**Data Structures and Algorithm**

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

**การทดลองที่ 6 :** AVL Tree และ B-Tree

**จุดประสงค์**

1. นักศึกษาเข้าใจการทำงานของ AVL Tree และ B-Tree ได้

**ตอนที่ 1 :** สร้าง AVL Tree

1. นักศึกษาใช้เว็บไซต์สำหรับสร้าง AVL Tree

|  |
| --- |
| <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html> |

1. อธิบายการทำงานของ AVL Tree เมื่อมีการ insert / delete ตามรายการต่อไปนี้

| **No.** | **การทำงาน** | **การทำงานใน Tree (ต่อท้าย node อะไร / มีการทำ rebalance อย่างไรบ้าง)** |
| --- | --- | --- |
| 1 | Insert 50 | 50 เป็น root node |
| 2 | Insert 30 | 30 อยู่ฝั่งซ้ายของ 50 |
| 3 | Insert 40 | 40 น้อยกว่า 50 และมากกว่า 30 จะอยู่ฝั่งขวาของ 30 และทำการ rebalancing  แบบ right subtree of left son of x เอา 40 เป็น root  X=50, y=30, z=40 |
| 4 | Insert 20 | 20 อยู่ซ้ายของ 40 และอยู่ซ้ายของ 30 |
| 5 | Insert 35 | 35 อยู่ซ้ายของ 40 และอยู่ขวาของ 30 |
| 6 | Insert 45 | 45 อยู่ขวาของ 40 และอยู่ซ้ายของ 50 |
| 7 | Insert 12 | 12 อยู่ทางซ้ายของ 20 |
| 8 | Insert 15 | เอา 15 ไปอยู่ทางขวาของ 30 ใช้การ rebalancing แบบ Right subtree of left son of y |
| 9 | Delete 45 | เอา 30 ขึ้นมา ย้าย 40 ไปเป็น node ขวา  Rebalancing แบบ left subtree of left son of x |
| 10 | Insert 8 | เอา 8 ไปอยู่ซ้ายของ 12 |
| 11 | Delete 15 | เอา 15 ออก เอา 12 ขึ้นมาแทน |
| 12 | Insert 80 | เอา 80 ไปอยู่ด้สนขวา 50 |
| 13 | Insert 100 | เอา 80 เลือนขึ้นไปทางขวาของ 40 แล้ว  เอา 100 ไปอยู่ทางขวาของ 80 Right subtree of right |
| 14 | Delete 40 | เอา 35 ขึ้นไปก่อน และเกิดไม่ balance  เลยต้องทำ rebalancing ด้วย วิธี Right subtree of right |
| 15 | Insert 55 | เอา 50 ไปอยู่ขวาของ 80 ทำ rebalancing แบบ Right subtree of left |
| 16 | Delete 100 | เอา 50 ไปอยู่ขวาของ 30 ทำ rebalancing  แบบ left subtree of left |

1. จากการทำงานดังกล่าว การ reblance ของ AVL Tree เกิดขึ้นในกรณีใด และมีขั้นตอนการ rebalance อย่างไรบ้าง

กรณีที่จะทำให้เราต้อง rebalance มีอยู่ 4 กรณี

กรณี 1 เพิ่มโหนดใหม่เข้าไปที่ subtree ด้านซ้าย ของโหนดลูกด้านซ้าย

กรณี 2 เพิ่มโหนดใหม่เข้าไปที่ subtree ด้านขวา ของโหนดลูกด้านซ้าย

กรณี 3 เพิ่มโหนดใหม่เข้าไปที่ subtree ด้านซ้ายของโหนดลูกด้านขวา

กรณี 4 เพิ่มโหนดใหม่เข้าไปที่ subtree ด้านขวาของโหนดลูกด้านขวา

ทั้ง 4 กรณีนี้อาจจะทำให้ length ด้านซ้ายมีความยาวต่างกับ length ด้านขวาเท่ากับ 2 ก็ได้ แต่กรณีที่ 1 กับ 4 เราเรียกว่าเป็น outside เพราะอยู่ขอบนอก แบบ outside นี้จะ rebalance โดยวิธี single-rotation ส่วนกรณีที่ 2 กับ 3 เราเรียกว่าเป็น inside เพราะอยู่ด้านใน แบบ inside นี้จะ rebalance โดยวิธี double-rotation

