Great_boleros_of_fire

Subtask 1 :

ในส่วนของปัญหาย่อยนี้สามารถแก้ด้วย brute force algorithm โดยจะทำการเรียงสับเปลี่ยนของลำดับในการลบท่าเต้นออก โดยในทุกการเรียงสับเปลี่ยนจะทำการตรวจสอบว่าในระหว่างการลบท่าเต้นออก เป็นการลบท่าเต้นแบบที่ 1 แบบที่ 2 หรือ แบบที่ 3 นั่นคือต้องตรวจสอบด้วยว่าในขณะการดำเนินการ ท่าเต้นไหนอยู่หน้าสุด และ ท่าเต้นไหนอยู่หลังสุด ซึ่งสามารถทำได้ใน complexity $O(n^2 n!)$ โดยหากมีการจัดการกับอัลกอริทึมที่ดี จะสามารถทำได้ภายในเวลาที่กำหนด

Subtask 2 :

ในส่วนของปัญหาย่อยนี้ สามารถสังเกตได้ว่าควรลบท่าเต้นแบบที่ 1 และ 2 ออกก่อน แล้วค่อยลบท่าเต้นแบบที่ 3 เพราะจะทำให้ไม่เปลืองจำนวนครั้งในการลบท่าเต้นในแบบที่ 3 ซึ่งการลบท่าเต้นแบบที่ 1 และ 2 จะทำให้เหลือท่าเต้นตั้งแต่ท่าที่ l ถึง r เมื่อ $1 \leq l \leq r \leq n$ ซึ่งหากจำนวนท่าเต้นทั้งหมดของ “โบเลโรแห่งไฟ” มีค่าเป็น a จะได้ว่าต้องลบท่าเต้นแบบที่ 3 ทั้งหมด $r - l + 1 - a$ ท่า ซึ่งการหา l และ r สามารถทำได้ใน complexity $O(n^2)$ และการตรวจสอบว่าท่าเต้นชุดนี้มีโอกาสที่จะสร้าง “โบเลโรแห่งไฟ” ได้หรือไม่ สามารถทำได้ใน $O(n)$ ดังนั้นโดยรวมแล้วจึงใช้ complexity $O(n^3)$

Subtask 3 :

ในส่วนของปัญหาย่อยนี้ ใช้ไอเดียหลักจากปัญหาย่อยที่แล้ว โดยจะกำหนดให้ $d(l, r)$ แทนชุดท่าเต้นที่ประกอบด้วยตั้งแต่ท่าเต้นที่ l ถึงท่าเต้นที่ r เมื่อ $1 \leq l \leq r \leq n$ โดยถ้าหากพิจารณาตั้งแต่ $d(1, x), d(2, x+1), d(3, x+2), \ldots, d(n-x+1, n)$ เมื่อ $1 \leq x \leq n$ จะได้ว่าสามารถใช้ sliding window algorithm ในการตรวจสอบว่ามีโอกาสที่จะสร้าง “โบเลโรแห่งไฟ” ของแต่ละ $d(i, i+x-1)$ ได้หรือไม่ เมื่อ$1 \leq i \leq n-x+1$ ซึ่งขั้นตอนนี้สามารถทำได้ภายใน $O(n)$ ในขณะเดียวกันการหาค่า x ทุกค่าก็สามารถทำได้ใน $O(n)$ เช่นกัน ดังนั้นโดยรวมแล้วจึงใช้ complexity $O(n^2)$

Subtask 4 :

ในส่วนของปัญหาย่อยนี้ทำเหมือนกับปัญหาย่อยที่แล้ว แต่สามารถสังเกตได้ว่าถ้าหาก $d(l,r)$ มีโอกาสที่จะสร้าง “โบเลโรแห่งไฟ” ได้ จะได้ว่า $d(l,r+1)$ หรือ $d(l-1,r)$ ก็มีโอกาสที่จะสร้าง “โบเลโรแห่งไฟ” ได้เช่นกัน ดังนั้น ในส่วนของขั้นตอนการหาค่า x ของปัญหาย่อยที่แล้ว สามารถใช้ binary search algorithm เพื่อหาค่า x ที่ดีที่สุดหรือว่าน้อยที่สุดได้ ทำให้ในขั้นตอนนี้ทำได้ใน $O(lgn)$ ดังนั้นโดยรวมแล้วจึงใช้ complexity $O(nlgn)$

Counting_number

Subtask 1:

