาเฉลี่ยในการเพิ่มข้อมูลต่อครั้ง (มิลลิวินาที/ครั้ง) พบว่าโครงสร้างข้อมูลลิงค์ลิสต์ใช้เวลาเฉลี่ยในการเพิ่มข้อมูลต่อครั้งสูงขึ้นอย่างชัดเจนเมื่อชุดข้อมูลเพิ่มขึ้น โดยเฉพาะชุดข้อมูล 10,000 รายการซึ่งใช้เวลามากถึง 9,345.85 มิลลิวินาที ขณะที่โครงสร้างข้อมูลต้นไม้ใช้เวลาในการเพิ่มข้อมูลต่อครั้งน้อยกว่ามาก ใช้เวลาเพียง 264.28 มิลลิวินาที




**ภาพที่ 2** การเปรียบเทียบประสิทธิภาพการค้นหาข้อมูล

จากภาพที่ 2 เป็นการแสดงผลการเปรียบเทียบประสิทธิภาพสำหรับการค้นหาข้อมูลระหว่างโครงสร้างข้อมูลต้นไม้กับโครงสร้างข้อมูลลิงค์ลิสต์ โดยกราฟทางซ้ายมือแสดงการเปรียบเทียบอัตราการค้นหาข้อมูล (ครั้ง/วินาที) พบว่าโครงสร้างข้อมูลต้นไม้มีอัตราการค้นหาข้อมูลน้อยกว่าโครงสร้างข้อมูลลิงค์ลิสต์เพียงเล็กน้อยในทุกชุดข้อมูล แต่ความแตกต่างนั้นไม่มากนัก ดังนั้นการค้นหาข้อมูลในทั้งสองโครงสร้างนี้จึงใกล้เคียงกัน

กราฟทางขวามือแสดงการเปรียบเทียบเวลาเฉลี่ยในการค้นหาข้อมูลต่อครั้ง (มิลลิวินาที/ครั้ง) พบว่าโครงสร้างข้อมูลลิงค์ลิสต์ใช้เวลาเฉลี่ยในการค้นหาข้อมูลต่อครั้งน้อยกว่าโครงสร้างข้อมูลต้นไม้เพียงเล็กน้อยในทุกชุดข้อมูล ซึ่งแสดงให้เห็นว่าเวลาที่ใช้ในการค้นหาข้อมูลระหว่างทั้งสองโครงสร้างนั้นไม่แตกต่างกันมากจนส่งผลกระทบต่อประสิทธิภาพโดยรวม




**ภาพที่ 3** การเปรียบเทียบประสิทธิภาพการลบข้อมูล

จากภาพที่ 3 เป็นการแสดงผลการเปรียบเทียบประสิทธิภาพสำหรับการลบข้อมูลระหว่างโครงสร้างข้อมูลต้นไม้ กับโครงสร้างข้อมูลลิงค์ลิสต์ โดยกราฟทางซ้ายมือแสดงการเปรียบเทียบอัตราการลบข้อมูล (ครั้ง/วินาที) พบว่าโครงสร้างข้อมูลต้นไม้มีอัตราการลบข้อมูลน้อยกว่าโครงสร้างข้อมูลลิงค์ลิสต์เพียงเล็กน้อยในทุกชุดข้อมูลแต่ความแตกต่างนั้นไม่มากนัก ดังนั้นการลบข้อมูลในทั้งสองโครงสร้างนี้จึงใกล้เคียงกัน

กราฟทางขวามือแสดงการเปรียบเทียบเวลาเฉลี่ยในการลบข้อมูลต่อครั้ง (มิลลิวินาที/ครั้ง) พบว่าโครงสร้างข้อมูลลิงค์ลิสต์ใช้เวลาเฉลี่ยในการลบข้อมูลต่อครั้งน้อยกว่าโครงสร้างข้อมูลต้นไม้เพียงเล็กน้อยในทุกชุดข้อมูล แต่ความแตกต่างในเวลาที่ใช้ในการลบข้อมูลระหว่างทั้งสองโครงสร้างนั้นไม่แตกต่างมากจนส่งผลกระทบต่อประสิทธิภาพโดยรวม

