ปัญหา รูปแบบของลำดับตัวอักษรโบราณ (PatternOfCharSequence)

นักโบราณคดีค้นพบลำดับตัวอักษรของอาณาจักรโบราณ เมื่อศึกษาอย่างละเอียดนักโบราณคดีก็ทราบว่าสายอักขระมีการจัดเรียงอย่างมีรูปแบบ โดยเกิดจากลำดับตั้งต้นที่มีความยาวของตัวอักษรไม่น้อยกว่า 2 ตัวอักษร และตัวอักษรในลำดับตั้งต้นนี้จะไม่มีตัวอักษรซ้ำกันเลย แล้วลำดับตั้งต้นจะถูกดำเนินการเพื่อให้ได้ลำดับใหม่ที่มาต่อกับลำดับตั้งต้นหลายครั้ง (จำนวนครั้งในการสร้างลำดับใหม่จากลำดับตั้งต้นมีอย่างน้อย 2 ครั้ง)

สำหรับการดำเนินการมี 2 รูปแบบคือ

(1) left shift ตามจำนวนตัวอักษรโดยไม่เกินจำนวนตัวอักษรตั้งต้น เช่นสมมติรูปแบบตั้งต้นคือ ABCD คำสั่งเป็น < 1 หมายความว่าให้เลื่อนตัวอักษรไปทางซ้าย 1 ตัวอักษรแล้วตัวอักษรทางซ้ายจะมาอยู่ต่อท้ายทางขวาของข้อความ ได้ผลลัพธ์คือ

(2) left shift ตามจำนวนตัวอักษรโดยไม่เกินจำนวนตัวอักษรตั้งต้น เช่นสมมติรูปแบบตั้งต้นคือ ABCD คำสั่งเป็น > 1 หมายความว่าให้เลื่อนตัวอักษรไปทางซ้าย 1 ตัวอักษรแล้วตัวอักษรทางซ้ายจะมาอยู่ต่อท้ายทางขวาของข้อความ ได้ผลลัพธ์คือ

ปัญหาคือสิ่งที่นักโบราณคดีได้มาคือลำดับของอักษรที่ยาวมาก และไม่รู้ว่ารูปแบบตั้งต้นคืออะไร จึงไม่รู้ว่าใช้ตัวดำเนินการแบบใดจึงเกิดลำดับนี้ขึ้นมา โปรดช่วยนักโบราณคดีกลุ่มนี้แกะรูปแบบลำดับอักษรตั้งต้นและการดำเนินการ โดยให้หาลำดับตั้งต้นที่มีความยาวน้อยที่สุด (ตั้งแต่ 2 ตัวอักษรขึ้นไป) ลำดับที่นักโบราณคดีพบนี้จะมีการดำเนินการกับลำดับตั้งต้นอย่างน้อย 2 รอบมาต่อกัน คือ ลำดับตั้งต้น ต่อด้วยลำดับผลลัพธ์จากการดำเนินการครั้งที่ 1 ต่อด้วย ลำดับผลจากการดำเนินการครั้งที่ 2 ... ต่อด้วย ลำดับผลจากการดำเนินการครั้งที่ N

ข้อสังเกตุ ตัวอักษรในลำดับที่นักโบราณคดีพบจะต้องมีตัวอักษรทุกตัวในลำดับตั้งต้น

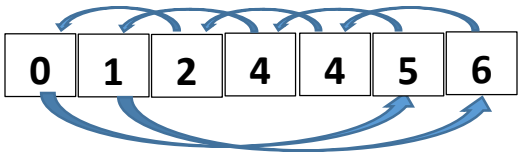
ตัวอย่างที่ 1 ลำดับคือ 012345623456014560123

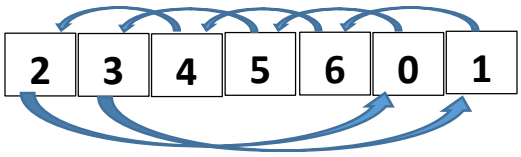
ลองทายลำดับตั้งต้นเป็นดังนี้

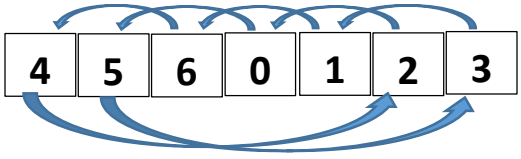
ลำดับตั้งต้นที่ทาย	ความเป็นไปได้	เหตุผล
01	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร 23456 ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
012	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร 3456 ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
0123	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร 456 ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
01234	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร 56 ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
012345	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร 6 ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
0123456	ได้	เพราะครอบคลุมทุกตัวอักษรในลำดับที่พบแล้ว
01234560	ไม่ได้	เพราะตัวอักษร 0 ซ้ำกัน 2 รอบ

ดังนั้นลำดับตั้งต้นคือ 0123456

ลองพิจารณาตัวอักษรลำดับถัดไปคือ 2345601 ทำให้เราทราบว่าตัวดำเนินการคือ left shift 2 ตัวอักษร







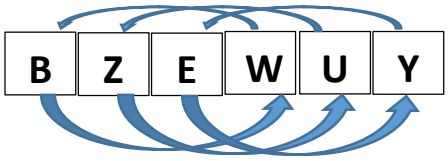
ตัวอย่างที่ 2 ลำดับคือ BZEWUYWUYBZEBZEWUYWUYBZE

