Editorial

ก่อสร้างทางเดิน (100 คะแนน)

output-only

ข้อนี้ เป็นโจทย์ output-only ที่ทำได้หลายวิธีมาก

Paper and Pen

วิธีพื้นฐานเลยคือนั่งทดแล้วเขียนไฟล์ .txt ใน notepad เอา วิธีนี้จะทำให้เสียเวลาสำหรับกรณีที่ N มีค่ามาก แต่ได้ผลลัพธ์ที่ค่อนข้างดีเทียบกับคอมพิวเตอร์

Random search

วิธีนี้คือสุ่มตารางมั่วมา แล้วเช็คดูว่าถูกหรือเปล่า หากไม่ถูกก็สุ่มใหม่เรื่อยๆ กรณี N มากจะใช้เวลานาน เพราะสุ่มเท่าไหร่ก็ไม่เจอตารางที่ดี (มันหายาก ความน่าจะเป็นต่ำ)

Iterative Deepening Search

หาก random ได้จำนวนคำตอบมั่ว เราสามารถ fix จำนวนกำแพงไว้ก่อนได้หรือไม่

วิธีนี้เป็นการหาคำตอบกรณีมีกำแพง K ช่อง เราค่อยๆเติมกำแพงทีละช่องโดยไม่ให้ผิดเงื่อนไข (จะผิดเงื่อนไขเมื่อการเติมกำแพงนั้นทำให้มีบางช่องที่ไม่สามารถเดินเข้าไปได้ เราสามารถตรวจสอบด้วย Flood-Fill Algorithm ได้) ต่อมาก็ไล่ไปเรื่อยๆ จนกว่าจะเจอ แต่ในเคสที่ใหญ่อาจไม่เจอเพราะจำนวน state มีมากเกินไป เราจึงตัดมาเพียงไม่กี่แบบเพื่อตรวจสอบ ซึ่งทำให้ได้คำตอบอาจไม่ดีที่สุด แต่พอทำได้ในเวลา วิธีนี้จะได้คะแนนประมาณ 70-80 คะแนน

Machine Learning

เราสามารถใช้เทคนิคทาง Machine Learning เช่น Reinforcement Learning (Q-Learning หรือ SARSA Algorithm) แต่เนื่องจาก State มีมากเกินไปจึงทำให้ใช้งานไม่ค่อยได้

Genetic Algorithm

เราสามารถใช้ Genetic Algorithm โดย encode state แล้วนำไป crossover กันในแต่ละรุ่น ทางผู้เขียนยังไม่ได้ทดลองใช้วิธีนี้ แต่คิดว่าน่าจะได้ผลลัพธ์ที่ค่อนข้างดี

Greedy (Hill Climbing with Heuristics) Algorithm

วิธีนี้อาจไม่ดีที่สุด แต่เป็นวิธีที่ทำให้เกิดผลลัพธ์ได้ดีในเวลาอันสั้น เราจะค่อยๆเติมกำแพงไปเรื่อยๆ แต่ในแต่ละขั้นตอนเราจะเลือกกำแพงในช่องที่ดีที่สุด โดยเราจะมี Heuristic สำหรับคำว่า ดีที่สุด คือ ช่องที่ทำให้ลดจำนวนช่องที่ไม่ได้ไปมากที่สุด กล่าวคือใน state ปัจจุบันจะมีบางช่องที่ยังไปไม่ถึง ให้มีทั้งหมด U ช่อง ให้ $h(x) : \mathbb{Z}_n^2 \rightarrow \mathbb{Z}_0^+$ แทนมูลค่า ในการเติมกำแพงในช่องตำแหน่ง x (เรามอง x เป็นคู่อันดับของตำแหน่งในตาราง) สมมติว่าหลังเติมกำแพงในช่องที่ x ไปแล้ว ทำให้มีบางช่องที่ยังไปไม่ถึงทั้งหมด U' ช่อง เราจะนิยาม $h(x)$ เป็น $U - U'$ เราจะพยายามเลือกช่องที่ทำให้ $h(x)$ มีค่าสูงที่สุดเท่าที่เป็นไปได้ หลังจากนั้นก็คำนวณ $h(x)$ ใหม่สำหรับทุกๆช่อง

