รายงานวิชา Algorithm Design and Analysis

เรื่อง

FARIDA – Princess Farida

ผู้จัดทำ

นายณัฐกิตติ์ จริตรัมย์ 5830300206

เสนอ

อาจารย์อดิศักดิ์ สุภีสุน

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา

Algorithm Design and Analysis รหัสวิชา 03603212

ภาควิชาวิศวกรรมคอมพิวเตอร์และสารสนเทศศาสตร์ คณะวิศวกรรมศาสตร์ศรีราชา

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

ภาคเรียนที่ 2 ปีการศึกษา 2562

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา Algorithm Design and Analysis รหัสวิชา 03603212 ภาคปลาย ปีการศึกษา 2562 จัดทำขึ้นเพื่อทำการศึกษา วิเคราะห์และออกแบบอัลกอริทึม พร้อมนำเสนอจากผลลัพธ์ความรู้ที่ได้รับในการศึกษาในรายวิชาดังกล่าว

โดยเนื้อหาของรายงานฉบับนี้จะกล่าวถึงปัญหาที่ชื่อว่า FARIDA - Princess Farida ที่อยู่ในเว็บไซต์ https://www.spoj.com/problems/FARIDA/ พร้อมนำเสนออัลกอริทึมที่ใช้แก้ปัญหา และบทวิเคราะห์อัลกอริทึมที่ใช้สำหรับในการแก้ปัญหาดังกล่าว ซึ่งโจทย์ปัญหานี้ใช้แนวคิดและความรู้ในเรื่องของ Dynamic Programming

ทางผู้จัดทำคาดหวังไว้เป็นอย่างยิ่งว่ารายงานฉบับนี้จะเป็นประโยชน์ให้กับผู้ที่ได้เปิดอ่านและกำลังศึกษาอัลกอริทึมที่เกี่ยวข้องให้ได้รับความรู้เพิ่มเติมจากรายงานฉบับนี้ และสุดท้ายนี้ถ้าเกิดผู้อ่านได้พบเจอความผิดพลาดใด ๆ ในส่วน ๆ ใดภายในรายงานฉบับนี้ ทางผู้จัดทำก็ขออภัยมา ณ ที่นี้ด้วย

นายณัฐกิตติ์ จริตรัมย์

ผู้จัดทำ

สารบัญ

FARIDA – Princess Farida

Memory Limit : 1,536 MB

Source Limit : 50,000 B

Time Limit : 1.237 s

กาลครั้งหนึ่งนานมาแล้วมีเจ้าหญิงที่แสนน่ารักชื่อว่าฟาริดาอาศัยอยู่ในปราสาทกับบิดา มารดา และคุณลุงของเธอ ระหว่างเส้นทางที่จะเดินทางมายังปราสาทนั้นมีสัตว์ประหลาดหลากหลายตัว สัตว์ประหลาดแต่ละตัวนั้นจะมีเหรียญทองอยู่จำนวนหนึ่ง ถึงแม้ว่ามันจะเป็นสัตว์ประหลาดแต่มันก็จะไม่ทำร้าย และกลับกันพวกมันจะมอบเหรียญทองให้กับคุณแทน แต่ก็ต่อเมื่อคุณไม่รับเหรียญจากสัตว์ประหลาดตัวก่อนหน้าตัวปัจจุบันโดยตรงเท่านั้น เพื่อที่จะแต่งงานกับเจ้าหญิงฟาริดา คุณจะต้องผ่านสัตว์ประหลาดทุกตัวและเก็บสะสมเหรียญให้ได้มากที่สุดเท่าที่จะทำได้ กำหนดจำนวนเหรียญที่สัตว์ประหลาดแต่ละตัวมี คำนวณหาจำนวนเหรียญสูงสุดที่คุณสามารถเก็บสะสมได้ระหว่างการเดินทางไปยังปราสาท