ในปัญหาย่อยนี้ให้ทำการสร้างสตริงลำดับตัวเลขความยาว 10 ทั้งหมด 3 กรณี คือเมื่อ a = 2 b = 3 c = 4 , a = 2 b = 3 c = 5 และ a = 3 b = 4 c = 5 ซึ่งในแต่ละคำถามจะตอบเป็นเลขโดดในลำดับที่ที่ถามของสตริง ซึ่งสามารถทำได้ใน Time Complexity $O(1)$ ต่อหนึ่งคำถาม ซึ่งจะใช้ Time Complexity โดยรวมคือ $O(N)$

Subtask 2:

ในปัญหาย่อยนี้ให้ทำการสร้างสตริงลำดับตัวเลขขึ้นมาใหม่ในแต่ละคำถาม โดยเราจะทำการไล่ตัวเลขตั้งแต่ 1 ถึง $\max(\mathrm{a},\mathrm{b},\mathrm{c}) \bullet x$ และทำการใส่ตัวเลขทีละหลักลงไปในสตริง ซึ่งจะตอบเป็นเลขโดดในลำดับที่ที่ถามของสตริง ซึ่งสามารถทำได้ใน Time Complexity $O(\max(\mathrm{a},\mathrm{b},\mathrm{c})\, x)$ ต่อหนึ่งคำถาม ซึ่งจะใช้ Time Complexity โดยรวมคือ $O(Nx \max(\mathrm{a},\mathrm{b},\mathrm{c}))$

Subtask 3:

ในปัญหาย่อยนี้จะทำคล้ายกับ Subtask 2 แต่เราจะทำการเลือกไล่เฉพาะตัวเลขที่หารด้วย $a\ b$ หรือ c ลงตัวเท่านั้น โดยเราจะทำการสร้างตัวแปร 3 ตัวคือ $i\ j$ และ k ซึ่งตัวแปรเหล่านี้จะมีหน้าที่เก็บค่าทวีคูณของ $a\ b$ และ c ตามลำดับ

เริ่มต้นเราจะกำหนดค่า $i = a, j = b$ และ $k = c$ และทำดังนี้

1) ให้ทำการหาว่าจากค่าที่เก็บของตัวแปร $i\ j\ k$ ค่าที่น้อยที่สุดคืออะไร

2) ให้ทำการใส่ตัวเลขของค่าที่น้อยที่สุดทีละหลักลงไปในสตริง

3) นำตัวแปรทุกตัวที่เก็บค่าที่น้อยที่สุดไป บวกด้วย a ถ้าหากเป็นตัวแปร i, บวกด้วย b ถ้าหากเป็นตัวแปร j และบวกด้วย c ถ้าหากเป็นตัวแปร k

4) ถ้าหากสตริงยังมีความยาวน้อยกว่า x อยู่ ให้กลับไปทำข้อ 1 ซ้ำใหม่

ซึ่งจะตอบเป็นเลขโดดในลำดับที่ที่ถามของสตริง โดยที่เราสามารถบอกได้ว่าทำซ้ำไม่เกิน $O(x)$ ครั้งแน่ๆ ทำให้ได้ว่า Time Complexity ต่อหนึ่งคำถามคือ $O(x)$ ซึ่งจะใช้ Time Complexity โดยรวมคือ $O(Nx)$

Subtask 4:

ในปัญหาย่อยนี้เราจะทำการนิยาม function $count(v)$ ขึ้นมา โดย function นี้จะมีหน้าที่รับข้อมูลตัวเลข v เข้ามาและส่งข้อมูลกลับว่ามีตัวเลขตั้งแต่ 1 ถึง v อยู่ทั้งหมดกี่ตัวที่หารด้วย a ลงตัว ซึ่งค่าที่จะส่งกลับของ function นี้ก็คือ $\left\lfloor \frac{v}{a} \right\rfloor$ นั่นเอง ส่วนวิธีการทำของ Subtask นี้จะมีสามขั้นตอนดังนี้

ขั้นตอนแรก : ให้หาว่าตัวเลขตัวก่อนสุดท้ายที่ถูกไส่เข้ามาในสตริงความยาว x เป็นตัวเลขกี่หลัก เช่น ถ้าว่าหาก $a = 3$ และ $x = 8$ สตริงก็จะเป็น 36912151 ซึ่ง 15 ก็คือตัวเลขตัวก่อนสุดท้ายที่ถูกไส่เข้ามาในสตริงและเป็นตัวเลข 2 หลัก ซึ่งจำนวนหลักของตัวเลขตัวก่อนสุดท้ายจะรับประกันว่ามีค่าไม่เกิน $\log_{10}(ax) = \log_{10}(10^6 \bullet 10^6) = \log_{10}(10^{12}) = 12$ หลักอย่างแน่นอน

