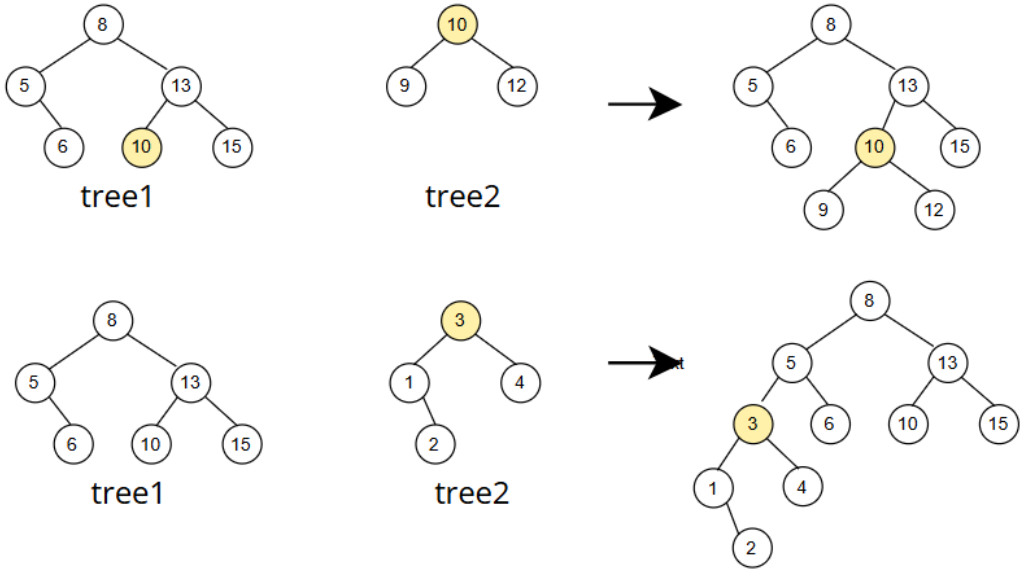
Tree Grafting (1 sec, 512 mb)

ในชีวิตจริง การตัดแต่งต้นไม้นั้นก็เป็นสิ่งที่สำคัญ นอกจากจะทำให้ต้นไม้ของเราแข็งแรง สวยงามแล้ว ยังสามารถขยายพันธุ์ต้นใหม่ได้อีกด้วย เทคนิคที่น่าสนใจก็จะมี 2 อย่างด้วยกัน คือ การต่อกิ่ง (grafting) หรือการนำต้นไม้อีกต้นหนึ่ง มาต่อกับต้นไม้ที่แข็งแรงกว่า เพื่อที่จะให้เติบโตได้ดีขึ้น และ*สวยงามที่สุด*

ในโจทย์ข้อนี้จึงจะให้น้อง ๆ ได้เป็นชาวสวนที่ต้องตัดแต่งต้นไม้ โดยการเขียนฟังก์ชั่น `void CP::map_bst ::modify(CP::map_bst &other)` ขึ้นมานั่นเอง โดยโจทย์จะให้ binary search tree มาสองต้น tree1 และ tree2 ที่เมื่อต่อกิ่ง (graft) เรียบร้อยแล้วจะ*สวยงามมากเลย* จนไม่ต้องทำการตัดกิ่ง (prune) ต่อ คือจะรับประกันว่าหลังจากต่อแล้วจะเป็น binary search tree แน่นอน

ในการต่อกิ่งจะใช้ฟังก์ชั่น `void CP::map_bst::graft(node* n, node* m, size_t size)` โดยจะเป็นการต่อราก (mRoot) ของ tree2 เข้าไปรวมอยู่ใน tree1 ในจุดที่เหมาะสมที่สุดดังนี้:

