

Assignment 09: การเรียงสับเปลี่ยน (Permutation)

คงรู้จักกันดีแล้วว่า การเรียงสับเปลี่ยน (permutation) คืออะไร ในที่นี้เราแทนการเรียงสับเปลี่ยนขนาด n ด้วยลิสต์ของจำนวน 1 ถึง n ซึ่งมีการเรียงสับเปลี่ยนที่ต่างกันทั้งหมด $n!$ แบบ เช่น เมื่อ $n = 3$ จะมี $3! = 6$ แบบดังนี้ $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$ และ $[3, 2, 1]$

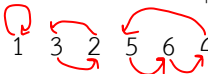
ให้ $p = [p_1, p_2, \dots, p_n]$ และ $q = [q_1, q_2, \dots, q_n]$ เป็นลำดับที่เกิดจากการเรียงสับเปลี่ยนขนาด n ตัว เราเรียก p ว่า "น้อยกว่า" q ก็ต่อเมื่อ มี $p_{\text{pos}} < q_{\text{pos}}$ โดย pos เป็นตำแหน่งน้อยสุดตำแหน่งแรกที่ $p_{\text{pos}} \neq q_{\text{pos}}$ เช่น $[1, 2, 3]$ น้อยกว่า $[3, 2, 1]$, $[1, 2, 4, 3]$ น้อยกว่า $[1, 3, 2, 4]$ เป็นต้น

อันดับของการเรียงสับเปลี่ยน

ด้วยนิยาม "น้อยกว่า" ข้างบนนี้ ทำให้เราสามารถจัดอันดับของการเรียงสับเปลี่ยนทุกรูปแบบได้ เช่น อันดับที่ 1 ถึง 24 ของการเรียงสับเปลี่ยนตัวเลข 1 ถึง 4 เป็นดังนี้ (อันดับที่ 1 คือการเรียงสับเปลี่ยนที่น้อยที่สุด)

อันดับ	การเรียงสับเปลี่ยน	อันดับ	การเรียงสับเปลี่ยน	อันดับ	การเรียงสับเปลี่ยน
1	[1, 2, 3, 4]	9	[2, 3, 1, 4]	17	[3, 4, 1, 2]
2	[1, 2, 4, 3]	10	[2, 3, 4, 1]	18	[3, 4, 2, 1]
3	[1, 3, 2, 4]	11	[2, 4, 1, 3]	19	[4, 1, 2, 3]
4	[1, 3, 4, 2]	12	[2, 4, 3, 1]	20	[4, 1, 3, 2]
5	[1, 4, 2, 3]	13	[3, 1, 2, 4]	21	[4, 2, 1, 3]
6	[1, 4, 3, 2]	14	[3, 1, 4, 2]	22	[4, 2, 3, 1]
7	[2, 1, 3, 4]	15	[3, 2, 1, 4]	23	[4, 3, 1, 2]
8	[2, 1, 4, 3]	16	[3, 2, 4, 1]	24	[4, 3, 2, 1]

วงวนในการเรียงสับเปลี่ยน

หากเราพิจารณาที่ตำแหน่ง i ใด ๆ ของการเรียงสับเปลี่ยน p โดยวิ่งจากตำแหน่ง i ไปที่ตำแหน่ง p_i แล้ววิ่งจากตำแหน่ง p_i ไปที่ตำแหน่ง p_{p_i} แล้วก็ไปที่ ตำแหน่ง $p_{p_{p_i}}$ ทำแบบนี้ไปเรื่อย ๆ จนกระทั่ง $p_{\dots p_i} = i$ ก็จะได้ 1 วงวน เช่น $[1, 3, 2, 5, 6, 4]$ ลองวิ่งตามตำแหน่งจะได้  ลูกศรแสดงการชี้จาก i ไปที่ p_i เรื่อย ๆ ได้วงวน $[1]$, $[3, 2]$ และ $[5, 6, 4]$ ซึ่งมีขนาด 1, 2 และ 3 ตามลำดับ

งานที่ต้องทำ

ให้เขียน 5 ฟังก์ชัน ดังนี้

- **order_of(permutation)** รับลิสต์ของการเรียงสับเปลี่ยน เพื่อหาว่า การเรียงสับเปลี่ยนนี้เป็นอันดับที่เท่าไร
- **permutation_at(order, n)** คืนลิสต์ที่แทนการเรียงสับเปลี่ยนอันดับที่ order ของการเรียงสับเปลี่ยนที่มี n ตัว ถ้าไม่สามารถหาการเรียงสับเปลี่ยนตัวที่ order ได้ ให้คืน None เช่น permutation_at(7,3) ต้องคืน None
- **next_permutation(permutation)** คืนลิสต์ที่แทนการเรียงสับเปลี่ยนอันดับถัดไปจาก permutation ที่ได้รับ ถ้าไม่มีอันดับถัดไป ให้คืน None
- **prev_permutation(permutation)** คืนลิสต์ที่แทนการเรียงสับเปลี่ยนอันดับก่อนหน้า permutation ที่ได้รับ ถ้าไม่มีอันดับก่อนหน้า ให้คืน None
- **longest_cycles(permutation)** คืนลิสต์ที่อยู่ในเก็บลิสต์ของวงวนที่มีขนาดใหญ่ที่สุดของ permutation โดยวงวนขนาดใหญ่สุดอาจมีหลายวงก็ได้ เช่น longest_cycles([7,1,5,3,4,6,2]) จะได้ [[3,5,4], [1,7,2]]
- หมายเหตุ: None เป็นค่าคงตัวใน Python การคืนค่า None คือคำสั่ง return None ไม่ใช่ return "None"

