Homework 4 Cellular Automata

Document version 1.1, 1 Sept 2021, 17:42

การบ้านคราวนี้คือการเขียนโปรแกรมเพื่อสร้าง cellular automata องค์ประกอบของโปรแกรมที่ใช้มี if และการดึงข้อมูลจาก List

เราเริ่มจากการเข้าใจ cellular automata ก่อน มันคือกฎเกณฑ์ที่กำหนดการเปลี่ยนแปลงของค่าในสิ่งที่เราเรียกว่า cell ในการบ้านคราวนี้เราใช้เซลล์แบบหนึ่งมิติ คือมีช่องเก็บของต่างๆเรียงกันเราใช้list เก็บ คือดูแล้วเป็น list หนึ่งมิติ cellular automata มีประโยชน์ในการอธิบายรูปร่างและปรากฏการณ์ต่างๆทางธรรมชาติ เช่นเดียวกับจำนวนกลีบของดอกทานตะวันเป็นไปตาม Fibonacci series [ref] เราเชื่อกันว่า cellular automata สามารถใช้อธิบายปรากฏการณ์น่าสนใจอย่างเช่น หิ่งห้อยกระพริบแสงพร้อมกัน หรือ กำเนิดจักรวาลได้

[ref] https://momath.org/home/fibonacci-numbers-of-sunflower-seed-spirals/

ตัวอย่าง cellular automata และคำอธิบายโดยละเอียดให้ไปอ่านจากเว็บไซต์นี้
https://math.hws.edu/eck/js/edge-of-chaos/CA-info.html
ซึ่งสามารถทดลองเล่นรูปแบบต่างๆได้ด้วย

ตอนนี้มาพูดถึงที่เราจะเขียน เซลล์ของเราเป็นขนาดความยาว 20 ช่องแต่ละช่องมีค่าไม่ 0 ก็ 1 การเปลี่ยนแปลงของค่าในช่องหนึ่งขึ้นอยู่กับเพื่อนบ้านทั้งสองข้าง ในการบ้านนี้กำหนดให้เพื่อนบ้าน = 5 นั่นก็คือดูไปทางซ้ายสองช่อง ทางขวาสองช่อง สมมติตอนแรกสุด เซลล์ของเรามีเป็นศูนย์หนึ่งสลับกัน ดังนี้
[0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]

เราสร้างกฎว่า 10101 จะทำให้เซลล์ตรงกลาง ในช่วงเวลาถัดไปมีค่าเป็นหนึ่ง

การรัน ก็คือการใช้กฎนี้กับค่าของเซลล์เริ่มต้นทุกๆเซลล์ เพื่อสร้างรูปแบบของเซลล์ในเวลาถัดไป เนื่องจากเซลล์ของเรามีค่า index ตั้งแต่ 0 ถึง 19 และจะต้องเผื่อเพื่อนบ้านทั้งซ้ายทั้งขวาสองช่อง ดังนั้นเซลล์ที่เราจะคำนวณ ก็คือเซลล์ที่มี index 2 ถึง 17

เริ่มต้น เราใช้กฎกับเซลล์ที่ 2 จะได้เป็นหนึ่ง ค่านี้ต้องบันทึกใน list อันใหม่ที่ตำแหน่งเดียวกัน เราทำไปเรื่อยเรื่อยทีละเซลล์จนถึง index ที่ 17 ก็จะได้ค่าเซลล์ใหม่ ครบอย่างนี้เรียกว่าทำงานครบหนึ่งรอบ เราทำอย่างนี้ซ้ำไปหลายรอบ ค่าของเซลล์ก็จะเปลี่ยนไปเรื่อยเรื่อย ทำให้เกิดรูปร่างต่างๆ

ตัวอย่างเช่น รัน โดย ใช้กฎ 10101 กับ list เริ่มต้นข้างบน หนึ่งครั้ง จะได้เอาท์พุต
[0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]

รัน ต่ออีกครั้งจะได้
[0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0]

ลักษณะรูปร่างต่างๆนี้ขึ้นอยู่กับค่าเริ่มต้นของเซลล์และกฎที่ใช้ กฎที่ใช้ก็อาจจะใช้มากกว่าหนึ่งกฎเกณฑ์ก็ได้ ในตัวอย่างในเว็บไซต์ที่กล่าวถึงมีกฎเกณฑ์ต่างๆเป็นร้อยรูปแบบ

เรามาดูโปรแกรมกัน

โปรแกรมหลักของเราก็คือ การคำนวณกฎไปที่แต่ละเซลล์ โดยอินพุตเป็น list เริ่มต้นและเอาท์พุตจะเป็น list ที่เสร็จแล้ว เราทำอย่างนี้ซ้ำๆกัน 19 รอบเพื่อจะผลิตรูปจะมีขนาด 20 × 20 (runrule, run20)

