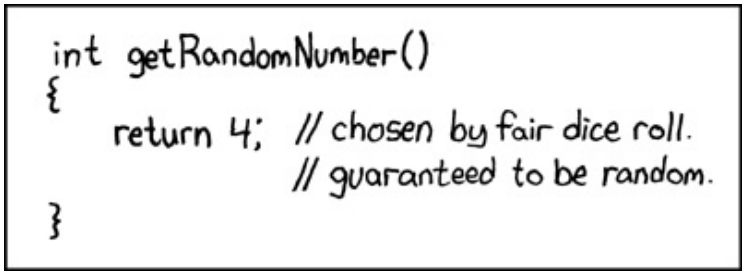
ลูกเต๋า

อะไร คือ ความสุ่ม (randomness)? คำถามนี้คงเป็นคำถามเชิงปรัชญาแสนล้ำลึก แต่ก็เป็นคำถามที่เรียบง่ายเหมือนกัน หากเราพิจารณาตัวเลข 4 เราอาจกล่าวได้ว่า 4 เป็นเลขที่มาจากการสุ่มจากการทอยลูกเต๋า

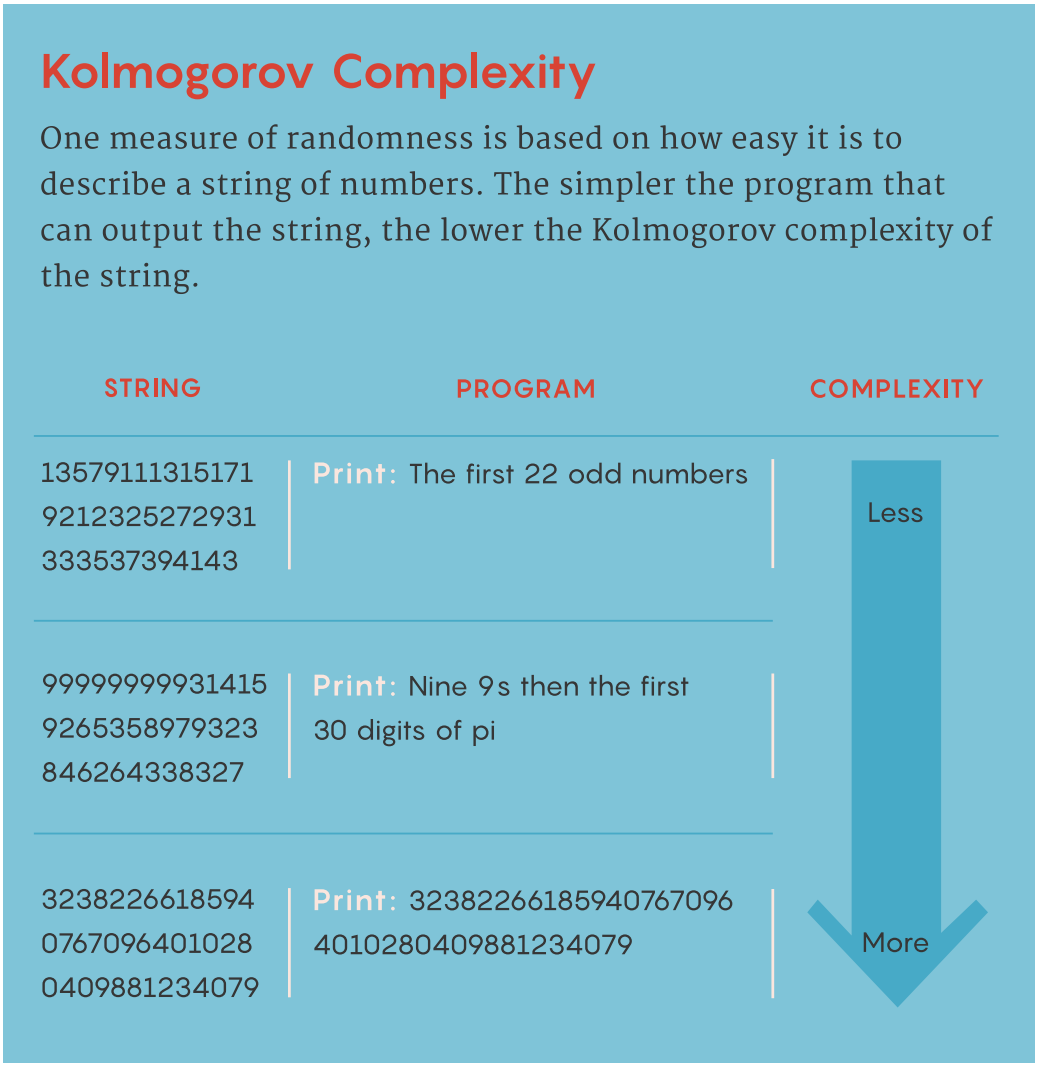
แล้วหากเราเขียนโปรแกรมสุ่มเลขแบบนี้ล่ะ?