วิธีนี้จะได้คะแนนประมาณ 95-98 คะแนน

Improved Greedy Hill Climbing with Heuristics Algorithm

เป็นวิธีอันเดียวกับอันที่แล้ว แต่ในแต่ละขั้นอาจมี $h(x)$ ที่มากที่สุดได้หลายค่า เราลองสุ่ม x จากทุก x ที่ทำให้ $h(x)$ มีค่ามากสุด แต่ผลประกฏว่าได้ผลลัพธ์แย่กว่าเดิม เราจึงตั้งสมมติฐาน (เดาอย่างมีหลักการ) ว่า x ที่ดี จะต้องอยู่ขอบๆ หากสุ่มมั่วจะแย่ เราจึง random เลือกระหว่าง x ที่อยู่บนซ้ายสุด กับ x ที่อยู่ล่างขวาสุด นั่นคือ $min(x)$ กับ $max(x)$ ไป ซึ่งทำให้ผลลัพธ์ดียิ่งขึ้น เมื่อ random หลายๆรอบจะสามารถหาคำตอบที่ดีได้

วิธีนี้จะได้คะแนนประมาณ 95-100 คะแนน

ประตูวิเศษ (100 คะแนน)

3.5 seconds, 256 megabytes

Subtask 1

มี 2 กรณี คือเริ่มประตูแรก กับเริ่มประตูที่สอง จะเริ่มประตูแรกได้ก็ต่อเมื่อ ($K = 1$ และ $P_2 = 2$) หรือ $P_1 = 2$ และจะเริ่มประตูที่สองได้ก็ต่อเมื่อ $K = 0$ และ $P_2 = 2$

Subtask 2

ตอบตำแหน่งของประตูที่มีหมายเลข N เพราะทุกประตูไม่สามารถพาไปไหนได้เลยนอกจากประตูเดิม

Subtask 3

ดูว่าประตูหมายเลข N อยู่ตรงไหน หากอยู่ไม่เกิน K นับจากประตู 1 สามารถตอบ 1 ได้เลย แต่หากอยู่เกิน K ให้ตอบตำแหน่งของประตู ลบด้วย K

Subtask 4

เงื่อนไขเดียวกันกับ Subtask 2 แต่สังเกตได้ยากกว่า

Subtask 5

พิจารณาเป็น graph ของ directed cycle หลายๆ อัน เราไล่ตั้งแต่ 1 ถึง N หากประตู ณ ตำแหน่งใดอยู่ใน cycle เดียวกับประตูหมายเลข N ตอบประตูแรกที่เจอ เพราะไม่ว่ายังไงเราสามารถเดินได้ทั้งหมด $K = N$ ซึ่งยาวกว่าขนาดของทุก cycle

Subtask 6

พิจารณาเป็น graph เช่นเดิม เรามองว่ามี state อยู่ทั้งหมด N states โดยใน state ที่ i จะเป็น graph ที่มีจุดเริ่มต้นที่เป็นไปได้ตั้งแต่ประตูตำแหน่ง 1 ถึง i เราค่อยๆวิเคราะห์ทุก state ทั้ง N state และทำการไล่ว่า แต่ละประตูใน state นั้นจะพาเราไปถึงประตูไหนได้บ้าง แล้วหา state ที่มีเลขน้อยที่สุดที่ทำให้เราสามารถไปถึงประตูหมายเลข N ภายในการใช้ประตูไม่เกิน K ครั้งได้

Time Complexity: $\mathcal{O}(N^3)$

Subtask 7

วิธีคล้าย Subtask 6 แต่เราค่อยๆทำแบบ Sequencial ไล่จาก state 1 ถึง N ในแต่ละ state จะมีเพียงประตูเดียวที่เพิ่มขึ้นมา เราคำนวณเฉพาะประตูนั้น เพราะประตูก่อนหน้าจะมีคำตอบเหมือนเดิม จะลดเวลาการทำงานอย่างมาก