การศึกษาและการเปรียบเทียบประสิทธิภาพของโครงสร้างข้อมูลต้นไม้และโครงสร้างข้อมูลลิงค์ลิสต์ พบว่า โครงสร้างข้อมูลต้นไม้ มีความสามารถในการจัดการข้อมูลที่มีความซับซ้อนและหลายระดับได้ดีกว่า โดยเฉพาะในกรณีของการจัดการข้อมูลที่มีความสัมพันธ์แบบลำดับชั้น ซึ่งตรงกับผลการศึกษาของ Afrati et al. (2014) ที่แสดงให้เห็นว่า Dremel ใช้เทคนิค semi-flattening ในการจัดการข้อมูลต้นไม้ในระบบฐานข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพ ซึ่งสามารถขยายตัวและเพิ่มประสิทธิภาพในการจัดการข้อมูลที่ซับซ้อนและมีหลายระดับได้ดีกว่าโครงสร้างข้อมูลแบบอื่น

นอกจากนี้ ผลการศึกษาของ Joshi et al. (2024) ได้ศึกษาเกี่ยวกับการใช้โครงสร้างข้อมูลลิงค์ลิสต์ในระบบการจัดการห้องสมุด โดยเฉพาะการใช้ Singly Linked List ที่ช่วยเพิ่มประสิทธิภาพในการเพิ่มหรือลบหนังสือ ติดตามประวัติการยืม และจัดการความพร้อมของหนังสือได้อย่างรวดเร็วและมีความยืดหยุ่น ซึ่งสนับสนุนข้อค้นพบของการใช้โครงสร้างข้อมูลลิงค์ลิสต์ในระบบที่ข้อมูลมีขนาดเล็กและไม่ซับซ้อน หรือกรณีที่การเพิ่ม/ลบข้อมูลมีความสำคัญมากกว่าการค้นหาข้อมูล

## 7. สรุปผลการวิจัย

สรุปผลการศึกษา

จากการศึกษาการเปรียบเทียบประสิทธิภาพระหว่างโครงสร้างข้อมูลต้นไม้กับโครงสร้างข้อมูลลิงค์ลิสต์ในการเพิ่มข้อมูลพบว่า โครงสร้างข้อมูลต้นไม้ มีประสิทธิภาพดีกว่าในทุกขนาดของชุดข้อมูล โดยเฉพาะในกรณีที่ข้อมูลมีขนาดใหญ่

1. ชุดข้อมูล 100 รายการ:
    - โครงสร้างข้อมูลต้นไม้: อัตราการเพิ่มข้อมูล 53.899 i/s
    - โครงสร้างข้อมูลลิงค์ลิสต์: อัตราการเพิ่มข้อมูล 43.210 i/s
2. ชุดข้อมูล 1,000 รายการ:
    - โครงสร้างข้อมูลต้นไม้: อัตราการเพิ่มข้อมูล 12.639 i/s
    - โครงสร้างข้อมูลลิงค์ลิสต์: อัตราการเพิ่มข้อมูล 10.032 i/s
3. ชุดข้อมูล 10,000 รายการ:
    - โครงสร้างข้อมูลต้นไม้: อัตราการเพิ่มข้อมูล 3.785 i/s
    - โครงสร้างข้อมูลลิงค์ลิสต์: อัตราการเพิ่มข้อมูล 0.107 i/s

ทั้งนี้ในการค้นหาและลบข้อมูลผลลัพธ์ของทั้งสองโครงสร้างไม่ต่างกันมากนัก โดยโครงสร้างข้อมูลลิงค์ลิสต์จะมีประสิทธิภาพดีกว่าเล็กน้อยในบางกรณี (ประมาณ 2-5%) แต่ไม่ส่งผลกระทบต่อประสิทธิภาพโดยรวม สรุปได้ว่า โครงสร้างข้อมูลต้นไม้เหมาะสมสำหรับการจัดการข้อมูลที่มีลำดับชั้นซับซ้อนและมีปริมาณมาก เช่น ข้อมูลพนักงานในองค์กร ที่ประกอบไปด้วยรหัสพนักงาน ชื่อ-นามสกุล ตำแหน่ง แผนก และบันทึกเหตุการณ์ต่าง ๆ ส่วนโครงสร้างข้อมูลลิงค์ลิสต์เหมาะสำหรับข้อมูลขนาดเล็กและไม่ซับซ้อน การศึกษาเหล่านี้ได้รับการสนับสนุนจากงานวิจัยต่างๆ ที่ได้กล่าวถึงการใช้โครงสร้างข้อมูลต้นไม้ในการจัดการข้อมูลที่มีหลายระดับ และการใช้โครงสร้างข้อมูลลิงค์ลิสต์ในการจัดการข้อมูลที่ไม่ซับซ้อน