- ทำการไล่ค่าลงจาก tree1 โดยอิงค่า mRoot ของ tree2
- ถ้ามี node ใน tree1 ที่มีค่าเท่ากับ mRoot ของ tree2
    - ถ้า node นั้นเป็น leaf node ให้ต่อกิ่งโดยแทนที่ node นั้นด้วย tree2 ทั้งต้นเลย
    - ถ้า node นั้นมีลูกอย่างน้อยหนึ่งด้าน ไม่ต้องต่อกิ่ง
- ถ้าไม่มีค่าที่เท่ากัน ให้ไล่ค้นตามกฎลงมาจนเจอตำแหน่งที่เหมาะสม แล้วต่อกิ่งโดยให้ mRoot ของ tree2 เป็นลูกของ node นั้นเลย
- หลังจากต่อกิ่งแล้ว **tree2 จะหายไป**



รูปที่ 1-2 ตัวอย่างการ graft ที่ไม่จำเป็นต้อง prune

ข้อบังคับ

- โจทย์ข้อนี้จะมีไฟล์ตั้งต้นมาให้ซึ่งประกอบด้วยไฟล์ map_bst.h, main.cpp และ student.h อยู่ ให้เขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ **ส่งเฉพาะไฟล์ student.h เท่านั้น**
    - ไฟล์ student.h จะต้อง**ไม่ทำการอ่านเขียนข้อมูลใด ๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใด ๆ**
- หากใช้ VS Code ให้ทำการ compile ที่ไฟล์ main.cpp

  **** main ที่ใช้จริงใน grader นั้นจะแตกต่างจาก main ที่ได้รับในไฟล์เริ่มต้น แต่จะทดสอบในลักษณะเดียวกัน ****

คำอธิบายฟังก์ชัน main

ขั้นตอนการทำงานของ main ที่ให้

- รับค่า n,m คือขนาดของ tree1 และ tree2 ตามลำดับ
- รับค่า key จาก vector ลงไปใน tree1 และ tree2 โดยใส่แบบใช้คำสั่ง insert
- เรียกคำสั่ง tree1.modify(tree2)
- printResult tree1 (เพื่อเทียบกับคำตอบจริง)

ในการให้คะแนนจริงของ main จะตรวจสอบ**ขนาด**ของต้นไม้ tree1 และลำดับ key ของต้นไม้ tree1 แบบ **inorder และ preorder** และอาจมีการ modify หลายครั้งในข้อเดียว

ตัวอย่างการทำงานของ main

ค่าที่รับเข้ามา	ค่าที่ส่งออก	คำอธิบาย
6 3 8 5 13 6 10 15 10 9 12	mSize: 8 Inorder: 5 6 8 9 10 12 13 15 Preorder: 8 5 6 13 10 9 12 15	**ตามรูปที่ 1**
6 4 8 5 13 6 10 15 3 1 2 4	mSize: 10 Inorder: 1 2 3 4 5 6 8 10 13 15 Preorder: 8 5 3 1 2 4 6 13 10 15	**ตามรูปที่ 2**
6 3 8 5 13 6 10 15 8 3 4	mSize: 6 Inorder: 5 6 8 10 13 15 Preorder: 8 5 6 13 10 15	**ไม่ได้ต่อกิ่ง เนื่องจากต้องแทนที่ node ที่มีลูกอยู่แล้ว**

ข้อมูลชุดทดสอบ

25% tree2 จะมีไม่เกิน 1 node และ modify 1 ครั้ง

25% modify 1 ครั้ง

50% ไม่มีเงื่อนไขอื่นใด

คำแนะนำเพิ่มเติม

- หลังจากทำการต่อกิ่งแล้ว ให้ทำลาย tree2 (เปลี่ยน pointer ของ tree2) เพื่อป้องกัน segmentation fault
- อย่าลืมเปลี่ยนค่า mSize เมื่อทำคำสั่ง graft (สามารถใช้ตัวแปรในฟังก์ชั่นได้)
- data type ที่ใช้ตรวจ ไม่ได้มีแค่ int และบางตัวไม่อาจใช้เครื่องหมายมากกว่าน้อยกว่าตรวจสอบได้

Author: daykungza