Input

บรรทัดแรกของการรับค่าคือจำนวนของชุดทดสอบ ในแต่ละชุดทดสอบจะเริ่มด้วยจำนวน N แทนจำนวนของสัตว์ประหลาด (0 <= N <= 10,000) บรรทัดถัดไปจะมีตัวเลข N ตัว แทนจำนวนเหรียญที่สัตว์ประหลาดแต่ละตัวมี (0 <= จำนวนเหรียญที่สัตว์ประหลาดแต่ละตัวมี <= 10^9) สัตว์ประหลาดจะถูกเรียงลำดับตามการเผชิญหน้าระหว่างเส้นทางไปยังปราสาท

Output

ในแต่ละชุดทดสอบให้แสดงผลลัพธ์ “Case C: X” โดยไม่มีโควท C คือลำดับชุดทดสอบ เริ่มต้นที่ 1 X คือจำนวนเหรียญสูงสุดที่คุณเก็บสะสมได้

Example

Input	Output
2 5 1 2 3 4 5 1 10	Case 1: 9 Case 2: 10

Input บรรทัดที่ 1 คือจำนวนชุดทดสอบที่จะทำการทดสอบ ตามตัวอย่างด้านบน เราจะทำการทดสอบทั้งหมด 2 ชุดทดสอบ จึงป้อนค่าเลข 2 ลงในบรรทัดแรกสุด ในแต่ละ 2 บรรทัดต่อ ๆ มาคือข้อมูลของชุดทดสอบแต่ล่ะชุด โดยที่บรรทัดแรกสุดของแต่ละชุดทดสอบ คือจำนวนสัตว์ประหลาดสำหรับชุดทดสอบนั้น ๆ และบรรทัดต่อมาคือจำนวนเหรียญที่สัตว์ประหลาดแต่ละตัวมี โดยลำดับการรับค่าคือลำดับเดียวกับสัตว์ประหลาดแต่ละตัวเริ่มต้นจากตัวแรก ตามตัวอย่างด้านบน จึงหมายความว่า บรรทัดที่ 2 คือจำนวนสัตว์ประหลาดของชุดทดสอบแรก มีทั้งหมด 5 ตัว เราจึงป้อนค่าเลข 5 ลงในบรรทัดที่ 2 ในส่วนของบรรทัดที่ 3 คือจำนวนเหรียญที่สัตว์ประแต่ละตัวในชุดทดสอบแรกมี โดยเราจะป้อนค่าตัวเลขทั้งหมด 5 ตัว ในบรรทัดเดียวกันโดยใช้ช่องว่างเป็นตัวแยกข้อมูลจำนวนเหรียญของสัตว์ประหลาดแต่ละตัว ข้อมูลบรรทัดที่ 3 จึงมีความหมายว่า สัตว์ประหลาดตัวที่ 1 มีเหรียญอยู่ 1 เหรียญ สัตว์ประหลาดตัวที่ 2 มีเหรียญอยู่ 2 เหรียญ สัตว์ประหลาดตัวที่ 3 มีเหรียญอยู่ 3 เหรียญ สัตว์ประหลาดตัวที่ 4 มีเหรียญอยู่ 4 เหรียญ และสัตว์ประหลาดตัวที่ 5 มีเหรียญอยู่ 5 เหรียญ บรรทัดที่ 4 ต่อมาคือการใส่ค่าของชุดทดสอบชุดที่ 2 โดยชุดทดสอบนี้จะมีแค่สัตว์ประหลาด 1 ตัว และสัตว์ประหลาดตัวนี้มีเหรียญอยู่จำนวน 10 เหรียญ ดังข้อมูลที่ใส่ในบรรทัดที่ 5