- งานวิจัยโดย Afrati et al. (2014) แสดงให้เห็นว่าโครงสร้างข้อมูลต้นไม้สามารถจัดการข้อมูลที่มีหลายระดับได้อย่างมีประสิทธิภาพในระบบ Dremel ซึ่งช่วยยืนยันถึงความเหมาะสมของโครงสร้างข้อมูลต้นไม้ในการจัดการข้อมูลที่มีความซับซ้อน
- งานวิจัยโดย Joshi et al. (2024) สนับสนุนการใช้โครงสร้างข้อมูลลิงค์ลิสต์ในระบบการจัดการห้องสมุดที่ต้องการความยืดหยุ่นในการเพิ่มหรือลบข้อมูล โดยแสดงให้เห็นถึงประสิทธิภาพในการจัดการข้อมูลที่ไม่ซับซ้อนในระบบต่างๆ

**8. ข้อเสนอแนะ**

ควรมีการศึกษาการปรับปรุงโครงสร้างข้อมูลให้มีประสิทธิภาพยิ่งขึ้น เช่น การประยุกต์ใช้โครงสร้างข้อมูลแบบ Balanced Tree หรือ Doubly Linked List หรือข้อมูลที่มีความซับซ้อนสูงขึ้น อีกทั้งควรทำการเปรียบเทียบกับโครงสร้างข้อมูลประเภทอื่น ๆ เช่น Hash Graph เพื่อให้ผลการวิจัยมีความครอบคลุมและสามารถประยุกต์ใช้ได้อย่างมีประสิทธิภาพในสถานการณ์ที่หลากหลายยิ่งขึ้น

**9. กิตติกรรมประกาศ**

งานวิจัยสำเร็จลุล่วงได้ด้วยดี เนื่องจากได้รับความกรุณาอย่างสูงจากผศ.ก่องกาญจน์ ดุลยไชย ประธานที่ปรึกษาโครงงาน ผศ.ดร.พาสน์ ปราโมกข์ชน และ ผศ.ดร.สมนึก สินธุปวน กรรมการที่ปรึกษาโครงงาน ที่ให้คำแนะนำและข้อคิดเห็นต่าง ๆ อันเป็นประโยชน์อย่างยิ่งต่องานวิจัย ขอขอบคุณพี่พนักงานบริษัท บานาน่าโคดดิ้ง จำกัด ที่ให้คำแนะนำการจัดเก็บข้อมูลทรัพยากรบุคคล ขอขอบคุณบุคลากรสาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้ ที่ให้ความรู้และคำปรึกษาด้านการจัดทำเอกสารโครงงาน รวมถึงสนับสนุน ตรวจสอบและรายงานผลความคืบหน้าการทำสหกิจศึกษา ตลอดจนการปรับปรุงแก้ไขข้อบกพร่องต่างๆ จนทำให้งานวิจัยครั้งนี้สำเร็จลุล่วงด้วยดี ผู้จัดทำขอขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

**10. เอกสารอ้างอิง**

Afrati, F. N., Delorey, D., Pasumansky, M., & Ullman, J. D. (2014). **Storing and Querying Tree-Structured Records in Dremel.** Retrieved from https://research.google/pubs/storing-and-querying-tree-structured-records-in-dremel/

Božinovski, A., Tanev, G., Stojčevska, B., Pačovski, V., & Ackovska, N. (2017). **Time Complexity Analysis of the Binary Tree Roll Algorithm.** Retrieved from https://www.researchgate.net/publication/321073392_Time_Complexity_Analysis_of_the_Binary_Tree_Roll_Algorithm

Fetaji, M., Ebibi, M., & Fetaji, B. (2021). **Measuring Algorithms Performance in Dynamic Linked List and Arrays.** Retrieved from https://www.researchgate.net/publication/356942264_Measuring_Algorithms_Performance_in_Dynamic_Linked_List_and_Arrays

Joshi, N., Satpute, N., Walgude, K., Korpad, D., & Ransing, E. (2024) **Linked List Data Structure and Library Management System of the AVL and red-black trees.** Retrieved from https://www.researchgate.net/publication/380886329_Linked_List_Data_Structure_and_Library_Management_System

Samoladas, D., Karras, C., Karras, A., Sioutas, S., & Theodorakopoulos, L. (2023). **Tree Data Structures and Efficient Indexing Techniques for Big Data Management: A Comprehensive Study.** Retrieved from https://www.researchgate.net/publication/369621229_Tree_Data_Structures_and_Efficient_Indexing_Techniques_for_Big_Data_Management_A_Comprehensive_Study