1. เลขสวย (Beautiful Number)

เฉลยแบบที่ 1 : ได้คะแนนเต็ม

ข้อนี้เราสามารถเขียนวนลูป 3 ชั้นได้ โดยรอบชั้นแรกเป็นการวน A ถึง B ส่วนชั้นที่ 2 เป็นการดึงค่าออกมาแต่ละหลัก ส่วนชั้นที่ 3 ก็จะเป็นการเช็คว่าค่าที่ดึงออกมานั้นอยู่ใน Chosen Number หรือไม่ เราสามารถ optimize โค้ดได้ด้วยการใส่ break คือเมื่อพบว่าตัวเลขตัวนี้เป็น Chosen Number แล้วก็ไม่ต้องทำการคำนวณต่อ ก็นับเลขต่อไปได้เลย

เฉลยแบบที่ 2 : ได้คะแนนเต็ม

Bucket เป็นอีกวิธีหนึ่งที่เร็วกว่าวิธีแรกเล็กน้อยคือ ขั้นแรกเรานำ Chosen Number มาเก็บลงใน Bucket แล้ว วน loop จาก A ไป B ดึงค่ามากแต่ละหลักใส่ Bucket ลงอีกตัว แล้วนำ Bucket ทั้งสองตัวนี้มาเปรียบเทียบกัน ซึ่งวิธีนี้ใช้ loop เพียงแค่ 2 ชั้น

เฉลยแบบที่ 3 : ได้คะแนนเต็ม

Bitwise Operator เป็นวิธีที่เร็วที่สุดใน 3 วิธี โดยเราจะกำหนดให้ตอนแรกกำหนดตัวแปร Chosen เป็น 0000000000 ในฐาน 2 ถ้าสมมุติเลข Chosen Number เป็น 02589 ก็ทำการมาร์คเลข 1100100101 ลงในตัวแปรชื่อ Chosen หลังจากนั้นเราก็จะวน A ถึง B แล้วเอาเลขแต่ละตัวมาทำกระบวนการเดียวกัน นำเลขที่มาร์คเรียบร้อยแล้วมา & (And) กันในระดับบิตกับตัวเลข Chosen ถ้าหาก & กัน ถ้า & กันแล้วไม่เป็น 0 แสดงว่าตัวเลขนั้นเป็น Beautiful Number

ดูการ implement เพิ่มเติมได้ใน source code

2. เกมลบคำ (Word Remove)

คีย์หลัก

ข้อนี้เราจะต้องพลิกแพลงโจทย์สักเล็กน้อย โดยโจทย์ยื่นมาให้ว่าเราต้องเลือกกติกาใดกติกาหนึ่ง แต่การคำนวณจริงนั้นเราต้องคิดทั้งสองกติกาพร้อมกัน แล้วดูว่ากติกาไหนได้ทางเลือกที่ดีกว่าก็ตอบกติกานั้น และคีย์หลักอีกข้อคือ โจทย์ยื่นข้อเสนอมาให้ว่า คุณต้องหาคำที่ตัดออกน้อยสุดกี่ครั้ง แต่ถ้าเราลองมองมุมกลับเป็น ในตารางมีคำที่ซ้ำกันมากที่สุดกี่คำ แล้วเวลาตอบก็เอา N ลบด้วยจำนวนคำที่ซ้ำกันมากที่สุด ก็จะทำให้ปัญหาข้อนี้ง่ายขึ้นเท่าตัว

เฉลยแบบที่ 1 : ได้ 90 คะแนน

เราใช้อัลกอรึทึ่ม O(N^2) ในการหาคำซ้ำว่ามีคำที่ซ้ำกันทั้งหมดกี่คำ โดยคำแรกก็เทียบคำที่อยู่ด้านหลังทั้งหมดแล้วนับ คำที่สองเทียบคำที่อยู่ด้านหลังทั้งหมดแล้ว ไปเรื่อยๆ จะพบว่าวิธีนี้ยังช้าอยู่ เพราะใช้ Big O โดยรวมเกือบ N^3

เฉลยแบบที่ 2 : คะแนนเต็ม

หลักการเหมือนแบบที่ 1 แต่นำมา Optimize ให้เร็วขึ้นด้วยการ Sort คำตาม Dictionary จะทำให้คำที่เหมือนกันเกาะกลุ่มเดียวกัน เมื่อเกาะกลุ่มกันแล้วการหาจำนวนคำซ้ำในเวลาเพียงแค่ O(N) เท่านั้นบวกกับเวลา Sort อีก N log N ทำให้ข้อนี้ใช้เวลาในการแก้ปัญหา เลยใช้เวลารวม O(N^2 log N)