Output แต่ละบรรทัดคือผลลัพธ์ของชุดทดสอบแต่ละชุด โดยผลลัพธ์จะเรียงลำดับตามลำดับของชุดทดสอบที่เราทดสอบ ตามตัวอย่างด้านบน บรรทัดที่ 1 คือผลลัพธ์ของชุดทดสอบที่ 1 เราจะให้ผลลัพธ์แสดงประโยคออกมาว่า Case 1: 9 หมายความว่าชุดทดสอบที่ 1 มีจำนวนเหรียญที่เก็บสะสมได้สูงสุดคือ 9 เหรียญ บรรทัดที่ 2 ก็คือผลลัพธ์ของชุดทดสอบที่ 2 มีความหมายว่าชุดทดสอบที่ 2 มีจำนวนเหรียญที่เก็บสะสมได้สูงสุดคือ 10 เหรียญ

อัลกอริทึม

```
1 : procedure main()
2 :     N[0, . . . , n] ⟵ empty array
3 :     for each n do
4 :         A[0, . . . , t] , B[0, . . . , t] ⟵ A and B are empty array
5 :         B[0]  ⟵ A[0]
6 :         B[1] ⟵ most (B[0] , A[1])
7 :         for j = 2, . . . , t - 1 do
8 :             B[j] ⟵ most (B[j - 1] , A[j] + B[j – 2])
9 :         N[n] ⟵ B[t - 1]
10 :    for each n do
11 :        print “Case ” n + 1 “: ” N[n]
12 : end procedure
13 : Return main()
```

หลังจากสร้าง main ในบรรทัดที่ 1 แล้วบรรทัดที่ 2 เราจะทำการกำหนดอาเรย์ N ที่มีขนาด n ขึ้นมาเพื่อใช้เก็บคำตอบของชุดทดสอบ n ชุด บรรทัดที่ 3 คือใช้ลูปในการหาคำตอบของชุดทดสอบแต่ละชุด ในชุดทดสอบแต่ละชุดเราจะทำการกำหนดอาเรย์ A และ B ขึ้นมา โดยอาเรย์ A ใช้เก็บจำนวนเหรียญของสัตว์ประหลาดแต่ละตัวตามลำดับที่เผชิญหน้า ส่วนอาเรย์ B ใช้เก็บจำนวนผลลัพธ์เหรียญที่มากที่สุดเมื่อเผชิญหน้ากับสัตว์ประหลาด ณ ลำดับนั้น บรรทัดที่ 5 คือการใส่ค่าในอาเรย์ B ตำแหน่งที่ 1 ให้มีค่าเท่ากับอาเรย์ A ตำแหน่งที่ 1 เนื่องจากเป็นค่าเริ่มต้น บรรทัดที่ 6 ให้เราเทียบว่าค่าในอาเรย์ B ตำแหน่งที่ 1 หรือก็คือจำนวนเหรียญที่มากที่สุดที่เก็บได้หลังจากเผชิญหน้ากับสัตว์ประหลาดตัวที่ 1 กับค่าในอาเรย์ A ตำแหน่งที่ 2 หรือก็คือจำนวนเหรียญของสัตว์ประหลาดตัวที่ 2 ถ้าค่าไหนมากกว่ากว่ากัน อาเรย์ B ตำแหน่งที่ 2 จะเก็บค่านั้นไว้ บรรทัดที่ 7 คือการใช้ลูปตั้งแต่สัตว์ประหลาดตัวที่ 3 ขึ้นไปจนถึงตัวสุดท้าย โดยแต่ละลูปจะเปรียบเทียบระหว่างค่า ของอาเรย์ B ตำแหน่างก่อนหน้าตำแหน่งปัจจุบัน 1 ลำดับ กับ ค่าอาเรย์ A ตำแหน่งปัจจุบัน รวมกับ ค่าอาเรย์ B ตำแหน่งก่อนหน้า 2 ลำดับ ว่าค่าไหนมีค่ามากกว่ากัน อาเรย์ B ตำแหน่งปัจจุบันจะเก็บค่านั้นไว้ หลังจากเปรียบเทียบทั้งหมดจนถึงสัตว์ประหลาดลำดับสุดท้าย เราจะใช้อาเรย์ N ตำแหน่งเดียวกับลำดับชุดทดสอบเก็บค่าของอาเรย์ B ตำแหน่งสุดท้ายไว้ หลังจากทำการคำนวนผลลัพธ์ของชุดทดสอบทั้งหมด เราก็จะวนลูปเพื่อแสดงค่าทั้งหมดตามบรรทัดที่ 10 และ 11

