โจทย์คู่ตัวเลขเด่นจัดเป็นข้อที่ยาก อัลกอริทึมที่ต้องใช้คือ Divide and Conquer และวิธีการจะมีลักษณะคล้ายการทำ merge sort [ขอขอบคุณอาจารย์ประมุข ขันเงิน มหาวิทยาลัยเกษตรศาสตร์สำหรับคำชี้แนะวิธีแก้โจทย์มา ณ ที่นี้ด้วย]

แนวคิด

เนื่องจากเลขคู่เด่นจะมี a_i > a_j แต่ b_i < b_j ลักษณะของคู่ตัวเลขเด่นจึงเหมือนมีการเรียงจากมากไปน้อยและน้อยไปมากในคราวเดียวกัน ถ้าหากเราจัดเรียงข้อมูลตามค่า a ก่อนและเราสามารถหาได้ว่าค่า b คู่ไหนบ้างที่ไม่ได้เรียงลำดับอย่างถูกต้อง กล่าวคือมีการสลับลำดับ (inversion) เราก็จะรู้ได้ว่าคู่ตัวเลขนั้นคือคู่ตัวเลขเด่น ปัญหาที่ต้องพิจารณาจึงมีอยู่ว่า ทำอย่างไรเราจึงจะหาการสลับลำดับของค่า b ได้อย่างรวดเร็ว พร้อม ๆ กับหาผลบวกของ a ในคู่ตัวเลขเด่นได้อย่างรวดเร็วด้วย

สำหรับการหาการสลับลำดับค่า b อย่างรวดเร็วนั้นเราทำได้ด้วยการอาศัยพฤติกรรมของ merge sort เพราะใน merge sort นั้นจะมีการเปรียบเทียบ ‘ค่าที่ดีที่สุด’ จากชุดตัวเลขสองชุด และถ้าค่าที่ดีที่สุดจากชุดที่สองดีกว่าชุดที่หนึ่งก็แสดงว่าค่าที่ดีที่สุดจากชุดที่สองนั้นดีกว่าค่าทุกค่าในชุดที่หนึ่งด้วยทำให้เราสามารถสรุปการสลับค่า b ได้อย่างรวดเร็ว ดังแสดงให้เห็นได้จากตัวอย่างข้างล่างนี้

1. สมมติให้ข้อมูลเข้ามีเลขแปดคู่ดังนี้
   (2, 1) (7, 6) (9, 3) (18, 4) (3, 2) (5, 5) (1, 7) (4, 8)

2. เรียงคู่จากน้อยไปมากตามค่า a ด้วย qsort
   (1, 7) (2, 1) (3, 2) (4, 8) (5, 5) (7, 6) (9, 3) (18, 4)

3. นำค่า b มาจัดเรียงด้วยวิธีแบบ merge sort แต่เราต้องคอยสังเกตเวลาที่พบ inversion ด้วย ในที่นี้การจัดเรียงที่เกิด inversion จะถูกบันทึกด้วยการใช้ ‘ กับตัวเลขที่จัดเรียงแล้ว

Iteration	**ค่าในอาเรย์**	**คำอธิบายเพิ่มเติม**
0	(7), (1), (2), (8), (5), (6), (3), (4)	ค่า b เริ่มต้นเป็นไปตามการจัดเรียงที่ได้จากขั้นตอนที่สอง วงเล็บเป็นการบอกชุดของตัวเลขที่ต้องพิจารณาใน merge sort
1	(1’, 7), (2, 8), (5, 6), (3, 4)	การเปรียบเทียบชุดตัวเลขที่ติดกันเป็นคู่ ๆ ไปแบบ merge sort ทำให้เกิดการจับกลุ่มค่าเป็นชุดตัวเลขที่ใหญ่ขึ้น ในขั้นตอนนี้พบการสลับค่าเพียงตำแหน่งเดียวเท่านั้นคือค่า

		1 กับ 7
2	(1, 2', 7, 8), (3', 4', 5, 6)	พบ inversion ระหว่าง 2 กับ 7, 3 กับ 5 และ 6, และ 4 กับ 5 และ 6 ขอให้สังเกตด้วยว่าแท้จริงแล้วตอนที่พบว่า 3 สลับกับ 5 เราไม่จำเป็นต้องเปรียบเทียบ 3 กับ 6 เพราะเราสรุปได้ทันทีว่าเลขทุกตัวในชุดเดียวกับ 5 ที่ตามหลังมันมาต้องมีการสลับลำดับด้วยแน่นอน และสิ่งเดียวกันนี้ก็เกิดขึ้นกับการสลับลำดับระหว่าง 4 กับ 5 ด้วย
3	(1, 2, 3', 4', 5', 6', 7, 8)	พบว่า 3, 4, 5, และ 6 สลับลำดับกับ 7 และ 8 (เช่นเดิม แท้จริงแล้วเราไม่ต้องเปรียบเทียบกับเลข 8 เราก็รู้ว่ามีการสลับลำดับ)

ในการหาและสรุปการสลับลำดับนั้นใช้เวลาเท่ากับ merge sort คือแต่ละ iteration ใช้ O(n) และมีจำนวน iteration ทั้งหมด O(log n) ดังนั้นขั้นตอนการหาการสลับลำดับจะใช้เวลา O(n log n) ปัญหาที่เหลือจึงมีอยู่ว่าเราจะหาผลบวกของค่า a อย่างมีประสิทธิภาพได้อย่างไร

เรื่องการหาผลบวกอย่างรวดเร็วทำได้โดยการหาผลบวกของค่า a ในชุดข้อมูลทางซ้ายเก็บไว้ก่อน โดยให้บวกเป็นค่าสะสมจากขวาไปซ้าย เช่น ในชุดข้อมูล (1, 2, 7, 8) ที่แสดงไว้ใน Iteration 2 จะมีผลบวกค่า a สะสม คือ (10, 8, 5, 4) ค่านี้นำมาใช้ใน Iteration 3 ได้ กล่าวคือตอนที่เราพบว่า 3 สลับลำดับกับ 7 เราก็จะได้ผลบวกของค่า a ที่เกี่ยวข้องกับการสลับลำดับครั้งนี้คือ 5 + (9 x 2) และตอนที่พบว่า 4 สลับลำดับกับ 7 เราก็จะได้ผลบวกค่า a ที่เกี่ยวข้องกับการสลับลำดับครั้งนี้คือ 5 + (18 x 2) สังเกตด้วยว่าเราสามารถหยิบผลบวกเดิมมาใช้ได้เลยทันที ไม่ต้องบวกซ้ำ ส่วนตัวคูณของค่า a ในทางขวามือก็คำนวณได้ทันทีใน O(1)

จะเห็นได้ว่าภาระการคำนวณที่ต้องทำในการหาผลบวกค่า a ในแต่ละ Iteration นั้นแจกแจงได้ดังนี้

1. O(n) จากการหาผลบวกสะสมเก็บไว้
2. O(n) จากการหยิบค่าผลบวกสะสมไปใช้ เพราะการหยิบใช้แต่ละครั้งใช้เวลาเป็น O(1) และทำเป็นอย่างมาก O(n) ครั้งในแต่ละ iteration

จากวิธีข้างต้นสรุปได้ว่าภาระงานในการหาการสลับลำดับและหาผลรวมค่า a รวมในทุก iteration คือ O(n log n)

เรียบเรียงโดย ภิญโญ แท้ประสาทสิทธิ์, 7 พฤษภาคม 2554