Heap Count Common Ancestors
(1 วินาที, 512mb)

CP::priority_queue นั้นเป็นโครงสร้างข้อมูลแบบ Binary Heap ซึ่งในที่นี้จะ implement โดยใช้ Array-Based Representation จงเพิ่มฟังก์ชัน size_t count_common_ancestors(size_t i, size_t j) const ให้กับ CP::priority_queue ฟังก์ชันนี้จะรับ **index** สองค่า คือ i และ j ซึ่งแทนตำแหน่งของปมใน heap และจะต้องคืนค่า **จำนวน (count)** ของปมทั้งหมดที่เป็น **บรรพบุรุษร่วม (Common Ancestors)** ของปมทั้งสอง

นิยาม: บรรพบุรุษ (Ancestor) ของปม i หมายถึงปมใด ๆ ที่อยู่บนเส้นทางจากปม i กลับไปยังปมราก (index 0)
หมายเหตุ: ปม i ถือว่าเป็นบรรพบุรุษของตัวมันเองด้วย

ตัวอย่าง

หาก priority_queue มีข้อมูล (ตามลำดับ index ใน array) เป็น [100, 50, 40, 20, 10, 30, 35]

- count_common_ancestors(3, 4):
    - บรรพบุรุษของ index 3 (ค่า 20) คือ: {3, 1, 0}
    - บรรพบุรุษของ index 4 (ค่า 10) คือ: {4, 1, 0}
    - บรรพบุรุษร่วมคือ: {1, 0} (ปมที่มีค่า 50 และ 100)
    - ฟังก์ชันต้องคืนค่า **2**
- count_common_ancestors(4, 5):
    - บรรพบุรุษของ index 4 (ค่า 10) คือ: {4, 1, 0}
    - บรรพบุรุษของ index 5 (ค่า 30) คือ: {5, 2, 0}
    - บรรพบุรุษร่วมคือ: {0} (ปมที่มีค่า 100)
    - ฟังก์ชันต้องคืนค่า **1**
- count_common_ancestors(3, 1):
    - บรรพบุรุษของ index 3 (ค่า 20) คือ: {3, 1, 0}
    - บรรพบุรุษของ index 1 (ค่า 50) คือ: {1, 0}
    - บรรพบุรุษร่วมคือ: {1, 0}
    - ฟังก์ชันต้องคืนค่า **2**
- count_common_ancestors(2, 2):
    - บรรพบุรุษของ index 2 (ค่า 40) คือ: {2, 0}
    - บรรพบุรุษร่วมคือ: {2, 0}
    - ฟังก์ชันต้องคืนค่า **2**

ข้อบังคับ

- ฟังก์ชันนี้ถูกประกาศเป็นแบบ const ซึ่งหมายความว่าฟังก์ชันนี้จะต้องไม่ทำการเปลี่ยนแปลงค่าใด ๆ ใน priority_queue
- รับประกันว่าค่า i และ j ที่รับเข้ามา จะอยู่ในช่วง 0 ถึง size() - 1 เสมอ
- โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ Code::Blocks ให้ ซึ่งในไฟล์โปรเจ็คดังกล่าวจะมีไฟล์ priority_queue.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

คำอธิบายฟังก์ชัน main()

main จะอ่านข้อมูลมาหลายบรรทัด ตามรูปแบบนี้:

1. บรรทัดแรกประกอบด้วยจำนวนเต็ม n และ m (โดย m คือจำนวนคำสั่ง)
2. บรรทัดที่สองประกอบด้วยจำนวนเต็ม n ตัว ซึ่งจะถูกนำไป push ใส่ใน priority_queue
3. หลังจากนั้นอีก m บรรทัด แต่ละบรรทัดประกอบด้วยจำนวนเต็ม 2 ตัว คือ i และ j
4. main จะเรียก count_common_ancestors(i, j) แล้วทำการพิมพ์ค่า (จำนวน) ที่ได้คืนมาออกทางหน้าจอ