(ภาพจาก xkcd)

แบบนี้ใครก็พูดได้สินะ ว่าเลขอะไรก็เป็น **เลขสุ่ม (random number)** ทั้งนั้น!

แท้จริงแล้วมันไม่ง่ายอย่างนั้นหรอก เรามีการศึกษาถึงคำถามว่า "จำนวนจำนวนหนึ่ง จะสามารถอธิบายได้อย่างง่ายที่สุดได้อย่างไร" (ดูภาพด้านล่างประกอบ) เราใช้สิ่งที่เรียกว่า **Kolmogorov Complexity**



(ภาพจาก Merrill Sherman/Quanta Magazine; บทความดัดแปลงจาก Quanta Magazine; ไม่มีลิงก์เนื่องจากอาจส่งผลกระทบต่อการแข่งขัน, จะมีการให้ลิงก์บทความหลังจบการแข่งขัน)

จากภาพจะเห็นได้ว่า:

- สำหรับสตริงตัวเลข `13579111315171921232527293133353739414 3` เราสามารถอธิบายได้ด้วยโปรแกรม "ส่งออกจำนวนคี่ 22 ตัวแรก"
- สตริงตัวเลข `99999999931415926535897932384626433832 7` เราสามารถอธิบายได้ด้วยโปรแกรม "ส่งออกเลข 9 เก้าตัวต่อด้วยเลขโดด 30 ตัวแรกของ π"
- สตริงตัวเลข `32382266185940767096401028040988123407 9` เราสามารถอธิบายได้ด้วยโปรแกรม "ส่งออก 32382266185940767096401028040988123407 9"

ในข้อนี้เราจะพิจารณา Kolmogorov Complexity ของจำนวนเต็มแต่ละตัว ตามรูปแบบที่กำหนดไว้ในแต่ละปัญหาย่อย โดยงานของคุณคือ ให้เขียนโปรแกรม A รับข้อมูลนำเข้าเป็นจำนวนเต็ม C แล้วส่งออกโปรแกรม B ที่เมื่อทำการเรียกใช้งานโปรแกรม B แล้วจะส่งออกจำนวนเต็ม C ที่กำหนดให้ โดยเป้าหมายคือให้โปรแกรม B มีความยาว**สั้นที่สุดเท่าที่เป็นไปได้**

ภาษา Assfuck

เพื่อความง่าย เราจะพิจารณาภาษา Assfuck สำหรับโปรแกรม B ซึ่งดัดแปลงมาจากภาษา Brainfuck โดยภาษา

Assfuck จะมีวิธีการทำงานดังนี้

เริ่มต้นให้พิจารณาว่ามีแถวลำดับ (array) ขนาดไม่จำกัด (สามารถเดินไปทางซ้ายและทางขวาได้เสมอ) ที่ทุกช่องมีค่าเป็นศูนย์ และตัวโปรแกรมจะมีหัวอ่าน/เขียนอยู่ที่ช่องช่องหนึ่ง เรียกเป็นช่องเริ่มต้น ต่อมาคุณสามารถบังคับให้หัวอ่านเดินไปทางซ้าย เดินไปทางขวา เพิ่มค่าช่องนั้น ลดค่าช่องนั้น และทำโปรแกรมแบบวนซ้ำ

โดยจะมี syntax ดังนี้

สัญลักษณ์	ความหมาย
+	เพิ่มค่าช่องปัจจุบันไปหนึ่งหน่วย
–	ลดค่าช่องปัจจุบันไปหนึ่งหน่วย
<	เลื่อนไปทางช่องด้านซ้าย
>	เลื่อนไปทางช่องด้านขวา
[และ]	เมื่อเข้าสู่วงเล็บสี่เหลี่ยมหนึ่งชั้น จะทำการตรวจสอบว่าช่องปัจจุบันเป็นศูนย์หรือไม่ หากเป็นศูนย์จะข้ามจาก [ไปยัง] (ข้ามคำสั่งทั้งหมดที่อยู่ข้างใน) หากไม่เป็นศูนย์ จะทำการปฏิบัติตามคำสั่งที่อยู่ข้างในจนครบตามลำดับแล้ววนกลับมาพิจารณาจุดเริ่มต้นอีกครั้ง