**ตอนที่ 2 :** สร้าง B-Tree

1. ให้นักศึกษาใช้เว็บไซต์สำหรับสร้าง B-Tree แล้ววาดลักษณะของ Tree โดยใช้ค่า Max Degree = 4

|  |
| --- |
| <https://www.cs.usfca.edu/~galles/visualization/BTree.html> |

1. อธิบายการทำงานของ B-Tree เมื่อมีการ insert / delete ตามรายการต่อไปนี้

| **No.** | **การทำงาน** | **การทำงานใน Tree (ต่อท้าย node อะไร / มีการทำ balance อย่างไรบ้าง)** |
| --- | --- | --- |
| 1 | Insert 50 | 50 เป็น node แรก |
| 2 | Insert 80 | อยู่โหนดเดียวกับ 50 |
| 3 | Insert 20 | ไปต่อข้างหน้า 50 อยู่โหนดเดียวกัน |
| 4 | Insert 100 | 50 เป็น root node 100ไปอยู่กับ80 และ 20  เรียง 20 80 100 |
| 5 | Insert 35 | 35 ไปอยู่ node เดียวกับ 20 ด้านซ้าย |
| 6 | Insert 120 | 120 ไปอยู่โหนดเดียวกับ 80,100 ด้านขวาของ root 50 |
| 7 | Insert 150 | 100 จะไปอยู่กับ 50 ด้านบน 80 เป็น โหนดใหม่ อยู่กลาง 120,150 อยู่ด้านซ้าย |
| 8 | Insert 110 | 110 ไปอยู่โหนดเดียวกับ 120 150 ทางด้านซ้ายของรูทโหนด |
| 9 | Delete 80 | จะเอาฝั่งน้อยขึ้นด้านขวาคือ 35 ที่อยู่กับ 20 ขึ้นมาเป็น รูท 50 แตกเป็นโหนดใหม่ |
| 10 | Insert 8 | 8 ไปอยู่ข้าง01หน้าโหนดเดียวกับ 20 ด้านซ้าย |
| 11 | insert 130 | เอา120 ขึ้นไป รูทโหนด 130 150 ไปอยู่ โหนดขวา ของ รูท 110 อยู่กลางขวาของ รูท |
| 12 | Insert 30 | ไปอยู่ด้านซ้ายของ รูท ซึ่งมี 8 20 |
| 13 | Insert 25 | 35 เป็นรูทตัวเดียว 20 เป็น โหนด ซ้าย 35  8 เป็นลูกของ 20 ทางซ้าย 25 30 ไปอยู่ 20 ดานขวา |
| 14 | Delete 120 | เอา 130 ขึนไปแทน |
| 15 | Delete 100 | เอา 50 ไปรวมอยู่ โหนดเดียวกับ 110 |
| 16 | Insert 100 | เอา 100 ไปรวมอยู่ โหนดเดียวกับ 50 100 |

1. จากการทำงานดังกล่าว การ balance ของ B-Tree เกิดขึ้นในกรณีใด และมีขั้นตอนการ balance อย่างไรบ้าง

คือการทำให้โหนดด้านล่างเต็มก่อน ถ้าล้น จะทำการ splite new node แบบ n/2 ก่อนแล้วจะเอา

ตำแหน่งเลลือนขึ้นไปเป็นโหนด ด้านบน แต่ถ้าทำการลบจะเลือกฝั่งที่น้อยกว่านั้นขึ้นไปแทนตำแหน่0งที่ถูกลบ