พิสูจน์อัลกอริทึม

Definitions พิสูจน์ว่าอัลกอริทึมนี้ให้คำตอบที่มีค่ามากที่สุดและไม่ขัดแย้งกับเงื่อนไขที่กำหนด

Theorem ค่าของตำแหน่ง B[t - 1] คือจำนวนเหรียญที่มากที่สุดของชุดทดสอบ (Proof by Induction)

Base Case เมื่อกำหนดให้ t = 5 จะทำให้อาเรย์ A และอาเรย์ B มีสมาชิกทั้งหมด 5 ตัว โดยถ้ากำหนดให้สมาชิกในอาเรย์ A ทั้ง 5 ตัว คือ {1, 2, 3, 4, 5} จาก B[0] = A[0] จะทำให้อาเรย์ B ตำแหน่งแรกมีค่าเดียวกับอาเรย์ A ตำแหน่งแรกเช่นเดียวกัน คือ 1 และ B[1] คือค่าที่มากสุดระหว่าง B[0] ที่มีค่าเท่ากับ 1 และ A[1] ที่มีค่าเท่ากับ 2 จึงทำให้ B[1] มีค่าเท่ากับ 2 จึงได้ว่า

เมื่อ A[0] = 1 จะทำให้ B[0] = 1

เมื่อ A[1] = 2 จะทำให้ B[1] = 2

- เนื่องจาก A[1] มีค่ามากกว่า B[0] จึงทำให้ B[1] = A[1]

จาก Theorem ที่กล่าวไว้ว่าตำแหน่ง B[t - 1] คือตำแหน่งที่จำนวนเหรียญมีค่ามากที่สุด เมื่อ t = 5 จะได้ว่าตำแหน่ง B[t - 1] ก็คือ B[5 - 1] = B[4] และจากเงื่อนไขบรรทัดที่ว่า

for j = 2, . . . , t - 1 **do**

B[j] ⟵ most (B[j - 1] , A[j] + B[j – 2])

จะทำให้ได้ว่า

เมื่อ A[2] = 3 จะทำให้ B[2] = 4

- เนื่องจาก B[2] = ค่าที่มากที่สุดระหว่างระหว่าง B[1] = 1 กับ A[2] + B[0] = 3 + 1 = 4 จึงทำให้ B[2] = 4

เมื่อ A[3] = 4 จะทำให้ B[3] = 6

- เนื่องจาก B[3] = ค่าที่มากที่สุดระหว่างระหว่าง B[2] = 4 กับ A[3] + B[1] = 4 + 2 = 6 จึงทำให้ B[3] = 6

เมื่อ A[4] = 5 จะทำให้ B[4] = 9

- เนื่องจาก B[4] = ค่าที่มากที่สุดระหว่างระหว่าง B[3] = 6 กับ A[4] + B[2] = 5 + 4 = 9 จึงทำให้ B[4] = 9

ดังนั้น B[t - 1] เมื่อ t = 5 จะได้ว่า B[5 - 1] = B[4] และพบว่า B[4] มีจำนวนเหรียญที่มากที่สุดคือ 9 เหรียญ

Inductive Step เมื่อกำหนดให้ t = n จะมีความหมายว่าตำแหน่งที่ B[n - 1] คือตำแหน่งที่มีจำนวนเหรียญมากที่สุด และจากเงื่อนไขที่กำหนด จากบรรทัดที่ว่า

for j = 2, . . . , t - 1 **do**

B[j] ⟵ most (B[j - 1] , A[j] + B[j – 2])

จะได้ว่า

for j = 2, . . . , n - 1 **do**

B[j] ⟵ most (B[j - 1] , A[j] + B[j – 2])

