โจทย์ระเบิดมหาประลัยต้องการวิธีแก้ปัญหาที่มีประสิทธิภาพ แต่เป็นโจทย์ที่ผู้เข้าสอบสามารถเขียนโปรแกรมให้เสร็จได้อย่างรวดเร็วด้วยการใช้ stack [ขอขอบคุณอาจารย์ประมุข ขันเงิน มหาวิทยาลัยเกษตรศาสตร์ อดีตนักเรียนทุนเล่าเรียนหลวงและเหรียญเงินโอลิมปิกนานาชาติสำหรับคำชี้แนะวิธีแก้ปัญหาที่มีประสิทธิภาพมา ณ ที่นี้ด้วย]

แนวคิด

ขออธิบายแนวคิดจากตัวอย่างที่สองในโจทย์ซึ่งมีระเบิดอยู่เจ็ดพิกัดจุดดังนี้

(1, 2) (2, 4) (4, 1) (7, 3) (5, 5) (6, 6) (3, 7)

เราแก้ปัญหาได้ดังนี้คือ

1. เรียงข้อมูลตามพิกัด *x* จะได้พิกัดดังนี้
   (1, 2) (2, 4) (3, 7) (4, 1) (5, 5) (6, 6) (7, 3)
   เวลาในการคำนวณ O(n log n)

2. เนื่องจากระเบิดจะไม่อยู่ในจุดตำแหน่งมหาประลัยทันทีที่มีระเบิดที่มีทั้งค่า *x* และ *y* มากกว่ามันเพียงอันเดียว ดังนั้นถ้าเราไล่ค่าในข้อหนึ่งจากซ้ายไปขวาเราจะได้ว่าค่าที่จุดทางขวามีค่าพิกัด *x* มากกว่าจุดทางซ้ายแน่ ๆ เราเพียงคอยตรวจค่า *y* ก็พอ

3. เราจะเริ่มพิจารณาจากจุดที่หนึ่งไปจุดที่ n ทีละจุด และเราจะสร้าง stack ที่เก็บจุดมหาประลัยจากการพิจารณาจากจุดที่ 1 ถึงจุดที่ k ไว้ ดังนั้นในตอนแรกเมื่อค่า k = 1 และเราจะมีจุด (1, 2) อยู่ใน stack และเป็นจุดมหาประลัยสำหรับค่า k นี้

4. เมื่อเพิ่มค่า k ขึ้นก็ให้เอาจุดที่ k มาเปรียบเทียบกับค่าบนสุดใน stack
   ถ้าจุดที่ k มีค่า *y* มากกว่า แสดงว่าจุดบนสุดของ stack ไม่ใช่ตำแหน่งมหาประลัยเพราะว่าเรารู้มาก่อนแล้วว่าค่า *x* ของจุดทางขวามีค่ามากกว่าจุดทางซ้าย เราสามารถตัดจุดที่อยู่บนสุดของ stack ทิ้งไปได้ทันทีแล้วเปรียบเทียบทำนองนี้ไปเรื่อย ๆ กับตัวบนสุดของ stack แต่ถ้าจุดที่k มีค่า *y* น้อยกว่าให้เก็บมันไว้ใน stack เพราะว่าไม่มีการบดบังกับจุดทุกจุดใน stack และจุดที่ k ต่างก็เป็นจุดมหาประลัย ณ ค่า k ปัจจุบันอยู่
   สังเกตด้วยว่าค่า *y* ของจุดต่าง ๆ ใน stack จะถูกเรียงจากมากไปน้อยเสมอ เพราะถ้ามีการเพิ่มขึ้นของค่า *y* แสดงว่ามีการบดบังกันเกิดขึ้น ซึ่งจากวิธีการที่อธิบายไปเรื่องนี้จะเป็นไปไม่ได้เพราะของที่อยู่ใน stack

ที่มีค่า *y* น้อยจะต้องถูกเอาออกมาทิ้งก่อนที่จะใส่ของที่มีค่า *y* มากเข้าไป

หากพิจารณาตามตัวอย่างที่สองในปัญหา เราก็จะได้สิ่งที่อยู่ใน stack ณ ค่า k ต่าง ๆ ดังนี้

k	Stack
1	(1, 2)
2	(2, 4)
3	(3, 7)
4	(3, 7) (4, 1)
5	(3, 7) (5, 5)
6	(3, 7) (6, 6)
7	(3, 7) (6, 6) (7, 3)

เนื่องจากจุดที่ถูกกำจัดออกจาก stack ไปแล้วจะไม่ถูกนำมาพิจารณาอีก เราจะได้ว่าขั้นตอนการเปรียบเทียบนี้ใช้เวลา O(n) และสรุปเวลาในการทำงานทั้งหมดได้เป็น O(n log n)

เรียบเรียงโดย ภิญโญ แท้ประสาทสิทธิ์, 7 พฤษภาคม 2554