รอบแรกอินพุตเป็น list เริ่มต้นที่เรากำหนด จะได้เอาท์พุตเป็น list ใหม่ รอบต่อไป ก๊อปปี้จาก list ใหม่ ไปใส่ list อินพุตเพื่อเตรียมทำซ้ำ จนได้ 19 รอบก็จะได้รูป 20 x 20

โปรแกรมหลักที่คุณต้องเขียนก็คือการคำนวณผลของกฎที่เซลล์เซลล์หนึ่ง (rule1)

ยกตัวอย่างเช่นเพื่อนบ้านของเราขนาด 5 เราใส่กฎสองข้อคือรูปแบบ 10100 เอาท์พุตจะเป็นหนึ่ง หรือ 01000 เอาท์พุตจะเป็นหนึ่ง เอาผลลัพธ์ของแต่ละกฎมา or กัน (การ or คือ อันใดอันหนึ่งเป็น 1 ผลจะเป็น 1) จะเขียนอย่างนี้คุณต้องใช้การดึงค่าของเซลล์ทีละช่อง มาตรวจสอบโดยใช้ if และในเงื่อนไขก็ใช้ logical expression ประเภท and or

ข้อสังเกต ถ้ามีหลายกฎ ลำดับการใช้กฎไม่สำคัญ ใครก่อนก็ได้ผลเหมือนกัน

อินพุตของฟังก์ชันนี้ (rule1) ก็มีค่า list และ index ช่องที่กำลังพิจารณา เอาท์พุตของฟังก์ชันคุณก็จะเป็นค่าไม่ 0 หรือ 1

งานของคุณมีสองส่วน ส่วนแรกคือ
1)โปรแกรมของคุณ รับ อินพุต เป็น list เริ่มต้น เป็น list ของเลข 0/1 ยาว 20 ช่อง แล้ว รัน cellular automata นี้ กับ กฎ 10101, 01000 ซ้ำ 19 รอบ ทุกรอบ พิมพ์ list ผลลัพธ์ออกมา จะดูเป็นรูป ขนาด 20 x 20

ตัวอย่าง จาก list เริ่มต้น p0 = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
และกฎ สองข้อ 10101, 01000 รัน 4 รอบ แล้วจะได้
[0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
[0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0]

เพื่อให้รูปสวยขึ้น ใช้ prettyprint() แทน print จะพิมพ์โดยไม่มีช่องว่าง ใน prettyprint ผมใช้สิ่งที่เรียกว่า list comprehension ซึ่งคุณจะได้เรียนในเร็วๆนี้ ตัวอย่าง

```
p0 = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
prettyprint(p0)
>>
01010101010101010101
```

2)ส่วนที่สองของงาน คือ ให้คุณลองสร้าง กฎ ด้วยตัวเอง พร้อมกับ list เริ่มต้น รัน 19 รอบ สร้าง รูปแปลกๆที่น่าสนใจออกมา ส่วนนี้ไม่ต้องรับอินพุต สร้าง รัน พิมพ์เอาท์พุตด้วย prettyprint() ออกมาเลย

การส่งการบ้าน

เขียนโปรแกรมที่รับอินพุต เป็น list เริ่มต้น ใช้กฎ 10101, 01000 รัน 19 พิมพ์ list ทุกรอบ ได้รูป 20 x 20 ต่อมา วิ่งโปรแกรมส่วนที่สอง ใช้ prettyprint พิมพ์ รูป 20 x 20 ออกมา ทั้งสองอย่างนี้ เรียกจาก main()

ตัวอย่างเพิ่มเติม

p0 = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
กฎ 10101, 01000 จะได้

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
[0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

p0 = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]

กฎ 00000, 11111 จะได้

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

```
# Homework 4: 1D cellular automata

# apply rule1 19 times
def run20():
  print(p0)
  pin = list(p0)
  pout = runrule(pin)
  print(pout)
  pin = list(pout)
  pout = runrule(pin)
  . . .

# cell index 0..19
# neibourhood 5
# apply rule to cell index 2..17
def runrule(p):
  pp = 20*[0] # initialise output
  pp[2] = rule1(p,2)
  pp[3] = rule1(p,3)
  . . .
  return pp

def rule1(p,i):
  . . .
  return 0

def prettyprint(p):
  s = "".join([str(e) for e in p])
  print(s)

# p0 = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
def main():
  p0 = input()
  run20()

main()

# -----------------
```