Segmented Vector

(1 sec, 512mb)

คลาส CP::SegmentedVector เป็นคลาสที่พัฒนาต่อยอดมาจาก CP::vector โดยมีความแตกต่างหลักๆ ดังนี้

1) เดิมใน CP::vector ตัวแปร mData มีประเภทเป็น T* (อาเรย์ชั้นเดียว) แต่ใน CP::SegmentedVector จะเปลี่ยนเป็น T** เพื่อให้เก็บข้อมูลในรูปของอาเรย์สองมิติ (ให้มองเป็นการแบ่งข้อมูลออกเป็น block ย่อยๆ แทนที่จะเก็บไว้ในอาเรย์ตัวเดียว)
2) size_t mSize แทนจำนวนข้อมูลทุกตัวที่เก็บภายใน mData ยกตัวอย่างเช่น หาก mData มีขนาด [4][3] (คือมี 4 block แต่ละ block มีขนาด 3 ช่อง) และทุกช่องมีข้อมูลครบทั้งหมด ค่า mSize จะเท่ากับ 12
3) size_t blockSize แทน**ขนาดของ block** หรือขนาดมิติที่สองของ mData
4) size_t mCap แทน capacity ของ mData โดยจะวัดจาก**ขนาดมิติที่ 1 ของ mData เท่านั้น** (จำนวน block) ไม่ได้หมายถึงจำนวนช่องข้อมูลทั้งหมดในหน่อยความจำเหมือนใน CP::vector
5) นอกนั้นทุกอย่างเราจะนิยามให้เหมือนกัน (แนะนำให้อ่าน segmented_vector.h ที่แนบไปให้เพื่อความเข้าใจที่มากขึ้น)

ตัวอย่าง เปรียบเทียบข้อแตกต่างระหว่าง mData ใน CP::vector และ mData ใน CP::SegmentedVector ที่เก็บข้อมูลชุดเดียวกัน

- mData = {1, 2, 3, 3, -1, 6, 7, X} (CP::vector, mSize = 7, mCap = 8)
- mData = {{1, 2}, {3, 3}, {-1, 6}, {7, X}, {X, X}} (CP::SegmentedVector, mSize = 7 , mCap = 5, blockSize = 2)

การเข้าถึงข้อมูลใน CP::SegmentedVector นั้นให้ถือว่าใช้หลักการเดียวกันกับ CP::vector นั่นคือ หากต้องการเข้าถึงข้อมูลที่มีค่าเป็น -1 ให้เข้าถึงด้วย index ที่ 4

อย่างไรก็ตามไฟล์ segmented_vector.h ที่แนบไว้ให้ ไม่ได้มีฟังก์ชันพื้นฐานอยู่ครบถ้วน จึงอยากให้นิสิตช่วยเขียนการทำงานของฟังก์ชันต่อไปนี้ให้ครบถ้วนในไฟล์ student.h

1) ~Segmented Vector();
   Destructor สำหรับ class CP::SegmentedVector
2) void expand(size_t capacity);
   ฟังก์ชันสำหรับขยายขนาด capacity ของ mData
3) insert(int index, const T& element);
   ฟังก์ชันสำหรับแทรกข้อมูล element เข้าไปยังตำแหน่งที่ index
4) T& operator[](int index) const;
   ฟังก์ชันสำหรับเข้าถึงข้อมูลตำแหน่งที่ index

(มีต่อหน้าถัดไป)

ข้อบังคับ

- **ให้จองพื้นที่ใน memory ด้วย “new” และคืนพื้นที่ ด้วย “delete” เท่านั้น**
- โจทย์ข้อนี้จะมีไฟล์ตั้งต้นมาให้ ซึ่งประกอบด้วยไฟล์ segmented_vector.h, main.cpp และ student.h อยู่ ให้นิสิตเขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ส่งเฉพาะไฟล์ student.h เท่านั้น
- ไฟล์ student.h จะต้องไม่ทำการอ่านเขียนข้อมูลใดๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใดๆ
- หากใช้ VS Code ให้ทำการ compile ที่ไฟล์ main.cpp
  **** main ที่ใช้จริงใน grader นั้นจะต่างจาก main ที่ได้รับในไฟล์โปรเจ็กต์เริ่มต้น แต่จะทำการทดสอบในลักษณะเดียวกัน ****

คำอธิบายฟังก์ชัน main

main() จะอ่านข้อมูลนำเข้าดังนี้

- บรรทัดแรก เป็นจำนวนเต็ม T ระบุจำนวนรอบของการทดสอบ

สำหรับแต่ละรอบของการทดสอบ

- อ่านค่าจำนวนเต็มหนึ่งจำนวนคือ bs
- ทำการสร้าง SegmentedVector โดยกำหนดให้ blockSize เริ่มต้นเท่ากับ bs

หลังจากนั้นจะทำการรับคำสั่งจากข้อมูลนำเข้าแบบต่อเนื่องจนกว่าจะได้รับคำสั่ง “q” เพื่อออกจากการทำงาน โดย operation ที่สามารถใช้ทดสอบได้มีทั้งหมดดังนี้

- “si” หมายถึง แสดงจำนวนข้อมูลปัจจุบันที่มีอยู่ในเวกเตอร์
- “ac X” หมายถึง แสดงค่าข้อมูลในตำแหน่งที่ X ของเวกเตอร์
- “pb X” หมายถึง แทรกข้อมูล X ไปที่ตำแหน่งท้ายสุดของเวกเตอร์
- “in pos X” หมายถึง แทรกข้อมูล X ไปที่ตำแหน่ง pos ของเวกเตอร์
- “po” หมายถึง ลบข้อมูลที่ตำแหน่งสุดท้ายออกจากเวกเตอร์
- “q” หมายถึง ออกจากการทำงาน

เมื่อออกจากการทำงานในแต่ละรอบของการทดสอบ โปรแกรมจะแสดง capacity, จำนวนข้อมูล และข้อมูลทุกตัวในเวกเตอร์ ตามลำดับ

ชุดข้อมูลทดสอบ

- 15% blockSize = 1
- 25% ไม่มีการเรียกใช้ insert (สำหรับชุดข้อมูลทดสอบนี้ ฟังก์ชัน push_back ในระบบตรวจจะสามารถใช้งานได้โดยตรงโดยไม่ต้องผ่านการ insert)
- 15% จำนวน operation รวมไม่เกิน 500
- 20% ไม่ตรวจสอบความถูกต้องของการจองพื้นที่และคืนพื้นที่ใน memory
- 25% ไม่มีเงื่อนไขเพิ่มเติม

(มีตัวอย่างหน้าถัดไป)

ตัวอย่างการทำงานของ main

ข้อมูลนำเข้า	ข้อมูลส่งออก
1 3 si pb 1 pb 2 pb 3 in 1 10 po si q	0 3 Final Vector: mCap = 2 mSize = 3 1 10 2 ----------
2 1 in 0 5 in 0 4 pb 3 si po q 10 pb 10 pb 20 pb 30 in 2 5 si ac 3 q	3 Final Vector: mCap = 4 mSize = 2 4 5 ---------- 4 30 Final Vector: mCap = 1 mSize = 4 10 20 5 30 ----------