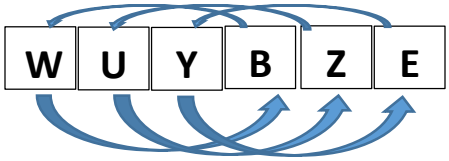
ลองทายลำดับตั้งต้นเป็นดังนี้

ลำดับตั้งต้นที่ทาย	ความเป็นไปได้	เหตุผล
BZ	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร EWUY ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
BZE	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร WUY ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
BZEW	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร UY ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
BZEWU	ไม่ได้	เพราะในลำดับที่พบมีตัวอักษร Y ทำให้ลำดับตั้งต้นนี้ไม่ครอบคลุม
BZEWUY	ได้	เพราะครอบคลุมทุกตัวอักษรในลำดับที่พบแล้ว
BZEWUYW	ไม่ได้	เพราะตัวอักษร W ซ้ำกัน 2 รอบ

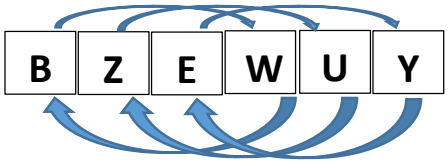
ดังนั้นลำดับตั้งต้นคือ BZEWUYW

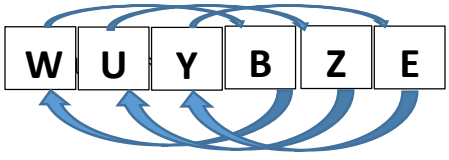
ลองพิจารณาตัวอักษรลำดับถัดไปคือ WUYBZE ทำให้เราทราบว่าตัวดำเนินการคือ left shift 3 ตัวอักษร

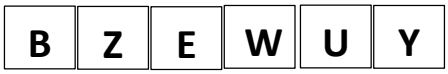




หรือ ตัวดำเนินการคือ right shift 3 ตัวอักษร







รูปแบบข้อมูลเข้า

บรรทัดแรก	ตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่หรือตัวเลขหรือสัญลักษณ์เรียงต่อกันโดยไม่มีช่องว่าง มีความยาวไม่เกิน 1000 ตัวอักษร และมีลำดับใหม่ที่เกิดจากลำดับตั้งต้นไม่น้อยกว่า 2 ลำดับ และไม่เกิน 20 ลำดับ

รูปแบบผลลัพธ์

ตัวดำเนินการ จำนวนกี่ตัว ถ้าลำดับที่ต่อกันไม่ได้มีการเรียงเป็นรูปแบบตามตัวดำเนินการให้พิมพ์ INVALID

บรรทัดแรก	ข้อความที่เป็นลำดับตั้งต้นที่ปรากฏในลำดับที่นักโบราณคดีค้นพบ
บรรทัดที่สอง	ตัวดำเนินการ อาจเป็น Left shift ใช้สัญลักษณ์ < ต่อด้วยตัวเลข Right shift ใช้สัญลักษณ์ > ต่อด้วยตัวเลข ให้ตอบตัวดำเนินการที่ตัวเลขน้อยกว่า ถ้าตัวเลขเท่ากันให้ตอบ “<” ถ้าไม่เปลี่ยนแปลงให้ตอบ =

ตัวอย่าง

ข้อมูลเข้า	ผลลัพธ์
ABCDBCDACDABDABCABCD	ABCD < 1
ABCDCDABABCDCDAB	ABCD < 2
ZXCDSAZXCDSAZXCDSA	ZXCDSA =
4563217890890456321721789045 63	4563217890 > 3
987669876897	INVALID

เมธอดที่น่าสนใจในคลาส String

Method signature	คำอธิบาย
char charAt(int)	คือค่าตัวอักษร ณ ตำแหน่งที่ i เช่น “Silpakorn”.charAt(4) จะได้ค่า ‘a’
int compareTo(String)	เปรียบเทียบสตริงเจ้าของเมธอด กับ สตริงในวงเล็บ ให้สตริงเจ้าของเมธอดเป็น text1 และสตริงในวงเล็บเป็น text2 เช่น “ANT”.compareTo(“CAT”) จะได้ -2 เพราะตัว A ลบ ตัว C เมธอดนี้จะเปรียบเทียบรหัสของตัวอักษรตามตำแหน่งตัวต่อตัว
int compareToIgnoreCase(String)	เปรียบเทียบสตริงเจ้าของเมธอด กับ สตริงในวงเล็บ ให้สตริงเจ้าของเมธอดเป็น text1 และสตริงในวงเล็บเป็น text2 เช่น “ant”.compareTo(“CAT”) จะได้ -2 เพราะตัว a ลบ ตัว c เมธอดนี้จะเปรียบเทียบรหัสของตัวอักษรตามตำแหน่งตัวต่อตัวโดยไม่สนใจตัวเล็กตัวใหญ่
int indexOf(String)	คืนค่าตัวแหน่งของ String ในวงเล็บที่ปรากฏในสตริงเจ้าของเมธอด ถ้ามีหลายตำแหน่งให้คืนตำแหน่งแรกที่ปรากฏ เช่น “MISSISIPPI”.indexOf(“SI”) จะได้ค่า 3
int indexOf(String,int)	คืนค่าตัวแหน่งของ String ในวงเล็บที่ปรากฏในสตริงเจ้าของเมธอด โดยตำแหน่งเริ่มต้นที่หาให้เริ่มจากค่า int ที่ปรากฏในวงเล็บที่เป็นอากิวเมนต์ตัวที่สอง ถ้ามีหลายตำแหน่งให้คืนตำแหน่งแรกที่ปรากฏ เช่น “MISSISIPPI”.indexOf(“SI”, 4) จะได้ค่า 5
String substring(int i)	คืนค่าสตริงที่ตัดตั้งแต่ตำแหน่ง i ถึงจบสตริง เช่น “MISSISIPPI”.substring(3) จะได้ค่า “SISIPPI”
String substring(int i0,int i1)	คืนค่าสตริงที่ตัดตั้งแต่ตำแหน่ง i0 ถึง i1-1 เช่น “MISSISIPPI”.substring(3,7) จะได้ค่า “SISI”