<pre># Prog-09: Permutation # Fill in your ID & Name # ... # Declare that you do this by yourself import math def order_of(permutation): return ??? #----- def permutation_at(order, n): return ??? #----- def next_permutation(permutation): return ??? #----- def prev_permutation(permutation): return ??? #----- def longest_cycles(permutation): return ??? #-----</pre>	<pre>def main(): while True: x = input().split() cmd = x[0] p = [int(e) for e in x[1:]] if cmd == 'O': print(order_of(p)) elif cmd == 'A': print(permutation_at(p[0], p[1])) elif cmd == 'N': print(next_permutation(p)) elif cmd == 'P': print(prev_permutation(p)) elif cmd == 'C': print(longest_cycles(p)) elif cmd == 'Q': return return #----- main()</pre>
---	--

download code นี้ได้

ไม่ใช่ตัวแปรใด ๆ ที่อยู่นอกฟังก์ชัน

ไม่เพิ่ม ลบ หรือ เปลี่ยนแปลง บริเวณคำสั่ง สีแดง เต็มขาด

ปฏิบัติตามอย่างเคร่งครัด

- ห้ามเขียนคำสั่งในฟังก์ชันที่ใช้ตัวแปรที่อยู่นอกฟังก์ชัน (หรือที่เรียกว่าตัวแปร global)
- อนุญาตให้เพิ่มคำสั่งในบริเวณพื้นที่สีขาวเท่านั้น จะเขียนฟังก์ชันเพิ่มเติมก็ได้
- ตั้งชื่อแฟ้ม ให้ถูกต้องตามที่เขียนใน CourseVille
- เพิ่มเติมข้อมูลใน comment ต้นโปรแกรมให้ตรงตามความจริง
- ก่อนส่งโปรแกรม ควรส่งงานโปรแกรมที่ส่งอีกครั้ง ว่าทำงานได้ตามที่ต้องการ
- ไม่อนุญาตให้ import คลังคำสั่งใด ๆ เพิ่มเติม (นอกจาก math ที่เขียนให้แล้ว)

ตัวอย่าง

Input	Output
O 1 2 3 4 5 O 5 4 3 2 1 O 3 1 4 5 2 O 10 9 1 11 3 4 12 6 7 2 5 8 Q	1 120 52 388570261
A 1 5 A 120 5 A 52 5 A 388570261 12 A 121 5 Q	[1, 2, 3, 4, 5] [5, 4, 3, 2, 1] [3, 1, 4, 5, 2] [10, 9, 1, 11, 3, 4, 12, 6, 7, 2, 5, 8] None
N 1 2 3 4 5 6 7 N 7 6 5 4 3 2 1 N 10 9 1 11 3 4 12 6 8 7 5 2 Q	[1, 2, 3, 4, 5, 7, 6] None [10, 9, 1, 11, 3, 4, 12, 7, 2, 5, 6, 8]
P 1 2 3 4 5 6 P 6 5 4 3 2 1 P 10 9 1 11 3 4 12 7 2 5 6 8 Q	None [6, 5, 4, 3, 1, 2] [10, 9, 1, 11, 3, 4, 12, 6, 8, 7, 5, 2]
C 1 2 3 4 5 6 C 6 5 4 3 2 1 C 6 1 2 3 4 5 C 9 6 2 11 12 10 8 7 5 3 4 1 Q	[[1], [2], [3], [4], [5], [6]] [[4, 3], [5, 2], [6, 1]] [[6, 5, 4, 3, 2, 1]] [[6, 10, 3, 2], [9, 5, 12, 1]]
O 19 18 3 21 22 24 27 15 13 28 17 7 6 5 11 14 A 5674182293293315513954093042784 29 C 19 18 3 21 22 24 27 15 13 28 17 7 6 5 11 14 Q	12 20 25 26 10 4 8 2 29 9 1 23 16 หาคำตอบเอง

หมายเหตุ: สำหรับผลลัพธ์ของ longest_cycle อาจมีลำดับที่ไม่เหมือนในตัวอย่างก็ได้ เช่น

[[6, 10, 3, 2], [9, 5, 12, 1]] อาจเป็น [[1, 9, 5, 12], [3, 2, 6, 10]] หรือแบบอื่นก็ได้

คำเตือน

- ข้อมูลที่ใช้ทดสอบจะมีขนาดใหญ่ เช่น หากการเรียงสับเปลี่ยนที่มีตัวเลขเป็นร้อยตัว หรือหาอันดับที่เป็นเลขร้อยหลัก จึงไม่ควรใช้วิธีการค่อย ๆ แจกการเรียงสับเปลี่ยนไล่อันดับจาก 1 ไปเรื่อย ๆ จะช้ามาก ๆ เวลาที่ทำงานในแต่ละฟังก์ชันควรจะเร็วมาก ๆ เป็นเสียของเสียวินาที กับการเรียงสับเปลี่ยนเป็นร้อยตัว หรืออันดับเป็นร้อยหลัก