Time Complexity: $\mathcal{O}(N^2)$

Official Solution

จาก Subtask 6, 7 เราจะหาประตูตำแหน่งที่น้อยที่สุด ที่ทำให้ไปถึงหมายเลข N ไม่เกิน K ครั้งได้ สังเกตว่าเราสามารถ Binary Search หาว่าต้องไปถึงประตูที่เท่าไรจึงจะสามารถไปหา N โดยใช้ไม่เกิน K ครั้ง ในแต่ละครั้งที่ Binary Search เราสามารถใช้วิธีแบบ Subtask 6 แต่ละทำให้ Time Complexity เป็น $\mathcal{O}(N^2 \log N)$ แต่หากเราใช้วิธี Breadth-first search จะทำให้ลดเวลาลง

Time Complexity: $\mathcal{O}(N \log N)$

Alternative Solution

เราสามารถมองว่า การไปให้ถึงประตูหมายเลข N โดยใช้ประตูไม่เกิน K ครั้ง มองย้อนกลับได้ว่า ย้อนจาก N ไป K รอบ มีประตูไหนบ้าง เหตุผลที่เราสามารถมองแบบนี้ได้เป็นเพราะค่า D ไม่ซ้ำกันเลย เมื่อเรามองย้อนกลับได้แล้วจะรู้ว่าเริ่มประตูไหนได้บ้าง ก็ตอบประตูที่อยู่ตำแหน่งน้อยที่สุด

Time Complexity: $\mathcal{O}(N)$

ลองชุด (100 คะแนน)

4.5 seconds, 512 megabytes

สำหรับข้อนี้ เราจะเขียนในรูปแบบคณิตศาสตร์คือ มีลำดับ S_i มาให้ตั้งแต่ $1 \le i \le N$ จะมีคำถาม Q คำถาม แต่ละคำถามจะระบุค่า L, R เราจะต้องตอบค่าของ $\sum_{i=L+1}^{R} \sum_{j=L}^{i-1} S_i \times S_j$

Subtask 1

นำค่าที่ได้มาคูณกัน เพราะมีการจับได้เพียงคู่เดียว

Subtask 2

สำหรับแต่ละคำถาม ไล่จับทุกคู่ที่เป็นไปได้ เนื่องจากมี Q คำถาม แต่ละคำถามมี $1 \le L < R \le N$ แสดงว่ามีได้มากสุด $\binom{N}{2}$ คู่

Time Complexity: $\mathcal{O}(QN^2)$

Subtask 3

สามารถทำได้แบบเดียวกับ Subtask 2 แต่สำหรับแต่ละคำถามต้องหยิบเฉพาะคู่ที่ต้องการนำมาคิดเท่านั้น แต่ละคำถามจึงมีได้มากสุด $\binom{4}{2} = 6$ คู่เท่านั้น

Time Complexity: $\mathcal{O}(Q)$

Subtask 4

สำหรับปัญหาย่อยนี้ จะต้องพิจารณามากขึ้น สำหรับแต่ละคำถาม เราจะมีเวลาไม่มาก จึงไม่สามารถใช้อัลกอริทึมระดับ $\Theta(N^2)$ ได้ แต่จะต้องใช้อัลกอริทึมที่มีประสิทธิภาพมากกว่านั้น

สังเกตว่า แต่ละคำถาม เราจับคู่ทุกคู่ที่เป็นไปได้มาตอบ เราจะแตกคำถามออกเป็น "สำหรับตัวที่ i ใดๆจาก L ถึง R หากให้ตัวนั้นเป็นตัวท้าย แล้วจะได้ผลลัพธ์เป็นอย่างไร" เราจะสังเกตได้ว่า $\sum_{i=L+1}^{R} \sum_{j=L}^{i-1} S_i \times S_j = \sum_{i=L+1}^{R} S_i \times f(i)$ เมื่อ $f(i)$ แทนผลรวมของ S_j สำหรับทุก $L \le j < i$