หมายเหตุ

ข้อนี้ให้ระวังเรื่องบั๊กให้ดีๆ ไม่ว่าจะกรณีที่ตารางทั้งตารางนั้นคำซ้ำทุกตัว ต้องตอบ 0 หรือกรณีที่สามารถใช้กติกาของทั้งคู่เล่นได้พร้อมกันก็ตอบ both และอีกกรณีหนึ่งที่หลายคนอาจจะพลาดได้ง่ายๆคือ กรณีที่ทั้งตารางนั้นไม่มีคำซ้ำกันเลย จำนวนครั้งน้อยที่สุดในการตัดคำเป็น N-1 เพราะว่าพอตัดเหลือ 1 ตัวแล้ว ไอ้ที่เหลือ 1 ตัวนั้นแหละมันซ้ำกัน เพราะฉะนั้นระวังตรงนี้ด้วย ข้อนี้จึงเป็นข้อเฉือนคะแนนได้เป็นอย่างดี

3. งอ (Bend)

เฉลยแบบที่ 1 : ได้ 30-35 คะแนน

Recursive เพียวๆวนกรณีที่สามารถวางท่อน้ำได้ทุกกรณีแล้วคำนวณหาจำนวนครั้งในการงอ ซึ่งวิธีนี้จะช้ามากๆแล้วได้คะแนนประมาณ 30-35 คะแนน

เฉลยแบบที่ 2 : 80-85 คะแนน

เราหาว่าในช่อง i,j นั้น มีท่อแบบไหนบ้างที่เชื่อมมายังท่อ i,j โดยเราแบ่งรูปแบบการวางท่อเป็น 2 กรณีคือการวางแบบแนวตั้งและแนวนอน ซึ่งจะเห็นได้ว่าน้ำมันไหลได้ 3 ทิศทาง เพราะฉะนั้นการที่น้ำจะไหลมาช่อง i,j ได้นั้น เราต้องดูว่าท่อด้านบน ท่อทางด้านซ้ายและด้านขวาท่อไหนบ้างที่มีโอกาสเชื่อมต่อมายังช่อง i,j ซึ่งวิธีนี้จะเสียเวลาเล็กน้อยตอนที่ต้องหาทุกกรณีที่ต้อง loop หาท่อที่มาเชื่อมต่อยังท่อตัวอย่าง ซึ่ง Big O นั้นประมาณเกือบ O(K^2) เมื่อ K คือ N^M

เฉลยแบบที่ 3 : คะแนนเต็ม

Dynamic Programming โดยประยุกต์จากเฉลยแบบที่ 2 นั่นคือ เราไม่จำเป็นต้องคำนวณท่อทุกท่อที่มาเชื่อมต่อกับช่อง i,j แต่เราคำนวณเพียงแค่ 3 ช่องพอ นั่นคือช่อง i-1,j (ช่องบน) i,j+1 (ช่องขวา) i,j-1 (ช่องซ้าย) แล้วแยกเป็น 2 กรณีคือ กรณีที่ช่อง i,j วางท่อน้ำแบบแนวนอน กับแบบแนวตั้ง ซึ่งวิธีนี้ใช้เวลาเพียงแค่ O(N^M) เท่านั้น

4. วงเล็บ (Parentheses)

เฉลยแบบที่ 1 : ได้ 40 คะแนน

วิธีนี้จะเป็นจำลองเครื่องวงเล็บเปิด/ปิด ทุกชนิด แล้วมาไล่เช็คแต่ละตัวว่าตัวไหนบ้างที่ถูกต้องตามเงื่อนไข ซึ่งวิธีนี้ช้ามาก

เฉลยแบบที่ 2 : คะแนนเต็ม

เราจะตัดเคสที่ไม่จำเป็นออกไปโดยการเอาวงเล็บเปิดมาเก็บไว้ใน stack กรณีที่เราสามารถเติมวงเล็บเปิดต่อไปได้โดยตรงตามเงื่อนไขก็เติม หากเราไม่สามารถเติมได้ก็ไม่เติมต่อ เช่นเดียวกันกับวงเล็บเปิด แต่วงเล็บเปิดจะไปอิงกับวงเล็บเปิดที่เก็บไว้ใน stack เพื่อดูว่า เราควรปิดตัวไหนและตรงตามกฎ ซึ่งวิธีนี้จะสามารถบั่นทอนเคสอื่นๆที่ไม่เกี่ยวข้องได้เยอะมาก เพราะทุกครั้งที่มีการ Recursive หากการเติมวงเล็บทำให้ผิดกฎก็ไม่ทำต่อไปเลย วิธีนี้จึงได้คะแนนเต็ม

เฉลยแบบที่ 3 : คะแนนเต็ม

หลักการเดียวกับเฉลยแบบที่ 2 แต่แทนที่จะเก็บวงเล็บเปิดไว้ใน stack เราสร้างตัวแปรเพิ่มขึ้นมา 3 ตัวเช่น pa,pb,pc เพื่อเก็บจำนวนวงเล็บเปิดที่เราเติมเข้าไป วิธีนี้ประสิทธิภาพเวลาอาจจะไม่แตกต่างกับวิธีที่ 2 เท่าไรนัก แต่ช่วยจัดระเบียบโค้ดเราและทำให้เราประหยัดเวลาในการเขียนโค้ดได้มากเลยทีเดียว