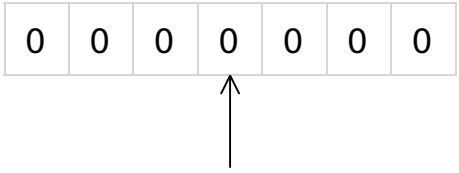
หมายเหตุ

- ช่องแต่ละช่องสามารถเก็บจำนวนเต็มขนาดภายใต้ตัวแปร `int` ของภาษา C++
- จำนวนการดำเนินการคำสั่งทั้งหมดนั้นไม่จำกัด แต่จะจำกัดตามเวลาที่ใช้
- จำนวนช่องใน array จะมี 2\,000,000 ช่อง ช่องปัจจุบันจะอยู่ตรงกลาง array
- เพื่อให้สะดวกต่อการใช้งาน สามารถแก้โค้ดจาก `grader.cpp` เพื่อสร้าง simulator ได้
- เพื่อความปลอดภัย โปรดระวังการเรียกใช้งานหน่วยความจำที่มากเกินไปหากเขียนข้อมูลนำเข้าในบางรูปแบบ (เช่น `[>+]`)
- ผลลัพธ์ของการทำงานคือค่าของช่องที่หัวอ่านอยู่เมื่อจบการทำงานแล้ว

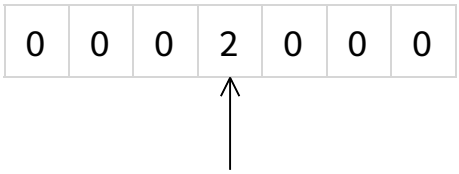
ตัวอย่างการทำงานของโปรแกรมภาษา Assfuck

ยกตัวอย่างการทำงานของโปรแกรม `++>->+++>--<-<->[->+<]>+` ได้ดังนี้

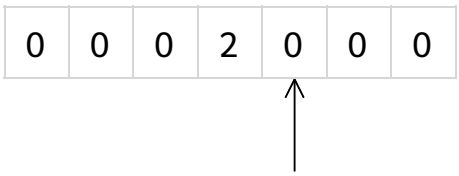
เริ่มต้นจะมีแถวลำดับอนันต์ดังนี้ โดยลูกศรแทนหัวอ่าน/เขียน (จริง ๆ แล้วเป็นแถวลำดับอนันต์ แต่เพื่อการแสดงผลจึงตัดมาเฉพาะส่วนที่ใกล้กับหัวอ่าน/เขียน)



- ด้านซ้ายของโปรแกรมมีคำสั่ง `++` เมื่อดำเนินการแล้วจะได้ผลลัพธ์ดังนี้



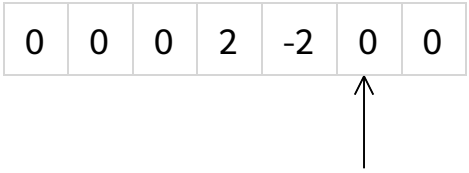
- ต่อมามีคำสั่ง `>` ซึ่งเมื่อดำเนินการแล้วได้ผลลัพธ์ดังนี้



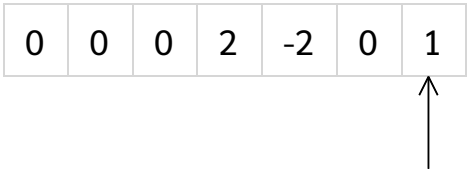
- หลังจากนั้นจะมีคำสั่ง `->+++>--<-<->` ซึ่งเมื่อดำเนินการแล้วจะได้ผลลัพธ์ดังนี้

- ต่อมามีคำสั่ง `[->+<]` โดยจะเริ่มจากการตรวจสอบว่าช่องปัจจุบันเป็นศูนย์หรือไม่ เนื่องจากช่องปัจจุบันเป็น 2 ซึ่งไม่ใช่ 0 จึงทำการดำเนินการภายในลูป ซึ่งก็คือ `->+<` ซึ่งเมื่อดำเนินการเสร็จแล้วจะได้ผลลัพธ์ดังนี้

- ต่อมาจะกลับมาพิจารณา `[->+<]` ใหม่อีกครั้งหลังครบลูปแล้ว ซึ่งเนื่องจากช่องปัจจุบันยังไม่เป็นศูนย์ จึงทำซ้ำอีกรอบ ได้ผลลัพธ์ดังนี้



- ในตอนนี้ได้ผลลัพธ์เป็นศูนย์แล้ว จึงจบการทำลูป
- ต่อมามีคำสั่ง `>+` ก็จะเลื่อนไปด้านขวา แล้วเพิ่มค่า จึงได้ `1` ดังนี้