เมื่อตำแหน่ง B[n - 1] = B[j] จะพบว่า ค่าของตำแหน่ง B[n – 1] มีค่าเท่ากับค่าที่มากที่สุดระหว่างค่าของตำแหน่ง B[n - 2] กับค่าผลรวมของตำแหน่ง A[n - 1] + B[n - 3] ซึ่งมาจากเงื่อนไขที่กำหนดและจะได้ว่าทุก ๆ ตำแหน่งของอาเรย์ B ตั้งแต่ n – 1 ลงไปจนถึง 2 นั้นจะเลือกค่าที่มากที่สุดหรือเท่ากับมาเก็บไว้ตลอด โดยค่าแรกสุดของ B[2] มาจากค่าที่มากที่สุดระหว่าง B[1] กับ A[2] + B[0] ซึ่งเมื่อไล่ย้อนกลับขึ้นจะพบว่าทุกๆค่าตั้งแต่ตำแหน่ง B[2] ขึ้นไปจนถึง B[n - 1] นั้นสุดท้ายแล้วจะได้ผลลัพธ์ออกมาว่า ทุกตำแหน่งของอาเรย์ B จะมีค่ามากกว่าหรือเท่ากับค่าก่อนหน้าของตำแหน่งปัจจุบันเสมอ ทำให้ตำแหน่ง B[n - 1] ที่เป็นตำแหน่งสุดท้ายจึงมีค่ามากที่สุด

ดังนั้นเมื่อ B[n - 1] คือตำแหน่งที่มีจำนวนเหรียญมากที่สุดของชุดทดสอบ เมื่อ B[n - 1] = B[t - 1] จึงทำให้ตำแหน่ง B[t - 1] เป็นตำแหน่งที่มีจำนวนเหรียญมากที่สุดที่เป็นไปตามเงื่อนไขของโจทย์

วิเคราะห์เวลาการทำงานของอัลกอริทึม

```
1 :  procedure main()
2 :      N[0, . . . , n] ⟵ empty array ---------- O(1)
3 :      for each n do ---------- O(n)
4 :          A[0, . . . , t] , B[0, . . . , t] ⟵ A and B are empty array ---------- O(1)
5 :          B[0]  ⟵ A[0] ---------- O(1)
6 :          B[1] ⟵ most (B[0] , A[1]) ---------- O(1)
7 :          for j = 2, . . . , t - 1 do ---------- O(n)
8 :              B[j] ⟵ most (B[j - 1] , A[j] + B[j – 2]) ---------- O(1)
9 :          N[n] ⟵ B[t - 1] ---------- O(1)
10 :     for each n do ---------- O(n)
11 :         print “Case ” n + 1 “: ” N[n] ---------- O(1)
12 : end procedure
13 : Return main()
```

พิจารณาจากอัลกอริทึมด้านบนจะพบว่าทุกบรรทัดนอกจากบรรทัดที่ 3, 7 และ 10 จะทำงานเสร็จในเวลาเพียงแค่ O(1) เท่านั้น ในขณะที่ลูป for ของบรรทัดที่ 3 จะมีเวลาการทำงานไม่เกิน O(n) รอบ และขณะเดียวกันลูป for บรรทัดของที่ 7 ที่อยู่ในลูป for ของบรรทัดที่ 3 ก็จะมีเวลาการทำงานไม่เกิน O(n) รอบเช่นเดียวกันกัน ทำให้เวลาการทำงานของบรรทัดที่ 3 ถึง 9 จบการทำงานในเวลา $O(n^2)$ เพียงเท่านั้น ในลูป for บรรทัดที่ 10 เองก็จะมีการทำงานไม่เกิน O(n) เช่นกัน จึงสรุปได้ว่าเวลาการทำงานของอัลกอริทึมนี้ใช้เวลาในการทำงานไม่เกิน $O(n^2) + O(n) = O(n^2)$