ผลรวมหลักของเลขทั้งหมดที่มี i หลักจะมีค่าเท่ากับ $i \bullet [count(10^i - 1) - count(10^{i-1} - 1)]$ ซึ่งเราจะหาว่าตัวเลขตัวก่อนสุดท้ายเป็นตัวเลขกี่หลักได้จากการไล่หาผลรวมต่อเนื่องตั้งแต่หลักที่ 1 จนถึงหลักที่ i โดยที่ i คือจำนวนหลักที่น้อยที่สุดที่ทำให้ผลรวมต่อเนื่องตั้งแต่หลักที่ 1 จนถึงหลักที่ i มีค่ามากกว่าหรือเท่ากับ x

ขั้นตอนสอง : ให้หาว่าตัวเลขตัวก่อนสุดท้ายที่ถูกไส่เข้ามาในสตริงความยาว x เป็นตัวเลขอะไร โดยเรารู้ว่าตัวเลขตัวก่อนสุดท้ายนี้จะอยู่ในช่วง 10^{i-1} ถึง $10^i - 1$ แน่นอน

สมมติให้ตัวเลขตัวก่อนสุดท้ายที่ถูกไส่เข้ามาในสตริงเท่ากับ j เราจะสามารถที่จะหาค่า j ได้จากกระบวนการ binary search algorithm

ขั้นตอนสาม : ให้หาว่าเลขโดดในลำดับที่ x ของสตริงคือเลขโดดอะไร โดยในตอนนี้เรารู้แล้วว่าตัวเลขตัวก่อนสุดท้ายคือ j ดังนั้นถัดจากตัวเลข j ยังขาดเลขโดดอยู่อีก $(x -$ ผลรวมหลักทั้งหมดตั้งแต่ 1 ถึง $j)$ ตัว

สมมติให้ค่า $(x -$ ผลรวมหลักทั้งหมดตั้งแต่ 1 ถึง $j)$ เท่ากับ k เราจะสามารถที่จะหาได้ว่าเลขโดดในลำดับที่ x ของสตริงลำดับตัวเลขก็คือ ลำดับที่ k ของตัวเลข $j + 1$

ซึ่งคำตอบจะเป็นเลขโดดในลำดับที่ k ของตัวเลข $j + 1$ ซึ่งสามารถทำได้ใน Time Complexity $O(\log_{10} ax + \log_2 ax)$ ต่อหนึ่งคำถาม ซึ่งจะใช้ Time Complexity โดยรวมคือ $O(\mathrm{N}\ (\log_{10} ax + \log_2 ax))$

Subtask 5:

ในปัญหาย่อยนี้วิธีการทำทั้งสามขั้นตอนจะเหมือนกับ Subtask 4 แต่ว่าในส่วนของ function $count(v)$ จะต่างกัน โดย function $count(v)$ ของ Subtask 5 นี้จะมีหน้าที่รับข้อมูลตัวเลข v เข้ามาและส่งข้อมูลกลับว่ามีตัวเลขตั้งแต่ 1 ถึง v อยู่ทั้งหมดกี่ตัวที่หารด้วย a b หรือ c ลงตัว ซึ่งเราจะใช้ความรู้ในเรื่องของ หลักการเพิ่มเข้าและตัดออก ที่ว่า $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$ มาใช้ในการนับ ทำให้ค่าที่จะส่งกลับของ function นี้ก็คือ $\left\lfloor \frac{v}{a} \right\rfloor + \left\lfloor \frac{v}{b} \right\rfloor + \left\lfloor \frac{v}{c} \right\rfloor - \left\lfloor \frac{v}{lcm(a,b)} \right\rfloor - \left\lfloor \frac{v}{lcm(a,c)} \right\rfloor - \left\lfloor \frac{v}{lcm(b,c)} \right\rfloor + \left\lfloor \frac{v}{lcm(a,b,c)} \right\rfloor$ นั่นเอง โดยที่ function $lcm()$ หรือค.ร.น.ของ i และ j ก็สามารถหาได้จากการเอา $\frac{i \bullet j}{\gcd(i,j)}$ โดยที่ function $gcd()$ หรือห.ร.ม. ก็สามารถหาได้จากขั้นตอนวิธีแบบยูคลิด

