BST Trim

(1 sec, 512mb)

จงเพิ่มบริการ void CP::map_bst::trim(int depth) ให้กับ CP::map_bst โดยฟังก์ชันนี้จะ “ลบ” ข้อมูลทุกตัวในต้นไม้ที่มีความลึก**มากกว่า depth** โดยกำหนดให้ปมรากมีความลึกเป็น 0 ปมลูกของรากมีความลึกเป็น 1 ปมลูกของลูกของรากมีความลึกเป็น 2 และเป็นเช่นนี้ไปเรื่อย ๆ

ถ้า depth มีค่าเป็น -1 หมายความว่าต้องลบทุกปมในต้นไม้

ข้อบังคับ

- ในโจทย์ข้อนี้ ห้ามทำการสร้างปมของ BST ใหม่ (ซึ่งหมายความว่าจะไม่สามารถใช้วิธีสร้างต้นไม้ใหม่แล้วค่อย ๆ เติมปมเข้าไปได้) แต่อย่างไรก็ตาม 60% ของ testcase จะอนุญาตให้สร้างปมของ BST ใหม่ได้
- โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ Code::Blocks ให้ ซึ่งในไฟล์โปรเจ็คดังกล่าวจะมีไฟล์ map_bst.h, main.cpp และ student.h อยู่ ให้นิสิตเขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ส่งเฉพาะไฟล์ student.h เท่านั้น
  - ในไฟล์ student.h ดังกล่าวจะต้องไม่ทำการอ่านเขียนข้อมูลใด ๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใด ๆ
- หากใช้ VS Code ให้ทำการ compile ที่ไฟล์ main.cpp

**** main ที่ใช้จริงใน grader นั้นจะแตกต่างจาก main ที่ได้รับในไฟล์โปรเจ็กต์เริ่มต้นแต่จะทำการทดสอบในลักษณะเดียวกัน ****

คำแนะนำ

ข้อนี้สามารถทำได้ง่าย ๆ โดยเขียนโปรแกรมแบบ Recursive และเพื่อความสะดวก ข้อนี้ได้เตรียมโครงของฟังก์ชัน my_recur(node* n,int level,int tmp) เพื่อใช้ในการเขียน recursive ไว้ให้ โดยฟังก์ชันนี้มีโครงอยู่ใน student.h แล้ว นิสิตสามารถเขียนและเรียกใช้งานฟังก์ชันนี้ได้เลย

คำอธิบายฟังก์ชัน main

main() จะอ่านข้อมูลมา 2 บรรทัด คือ

- บรรทัดแรกประกอบด้วยจำนวนเต็ม N และ K ซึ่งระบุจำนวนข้อมูลใน map และค่าความลึกที่ต้องการจะตัด
- บรรทัดที่สองประกอบด้วยจำนวนเต็ม N ตัวคือข้อมูลที่จะใส่เข้าไปใน CP::map_bst ตามลำดับ

หลังจากนั้น main จะเรียก trim(K) และ ทำการพิมพ์ข้อมูลทั้งหมดใน map ออกมาด้วยฟังก์ชัน print()

ชุดข้อมูลทดสอบ

- 20% ต้นไม้มีความลึกไม่เกิน 2 และค่า K เป็น 0 หรือ 1 เท่านั้น สามารถสร้างปมใหม่ได้
- 40% สามารถสร้างปมใหม่ได้
- 40% ไม่มีเงื่อนไขอื่น ๆ

(--- มีตัวอย่างอยู่ในหน้าถัดไป ---)

ตัวอย่าง

ข้อมูลนำเข้า	ผลลัพธ์ของ m.print() หลังจากเรียก m.trim(K)
9 2 5 6 7 8 9 1 2 3 4	======== size = 5 ========= ---- 7:2 -- 6:1 5:0 ---- 2:6 -- 1:5
12 5 70 30 100 80 130 150 140 160 147 143 145 144	======== size = 9 ========= -------- 160:7 ------ 150:5 ---------- 147:8 -------- 140:6 ---- 130:4 -- 100:2 ---- 80:3 70:0 -- 30:1