เราเก็บตัวแปรเพิ่มเติมแทนค่า $f(i)$ ณ สถานะใดๆ และค่อยๆปรับไปเรื่อยๆ (เปลี่ยน $f(i)$ เป็น $f(i+1)$ โดยการเพิ่มค่า S_i เข้าไปในตัวแปร) แต่ละครั้งที่ถาม จึงใช้เพียงเวลา $\mathcal{O}(N)$ เท่านั้น

Time Complexity: $\mathcal{O}(QN)$

Subtask 5

สามารถใช้วิธีเดียวกับ Subtask 4 ได้ แต่เพราะ Subtask 4 อาจมีวิธีอื่นๆ เช่นวิธี $\mathcal{O}(N^2 + Q\log N)$ จึงมี Subtask 5 ที่ให้เฉพาะ $\mathcal{O}(QN)$ เท่านั้น

Subtask 6

หาก $0 \le S_i \le 1$ จะมีสมบัติที่ว่า $S_i \times S_j = 1$ ก็ต่อเมื่อ $S_i = S_j = 1$ มิฉะนั้นจะได้เป็น 0 จึงได้ว่าโจทย์คือให้นับว่าภายในแต่ละช่วงที่ถามนั้นสามารถจับคู่เลข 1 ได้ทั้งหมดกี่วิธี โดยเราสามารถสร้าง $T_i = T_{i-1} + S_i$ (จะนิยามเฉพาะ $0 \le i \le N$ เท่านั้นและกำหนด $T_0 = 0$) แล้วหาจำนวนเลข 1 ในช่วง L ถึง R ได้โดยจะมีค่าเป็น $T_R - T_{L-1}$ เมื่อต้องการนับจำนวนคู่ จะได้เท่ากับ $\binom{T_R - T_{L-1}}{2}$ คู่

Time Complexity: $\mathcal{O}(N+Q)$

Subtask 7

ใช้วิธีการสังเกตของ Subtask 4 และ 6 รวมกัน คือการหาคำตอบในช่วง L ถึง R นั้น ไม่จำเป็นต้องใช้อะไรเยอะ ใช้แค่ค่าบางค่าพอ และเราจัดรูปนิพจน์คำถามจาก $\sum_{i=L+1}^{R}\sum_{j=L}^{i-1} S_i \times S_j$ เราจะนิยามค่า $U = \sum_{i=L}^{R} S_i^2$ และค่า $T = \sum_{i=L}^{R} S_i$

สังเกตค่า T^2 จะได้ว่ามีค่าเท่ากับ ผลรวมของผลคูณทุกคู่โดยนับซ้ำ (ซึ่งก็คือ $\sum_{i=L}^{R}\sum_{j=L}^{R} S_i \times S_j$) เมื่อเราลบค่า U ออกจาก T^2 เราจะได้ผลรวมของผลคูณทุกคู่โดยไม่มีกรณี $i = j$ (ซึ่งก็คือ $\sum_{i=L}^{R}\sum_{\substack{j=L\\ j\neq i}}^{R} S_i \times S_j$) ต่อมาเรานำค่าที่ได้ไปหารด้วย 2 เพราะการจับคู่โดยปราศจาก $i = j$ สำหรับคู่ x, y ใดๆจะมีคู่ที่ $i = x$ กับ $j = y$ และมีคู่ที่ $i = y$ กับ $j = x$ นับรวมอยู่ด้วย

จึงสรุปได้ว่า คำตอบคือ $\frac{T^2 - U}{2}$

เราคำนวณค่า U และ T ของช่วงใดๆ ได้ด้วยวิธีเดียวกับ Subtask 6 ทำให้สามารถตอบคำถามได้ในทันที

Time Complexity: $\mathcal{O}(N+Q)$