และหลังจากนั้นก็ทำตามทั้งสามขั้นตอนตามลำดับเหมือนกับ Subtask 4 ซึ่งจะสามารถทำได้ใน Time Complexity $O(\log_{10} \max(a,b,c)x + \log_2 \max(a,b,c)x)$ ต่อหนึ่งคำถาม ซึ่งจะใช้ Time Complexity โดยรวมคือ $O(\mathrm{N}\ (\log_{10} \max(a,b,c)x + \log_2 \max(a,b,c)x)$

Cheat 1

Subtask 1 :

ในส่วนของปัญหาย่อยนี้ เราจะเปลี่ยนคำถามในแต่ละคำถามเป็น หากเราเลือกใช้เส้นทางในช่วง l ถึง $\mathrm{r}-1$ เราจะหยุดที่ห้องน้ำที่ a โดยที่ให้ $\max(\mathrm{dis}(\mathrm{l},\mathrm{a}),\mathrm{dis}(\mathrm{a},\mathrm{r}))$ มีค่าน้อยที่สุดเท่าที่จะทำได้ ในกรณีนี้เราจะลูปค่า a ตั้งแต่ห้องน้ำที่ 1 ถึง n และจะคำนวณ $\mathrm{dis}(\mathrm{a},\mathrm{b})$ ได้โดยการลูปบวกระยะทางเส้นทางตั้งแต่เส้นทางที่ a ถึง $\mathrm{b}-1$ โดยจะสามารถทำฟังก์ชัน dis ได้ใน Time Complexity $O(n)$ ต่อการเรียก 1 ครั้ง และจะเรียกทั้งหมด $2n$ ครั้งต่อการลูปค่า a ทั้งหมด ซึ่งสามารถทำได้ใน Time Complexity $O(2n^2)$ ต่อ 1 คำถาม ทำให้ Time Complexity โดยรวมคือ $O(2n^2\mathrm{q})$

Subtask 2 :

ในส่วนของปัญหาย่อยนี้ เราจะเปลี่ยนวิธีการคำนวณค่าฟังก์ชัน dis โดยเราจะทำการสร้างเป็นอาเรย์ d ซึ่งเราจะนิยาม $\mathrm{d}[\mathrm{i}]$ คือผลรวมของความยาวของเส้นทางตั้งแต่ 1 ถึง i และจะคำนวณ $\mathrm{dis}(\mathrm{a},\mathrm{b})$ ได้โดยการนำ $\mathrm{d}[\mathrm{b}-1]-\mathrm{d}[\mathrm{a}-1]$ ซึ่งสามารถคำนวณได้ใน Time Complexity $O(1)$ ต่อการเรียก 1 ครั้ง และจะเรียกทั้งหมด $2n$ ครั้งต่อการลูปค่า a ทั้งหมด ซึ่งสามารถทำได้ใน Time Complexity $O(2n)$ ต่อ 1 คำถาม ทำให้ Time Complexity โดยรวมคือ $O(2n\mathrm{q})$

Subtask 3 :

ในส่วนของปัญหาย่อยนี้ เราจะสังเกตว่าจะมีจุดหนึ่งที่ $\mathrm{dis}(\mathrm{l},\mathrm{a})\leq\mathrm{dis}(\mathrm{a},\mathrm{r})$ และ $\mathrm{dis}(\mathrm{l},\mathrm{a}+1)>\mathrm{dis}(\mathrm{a}+1,\mathrm{r})$ ซึ่งเราจะสามารถ binary search หาค่าจุด a ที่เข้าตามเงื่อนข้างต้น หาก $\mathrm{dis}(\mathrm{l},\mathrm{a})\leq\mathrm{dis}(\mathrm{a},\mathrm{r})$ เราจะปรับให้ไปดูทางฝั่งขวาของตัวที่ดูในปัจจุบัน นอกจากนั้นจะดูฝั่งซ้ายของตัวที่ดูในปัจจุบัน ซึ่งในตอนสุดท้าย เราจะสนใจเฉพาะ $\max(\mathrm{dis}(\mathrm{l},\mathrm{a}),\mathrm{dis}(\mathrm{a},\mathrm{r}))$ และ $\max(\mathrm{dis}(\mathrm{l},\mathrm{a}+1),\mathrm{dis}(\mathrm{a}+1,\mathrm{r}))$ หาก $\mathrm{a}+1\leq\mathrm{r}$ ซึ่งสามารถทำได้ใน Time Complexity โดยรวมคือ $O(\log(\mathrm{n}))$ ต่อ 1 คำถาม ทำให้ Time Complexity โดยรวมคือ $O(\mathrm{q}\log(\mathrm{n}))$