- ผลลัพธ์ของการดำเนินการโปรแกรมนี้จึงเป็น `1`

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันดังต่อไปนี้

```
string A(int T, int C)
```

- ฟังก์ชันนี้จะถูกเรียกหลายครั้ง (ไม่เกิน 100 ครั้งต่อข้อมูลทดสอบ) โดยรับจำนวนเต็ม T กับจำนวนเต็ม C ตามลำดับ
- จำนวนเต็ม T แทนความยาวสตริง `B` ที่สั้นที่สุดที่ผู้จัดการแข่งขันหาได้
- ฟังก์ชันนี้จะต้องคืนค่าสตริง `B` ซึ่งเป็นโปรแกรมภาษา Assfuck ประกอบด้วยอักขระ `+-<>[]` เท่านั้นความยาวไม่เกินหนึ่งล้านตัวอักษร
- ในเกรดเดอร์จริงจะมีการตรวจสอบความถูกต้องของโปรแกรม ซึ่งอาจจะใช้เวลารวมมากกว่า 10 วินาที ดังนั้นฟังก์ชันของคุณไม่ควรจะใช้เวลาทำงานนานนัก

ขอบเขต

- $1 \leq C \leq 1\,000$
- $C \neq 0$

ปัญหาย่อย

1. (9 คะแนน) $1 \leq C \leq 9$
2. (16 คะแนน) $10 \leq C \leq 99$
3. (40 คะแนน) $100 \leq C \leq 255$
4. (10 คะแนน) $C = 2^k$ สำหรับบางจำนวนเต็มบวก k
5. (25 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม

การให้คะแนน

สำหรับข้อนี้ หากผลลัพธ์ `B` เมื่อนำไปดำเนินการแล้วไม่ได้ผลลัพธ์เป็นค่า `C` จะได้ 0 คะแนนทันที

ต่อไปนี้จะสมมติว่าได้ผลลัพธ์ที่ถูกต้อง ในแต่ละปัญหาย่อย หาก S เป็นความยาวสตริง `B` ที่ผู้เข้าแข่งขันหาได้ และ T เป็นความยาวสตริง `B` ที่ผู้จัดการแข่งขันหาได้ แล้วในแต่ละข้อมูลทดสอบ ผู้เข้าแข่งขันจะได้คะแนนดังนี้

เงื่อนไข	อัตราส่วนคะแนนต่อคะแนนเต็มของข้อมูลทดสอบ
$S \le T$	1.0
$T < S \le 4T$	$e^{\frac{2}{3} - \frac{2S}{3T}}$
$4T < S \le 10T$	$0.17 - \frac{S}{100T}$
$10T < S \le 1\,000\,000$	0.07
$S > 1\,000\,000$	0.0

ในแต่ละปัญหาย่อย คะแนนของปัญหาย่อยจะเป็น **ค่าต่ำสุด** ของคะแนนของแต่ละข้อมูลทดสอบ คะแนนผลลัพธ์ท้ายสุดจะคิดจากทศนิยมสองตำแหน่ง คะแนนของแต่ละข้อมูลทดสอบจะเป็น **ค่าต่ำสุด** ของคะแนนของการเรียกฟังก์ชันแต่ละครั้ง

ตัวอย่าง

```
A(14, 16)
```

- หากผู้เข้าแข่งขันส่งออก `++++++++++++++++` จะได้ผลลัพธ์เป็น 16 ซึ่งเป็นคำตอบที่ถูกต้อง อย่างไรก็ตาม ในกรณีนี้จะได้ $S = 16$ และ $T = 14$ จะเข้าเงื่อนไข $T < S \le 4T$ ทำให้ได้คะแนน $e^{\frac{2}{3} - \frac{2 \times 16}{3 \times 14}}$ ซึ่งมีค่าประมาณ 0.909156442877
- หากผู้เข้าแข่งขันส่งออก `++++[>++++<-]>` จะได้ผลลัพธ์เป็น 16 เช่นกัน แต่ในกรณีนี้ $S = T = 14$ ผู้เข้าแข่งขันจึงได้รับคะแนน 1.0 ซึ่งเป็นคะแนนเต็มสำหรับข้อมูลทดสอบ

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลดังต่อไปนี้:

- บรรทัดที่ 1: T C

เกรดเดอร์ตัวอย่างจะส่งออกข้อมูลสองบรรทัด ดังนี้:

- บรรทัดที่ 1: B
- บรรทัดที่ 2: อัตราส่วนคะแนน

ข้อจำกัด

- Time limit: 15 second
- Memory limit: 512 MB